A Journey Through Computation
Natural and Human

Karl Fant

With gratitude to
Steve Johnson and Chris Nicol
for
listening and understanding

Table of Contents

Chapter 1:
Subtleties of primitivity

Sequence controlled steps of behavior, considered the fundamental model of computation, and Boolean logic, considered the most primitive computational behavior do not provide a complete and coherent accounting of computation.

1.1. Sequentiality.

"Computation is the evolution process of some environment by a sequence of "simple, local" steps." [1]

Sequential computation is represented one step at a time. Each step is a behavior that receives an input from a state environment and returns an output to the state environment evolving the state environment step by step. Each step completes and delivers its output before the next step begins ensuring that each next step receives a stable input from a stable state environment.

1.1.1. The innate concurrency of sequentiality

A sequential ordering of steps does not, in itself, specify a computation. A sequence of steps has to realize the correct flow of dependency relations through the state environment. Dependency of flow relations includes relations that can occur "all at once" or "in any order". The, "all at once", represents concurrency. The, "in any order", means that the concurrent steps can be mapped to a sequential ordering of steps. But "in any order" also means that there can be a multitude of different sequential orderings that correctly realize the flow of dependency relations of the computation. Further, there is a multitude of sequential orders that do not correctly realize the dependency relations.
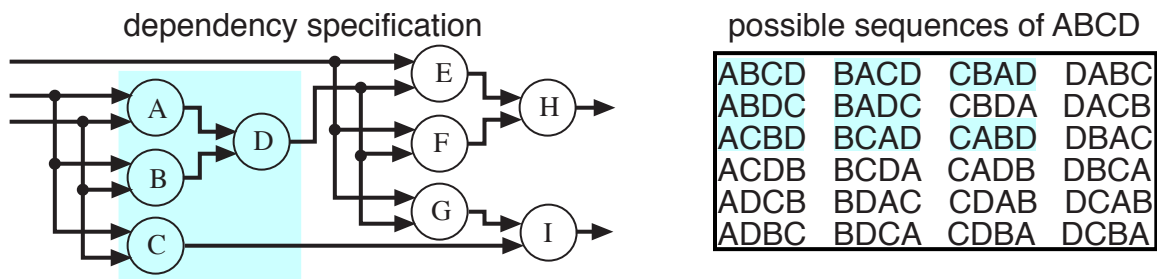
Figure 1.1. Sequencing concurrency

The left of Figure 1.1 represents a network of dependency relations. In the shaded region of the network A and B can occur in any order before D. If either A or B occur after D the sequence is incorrect. C can occur in any order with A, B and D. The right of Figure 1.1 represents all the possible sequences of A, B, C and D. The shaded sequences are the eight sequential orderings realizing the correct flow of dependency derived from the similarly shaded portion of the dependency network. The unshaded 16 possible sequences are all incorrect (D occurs before A or B or both). With such variety of sequence it can be difficult to be confident of the correctness of a specific sequential order and even more difficult to reliably perceive incorrectness. The only means of differentiating a correct sequential ordering from an incorrect sequential ordering in a typically enormous set of possible sequential orderings is to refer to the unique dependency of flow representation with its innate concurrent relations.

1.2.   QUANDARY 1: Dependency

Sequential ordering is necessarily derived from dependency relations so the expression of dependency relations with their innate concurrency must be considered to be more primitive than the expression of sequential order.

1.2.1.   The sequence controller

Steps do not sequence themselves. Sequentiality requires a sequence controller that actualizes each step in turn, connects it with the state environment and determines its completeness of output before beginning the next step. A sequence controller cannot control itself but can be represented as a sequence of more primitive steps controlled by a more primitive sequence controller forming a hierarchy of sequence controllers such as program, instructions, microcode and so on. However, there must be a most primitive sequence controller which cannot itself be sequence controlled.

1.3.   QUANDARY 2: Primitivity

Sequentiality cannot be primitive. Its most primitive sequence controller must be expressed, not in terms of a sequence of steps but directly in terms of dependency relations among primitive behaviors which dependency relations include innate concurrency relations.

That sequential interpretation provides a universal realizer of every possible computation does not necessarily mean that sequential interpretation represents a primitive essence of computation.

1.4.   Concurrency

Concurrency relations are conventionally viewed from simply insignificant

> *"All we would lose by the omission of "parallel processing" is speed, nothing fundamental."* [2]

to fundamentally unrealizable.

> *"The introduction of concurrency into computation opens Pandora's box to release the possibility of non determinacy and a host of other complications, including deadlock and livelock .... Events within concurrent systems do not necessarily occur at predetermined times, but may occur in different orders depending upon the relative speeds of different system parts. The usual assumption, which is also physically realistic, is that relative speeds cannot be controlled so precisely that the designer or programmer can depend upon them to enforce sequencing requirements."* [3]

1.5.   Boolean logic

The most primitive behaviors are conventionally understood to be Boolean logic behaviors because they are primitive to a mathematician, the arbitrarily capable agency at the heart of mathematics, who can correctly interpret the behavior of a Boolean network and its concurrent relations step by step with her pencil and paper. In the absence of the mathematician, a network of Boolean logic behaviors relying on the merits of its intrinsic logical behaviors, does not work.

Enlivened Boolean functions are continually responsive to their input and are continually asserting output dependent on the input. Two behaviors, each presenting an input to a third behavior, can behave independently and concurrently with different delays. The two input transitions to the third behavior can arrive at different times causing the third behavior to temporarily output an incorrect result transition (a glitch). This temporarily incorrect result transition will be presented to subsequent behaviors causing them to temporarily transition their output to incorrect result transitions which will race ahead though the network of behaviors asserting a chaos of incorrect transitions at the output of the network. The network and its Boolean behaviors cannot, on their own, determine amidst the chaotic transitioning of incorrect results when its input has completely and stably transitioned nor when the network output is completed with the correct result of the presented input. This is the Pandora's box of non determinacy mentioned in the quote above.

1.6.　　QUANDARY 3: Insufficient primitivity
Boolean logic behaviors, which are insufficiently expressive on their own behavioral merits to coordinate the innate concurrent relations in a network of dependently related Boolean logic behaviors, do not present a viable primitivity.

1.6.1.　　Hiding the chaos of concurrency
If the inputs to a Boolean logic network are held stably for long enough correct results eventually propagate through the behaviors and the network stabilizes asserting the complete and correct output for the presented input. The chaotic behavior of a Boolean logic combinational network can be hidden and remediated by isolating and bounding it with memories (registers - data lifeboats) controlled by a time interval long enough to allow the network to stabilize. At the beginning of the time interval an input memory presents the input to the network, the time interval waits long enough for the network to settle to a stable output then at the end of the time interval an output memory accepts the correct and stable output. The chaotic behavior of the Boolean network is isolated and hidden within the time interval and between the bounding memories.

Successive time intervals present a sequence of inputs to the network and produce a sequence of outputs. The chaotic Boolean logic network with the crutch of the bounding memories and time interval becomes a timed sequential step of determined computation.

1.6.2.　　Sequential steps of hidden concurrency
These memory and time interval bounded Boolean networks are composed into a network of bounded Boolean networks all timed with the same time interval long enough to accommodate the slowest to stabilize network (critical path). All of the bounded networks controlled by the same common time interval, behave concurrently (all at once) within each time interval. With the single common time interval precisely controlling the start and end of all the networks, the concurrently behaving bounded Boolean networks fulfill the above author's requirement of precisely controlled relative speeds for reliable concurrent behavior. With each successive time interval results flow from bounded Boolean network to bounded Boolean network. The innate and chaotic concurrency relations isolated inside each bounded Boolean network are sequence controlled by the timing interval and the memories. In this way a most primitive sequence

controller can be realized as a network of bounded Boolean networks all behaving concurrently and determinately to realize a single step of sequence control.

1.7.   QUANDARY 4: Concurrency myopia

Timed sequence control has vanquished the chaotic concurrency of Boolean networks enabling the construction of a most primitive sequence controller. Might sequence control be considered necessary and foundational if not primitive?

Yet, even with the veneer of sequence control, the representation of sequenced steps at all levels, still derives from and is preceded by dependency relations with their innate concurrency and the most primitive sequence controller is realized with time bounded Boolean logic networks within each of which lingers the chaotic concurrency of the most primitive dependency relations.

Despite the immense computational success of sequentiality and Boolean logic over the past 70 years they fail to completely and coherently account for computation. A complete and coherent accounting of computation must embrace the primitivity of dependency relations and their innate concurrency.

1.8.   Seeking a sufficient accounting

The straightforward goal of the journey is to discover an accounting of computational interaction that directly expresses dependency flow with its innate primitive concurrency relations and which evolves to encompass all forms of computational interaction. An accounting that provides a common characterization of interaction whether in a biological cell, in a digital computer or in a mathematician's head. An accounting capable of characterizing the mathematician herself as well as what she does with her pencil and paper. An accounting from which familiar forms of interaction, Boolean logic with its time interval and sequential interpretation with its notion of state, emerge through considerations quite different from those of their historic development. An accounting of computational interaction in terms of a primitivity complete and sufficient in itself with nothing hidden and in no need of any extrinsic support.

---

1. Avi Wigderson "Mathematics + Computation, The theory revolutionizing technology and science", (Princeton, New Jersey, Princeton University Press, 2019). p. 307.
2. Richard Feynman, "*Feynman Lectures On Computation*" (Reading, Massachusetts, Addison Wesley, 1996) p 4.
3. Charles Seitz ed., *Resources in Parallel and Concurrent Systems*, ACM Press, New York, 1991, introduction, page ix.

Chapter 2:
Condition Differentiation

The journey of discovery begins with the notion of differentness spontaneously and dependently interacting and changing as first principle in contrast to regarding a mathematician with pencil and paper as first principle.

2.1.   Differentness of condition

Condition is a common term representing differentness. Each differentness is represented as a condition that can interact its differentness with one or more other different conditions and change condition differentness. A condition is different from all other conditions in its propensity to spontaneously interact with specific other different conditions and to not interact with all remaining different conditions. Interacting conditions of differentness symmetrically and equally appreciating each others differentness by changing is the essence of computational interaction. The different after conditions are the appreciation the differentness of the interacting conditions. A differentness of condition that never interacts and never changes is not appreciable. A differentness of condition that changes spontaneously without interacting is not appreciated.

2.2.   Sameness of persistence

Persistence is a common term representing sameness. The sameness of a persistence spans and relates a specific after differentness as the appreciation of a specific before differentness of a specific interaction change. Without a spanning persistence a newly appearing after differentness is not an appreciation of any specific before differentness.

Sameness and differentness are codependent and collaborative. Codependent in the sense that sameness presents in relation to differentness and differentness presents in relation to sameness. Collaborative in the sense that the spanning sameness of persistence links the changing differentness conditions of interaction and the interaction propensities of differentness of condition links the sameness of persistences mutually and indefinitely renewing each's expressivity.

For this narrative persistence will be characterized as a carrier of condition differentness that asserts one at a time of two or more mutually exclusive condition differentnesses each with a specific interaction propensity. When an interaction occurs the persistence changes it asserted condition to the interaction result condition linking the after condition with the before condition through the sameness of the persistence and ensuring the beforeness and afterness of the interacting conditions with its mutually exclusive one at a time assertion of condition.

2.3.   Place of common association

Persistences indiscriminately associate in a place of common association which is itself a spanning persistence and which might be a gravity well, a cell membrane, a concentration gradient, a tide pool, a community and so on. When two or more asserted conditions encounter they either interact or do not interact. If the encountering conditions have no propensity to interact then nothing happens. Each persistence with its asserted condition moves on with no lingering consequences of the failure to interact. If the encountering conditions have a propensity to interact the conditions interact with the asserting persistences spontaneously changing their asserted condition. A molecule, for instance, is a persistence that asserts a chemical condition

that may spontaneously interact with and change its chemical condition only in combination with specific other chemical conditions.

The place of common indiscriminate association for this discussion is represented by a shaking bag, illustrated in Figure 2.1. The bag prevents contained persistences and their asserted conditions from wandering off and prevents external persistences from intruding. The shaking ensures that each asserted condition encounters all the other asserted conditions in the bag.
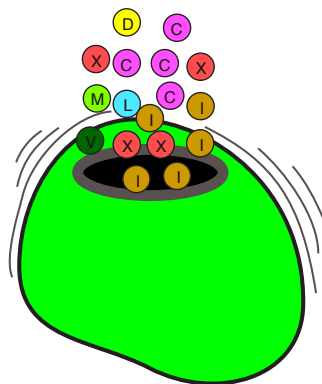
Figure 2.1.   Persistences and their conditions agitated in a place of common association.

2.3.1.   Interaction dependency

As conditions interact and change the interacting conditions disappear and different result conditions with different interaction propensities appear which interact further and change into further different result conditions forming progressions of dependent interactions within the bag represented by the different conditions and their specific interaction propensities.

2.3.2.   Interaction coordination

Interactions coordinate their dependency relations with completeness relations. An interaction begins with completeness of input, when all of the interacting conditions are sufficiently proximally associated. An interaction is completed when the interacting conditions disappear and the interaction result conditions appear. The appearance of an interaction result condition is sufficient to imply that the interacting conditions were completely present and that the interaction has completed.

2.3.3.   Pure condition differentiation

Because all specificity of interaction is in terms of differentness of conditions and their interaction propensities this form of interaction is called **pure condition differentiation**.

The familiar exemplar of pure condition differentiation is proteins in the cytoplasm of a biological cell. Each protein molecule is a persistence and the folding of each protein molecule manifests two or more unique configuration conditions determining its interaction with other molecular conditions. The warm cytoplasm and the cell membrane form the agitating place of common association.

2.4.   A familiar example of pure condition interaction: Roman numeral addition

Pure condition interaction is illustrated with Roman numeral addition considered without the subtractive principle: 9 is **VIIII** instead of **IX**, 40 is **XXXX** instead of **XL** and so on. A Roman numeral is represented by the **possible conditions**: **I**, **V**, **X**, **C**, **D**, **M**. Without the subtractive

principle Roman numerals are association independent. Place in relation does not contribute to differentiation. The subtractive principle which may have been invented by stone carvers as an economy measure is not intrinsic to Roman numerals.[1] For example:

XLII = LXII = ILXI = IILX = LIIX = LIXI = IXIL = XILI = XIIL = IIXL = … = 62

Roman numeral addition is expressed with the interaction propensities shown in Figure 2.2.

$$[I,I,I,I,I] \Rightarrow V$$
$$[V,V] \Rightarrow X$$
$$[X,X,X,X,X] \Rightarrow L$$
$$[L,L] \Rightarrow C$$
$$[C,C,C,C,C] \Rightarrow D$$
$$[D,D] \Rightarrow M$$

Figure 2.2.   Roman Numeral interaction fulfillment relations.

The conditions embody the interaction propensities. When two **V**s encounter they fulfill the interaction relation [**V,V**] and spontaneously change into **X**. Given two Roman numerals these interaction propensities will reduce them to a minimal single numeral representing their addition. The numbers 1878 and 122 are used as example.

MDCCCLXXVIII + CXXII = MM

The two example numerals are placed into a shaking bag as in Figure 2.3 below. The five **I**s will fulfill relation [**I,I,I,I,I**], and change into a **V**. There are then two **V**s that will change to an **X** resulting in five **X**s which will change to an **L** resulting in two **L**s which will change to a **C** resulting in five **C**s which will change to a **D** resulting in two **D**s which will change to an **M**. What remains in the bag are two **M**s. There is no interaction propensity with fulfillment [**M,M**] so no more interactions can occur and the addition is completed. Conditions associate, interact according to their propensities and the Roman numeral sum appears.
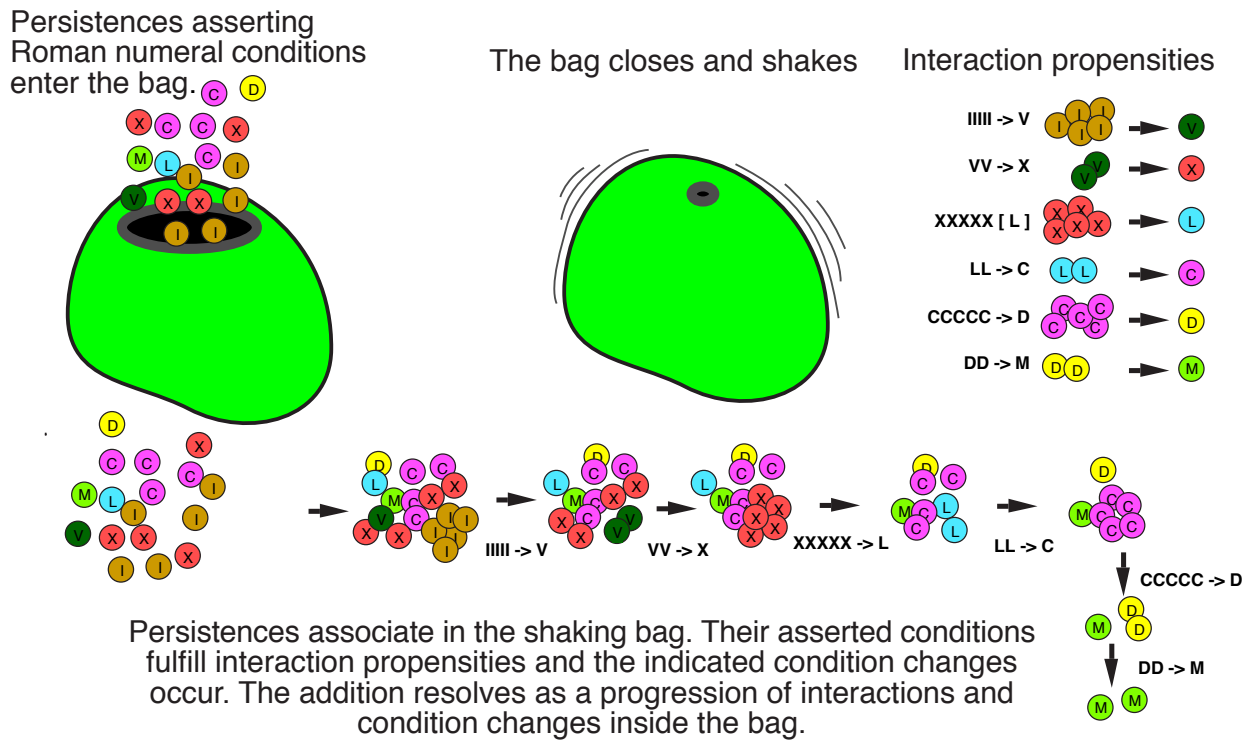
Figure 2.3.   Roman numeral addition in shaking bag.

A fully determined numerical interaction occurs as a progression of discriminate interactions of different conditions according to interaction propensities while conditions indeterminately associate inside the shaking bag

2.4.1.   Interaction incompleteness

But the addition interaction is not complete in itself. It cannot, itself, determine when its interaction is completed. As with a Boolean logic network the only way to determine the progress of the addition is with an external agency which, in this case, must open the bag and perform a complicated count of the conditions. When there are four or fewer of **I**, **X**, **C** and one or fewer of **V**, **L**, **D** the addition is completed.

2.4.2.   Completeness of expression

Intrinsically determining when a interaction is done requires that there be a necessarily last interaction propensity fulfilled. With the present form of the expression there might not even be a first interaction. **VI** + **XII** = **XVIII** is done with no interaction behavior at all. There must be a completeness of behavior at each stage of interaction to insure that every interaction propensity is fulfilled in an orderly progression to the necessarily last interaction.

Completeness of behavior requires a completeness of representation. The first question of an addition is, how many **I**s are present in the bag. The interaction must count the **I**s and somehow determine that it has considered all the **I**s and has not missed any **I**s in the bag. This counting can be accomplished if the number of **I**s in the bag to be counted is always the same. This constancy of quantity of Is is arranged with buffer conditions such that there is always the same number of each condition and its buffer condition in each Roman numeral. The corresponding lower case letter will be used as the buffer condition for each numeral condition as shown in Figure 2.4.

Numeral condition	Buffer condition
I	**i**
V	**v**
X	**x**
J	**l**
C	**c**
D	**d**
M	**m**

Figure 2.4.   Roman numerals with their associated buffer conditions.

The buffer condition **i** is used such that there is always exactly four of **I** and/or **i** in a Roman numeral: **iiii**, **Iiii**, **IIii**, **IIIi** or **IIII**. The buffer conditions serve a role similar to zero in place value numbers. When two numerals are added there will always be exactly eight of **I** and/or **i**. The criterion for completeness, the counting of **I/i,** can now be represented by the completeness fulfillment of each **I/i** Interaction propensity, shown in Figure 2.5 below, which is exactly eight **I/i** conditions. Each completeness of association fulfillment is an unordered association of conditions. There are no order relationships in the bag. The Interaction propensity fulfillment will occur only when the proximal association of all eight conditions is completely formed.

Interaction propensities `for` **`I,i`**
```
[i,i,i,i,i,i,i,i] => [i,i,i,i,v]
[I,i,i,i,i,i,i,i] => [I,i,i,i,v]
[I,I,i,i,i,i,i,i] => [I,I,i,i,v]
[I,I,I,i,i,i,i,i] => [I,I,I,i,v]
[I,I,I,I,i,i,i,i] => [I,I,I,I,v]
[I,I,I,I,I,i,i,i] => [i,i,i,i,V]
[I,I,I,I,I,I,i,i] => [I,i,i,i,V]
[I,I,I,I,I,I,I,i] => [I,I,i,i,V]
[I,I,I,I,I,I,I,I] => [I,I,I,i,V]
```

Interaction propensities `for` **`V,v`**
```
[v,v,v] => [v,x]
[V,v,v] => [V,x]
[V,V,v] => [v,X]
[V,V,V] => [V,X]
```

Figure 2.5.   Interaction propensities for I,i and V,v.

The result of the interaction of eight **I/i** is four **I/i** and one **V/v**. The one **V/v** is the carry to the **V/v** interaction. There is one of **V/v** in each Roman numeral and there will always be a carry condition of **V** or **v** so there will always be exactly three of **V/v** present after the carry. The **Vv** Interaction propensity fulillments require three of **V/v** in Figure 2.5 above ensuring that the **V/v** interaction occurs strictly after the **I/i** interaction. The **V/v** interaction produces one of **V/v** and one of **X/x**.

There will be four of **X/x** in each Roman numeral so adding two Roman numerals will involve eight **X/x** conditions and the carry **X/x** condition making exactly nine**X/x** conditions to form the **X/x** Interaction propensity fulfillments shown in Figure 2.6. Again, because of the carry, the **X/x** interaction will occur strictly after the **V/v** interaction.

```
            Interaction propensies for X,x
[x,x,x,x,x,x,x,x,x] => [x,x,x,x,l]
[X,x,x,x,x,x,x,x,x] => [X,x,x,x,l]              Interaction propensies for L,l
[X,X,x,x,x,x,x,x,x] => [X,X,x,x,l]                    [l,l,l] => [l,c]
[X,X,X,x,x,x,x,x,x] => [X,X,X,x,l]                    [L,l,l] => [L,c]
[X,X,X,X,x,x,x,x,x] => [X,X,X,X,l]                    [L,L,l] => [l,C]
[X,X,X,X,X,x,x,x,x] => [x,x,x,x,L]                    [L,L,L] => [L,C]
[X,X,X,X,X,X,x,x,x] => [X,x,x,x,L]
[X,X,X,X,X,X,X,x,x] => [X,X,x,x,L]
[X,X,X,X,X,X,X,X,x] => [X,X,X,x,L]
[X,X,X,X,X,X,X,X,X] => [X,X,X,X,L]
```

Figure 2.6.   Interaction propensities for X,x and L,l.

After the **X/x** interaction there will be three **L/l** conditions fulfilling the **L/l** Interaction propensities strictly after the **X/x** interaction. The **L/l** interaction produces one **L/l** condition and one **C/c** carry condition.

There will be four of **C/c** in each Roman numeral so adding two Roman numerals will involve eight conditions and the carry condition will make exactly nine **C/c** conditions to form the **C/c** interaction propensity fulfillments as shown in Figure 2.7. Again, the **C/c** interaction will occur strictly after the **L,l** interaction.

```
            Interaction propensies for C,c
[c,c,c,c,c,c,c,c,c] => [c,c,c,c,d]
[C,c,c,c,c,c,c,c,c] => [C,c,c,c,d]
[C,C,c,c,c,c,c,c,c] => [C,C,c,c,d]              Interaction propensies for D,d
[C,C,C,c,c,c,c,c,c] => [C,C,C,c,d]                    [d,d,d] => [d,m]
[C,C,C,C,c,c,c,c,c] => [C,C,C,C,d]                    [D,d,d] => [D,m]
[C,C,C,C,C,c,c,c,c] => [c,c,c,c,D]                    [D,D,d] => [d,M]
[C,C,C,C,C,C,c,c,c] => [C,c,c,c,D]                    [D,D,D] => [D,M]
[C,C,C,C,C,C,C,c,c] => [C,C,c,c,D]
[C,C,C,C,C,C,C,C,c] => [C,C,C,c,D]
[C,C,C,C,C,C,C,C,C] => [C,C,C,C,D]
```

Figure 2.7.   Interaction propensities for C,c and D,d.

After the **C/c** interaction there will be three **D/d** conditions fulfilling the **D/d** Interaction propensities strictly after the **C/c** interaction. The **D/d** interaction produces one **D/d** condition and one **M/m** carry condition.

The **M/m** interaction propensities, shown in Figure 2.8 pose a difficulty because **M** does not have an intrinsic maximal form. One can put as many **M**s as one likes in a Roman numeral and the only way to pre-determine how many **M**s there are is to limit the number of **M/m**s allowed in a Roman numeral. In this discussion the number of **M/m**s will be limited to five so that, with the carry, there are always exactly eleven of **M/m**. The **M/m** interaction occurs strictly after the **D/d** interaction and is the necessarily last interaction of the Roman numeral addition. The interaction produces five of **M/m** and the condition **Z** which indicates the completion of the addition.

Interaction propensities for **M,m**

```
[m,m,m,m,m,m,m,m,m,m,m] => [m,m,m,m,m,Z]
[M,m,m,m,m,m,m,m,m,m,m] => [M,m,m,m,m,Z]
[M,M,m,m,m,m,m,m,m,m,m] => [M,M,m,m,m,Z]
[M,M,M,m,m,m,m,m,m,m,m] => [M,M,M,m,m,Z]
[M,M,M,M,m,m,m,m,m,m,m] => [M,M,M,M,m,Z]
[M,M,M,M,M,m,m,m,m,m,m] => [M,M,M,M,M,Z]
[M,M,M,M,M,M,m,m,m,m,m] => [M,M,M,M,M,Z]
[M,M,M,M,M,M,M,m,m,m,m] => [M,M,M,M,M,Z]
[M,M,M,M,M,M,M,M,m,m,m] => [M,M,M,M,M,Z]
[M,M,M,M,M,M,M,M,M,m,m] => [M,M,M,M,M,Z]
[M,M,M,M,M,M,M,M,M,M,m] => [M,M,M,M,M,Z]
[M,M,M,M,M,M,M,M,M,M,M] => [M,M,M,M,M,Z]
```

Figure 2.8.   Interaction propensities for M,m.

2.4.3.   The new Roman numeral format

A modified Roman numeral is always exactly twenty conditions: four of **I** or **i,** one of **V** or **v**, four of **X** or **x**, one of **L** or **l**, four of **C** or **c**, one of **D** or **d**, and five of **M** or **m**. The number one is **mmmmmdcccclxxxxviiiI**. The number zero is **mmmmmdcccclxxxxviiii**. This is analogous to a 32 bit 2s complement binary number which is always 32 bits regardless of its magnitude.

The Roman numerals.
<div align="center">

MDCCCLXXVIII and **CXXII**

</div>

now become
<div align="center">

mmmmMDCCCcLxxXXViIII and **mmmmmdCccclxxXXviiII**

</div>

and the addition becomes:

 mmmmMDCCCcLxxXXViIII + **mmmmmdCccclxxXXviiII** = **mmmMMdcccclxxxxviiii**.

The addition process accepts two completely represented numerals and produces one completely represented numeral preserving the numeral representation convention.

The above numerals are formatted for readability but there is still no intrnsic association differentiation.
<div align="center">

mmmmMDCCCcLxxXXViIII = **mmimVxIMDxICXmXCICcL** = … = 1878

</div>

The only association relation is that all the conditions are collectively associated at a single place of common association.

2.4.4.   Order from chaos

As shown in Figure 2.9 the indiscriminately associating conditions of two Roman numerals, autonomously add themselves in a dependent progression of proximity of association completeness fulfillments and interactions to a necessarily last fulfillment and interaction which completes the sum and produces the condition **Z** singularly indicating the addition is completed. The **Z** condition might open the bag and spill the result or it might perform a coordination duty within the bag.
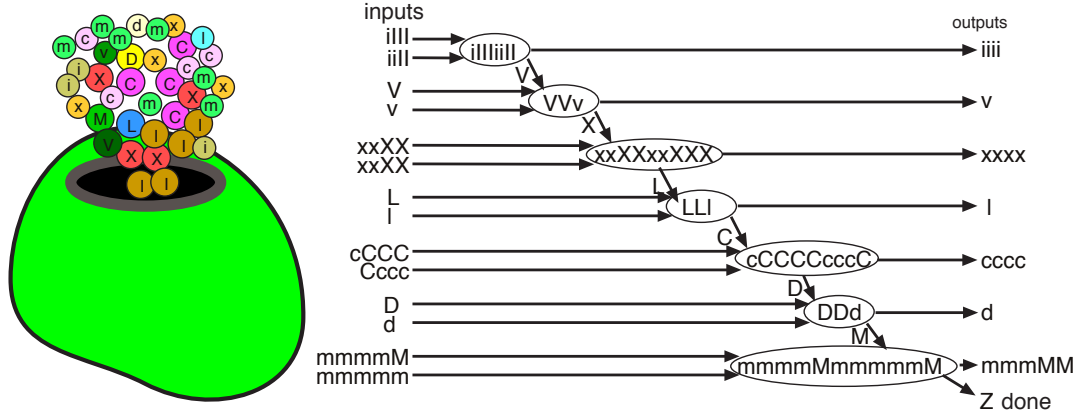
Figure 2.9.   Modified Roman numeral addition in shaking bag.

2.5.   Interlude: Pure condition differentiation

A pure condition interaction is persistences asserting differentness of conditions with different propensities of interaction indiscriminately associating within a single place of common association (the shaking bag). Inside the bag there is no agency of central influence and there is no agency of explicit control. The persistences and their asserted conditions behave individually and independently. The only influence on their behavior is the confinement of the bag and its shaking. There is no ambiguous behavior that needs to be isolated or hidden. There is no need of any assistance or influence from outside the bag.

2.5.1.   Search

The bag is not shaking to perform searches and realize interactions yet its shaking enables searches and realizes interactions. A persistence and its asserted differentness condition with its specific interaction propensity has no intrinsic agency to perform a search to fulfill its interaction propensity yet inside the shaking bag the condition will experience a journey of encountering many other conditions with which it does not interact and then find one condition with which it does interact.

2.5.2.   Dependency and completeness

Dependency of interaction relations are expressed by the differentness of the conditions and their specific propensities to interact with other different conditions. An interaction occurs only with complete proximate association of interacting differentnesses which coordinates the interaction behaviors. The appearance of the result differentness and the disappearance of the interacting differentnesses indicates, that the different interacting conditions were completely proximate, that the interaction has completed and that the result differentness is the correct resolution of the interacting differentnesses. This is the *completeness of input criterion*.

2.5.3.   Concurrency

There can be a multitude of pure condition interactions with mutually disjoint condition sets and interaction propensities concurrently realizing independent progressions of dependent condition interactions in a single frothing sea of conditions.

2.5.4.   Real pure condition expression

Pure condition is the expressional form of a biological cell which is a single place of common association filled with thousands of different protein conditions with specific interaction propensities supporting the intermingled expression of thousands of different

15   Copyright © 2024 by Karl Fant

interactions proceeding concurrently and continually without ambiguity. The cell as a whole in complete control of itself.

2.5.5. Place and time in the shaking bag

There is no referential where or when inside the bag. There is no consistent or coherent metric of temporal or spatial relations among the persistences, their conditions of differentnesses or among their interactions within the chaos of the shaking bag yet fully determined interaction occurs. Trying to relate the bag to an external metric of space or of time contributes nothing to either understanding or to realizing pure condition interaction.

2.5.6. A first principle fulfilled

In a pure condition expression, differentness spontaneously and dependently interacting and changing (first sentence of chapter) is a complete and sufficient accounting in itself of its interactions with no ambiguous behavior in need of being isolated or hidden and in no need of any extrinsic support from outside the bag.

The journey continues in Chapter 3 exploring the static association of persistences and their asserted conditions.

---

1. See "https://en.wikipedia.org/wiki/Roman_numerals".

Chapter 3:
Association Differentiation

In Chapter 2 persistences associated indiscriminately and conditions interacted specifically. All expression of differentness was in terms of differentness of condition with no differentiation in terms of association of persistences. In this chapter persistences will associate specifically and conditions will interact indiscriminately (all conditions interact with each other). All expression of differentiation will be in terms of differentness of association with no differentiation in terms of differentness of condition. Interaction will be determined by direct static association of persistences rather than by interaction propensity of conditions. Each persistence will be different from all other persistences not by the differentness of the conditions it asserts but by the differentness of its place in a network of static association relations.

3.1. The behavior of statically associated persistences

An individual persistence has no associational structure, is not directional, has no input or output, no top or bottom, no right or left. Associated persistences asserting conditions with compatible interaction propensities will be continually asserting a condition to each other, each continually fulfilling an interaction propensity and continually changing its asserted condition in response. Associated persistences whose asserted conditions do not continually and indiscriminately interact have no significance.

3.1.1. Undirected continual interaction

If the conditions of associated persistences and their interaction propensities are identical as shown at the left of Figure 3.1 then the conditions of the associated persistences will interact continually and indiscriminately in all directions even with themselves.

[N,N] -> D   [N,N] -> X   X -> D
[N,D] -> D   [N,D] -> Y   Y -> N
[D,N] -> D   [D,N] -> Y
[D,D] -> N   [D,D] -> Y

Figure 3.1.   Undirected interaction of associated persistences.

A persistence can avoid interacting with its own conditions by being responsive to one set of conditions and by asserting a different set of conditions. But if it associates with a second persistence that is responsive to its asserted conditions and which asserts condition to which it is responsive, as at the right of Figure 3.1, they are dependent on each other, feed back to each other and continually interact with each other.

3.1.2. Directionalizing interaction behavior

To avoid the continual mutual interaction requires a linear association of three persistences with complementary condition interaction propensities each responsive to and asserting a different set of conditions as illustrated in Figure 3.2.

$$[N,N] \rightarrow X \quad X \rightarrow S \quad S \rightarrow N$$
$$[N,D] \rightarrow Y \quad Y \rightarrow T \quad T \rightarrow D$$
$$[D,N] \rightarrow Y$$
$$[D,D] \rightarrow Y$$
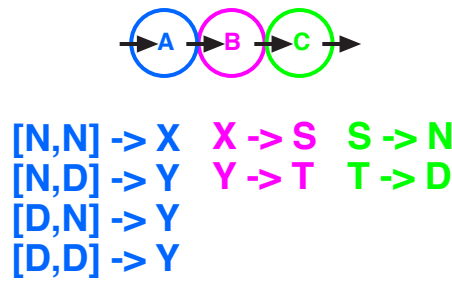
Figure 3.2.   Directionalizing the behavior of statically associated persistences.

- Persistence **A** is responsive only to conditions **N** and **D** and asserts only conditions **X** and **Y**.
- Persistence **B** is responsive only to conditions **X** and **Y** and asserts only conditions **S** and **T**.
- Persistence **C** is responsive only to conditions **S** and **T** and asserts only conditions **N** and **D**.
- Persistence **C** is dependent on persistence **B**
- Persistence **B** is dependent on persistence **A** and not dependent on persistence **C**
- Persistence **A** is not dependent on persistence **B** and not dependent on persistence **C**.
- The relations of dependency become directionalized.

Conditions **N** and **D** asserted by persistence **C** do not associate to persistence **A** and do not influence the responsive behavior of persistence **A**. The **X** and **Y** conditions and the **S** and **T** conditions of persistence **B** are buffering conditions that separate the persistence **C** assertion of **N** and **D** from the responsivity of persistence **A** to **N** and **D**. It does not matter what persistence **B**'s buffering conditions are as long as they connect **A** and **C** and are different from **N** and **D**.

The linearly associated persistences form a **directionalized association interaction behavior** sensitive to and asserting the same set of conditions **N** and **D**. Identical input and output conditions are different because they are at different isolated places of association in relation to the interaction behavior. The directionalized association interaction behavior establishes a new form of representing interaction differentness in terms of **differentness of place of association**.

3.1.2.1.   Practical directionalized interaction behaviors

A switch, Figure 3.3a, is a directionalized association interaction behavior. In Figure 3.3 the colors of the switches correspond to persistences **A**, **B** and **C**. An electromagnetic switch receives an input current condition that influences a magnetic field condition which influences a physical position condition which influences an output current condition which is isolated from the input current by interaction through two different condition domains.

a. Electromagnetic switch   b. Vacuum tube   c. Transistor
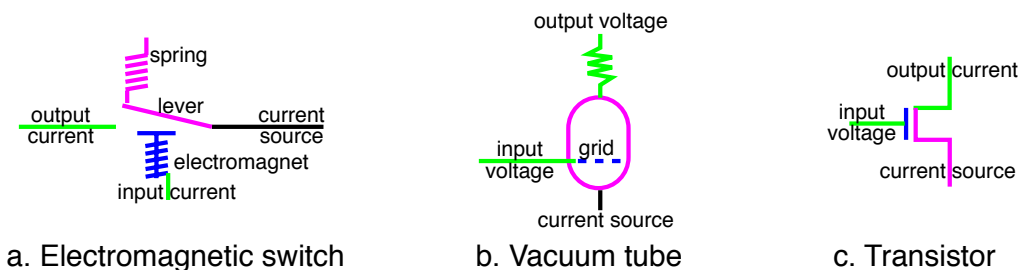
Figure 3.3.   Switches isolate and directionalize condition transition flow.

An electronic tube, Figure 3.3b receives a voltage condition which influences a charge condition on the grid which influences an electron flow condition through the vacuum which influences the output voltage condition on the wire which does not feed back to and influence the input voltage condition.

A transistor, Figure 3.3c, receives a voltage condition on the gate influencing the charge condition in the channel which influences the electron flow condition through the channel which influences the voltage condition at the output which does not feed back and influence the input voltage condition.

A neuron receives input on its dendrites and asserts output on its axon which output does not feed back through the neuron to influence its dendrites

3.1.3.   Differentiating interaction behavior

Directed interaction behaviors can be characterized in terms of the condition interaction propensities embodied by persistence **A** and the assertions of persistence **C**. As shown in Figure 3.4 directed interaction behaviors named **"one of"** and **"all of"** are defined in terms of the response of persistence **A** to condition combinations and the condition it asserts which plays through to persistence **C** which asserts the result condition for the interaction behavior.

Each type of interaction behavior can be represented with a unique graphic indicating each unique set of interaction propensities as at the bottom of Figure 3.4.
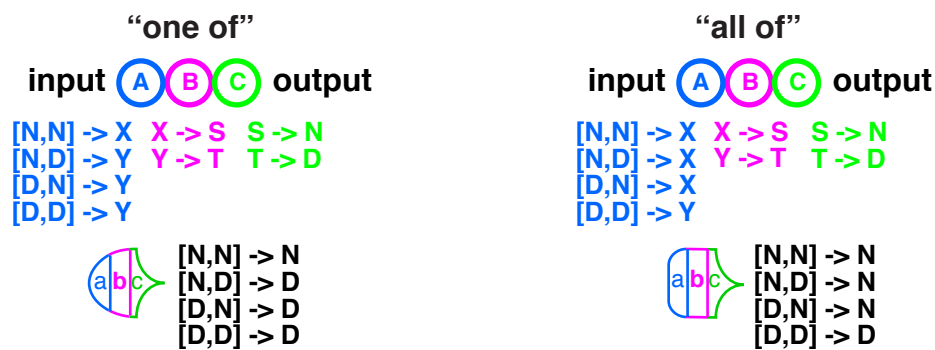
Figure 3.4.   Basic interaction behaviors

3.1.4.   Directed networks of directed interaction behaviors

Since every interaction behavior recognizes and asserts the same set of conditions and isolates its output from its input, associated interaction behaviors are assured to interact enabling the indiscriminate association of interaction behaviors output to input into directed networks of dependent interaction behavior. Dependency networks of directed interaction behaviors can be represented by stretching persistence **C** to associate with persistence **A** of other behaviors as on the left of Figure 3.5. Persistence **C** can be stylistically extended to represent association as on the right.

Primitive interaction behaviors
associated output to input

Primitive interaction behaviors
associated with extended output
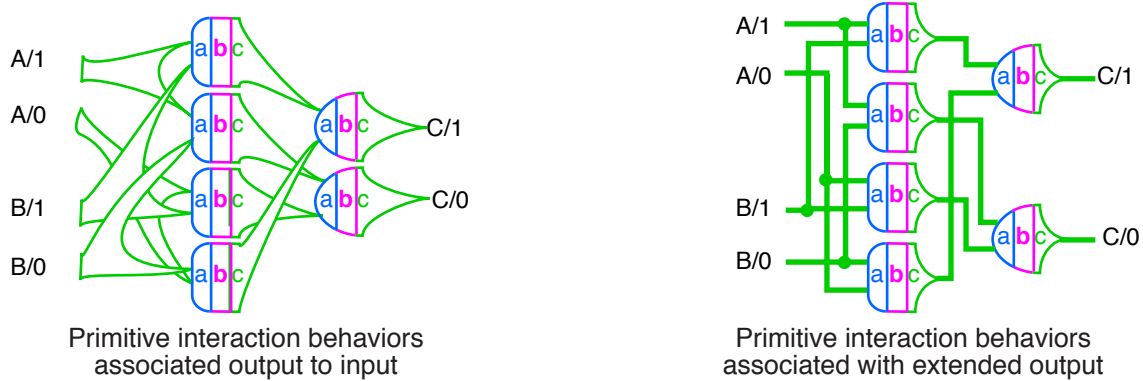
Figure 3.5.   Representations of directed networks of interaction behaviors.

3.1.5.   Association differentiation

- Each interaction behavior isolates its input from its output.
- The output of each interaction behavior in a network associates only to the inputs of other interaction behaviors.
- The **D** or **N** condition asserted by each interaction behavior is different from the identical **D** or **N** condition asserted by every other interaction behavior in the network by virtue of each behavior isolating its input from its output as shown in Figure 3.6.

isolated places of
association differentness

isolated places of
association differentness
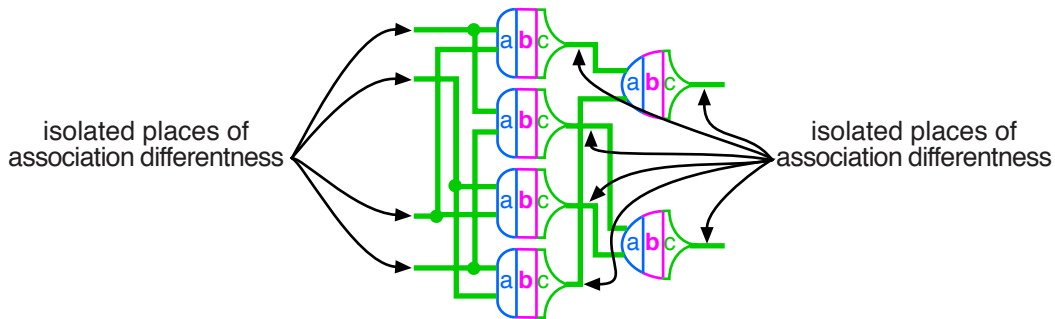
Figure 3.6.   Interaction behaviors isolating different places of association.

3.1.6.   Differentiating instances of interaction behavior

The coordination of interaction is accommodated here the same way it was accommodated in chapter 2, with completeness of interactive association. In chapter 2 coordination was expressed in relation to emptiness with the appearance of proximate completeness of interacting differentness conditions initiating an interaction and then the disappearance of the interacting differentness conditions and the appearance of the interaction result condition indicating the complete fulfillment of the interaction. This appearance and disappearance of completenesses bounds and differentiates instances of primitive interaction behavior (section 2.3.2).

For an association interaction behavior, continually presented with input conditions and continually asserting its output conditions there is no bounding of interaction conditions. No absence or emptiness separating one condition interaction from a next condition interaction as there is in the shaking bag. Conditions are continually presented to the input of a behavior and its output condition continually responds to its presented input conditions. This poses two difficulties.

- One: a next presentation of interaction conditions may require two or more inputs to transition. They will not transition simultaneously and the output can respond with a temporarily spurious condition to an incompletely transitioned input.
- Two: some input presentations may map to the same output condition and the transition of presented input from one interaction input to another may not cause the output to transition. If the output condition does not transition there is no intrinsic interaction behavior that differentiates instances of interaction.

These difficulties are identical to the Boolean logic difficulties of section 1.5 and, are resolved as follows.

3.1.6.1.   Define an explicit representation of emptiness

These difficulties are addressed by assigning one condition **N** to explicitly represent "**N**ot an interaction differentness" while assigning all the other available conditions, in this case only **D**, to represent "an interaction **D**ifferentness". Condition **N** represents the absence of, emptiness of, interaction differentness. Condition **D** represents the presence of interaction differentnesses. A transition from **N** to **D** represents an appearance of interaction differentness from an absence/emptiness of interaction differentness. A transition from **D** to **N** represents a disappearance of interaction differentness into a non differentiating emptiness of interaction differentness.

3.1.6.2.   Define two disjoint domains of completeness

Define a representation of "**D** completeness" which is a pattern of **D** assertion in a background of **N** assertion.

Define a representation of "completely **N**" which is all **N** assertions with no **D** assertions, i.e. completely empty of interaction differentness.

Interaction occurs with the appearance of **D** completeness.

3.1.6.3.   Require monotonic transition between completeness representations

Each interaction behavior will begin with a transition from completely **N** to **D** completeness, the appearance of completeness of interaction differentnesses. Each interaction behavior will end with, a transition from **D** completeness to completely **N**, the disappearance of the interaction differentnesses.

The monotonic transitioning between **D** completeness and completely **N** shown in Figure 3.7 unambiguously differentiates successive instances of **D** completeness.
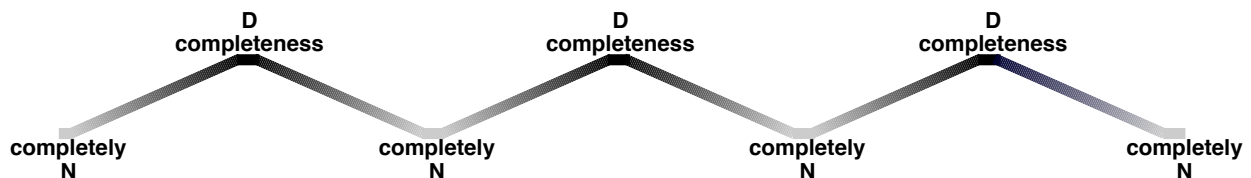
Figure 3.7.   Monotonic transition of condition completeness relations.

3.2.   Primitive interaction behaviors: sufficient expressivity

Define primitive interaction behaviors that transition their output only with presentation of input completeness fulfilling the completeness of input criterion (section 2.5.1). A primitive interaction behavior begins empty with inputs completely **N** and its output asserting **N**. When the inputs transition to **D** completeness the output transitions to **D** which is maintained until the input transitions to completely **N** at which point the output transitions to **N** which is maintained until

the input transitions to **D** completeness at which point the output transitions to **D** which is maintained until the input transitions to completely **N** and so on.

Primitive behaviors fulfilling the completeness of input criterion can be realized with primitive behavior interaction propensities arranged according to the interaction tables shown in Figure 3.8 which also shows the graphic representation and the textual representation of each interaction behavior. The "−" means that no output transition occurs. The transition of the output between **D** and **N** reflects the transition of the input between **D** completeness and completely **N**. Enclosing braces **{ }** indicate **"one of"** related behavior for which **D** completeness is one input at **D** and the rest at **N**. Enclosing brackets **[ ]** indicate **"all of"** related behavior for which **D** completeness is all inputs at **D** and none at **N**. Completely **N** is the same for all behaviors with all inputs at **N**, empty of interaction differentness. **<=** or **=>** indicates **"is dependent on"**
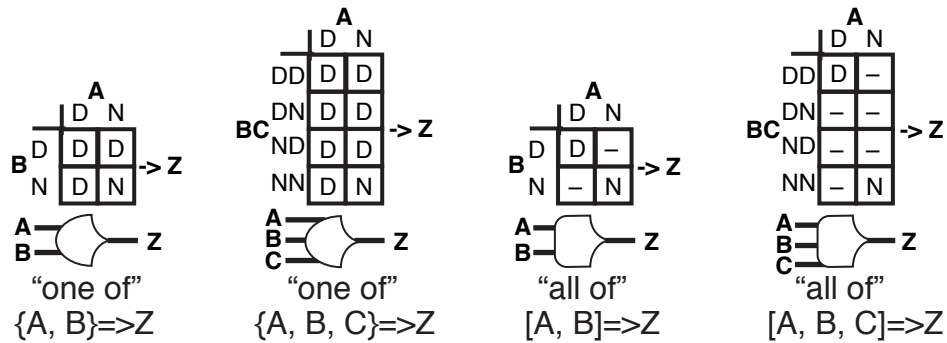
Figure 3.8.   Primitive interaction behavior interaction propensities.

The **"all of"** relation, *mutual inclusivity*

Z<=[A B C]

If **"all of"** A and B and C transition from **N** to **D** then **Z** will transition to **D**. When **"all of"** A and B and C have transitioned to **N** then **Z** will transition to **N**.

The **"one of"** relation, *mutual exclusivity*

Z<={A B C}

If **"one of"** A or B or C transitions from **N** to **D** then **Z** will transition to **D**. If a second input transitions to **D** then **Z**, already at **D**, does not transition. When **"all of"** A and B and C have transitioned to **N** then **Z** will transition to **N**.

Primitive interaction behaviors are differentiated only in terms of the appreciation of their input to **D** completeness. The transition of input to completely **N** is universal for all behaviors. The behavior of the primitive interaction behaviors is illustrated in Figure 3.9.
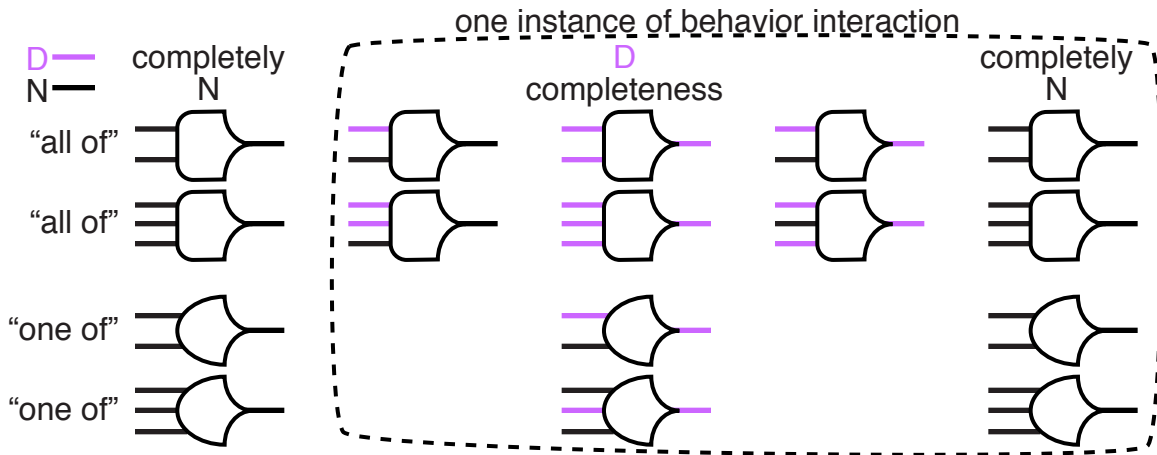
Figure 3.9. primitive behavior with monotonic completeness transition behavior.

The **"all of"** behavior is similar to the Muller C element which is generally viewed as a binary control operator in contrast to an interaction mapping behavior.[1] The **"one of"** behavior is identical to Boolean OR. The derivation of both elements here is quite different from their historical development.

3.2.1.1.   The difficulties of section 3.1.6 are resolved.

- Fulfilling the completeness of input criterion a primitive behavior does not assert spurious output transitions due to incompletely presented input.
- Because of the monotonic transitioning between **D** completeness and completely **N** and fulfillment of the completeness of input criterion (section 3.3.1) every primitive behavior interaction is unambiguously differentiated by the transition of its output to **D** followed by its transition to **N**.

3.2.1.   Non interaction behaviors

There are two primitive behaviors that do not participate in the interactions of differentnesses but that participate in the coordinaion of the dependent flow of interaction behavior.

The first behavior on the left of Figure 3.10 is referred to as **conversion** because it is used in network closure to convert between the two disjoint domains of completeness representation **D** and **N** (section 3.7). Conversion does not invert between two interaction differentnesses as a Boolean logic inversion does and never appears in an interaction dependency relation. The inverter symbol is still used because it is convenient and should be easily understood in context.

The single tilde ~ represents a conversion that is forced to **N** during initialization to allow an initial completely **N** wavefront to propagate through a network (Appendix D). The double tilde ⁓ represents a conversion that is not forced to **N** during initialization which is used only when initializing to **D** completeness (Appendix E).

The second behavior on the right of Figure 3.10 is the **mutex/arbiter**, represented textually as {{ }}, that enforces two otherwise uncoordinated wavefront flows to flow mutually exclusively. In Figure 3.10 **A** flows to **A** and **B** flows to **B** but only **"one at a time"** turning two uncoordinated flows into a coherent **"one of"** related flow (Appendix C and section 4.9.3). The arbiter can be viewed as an enforcing "one of" behavior. The mutex/arbiter is a well understood component of asynchronous design.[2]
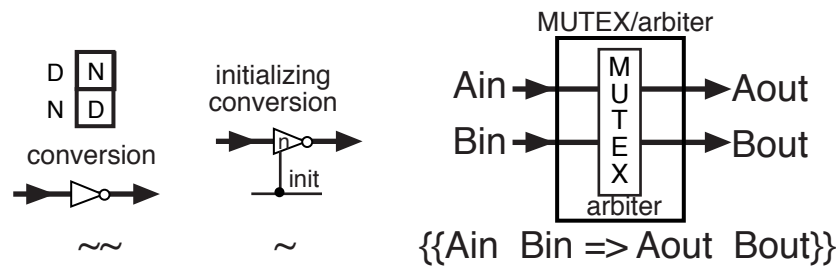
Figure 3.10. Non interacting flow coordination behaviors.

3.3.   INTERLUDE: Sufficiently expressive primitivity

Each primitive interaction behavior expresses one primitive step of mapping (interaction), condition holding behavior (memory) and input completeness appreciation (coordination). These primitive interaction behaviors along with monotonic transitioning between **D** completeness and completely **N** enable all that follows in terms of association differentiation. A static structure of differentness collaborates with a dynamic structure of differentness to stably express computation. It will be shown by the end of this chapter that these primitive behaviors are sufficiently expressive on their own intrinsic behavioral merits and in no need of any extrinsic assistance.

3.3.1.   Completeness of input criterion

A primitive behavior transitions its output only when its input is presented with **D** completeness or completely **N**. The transition of the output implies the presentation of completeness of input and the completion of the interaction behavior.

3.3.2.   Constant behavior

The primitive interaction behaviors are constant in that they always assert the same output differentness for the same presented completeness of input differentness allowing the primitive interaction behaviors to be indiscriminately referenced from and copied to anywhere and anywhen. In particular, the primitive interaction behaviors can be referenced and copied to compose networks of dependently related primitive behaviors.

3.3.3.   The environment

A primitive behavior is entirely dependent on the presentation of input from an environment external to itself. If there is no transition of input presented there is no behavior.

> The responsible environment
>
> If the input conditions presented by the environment to a primitive interaction behavior **monotonically transition between D completeness and completely N with an appropriate delay between transitions** then the output of the primitive interaction behavior will monotonically and correctly follow the transitioning of the input.
>
> The environment establishes the presentation of input differentnesses for each primitive interaction behavior and establishes the behavior interaction time. The mathematician with pencil and paper is an exemplar of a responsible environmental.

By the end of this chapter interactions will behave independently of the environment which will no longer have any behavioral responsibility.

3.3.4.   Primitive behavior space and time

A primitive interaction behavior performs one atomic *instance of interaction* of differentness behavior between atomic places of association in one *instance of interaction time* represented by the transition of a behavior's output from **N** to **D** and back to **N**. The succession of presentations of **D** completeness and completely **N** use and unuse the primitive behavior in different instances of interaction time extending the representation of differentness of the primitive behavior through time.

3.4.   The constant network: composing primitive behaviors

Primitive interaction behaviors linearly associated output to input form a network of dependency relations among the primitive interaction behaviors that always asserts the same output differentness for the same presented input differentness. A constant network is indiscriminately referencable from and copyable to anywhere and anywhen. In particular, constant networks can be referenced and copied to compose larger constant networks.

3.4.1.   Interaction dependency relations

Interaction dependency relations are expressed in terms of **mutually inclusivity** and **mutually exclusivity**. The completeness relations of mutual inclusivity [ ] (all at a time, **"all of"**) and mutual exclusivity { } (one at a time, **"one of"**) permeate interaction expression.

An interaction is a mutual inclusion (input completeness) of mutual exclusivities (individual input differentnesses) dependently interacting through a mapping relation to produce a mutual exclusivity (output) which proceeds on to participate in the mutual inclusivity of a next interaction.

In the pure condition expression of Chapter 2 each persistence mutually exclusively asserted one at a time of a range of possible different conditions. Interaction occurs when persistences asserting the interacting conditions are mutually inclusively proximately associated. The interaction produces one result differentness condition from a range of mutually exclusive possible result differentness conditions which proceeds to mutually inclusively associate with other conditions to interact and produce a next mutually exclusive result differentness and so on.

The primitive behaviors are minimal relations of mutual inclusivity and mutual exclusivity from which larger relations of mutual inclusivity and mutual exclusivity are constructed. Figure 3.11 shows a network of dependently associated **"one of"** and **"all of"** primitive behaviors.

3.4.2.   Differentness of place of association

In a network of primitive interaction behaviors associated output to input, the output of each interaction behavior is a unique differentness of place of association within the network as illustrated in Figure 3.11 (section 3.1.5).
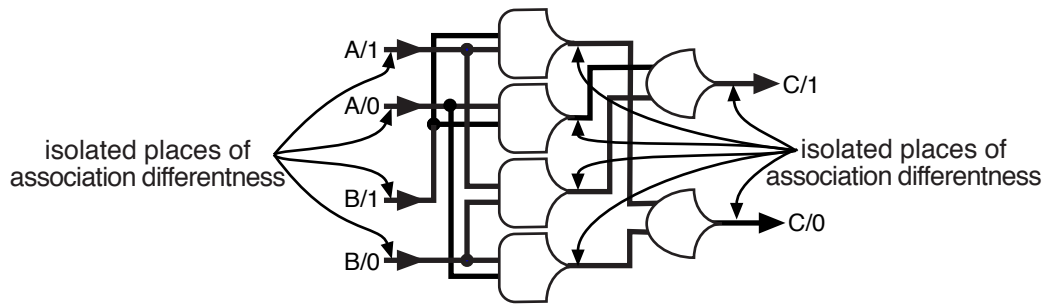
Figure 3.11. Differentness of place of association in a network of primitive interaction behaviors associated output to input.

If a primitive interaction behavior asserts condition **D** it expresses the differentness represented by its unique place of association. If a primitive interaction behavior asserts condition **N** it does not expresses the differentness represented by its unique place of association which is empty of differentness.

Condition **D** is the only condition expressing interaction differentness. There is no longer any differentiation of interaction differentness in terms of condition. This is *pure association differentiation* with all interaction differentness represented by differentness of place of association and specificity of interaction represented by direct association. This is in contrast to *pure condition differentiation* (Chapter 2) in which all interaction differentness is represented by differentness of condition and specificity of propensity of condition to interact.

3.4.3.   Localities of interaction differentness

Each place of association differentness is a primitive interaction differentness. Places of association differentness are associated to form localities of interaction differentness which are the places in the network between the associated primitive behaviors which represent the differentnesses that interact through the primitive behaviors. One differentness from a range of mutually exclusive interaction differentnesses is asserted when the primitive behavior outputs forming a locality transition from completely **N** to **D** completeness and then unasserted when the primitive behavior outputs transition to completely **N**.

A locality represents two differentnesses that determine its significance:

1.   A locality asserts a single differentness from a range of possible mutually exclusive differentnesses.
2.   A locality represents a unique place of differentness within the association network of dependency relations. A same differentness from a range of mutually exclusive differentnesses, the number 4 for instance, can be expressed by multiple localities and each expression is different by virtue of the differentness of place of association of the locality.

A locality's differentness of place of association and its asserted differentness together represent a unique significance within an interaction.

3.4.3.1.   Locality differentness

A, **B** and **C** in Figure 3.11 are localities of mutually exclusive differentness each of which can express **"one of"** its mutually exclusively differentnesses **"at a time"**. Each locality consists of two places of association differentness. Each place can assert **"one of"** conditions **D** or **N**.

Each locality expresses a mutually exclusive differentness with **"one of"** its places of association transitioning to **D** and the other place remaining at **N** (**D** completeness). A locality is empty of differentness when all of its places are asserting **N (**completely **N).**

Multi-rail and particularly dual-rail encoding has long been understood as a delay insensitive encoding for differentness (information) transfer with "**N**" typically referred to as a spacer state.[3] It has been less well understood as a fundamental aspect of interaction differentness.

A network of associated primitive behaviors is also a network of interacting localities.

3.4.3.2.   Expressing locality differentness

Differentnesses of a locality are expressed in terms of mutually exclusive, "**one of**", and mutually inclusive, **"all of"**, withiness relations. For instance the locality named **C** of Figure 3.11 can assert **"one of"** two places of association named **1** and **0** each of which can assert **"one of" D** or **N** is represented as two **"one of"** relations expressed as:

>   **C/{1 0}/{D N}**

The slash / indicates withiness distribution. The braces **{ }** indicate **"one of"** related. The above expression distributively expands to:

>   **C/{1/{D N} 0/{D N}}**

The three localities **A**, **B** and **C** of Figure 3.11 are expressed as

>   **A/{1 0}/{D N}   B/{1 0}/{D N}   C/{1 0}/{D N}**

Since every place of association can only assert **D** or **N** the **{D N}** term is a universal most primitive terminal and can be implied with a terminal dangling slash.

>   **A/{1 0}/   B/{1 0}/   C/{1 0}/**

Which expands to:

>   **A/{1 0}/{D N}   B/{1 0}/{D N}   C/{1 0}/{D N}**

A, **B** and **C** which all have the same locality structure can also be represented as:

>   **(A B C)/{1 0}/**

Which distributively expands to

>   **A/{1 0}/   B/{1 0}/   C/{1 0}/**

and so on.

The differentnesses of a locality are referenced as

>   **A/1   A/0   B/1   B/0**

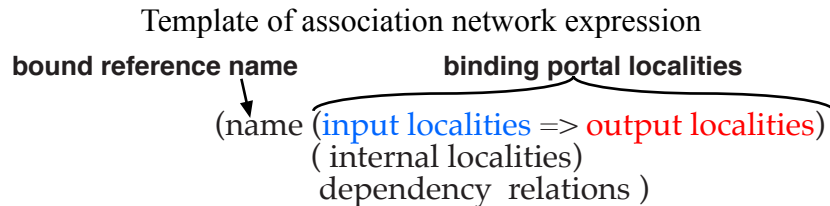and so on

>   **E/{3 2 1 0}/**

Expresses a locality named **E** with four mutually exclusive differentnesses named **3**, **2**, **1** and **0** represented as four places of association only one of which will assert **D** at a time.

Localities with larger ranges of interaction differentness are composed by associating localities with small ranges of interaction differentness (section 3.4.8.).

Localities mirror the persistences of pure condition differentiation of Chapter 2 as locus of interaction transition and bearer of mutually exclusive differentness.

3.4.4.   Expressing a constant network

A constant network expression, enclosed in parentheses, includes an expression *reference name*, a *binding portal* specifying the localities exposed to the *external environment* with their input to output dependency relation, the specification of referenced *internal localities* and the *dependency relations* among the localities and behaviors of the network.

Template of association network expression

bound reference name   **binding portal localities**

(name (input localities => output localities)
       (internal localities)
       dependency relations)

In Figure 3.11 locality **A** expressing a mutually exclusive differentness and locality **B** expressing a mutually exclusive differentness mutually inclusively associate to interact and produce the mutually exclusive differentness of locality **C**. The expression below completely expresses the network. To keep directionality conveniently visible input localities are shown in **blue** and output localities are shown in **red**. There are no internal localities in this expression. In Figure 3.12 the primitive behaviors **"one of"** and **"all of"** are referenced respectively with **{ }** and **[ ]** encompassing a list of input association places composed with nesting relations as with:

{[A/0 B/1] [A/1 B/0]}

bound reference name   **binding portal localities**

(net (A/{1 0}/ B/{1 0}/ => C/{1 0}/)
dependency relations { C/{ 1<={[A/0 B/1] [A/1 B/0]}
                              0<={[A/0 B/0] [A/1 B/1]} })
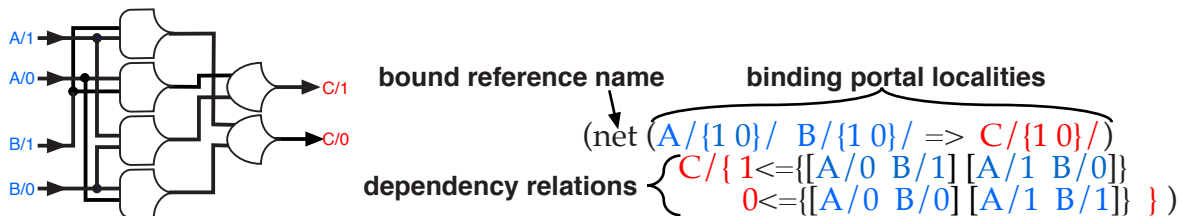
Figure 3.12. Network expression for Figure 3.11

The expression reference name is **net**. The binding portal expresses that output locality **C** is dependent on, **=>**, the input localities **A** and **B**. The specific dependency relations of the network are expressed by recapitulating the expression of output assertion locality **C** and nesting within the recapitulated expression the dependency relation, **<=**, for each component of the locality, **{1 0}**, on components of input localities **A** and **B** in terms of the primitive behaviors **"all of"** [ ] and **"one of"** { }. For instance, [A/0 B/1] represents an **"all of"** behavior dependent on inputs **A/0** and **B/1**. If both **A/0** and **B/1** assert **D** then the **"all of"** relation asserts **D**. **{[A/0 B/1] [A/1 B/0]}** represents a **"one of"** behavior dependent on **[A/0 B/1]** and **[A/1 B/0].** If **"one of"** the relations **[A/0 B/1]** or **[A/1 B/0]** asserts **D** then the **"one of"** relation **{[A/0 B/1] [A/1 B/0]}** will assert **D**.

If the term **{[A/0 B/1] [A/1 B/0]}** resolves to **D** then **C/1** transitions to **D**. If the term **{[A/0 B/0] [A/1 B/1]}** resolves to **D** then **C/0** transitions to **D**. Only one of the terms will resolve to **D**. The mutual exclusivity of the terms traces back to the mutually exclusivity of the differentnesses of locality **A**, the mutual exclusivity of the differentnesses of locality **B** and their cross association (section 3.4.7).

The expression of the localities and their dependency relations map directly to the network of Figure 3.11.

3.4.5.   The binding portal

There are three forms of dependency association relation. The binding portal serves as an association hub expressing all three association relations.

1. One to one relations
    The binding portal allows the network to be referenced externally and to associationally bind each portal locality with differently named external locality (section 3.4.5.1).
2. Many to one relations
    A binding portal typically maps multiple inputs to a single output (section 3.4.5.2).
3. One to many relations
    The binding portal associates each input locality by name correspondence to multiple places of association within the dependency expression (section 3.4.5.3).

3.4.5.1.   One to one reference association through the exposed binding portal

A network expression is exposed to reference by an *external environment* through its binding portal in terms of its expression reference name and the syntax structure of its binding portal. The only name within a network expression that is externally visible is the expression reference name. All other names within the expression can only be referenced from within the expression. The same names can be reused outside the expression without ambiguity.

External locality names are bound to internal locality names by corresponding syntax structure in relation to the expression reference name. There are three ways of referencing a network expression through its binding portal:

with full portal reference,

net (X Y => Z)

(net (A/{1 0}/ B/{1 0}/ => C/{1 0}/) .....)

with locality nesting,

Z<=net (X Y)

(net (A/{1 0}/ B/{1 0}/ => C/{1 0}/) .....)

or with portal nesting.

A(.....B(.......net (X Y).......))

(net (A/{1 0}/ B/{1 0}/ => C/{1 0}/) .....)
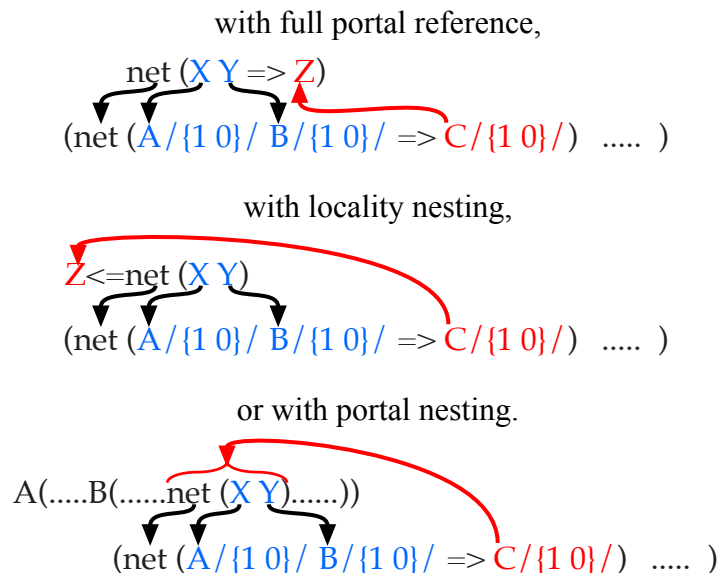
With full portal reference and locality nesting the output locality named **C** is referenced by the external locality named **Z** which can be further referenced distributing the differentness of locality **C**. Portal nesting is an unnamed one to one relation which does not support further reference to and distribution of the differentness of locality **C**.

All of these variations of reference are used in the examples.

3.4.5.2.   Many to one association in the binding portal

The many to one association of input to output in the binding portal.

$$(net\ (A/\{1\ 0\}/\ \ B/\{1\ 0\}/\ =>\ C/\{1\ 0\}/\ )$$

3.4.5.3.   One to many association into the dependency relations of a network

The binding portal distributes the differentness of each of its localities by name correspondence association into the expression of dependency relations.

bound reference name   **binding portal localities**

$$(net\ (A/\{1\ 0\}/\ \ B/\{1\ 0\}/\ =>\ C/\{1\ 0\}/\ )$$

dependency relations $\left\{ \begin{array}{l} C/\{\ 1<=\{[A/0\ \ B/1]\ [A/1\ \ B/0]\} \\ 0<=\{[A/0\ \ B/0]\ [A/1\ \ B/1]\}\ \}\ ) \end{array} \right.$

3.4.5.4.   The external environment

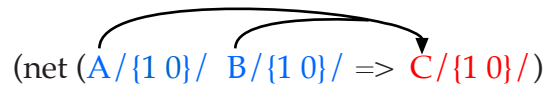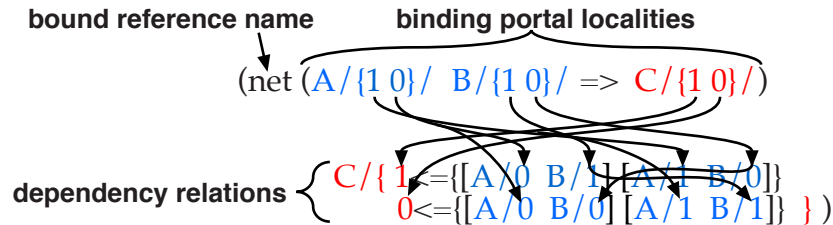A binding portal exposes a constant network to a presentation from an uncharacterized environment beyond its binding portal on which presentation the constant network is completely dependent for its liveness (no presentation no network behavior), its temporal reference (with the monotonic transitions between **D** completeness and completely **N**) and to provide variability of behavior (The constant network itself is constant. Its behavior varies only with the differentness of its presented input). The external environment is in complete control of the constant network through its binding portal.

3.4.5.5.   Network wholeness

The binding portal bounds the network with input and output and defines its wholeness of interaction behavior. This wholeness of network interaction behavior is not a wholeness of expression which extends through the binding portal into the environment not accounted by the constant network.

3.4.6.   Wavefronts of transition

A transition wavefront consists of behaviors transitioning their output in response to transitioning of presented input which transitioned outputs are presented to the input of subsequent behaviors which transition their output and so on. Flowing wavefronts of transition shuttle the weft of transition differentness through the warp of network differentness weaving the fabric of computation.

An interaction begins with a transition to completeness of presentation of one or more interacting differentnesses to the input of the exposed binding portal of a constant network which initiates a wavefront of transition flowing through the network to its binding portal output. This wavefront of transition flow from the completeness of presented input to the completeness of output must be unambiguous for all possible input presentations and all possible internal delay relations.

3.4.6.1.   The unambiguous wavefront of transition to D completeness

Figure 3.13 illustrates a transition to **D** completeness wavefront flowing through the constant network of Figure 3.11 followed by the transition to completely **N** wavefront. The constant network begins empty of interaction differentness with all localities completely at **N**, Figure

3.13a. The external environment transitions the input localities from completely **N** to **D** completeness. **A/0** transitions to **D** forming **D** completeness for locality **A**, Figure 3.13b, then **B/1** transitions to **D** forming **D** completeness for locality **B**, Figure 3.13c, forming network input **D** completeness. The inputs are held at **D** completeness as a wavefront of transitions from **N** to **D** begins flowing through the network. Because the inputs are held at **D** completeness, because of the completeness behavior of the primitive interaction behaviors and because there are only transitions from **N** to **D** with no transitions from **D** to **N** there are only correct transitions flowing through the network. There are no incorrect or spurious transitions: no glitching.

Due to the cross association, the output assertion of the network **C** transitioning to **D** completeness, Figure 3.13d, implies that all of the input has transition to **D** completeness, that the consequent transition to **D** wavefront has propagated through the network and that the transitioned output is the correct resolution of the presented input as well as the necessarily last transition to completeness of the wavefront flow through the network indicating completeness of interaction (section 3.4.7.2). The transition of the output to **D** completeness corresponds to the **Z** condition of the Roman Numeral example of Chapter 2

3.4.6.2.   The unambiguous wavefront of transition to completely N

The transition to **D** completeness wavefront is followed by the external environment presenting a transition to completely **N** to the input. **A/0** transitions to **N** forming completely **N** for locality **A**, Figure 3.13e, then **B/1** transitions to **N** forming completely **N** for locality **B**, Figure 3.13f. The inputs are held at completely **N** as a wavefront of transitions from **D** to **N** begins flowing through the network. Again, because the inputs are held at **N**, because of the completeness behavior of the primitive interaction behaviors and because there are only transitions from **D** to **N** with no transitions from **N** to **D**, there are only correct transitions flowing through the network. There are no incorrect or spurious transitions: no glitching.

Again, due to the cross association, when the output of the network **C** transitions to completely **N**, Figure 3.13g, it implies: that the input transition to **N** wavefront is complete, that the transition to **N** wavefront has propagated through the network and that the network is completely empty of interaction differentness (excepting orphans, see section 3.4.7.1 and 3.4.7.2).[4]
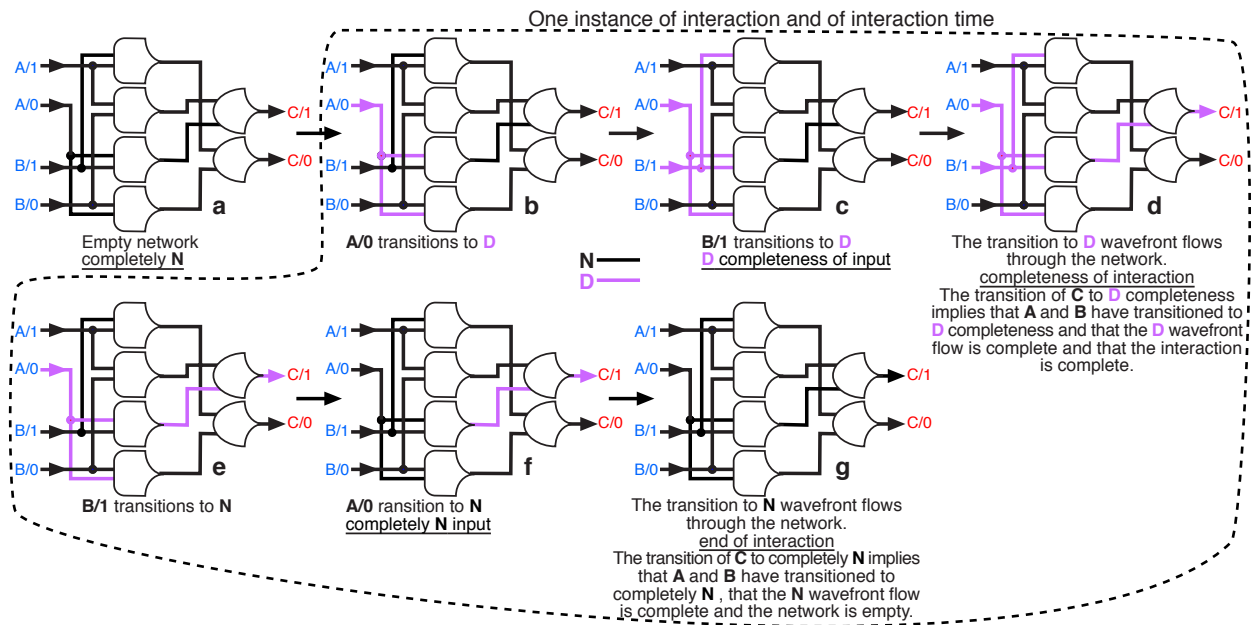
Figure 3.13. Flow of transition wavefronts through the network.

3.4.6.3.   Singular appreciability

With unambiguous wavefront flow through the network the transition of the network output locality to completeness is the only singularly appreciable event of a network interaction marking one instance of interaction and one instance of interaction time. It is the network counterpart of a temporal instant.

3.4.6.4.   Multiple inappreciability

All other wavefront transition events in the network, inputs that can present all at once or in any order and internal concurrent transition relations with differing delays that can occur all at once or in any order are not singularly appreciable but are temporally incoherent in the context of the constant network as a whole.

3.4.6.5.   Network ephemeral instances of interaction

A constant network can perform only one interaction *at a time*. All transition behavior in the network is subordinate to and dependent on presentation to the binding portal input and its consequent wavefront flow through the network. A transition to **D** completeness wavefront flowing through the network from its binding portal input presentation to its binding portal output transition to **D** completeness and flowing out of the network into the external environment is one **instance of interaction** for the network. Interaction result wavefronts don't necessarily end in the external environment. They are just unaccounted by the network after they flow out.

The following transition to completely **N** wavefront flowing through the network and transitioning the output to completely **N** empties the network of interaction differentness retaining no record, no history, of the instance of interaction.

With each transition to **D** completeness a network is used. With each transition to completely **N** the network is unused. With the next transition to **D** completeness the network is reused. Each instance of interaction occurs and then irretrievably disappears from the network.

3.4.7.   Recognizing presented input

To assert an output based on the differentness of its presented input a network must first appreciate the presented input differentness. The input to the network forms a locality of mutually exclusive differentness composed of the individual input localities. The mutually exclusive differentnesses for locality **A/{1 0}/** are named **1** and **0**. The mutually exclusive differentnesses for locality **B/{1 0}/** are named **1** and **0**. The two inputs cross associate to form a composite range of mutually exclusive differentnesses named, **00 01 10** and **11**, only one of which is presented at a time by the two input localities.

"one of" **A/1** or **A/0** will assert **D** and **"one of"** **B/1** or **B/0** will assert **D**. Only one of the cross associations will present **DD** which is appreciated by one of a rank of **"all of"** behaviors searching for the one fulfilling cross association. The one appreciating **"all of"** behavior, recognizing the presented input differentness, will transition its output to **D** determining whether **C/1** or **C/0** transitions to **D**.

The output of the rank of **"all of"** behaviors is itself a locality of mutually exclusive differentness which can be named and referenced explicitly expressing the cross association search as in Figure 3.14 instead of it being implied in the nesting relations of the network dependency expression as in the expression of section 3.4.4.
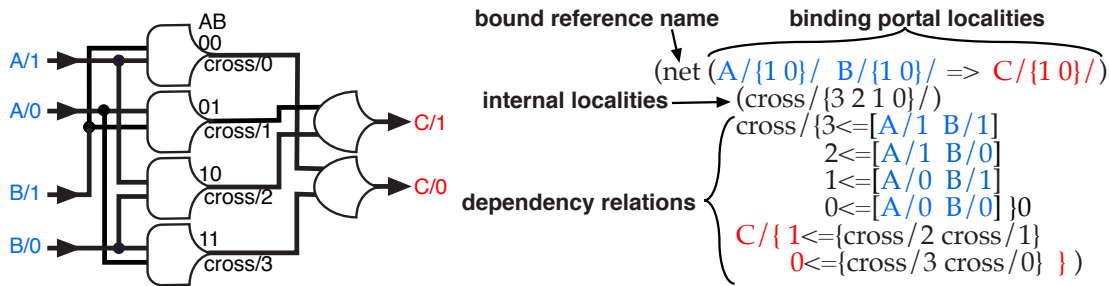
Figure 3.14. The network expression with explicitly named cross association locality

3.4.7.1.   Search failures - The orphans

As the transition to **D** wavefront propagates through a cross association search there are ineffective search branches that fail to form cross association completeness which are called orphans because they have lost their relations. In Figure 3.15a the effective search branches of the transition to **D** completeness wavefront are highlighted in purple and the ineffective orphan branches of the transition to **D** completeness wavefront are highlighted in green.
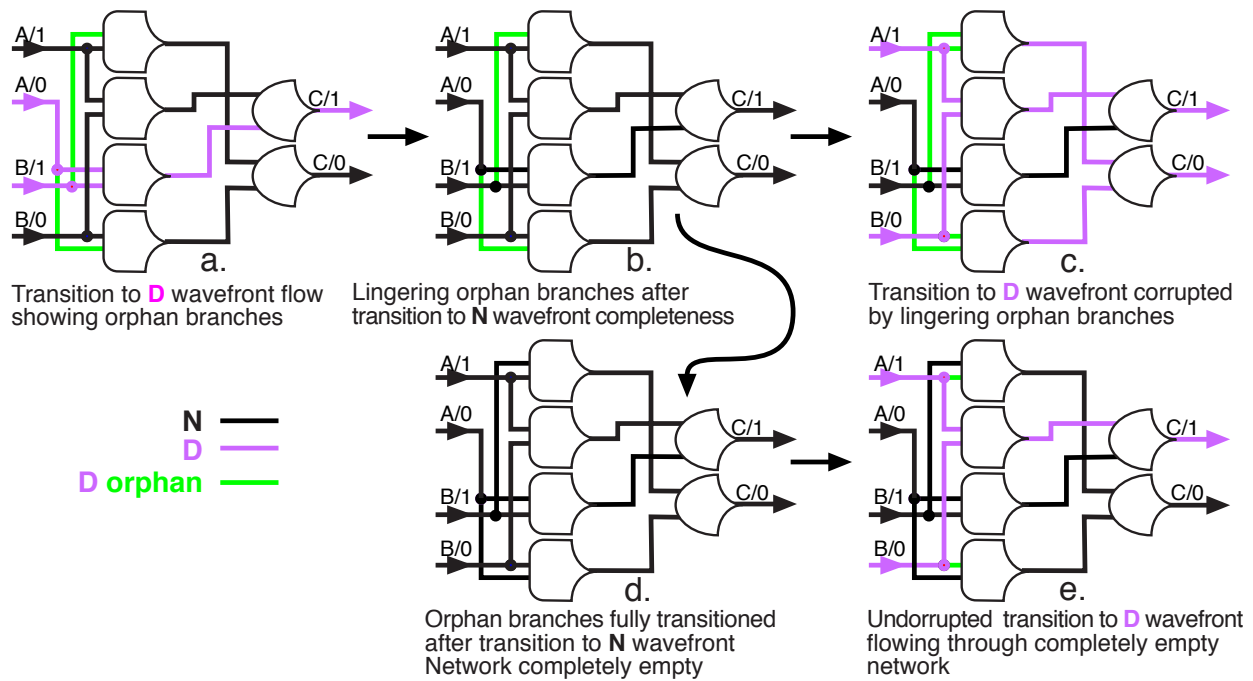
Figure 3.15. The orphans.

When the inputs presenting **D** completeness transition to completely **N** a wavefront of transition to completely **N** propagates through the constant network. When **C** transitions to completely **N** it implies that the input presentations have transitioned to completely **N** and that the effective search branches of the transition to **D** wavefront have transitioned to **N** but it does not imply that the ineffective orphan branches have transitioned to **N** which they may not have as in Figure 3.15b.

There must be an assumption that every orphan branch will transition to **N** before the next transition to **D** wavefront arrives at the branch. If the next transition to **D** wavefront arrives before the orphan branches transition to **N** the search is corrupted as in Figure 3.15.c. The differentnesses of locality **C** should never both be **D**. If the orphans transition to **N** before the next transition to **D** wavefront arrives as in Figure 3.15d the next transition to **D** wavefront is not corrupted and the result will be as in Figure 3.15 e.

It is shown in Appendix G, that if orphan paths are isolated to a branch and do not propagate through a primitive interaction behavior then orphan branches will always transition to **N** well before the next transition to **D** wavefront can arrive at the branch.

The orphan is similar to the Martin notion of the isochronic fork which has a more stringent relative delay requirement because it relates to undefined external relationships of which worst case behavior must be assumed.[5] The orphan delay sensitivity is less stringent because it relates well defined internal relationships.

Notice that in the pure condition expression of Chapter 2 search failures to form interactive completeness do not have lingering implications. A persistence and its asserted condition simply do nothing and continue searching (Section 2.3).

3.4.7.2.   The completeness of input criterion

The completeness of the output must imply the completeness of all of the presented input. The primitive behaviors fulfill the completeness of input criterion but a constant network does not fulfill the completeness of input criterion by virtue of its composition with completeness of

input fulfilling primitive behaviors. There can be networks that do not fulfill the completeness of input criterion.
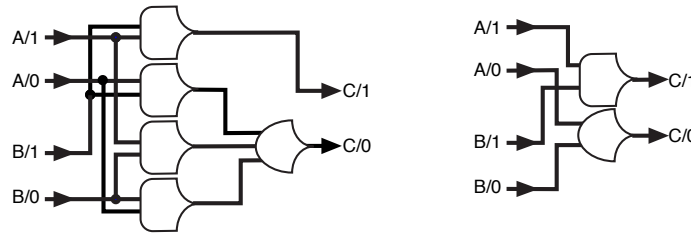
Figure 3.16. Fulfilling and not fulfilling the completeness of input criterion

The network on the left of Figure 3.16 fulfills the completeness of input criterion. The network on the right performs the same interaction mapping behavior asserting the correct result but it does not fulfill the completeness of input criterion. The output **C/0** can transition to **D** with just one input transitioned to **D** completeness.

The cross association not only recognizes the presented input but demands the completeness of the presented input. This demand for completeness of input ensures that the asserted output of the network will not transition until the input is completely presented. The singular transition of the network output to completeness indicates that the input of the network has transitioned to completeness, that the consequent wavefront has propagated completely through the network and that the asserted output is the correct result for the presented input for both the transition to **D** completeness and the transition to completely **N**.

3.4.8.   Composing larger localities of interaction differentness

The cross association search in which two input localities each with a small range of mutually exclusive differentnesses mutually inclusively combine to form a single locality with a larger range of mutually exclusive differentness inspires the means of expressing localities with large ranges of mutually exclusive differentness in terms of mutually inclusive combinations of localities with small ranges of mutually exclusive differentness. A place value number, for instance, is a locality of mutually inclusive range, **"all of"**, **[ ]** range of places each with a mutually exclusive, **"one of"** **{ }** range of digit diferentnesses.

A two bit binary number with a range of 4 mutually exclusive differentnesses is expressed as:

twobit/[1 0]/{1 0}/   which expands to   **twobit/[1/{1 0}/ 0/{1 0}/]** which expands to **twobit/[1/{1 0}/{D N}   0/{1 0}/{D N}]**   which expands to **twobit/[1/{1/{D N} 0/{D N}}   0/{1/{D N} 0/{D N}}]**

Examples of two bit locality are **qtob/B and btoq/A** in section 3.4.9.
A five bit binary number with a range of 32 mutually exclusive differentnesses is:

fivebit/[4-0]/{1 0}/   which expands to   **fivebit/[4 3 2 1 0]/{1 0}/**   which expands to
fivebit/[4/{1 0}/ 3/{1 0}/ 2/{1 0}/ 1/{1 0}/ 0/{1 0}/]   which expands to
fivebit/[4/{1 0}/{D N} 3/{1 0}/{D N} 2/{1 0}/{D N} 1/{1 0}/{D N} 0/{1 0}/{D N}]
    which expands to
fivebit/[4/{1/{D N} 0/{D N}}
        3/{1/{D N} 0/{D N}}
        2/{1/{D N} 0/{D N}}
        1/{1/{D N} 0/{D N}}
        0/{1/{D N} 0/{D N}}]

and so on. An example of five bit locality is **fivebitadder/A** in Section 3.4.10.2.
Bits are referenced as

twobit/1   fivebit/4   fivebit/3

Bit differentnesses are referenced as

twobit/1/0   fivebit/4/1   fivebit/3/0

and so on.

3.4.9.   Composing larger constant networks

Larger constant networks are composed from smaller constant networks by associating binding portal output to binding portal input. The first example is a composition of two component constant networks. Network **btoq** converts a two digit binary radix 2 representation to a one digit quaternary radix 4 representation. Network **qtob** converts quaternary back to binary.

btoq　　　**bound reference name**　　**binding portal localities**

A/1/1 → B/3
A/1/0 → B/2
A/0/1 → B/1　　**dependency relations**
A/0/0 → B/0

$(btoq(A/[1\ 0]/\{1\ 0\}/ \Rightarrow B/\{3\ 2\ 1\ 0\}/)$

$B/\{3<=[A/1/1\ A/0/1]$
$2<=[A/1/1\ A/0/0]$
$1<=[A/1/0\ A/0/1]$
$0<=[A/1/0\ A/0/0]\ \}\ )$
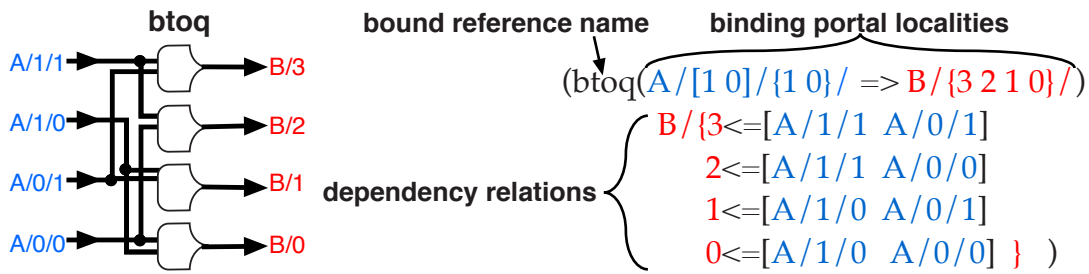
The components of locality **B/[3 2 1 0]** are dependent on relations among components of input locality **A[1 0]/{1 0}**.

qtob　　　**bound reference name**　　**binding portal localities**

A/3 → B/1/1
A/2 → B/1/0
A/1 → B/0/1　　**dependency relations**
A/0 → B/0/0

$(qtob(A/\{3\ 2\ 1\ 0\}/ \Rightarrow B/[1\ 0]/\{1\ 0\}/)$

$B/[\ 1\{1<=\{A/3\ A/2\}$
$0<=\{A/1\ A/0\}\}$
$0\{1<=\{A/3\ A/1\}$
$0<=\{A/2\ A/0\}\}\ ]\ )$
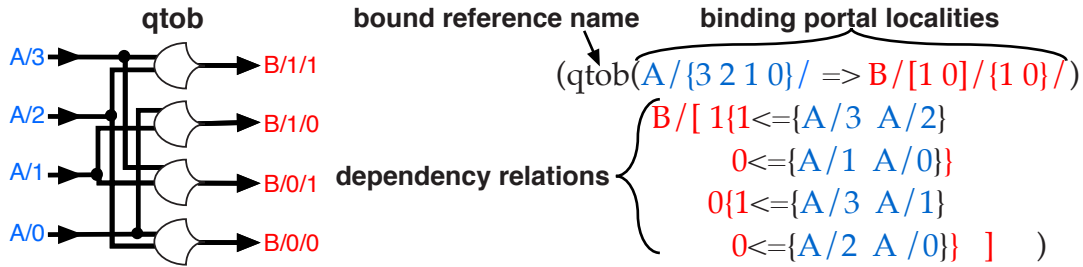
The components of locality **B/[1 0]/{1 0},** are dependent on relations among components of input locality **A[3 2 1 0]**.

The two component constant networks are composed into a larger constant network named **biggernet** by representing the dependency relation between binding portals with portal nesting, locality nesting or full portal reference.

Composing with portal nesting and locality nesting

bound reference name　　　　　　　**binding portal localities**

$(biggernet(X/[1\ 0]/\{1\ 0\}/ \Rightarrow Z/[1\ 0]/\{1\ 0\}/)$
dependency relations → $Z<=qtob(btoq(X))\ )$

Composing with locality nesting

bound reference name　　　　**binding portal localities**　　**internal localities**

$(biggernet\ (X/[1\ 0]/\{1\ 0\}/ \Rightarrow Z/[1\ 0]/\{1\ 0\}/)\ (Y/\{3\ 2\ 1\ 0\}/)$
dependency relations
$Y<=btoq(X)$
$Z<=qtob(Y)\ )$

Composing with full portal reference

bound reference name   **binding portal localities**   **internal localities**

(biggernet (X/[1 0]/{1 0}/ => Z/[1 0]/{1 0}/) (Y/{3 2 1 0}/)

dependency relations { btoq(X => Y)
                           qtob(Y => Z))

Locality **Z** as a whole is dependent on the asserted differentness of **qtob** which is dependent on the asserted differentness of **btoq** which is dependent on the presented differentness of **X** as a whole. The intermediate locality **Y** is also dependent as a whole. All three of the above dependency expressions specify the constant network of Figure 3.17.

btoq   qtob

X/1/1   Y/3   Z/1/1
X/1/0   Y/2   Z/1/0
X/0/1   Y/1   Z/0/1
X/0/0   Y/0   Z/0/0

Figure 3.17. Composed constant network.

3.4.9.1.   Associating exposed binding portals
    When binding portals are associated a portion of the binding portals and a portion of the previously uncharacterized external environments (section 3.4.5.4) just beyond the exposed binding portals become incorporated into and fully characterized within the composite network. The un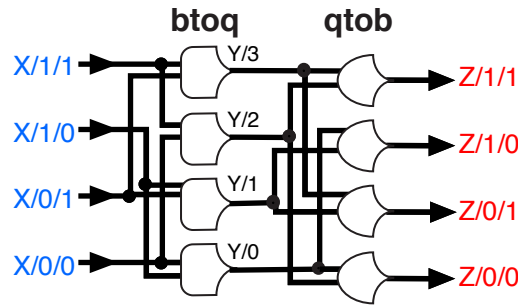associated portions of the binding portals form the exposed binding portal of the new composite network as a whole which retains the singular appreciability of its interaction as a whole. The newly component networks become subordinate to and dependent on the new exposed binding portal and its input wavefront of transition for their liveness and time (section 3.4.9.8).
    No matter how big and complex a composed constant network becomes it never outgrows the need for an exposed binding portal to an uncharacterized external environment in complete control of the constant network (section 3.4.5.4).

3.4.9.2.   Locality correspondence
    When an output locality of one network is associated to an input locality of another network the structure of the associated localities must correspond exactly in that they effectively express a same single locality, just with different names. The structure of locality **Z** must match the structure of locality **qtob/B**. The structure of locality **X** must match the structure of locality **btoq/ A**. The input locality structure of locality **qtob/A** must match the corresponding nested output locality structure of **btoq/B** to which the intermediate locality **Y** must also correspond.

3.4.9.3.   Inheritance of locality structure
    One approach to expressing locality structure correspondence is to require that all locality expressions be fully specified at all levels and that the structure of all dependently related localities correspond. Another approach is to allow a locality to be referenced by name only and to inherit its structure from an already fully expressed reference to the common locality. There

can be multiple references to the same locality and they must all correspond. If some references are partial they can be filled in from other references. The references among themselves must compose to a single expression of the locality. The **biggernet** dependency expression above can be rendered as:

<div align="center">

(biggernet (X -> Z)
       Z<=qtob(btoq(X))   )

</div>

Z inherits locality structure from **qtob/B/[1 0]/{1 0}/** resulting in

<div align="center">

Z/[1 0]/{1 0}/

</div>

X inherits its locality structure from **btoq/A/[1 0]/{1 0}/** resulting in

<div align="center">

X/[1 0]/{1 0}/

</div>

biggernet expanded with the inheritance is

<div align="center">

(biggernet (X/[1 0]/{1 0}/ -> Z/[1 0]/{1 0}/)
       Z<=qtob(btoq(X))   )

</div>

The **biggernet** dependency expression can also be rendered as:

<div align="center">

(biggernet (X -> Z) (Y)
      Y<=btoq(X)
      Z<=qtob(Y)   )

</div>

with the inheritance including **Y** inheriting from **btoq/B** and from **qtob/A**:

<div align="center">

(biggernet (X/[1 0]/{1 0}/ -> Z/[1 0]/{1 0}/) (Y/{3 2 1 0}/)
      Y<=btoq(X)
      Z<=qtob(Y)   )

</div>

Inheritance simplifies expression and reduces opportunities for errors.

3.4.9.4.   Composing constancy and completeness behavior

If each component constant network of a composite constant network is constant (section 3.4) and fulfills the completeness of input criterion (section 3.4.7.2) then the composite constant network as a whole will be constant and fulfill the completeness of input criterion. The point is illustrated in Figure 3.18 showing a larger network composed with the network of Figure 3.11 copied three times.

The component network expression is:

<div align="center">

(net(A/{1 0}/ B/{1 0}/ => C/{1 0}/)
          C/{1<={[A/0 B/1] [A/1 B/0]}
             0<={[A/0 B/0] [A/1 B/1]} }   )

</div>

The composed network expression is:

<div align="center">

(bignet(W X Y Z => C)
          C<=net(net(W X) net(Y Z))   )

</div>

Which expands with inheritance to:

(bignet(W/{1 0}/ X/{1 0}/ Y/{1 0}/ Z/{1 0}/ => C/{1 0}/)
C<=net(net(W X) net(Y Z))   )

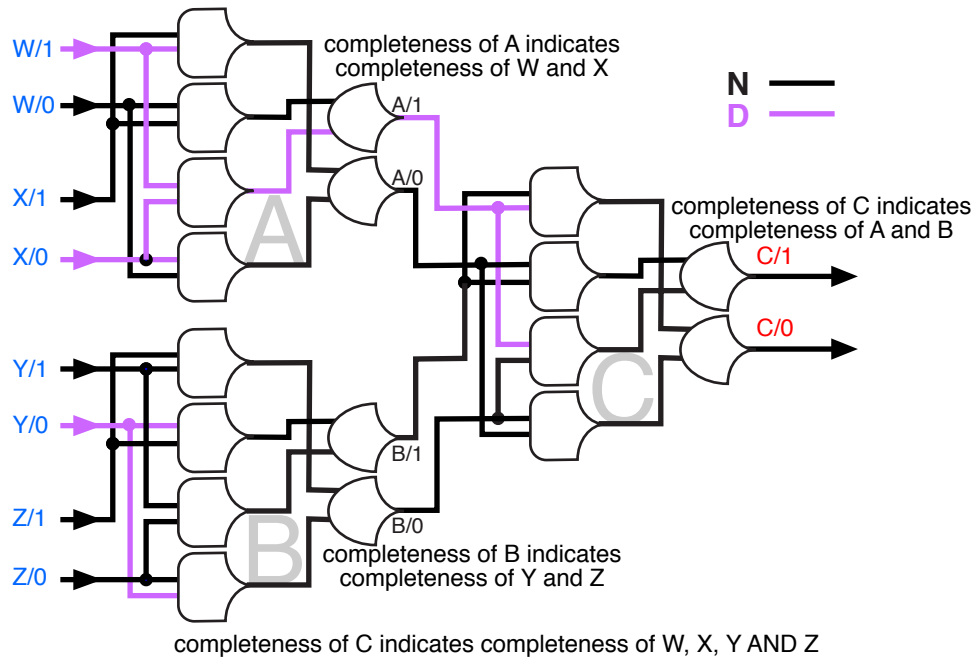completeness of C indicates completeness of W, X, Y AND Z

Figure 3.18. The composition of completeness relations in bignet.

3.4.9.5.   Constancy

The completeness of **C** is dependent through **net C** on the completeness of **A** which is dependent through **net A** on the completeness of **W** and **X** and on the completeness of **B** through **net B** which is dependent on the completeness of **Y** and **Z**. If **A** is constant in relation to **W** and **X** and **B** is constant in relation to **Y** and **Z** and **C** is constant in relation to **A** and **B** then **C** is constant in relation to **W**, **X**, **Y** and **Z**.

3.4.9.6.   The completeness of input criterion

In Figure 3.18 the completeness of **C** implies the completeness of its two inputs **A** and **B**. The completeness of **A** implies the completeness of **W** and **X**. The completeness of **B** implies the completeness of **Y** and **Z**. Then the completeness of **C** implies the completeness of **W**, **X**, **Y** and **Z**. If any of **W**, **X**, **Y** and **Z** are not completely transitioned **C** will not be completely transitioned. This is the case for both the transition to **D** completeness and for the transition to completely **N**.

3.4.9.7.   Concurrency

The transition to completeness of localities **A** and **B** are singularly appreciable events in relation to **net A** and **net B** respectively. However in relation to **bignet** they are independent and concurrent. The concurrent behaviors of **net A** and **net B** are coordinated by the completeness behavior of **net C**. Figure 3.18 illustrates a transition to **D** presentation that is not yet complete. **W**, **X** and **Y** have transitioned to **D** completeness but **Z** has not yet transitioned. **C** will transition to **D** completeness only after **Z** transitions to **D** completeness.

3.4.9.8.   Instances of interaction, wholeness and chaos

As constant networks are composed into larger composite constant networks the newly component networks become dependent on the wavefront of the composite network for their

input presentation. While each component network retains the same singularity of behavior in its own local context that it exhibited as a stand alone network each component singularity of behavior is now one singularity among many in the composite network as a whole. As an interaction wavefront flows through a composite network each component network and each component primitive behavior appreciates its own output transition to completeness marking its own local instance of interaction. Even though these instances of component interaction are all subordinate to and dependent on the flow of the composite network wavefront and are determined by dependency relations that converge to transition to completeness of the output locality of the composite network there is no singularly coherent relation among the collective component transition behaviors. The flowing wavefront that produces a singular transition event for the network as a whole (section 3.4.6.3) produces an incoherent chaos of local component events in relation to the network as a whole.

As a constant network grows larger the extent of this internal incoherence increases within the network while the transition to completeness of composite network's output locality remains the only singularly appreciable behavior in relation to the network as a whole marking one instance of interaction (sections 3.4.5.5 and 3.4.6.4).
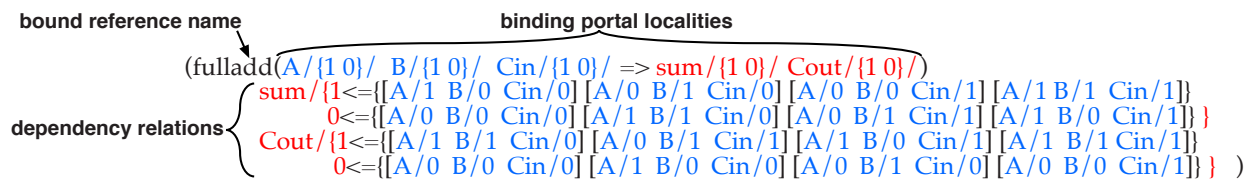
3.4.10.   5 bit adder constant network

The 5bit addition example illustrates a more complex constant network, that fulfills the completeness of input criterion and which can be indiscriminately referenced and copied.

3.4.10.1.   component

The first step is to express the full adder and compose a 5 bit number locality then to compose the 5 bit adder in terms of the full adder and 5 bit number localities.

The fulladd expression specifies the fulladd dependency network in the left of Figure 3.19.
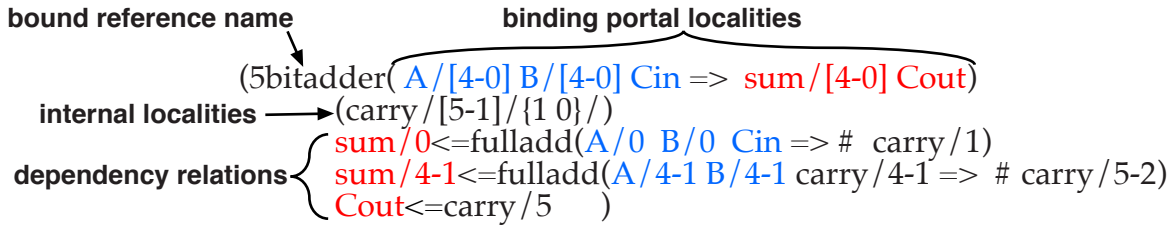
bound reference name   **binding portal localities**

$$(fulladd(A/\{1\,0\}/\ B/\{1\,0\}/\ Cin/\{1\,0\}/ => sum/\{1\,0\}/\ Cout/\{1\,0\}/)$$

dependency relations

$$sum/\{1<=\{[A/1\ B/0\ Cin/0]\ [A/0\ B/1\ Cin/0]\ [A/0\ B/0\ Cin/1]\ [A/1\ B/1\ Cin/1]\}$$
$$0<=\{[A/0\ B/0\ Cin/0]\ [A/1\ B/1\ Cin/0]\ [A/0\ B/1\ Cin/1]\ [A/1\ B/0\ Cin/1]\}\ \}$$
$$Cout/\{1<=\{[A/1\ B/1\ Cin/0]\ [A/0\ B/1\ Cin/1]\ [A/1\ B/0\ Cin/1]\ [A/1\ B/1\ Cin/1]\}$$
$$0<=\{[A/0\ B/0\ Cin/0]\ [A/1\ B/0\ Cin/0]\ [A/0\ B/1\ Cin/0]\ [A/0\ B/0\ Cin/1]\}\ \}\ )$$

sum is dependent on the dependency of its components, **[1 0]** on input localities **A**, **B** and **Cin** distributed by the binding portal. **Cout** is dependent on the dependency of its components, **[1 0]** on input localities **A**, **B** and **Cin** distributed by the binding portal.

3.4.10.2.   Composition

With **fulladd** an expression is encountered with two output assertions. This poses a difficulty with representing dependency with portal nesting as the portal nesting relation is one to one. Two versions of **fulladd** can be expressed, one with a **sum** output and one with a **Cout** output each of which can then be nested. Another approach with full portal reference is to designate the dependency of one output as nested with **#** in its syntactic position and to designate the dependency of the other output by name correspondence with a name in its syntactic position.

In the references to **fulladd** in the **5bitadder** expression the dependency relation of the assertion locality **sum** is designated to be nested, **#**, and the dependency relation of the assertion locality **Cout** is represented with name correspondence, **carry/x**.

The **5bitadder** expressed in terms of **fulladd** and the structure of the 5 bit number locality.

bound reference name   **binding portal localities**

(5bitadder(A/[4-0] B/[4-0] Cin => sum/[4-0] Cout)

internal localities ⟶ (carry/[5-1]/{1 0}/)

dependency relations ⟨ sum/0<=fulladd(A/0 B/0 Cin => # carry/1)
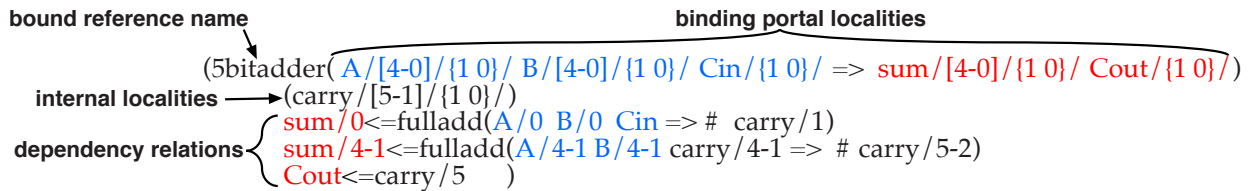sum/4-1<=fulladd(A/4-1 B/4-1 carry/4-1 => # carry/5-2)
Cout<=carry/5   )

3.4.10.3.   Inheritance

The locality expressions **A/[4-0]**, **B/[4-0]**, **sum[4-0]** and **Cout**, because they do not include the terminal dangling / or /**{D N}**, are not complete. **A/[4-0]** just specifies **"all of"** five things named **4**, **3**, **2**, **1**, and **0**. What the things are will be inherited by reference.
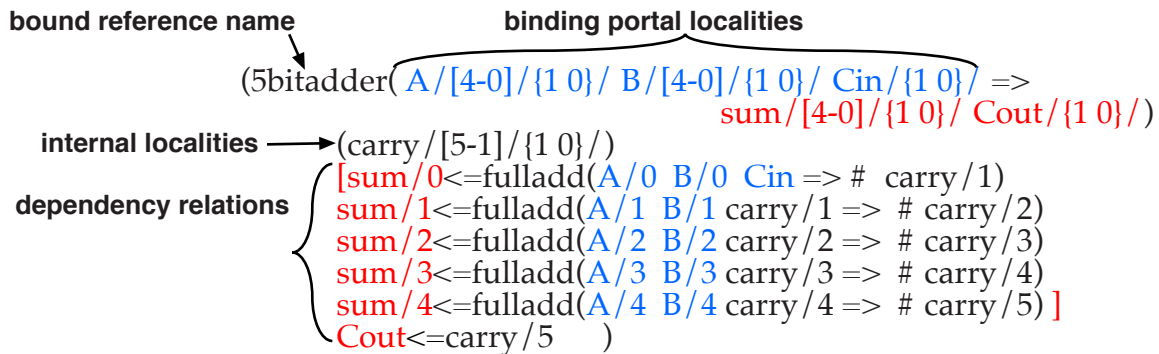
Each referential **A/4-1** expands to **A/4 A/3 A/2 A/1** with each **A/x** referencing an instance of **fulladd**. Each instance of **A/x** corresponds to **fulladd/A** and inherits from **fulladd/A** completing the referencing locality. The locality expression **A[4-0]** becomes **A[4-0]/{1 0}/** completely representing a 5 bit binary number. The inheritance could have been **A[4-0]/{2 1 0}/** representing a five trit number or **A[4-0]/{3-0}/** representing a five quat number.

The inheritance expansion from **fulladd**

bound reference name   **binding portal localities**

(5bitadder(A/[4-0]/{1 0}/ B/[4-0]/{1 0}/ Cin/{1 0}/ => sum/[4-0]/{1 0}/ Cout/{1 0}/)

internal localities ⟶ (carry/[5-1]/{1 0}/)

dependency relations ⟨ sum/0<=fulladd(A/0 B/0 Cin => # carry/1)
sum/4-1<=fulladd(A/4-1 B/4-1 carry/4-1 => # carry/5-2)
Cout<=carry/5   )

sum/[4-0] expands to **sum/4**, **sum/3**, **sum/2**, **sum/1** each referencing an instance o fulladd.

The above expression expands to:

bound reference name   **binding portal localities**

(5bitadder(A/[4-0]/{1 0}/ B/[4-0]/{1 0}/ Cin/{1 0}/ =>
sum/[4-0]/{1 0}/ Cout/{1 0}/)

internal localities ⟶ (carry/[5-1]/{1 0}/)

dependency relations ⟨ [sum/0<=fulladd(A/0 B/0 Cin => # carry/1)
sum/1<=fulladd(A/1 B/1 carry/1 => # carry/2)
sum/2<=fulladd(A/2 B/2 carry/2 => # carry/3)
sum/3<=fulladd(A/3 B/3 carry/3 => # carry/4)
sum/4<=fulladd(A/4 B/4 carry/4 => # carry/5)]
Cout<=carry/5   )

3.4.10.4.   Adder Network

The above expressions map to the **5bitadder** constant network in the right of Figure 3.19. The 5 bit adder constant network is composed in terms of **fulladd** component constant networks. The 5 bit number localities are composed in terms of 1 bit digit localities.
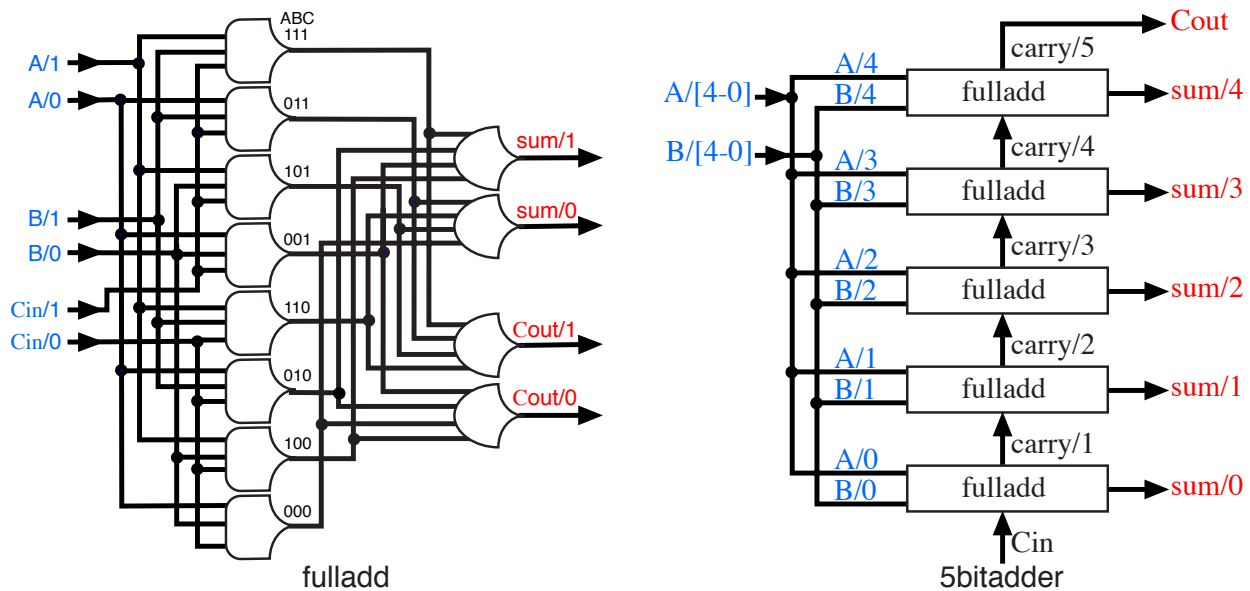
Figure 3.19. Fulladd network and 5 bit adder network.

3.4.10.5.   Constancy and completeness

Each **fulladd** is constant and fullfils the completeness criterion. Each successive **fulladd** in the **carry** chain of **5bitadder** requires the predecessor **carry** to meet completeness of input to transition its output. If any part of **A** or **B** is not complete then at least one **fulladd** will not achieve input completeness, not transition its **sum** and at least one part of the **5bitadder** **sum** will not transition to completeness. If the **5bitadder** **sum** transitions to completeness it implies that **A** and **B** and **Cin** have transitioned to completeness fulfilling the completeness criterion for the network as a whole.

The expression of a 5 bit multiplier constant network is presented in Appendix F.

3.5.   INTERLUDE: The constant network

A constant network is **a network of dependency relations** among primitive behaviors and localities of differentness with the primitive behaviors linearly and progressively associated output to input i.e. no circular or feedback associations (section 3.4.2). It is *constant* in that it always asserts the same completeness of output differentness for the same presented completeness of input differentness (section 3.4). It *fulfills the completeness of input criterion* in that the transition of its output to completeness implies that the presented input has transitioned to completeness, that the interaction is complete and that the output is the correct result for the presented input (section 3.4.7.2). It is *indiscriminately composeable* in that it, can be referenced from and copied to anywhere and anywhen, in particular to compose bigger constant networks (section 3.4.9).

A constant network manifests interaction as *wavefronts of differentness transition* flowing through the network coordinated in terms of *completeness relations* (section 3.4.6). A wavefront is initiated by the transition of input presented from an *external environment* to an *exposed binding portal* which bounds the network with input and output ports (section 3.4.5). The constant network is subordinate to and is controlled by the input presented from the external environment which provides its source of liveness (with presentation), determines its temporal

instance of interaction (with the monotonic transitions between **D** completeness and completely **N**) and provides its source of variability of behavior (The constant network itself is constant. Its behavior varies only with the differentness of its presented input).

The responsible environment

For a constant network to behave the environment is still **required to monotonically transition inputs to the network with an appropriate delay** between transition to **D** completeness and transition to completely **N** to allow each consequent transition wavefront to flow completely through the network. This responsibility **is now relegated to the input presentation of the exposed binding portal of the constant network as a whole**. If fulfilled for the constant network as a whole then the monotonic transitioning and delay requirements for the primitive interaction behaviors and component constant networks within the composite constant network are also fulfilled.

By the end of this chapter the network will be in complete control of itself in relation to a passively indifferent environment.

3.5.1. Constant network space
The constant network bounded by its exposed binding portal is the interaction space.

3.5.2. Constant network time
A constant network can perform only one interaction at a time. The transition of the its output locality to **D** completeness implying that the presented input is complete and that the output is the correct result of the presented input is the only singularly appreciable network behavior bounded by the exposed binding portal and framed by transition to completely **N** wavefronts coherently marking one instance of interaction and one instance of interaction time for the network as a whole (sections 3.4.6.5 and 3.4.6.3).

3.5.2.1. Extending association differentness through time
Even if successive interactions are identical in all aspects of transition behavior the interactions are different by virtue of differentness of instances of interaction time. Each instance of interaction, each reuse of a constant network, extends its expression of differentness of interaction through time. This is temporal differentiation (Chapter 4).

3.5.2.2. Ephemeral time
However, these instances of interaction are ephemeral (section 3.4.6.5), disappearing from the network, never associating and never interacting in the context of the network which cannot account this extension of expression of differentness of interaction.

3.5.2.3. Interaction incoherence
Constant network individual inputs can transition to completeness in varying orders including all at once. Wavefront transitions flowing through the network concurrency relations with varying delays can transition in varying orders including all at once (section 3.4.6.4 and 3.4.9.8). There is no singular referent of time or space that coherently relates them. Wavefront flow through the network is coordinated in terms of dependency completeness relations not in terms of temporal relations or spatial relations.

3.5.2.4. Wholeness
The exposed binding portal bounds the network providing its liveness and time and realizing the network's reuse determines the wholeness of the network, its instances of interaction and of

interaction time (sections 3.4.5.5 and 3.4.9.8). This wholeness of constant network is not wholeness of expression which leaks through the binding portal into the external environment not accounted by the constant network.

3.5.3.   Nothing new

A constant network is a linear progression of specifically interacting differentnesses represented as a directed network of dependency relations among the primitive behaviors defined in section 3.2. Nothing new has been introduced either abstractly or substantially.

3.5.4.   The transcendental view

Because a constant network always asserts the same result differentness for the same presented input differentness, a constant network, however complex, can be abstractly characterized as a single step mapping transcending and ignoring its internal structure and behavior which includes intrinsic concurrent dependency relations. This equates the network with a single step, single place primitive behavior. This reduction of complexity to primitivity rather overlooks the emergence of complexity from primitivity.

3.6.   QUANDARY 5: The environment

On the one hand the external environment can be incorporable into a constant network. On the other hand the external environment is never fully incorporated. What and where is this inscrutable external environment that is in complete control of the constant network, residing always on the other side of its exposed binding portal, undoubtably there, but always receding just beyond grasp like the never findable end of the rainbow?

This binding portal exposed to an uncharacterized environment is intrinsic to constant networks. If a constant network were not completely dependent on its presented input it would not be indiscriminately referenceable and copyable. But the quandary remains. *Key aspects of a constant network, its variability of behavior, its time and its very liveness lie beyond its expressivity.*

3.7.   The oscillation network: self regulation

A constant network fulfilling the completeness of input criterion can use the completeness transitioning of its output to self regulate wavefront flow into the network. In Figure 3.20 the completeness of the output transition (in this case the transition of **"one of"** C/0 or C/1 to **D** or the transition of **C** to **"all of"** N) is appreciated by a **"one of"** behavior which reduces the completeness of output to a single place of association C.comp with condition **D** representing **D** completeness and condition **N** representing completely **N**. C.comp is the singular appreciation of the singularly appreciable completeness of the output locality. This singular appreciation of completeness of interaction corresponds to the **Z** condition of the Roman numeral example of Chapter 2 (section 2.4.4).
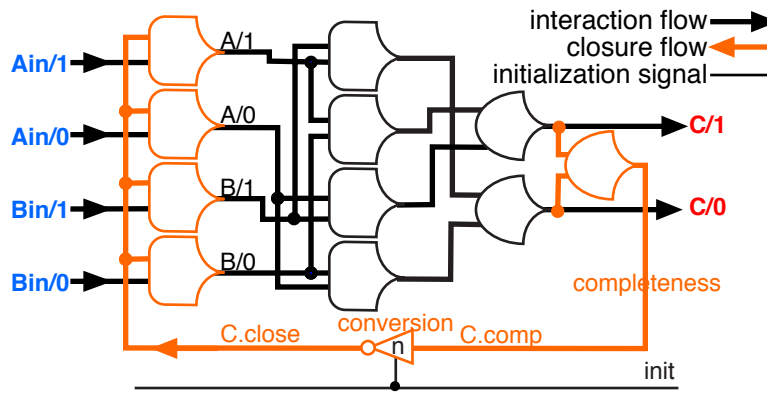
Figure 3.20. Self regulating oscillation network.

The dependency relations for appreciating and reducing the completeness of a locality to a single place of association are contained in the expression of each locality (section 3.9.2).

3.7.1.   Regulating network input

The appreciation of output completeness **C.comp** is converted ~**C.comp**=>**C.close** and closes with the input through **"all of"** behaviors to allow the next transition wavefront into the network. With the network empty and its output completely **N** **C.comp** will be **N** and ~**C.comp**=>**C.close** will be **D** which will enable a wavefront of transition to **D** completeness to flow through the rank of **"all of"** behaviors into the constant network. **C.close** will remain **D** with the **"all of"** behaviors maintaining the **D** completeness wavefront until the wavefront has propagated all the way through the network and transitioned the output to **D** completeness. If **Ain** and **Bin** transition to completely **N** they will wait for **C.comp** to transition to **N**.

When the output transitions to **D** completeness **C.close** transitions to **D** and ~**C.comp**=>**C.close** transitions to **N** allowing a transition to completely **N** wavefront into the constant network. If the input is not completely **N** the rank of **"all of"** behaviors will wait for the input to transition to completely **N**. **C.close** will remain **N** with the **"all of"** behaviors maintaining the **N** wavefront until the wavefront has propagated all the way through the network and transitioned the output to completely **N**. If **Ain** and **Bin** transition to **D** completeness they will wait for **C.close** to transition to **D**.

When the output transitions to **N** completeness **C.comp** transitions to **N** and ~**C.comp**=>**C.close** transitions to **D** allowing a transition to **D** completeness wavefront into the constant network. If the input is not **D** completeness the rank of **"all of"** behaviors will wait for the input to transition to **D** completeness. When the transition to **D** completeness wavefront arrives it passes through the **"all of"** behaviors into the constant network. **C.close** will remain **D** with the **"all of"** behaviors maintaining the **D** completeness wavefront until the wavefront has propagated all the way through the network and transitioned the output to **D** completeness. If **Ain** and **Bin** transition to completely **N** they will wait for **C.close** to transition to **N**.

When the output transitions to **N** completeness **C.comp** transitions to **N** and ~**C.comp**=>**C.close** transitions to **D** allowing a transition to **D** completeness wavefront into the constant network.

And so on.

The next input transition is not allowed into the network until the current transition wavefront has completely propagated through the network and transitioned the output. The closure, recognizing the singularly appreciable event of the output of the constant network transitioning
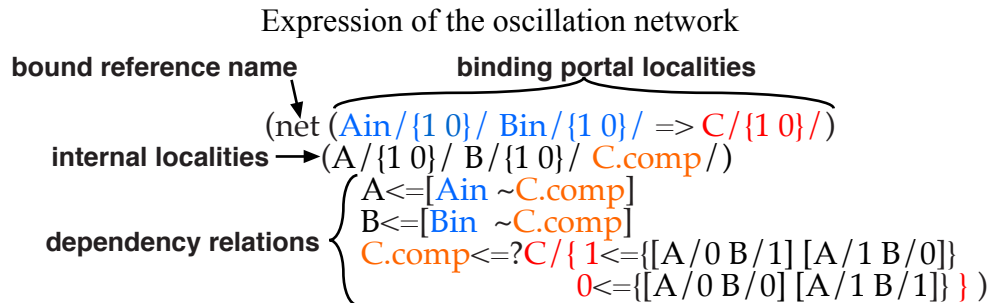
to completeness manages the monotonic transitioning of input presentations regulating the flow of wavefronts into the network.

3.7.2.   Expressing the oscillation network

The closed network with a single converted differentness appreciating the monotonic transitions or oscillations between two disjoint completeness representations is similar to a spontaneously alive binary ring oscillator circuit with a single inversion. So the network is called an *oscillation network*.

The **black** paths in Figure 3.20 are interaction wavefront flow paths. The **orange** paths are closure flow paths. The initializing converter allows the network to be initialized to all **N**, empty of interaction differentness ready for the first transition to **D** completeness (appendix section D.1.1.)

The network expression with closure introduces two syntax elements. The tilde ~ represents conversion of **D** to **N** or of **N** to **D** (Section 3.2.1). The conversion occurs only in closure, never in an interaction wavefront. The **?** before a locality name indicates the completeness reduction of the referenced locality to a single place of association of the same name with a **.comp** suffix, **name.comp**.

Expression of the oscillation network

bound reference name   **binding portal localities**

$$(\text{net } (\text{Ain}/\{1\ 0\}/\ \text{Bin}/\{1\ 0\}/\ => C/\{1\ 0\}/)$$

internal localities $\rightarrow (A/\{1\ 0\}/\ B/\{1\ 0\}/\ C.\text{comp}/)$

dependency relations
$$\begin{cases} A <= [\text{Ain} \sim C.\text{comp}] \\ B <= [\text{Bin} \sim C.\text{comp}] \\ C.\text{comp} <= ?C/\{\ 1 <= \{[A/0\ B/1]\ [A/1\ B/0]\} \\ \qquad\qquad\qquad 0 <= \{[A/0\ B/0]\ [A/1\ B/1]\}\ \} \ ) \end{cases}$$

C.comp is dependent on the completeness of **C** the completeness of which is dependent on the components of **A** which are dependent on **Ain** and conversion of **C.comp** and on the components of **B** which are dependent on **Bin** and conversion of **C.comp**.

3.8.   INTERLUDE: Marking time with the oscillation network

Closing a constant network with the converted output locality completeness enabling the network input creates a spontaneously alive, self regulating network continually striving to transition between **D** completeness and completely **N** performing the interaction of its encompassed network on successive presentations of completeness transition with its own appropriate delay.

The encompassed interaction network provides *interaction behavior*. The input **"all of"** enable rank provides *memory behavior*. The completeness of input criterion and the completeness closure provide *coordination behavior*. These are the same properties exhibited by the primitive behaviors in section 3.3. At this point the oscillation network can be considered an abstract primitivity. All networks from now on will be composed of oscillation networks.

The encompassed interaction network must be constant. If a wavefront in the network flows anywhere other than to contribute to the output of the network the completeness of the output of the network will not imply the status of the non contributing wavefront.

<div style="border:1px solid black">

The responsible environment

An oscillation network **self regulates the monotonic transitioning delay** of its input between **D** completeness and completely **N**. Prior to this point the environment determined the delay between presentation transitions in terms of a conventional time metric. The environment still has an imposed requirement for monotonically transitioning presented input with an appropriate delay between transitions but now the **oscillation network itself determines the appropriate delay between transitions of input presentation**. The responsibility of the environment is now to honor the closure by next transitioning the input only after each transition wavefront is accepted with closure by the network.

The environment is still a source of variability and monotonic transitioning but it is no longer in complete control. The network itself is now in control of its own time. There is no longer any appeal to a conventional metric of time.

</div>

3.8.1. Oscillation network space and time

The constant network encompassed by the closure is the interaction space. The transitions of the output completeness between **D** completeness and completely **N** are the appreciable boundaries of instances of interaction and differentness of interaction time in relation to the encompassed constant network. The oscillation network appreciating these boundaries serves as an *escapement mechanism* establishing a nonuniform metric of interaction time for its encompassed constant network in relation to its flowing wavefronts of transition. The oscillation network is now in control of its own interaction time and there is no longer any appeal to a conventional metric of time. Network interaction time becomes a metric of time in itself unrelated to any conventional metric of time.

From this point on networks will be in control of their own time.

3.8.2. Nothing new

An oscillation network is an emergent behavior expressed solely in terms of dependently associating the defined primitive behaviors (section 3.2). Nothing new has been introduced either abstractly or substantially.

3.8.3. No external metrics

There is no coherent reference frame or external metric of space or of time relative to an oscillation network and trying to impose an external metric onto the oscillation network contributes nothing to either the understanding of or the effective realization of the oscillation network.

3.8.4. The transcendental view

The closure network can be removed and ignored and the constant network within the closure network viewed as a single step mapping behavior if one wishes.

3.9. The pipeline network: composing self regulation

A pipeline network is a structure of linked oscillation networks coordinating their wavefront flow with closure relations. Placing the output completeness determination of oscillation network **A** after the input enable closure of oscillation network **B** links the two oscillation networks coordinating wavefront flow from oscillation network **A** to oscillation network **B**.

Figure 3.21 shows a **pink** an **orange** and a **blue** oscillation network. They are linked by placing the completeness of the **pink** network after the enable of the **orange** network and by placing the completeness of the **orange** network after the enable of the **blue** network forming a pipeline of linked oscillation networks.
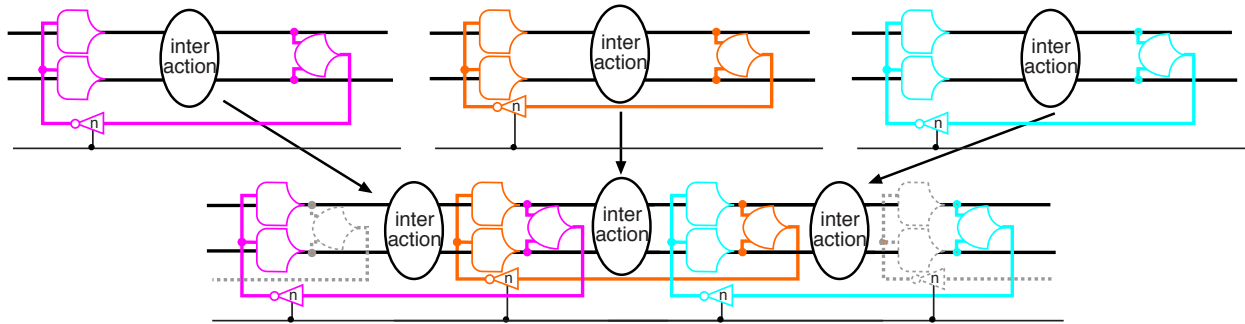
Figure 3.21. linking oscillation networks.

The wavefront accepted by the **pink** network from a previous network flows to the enable of the **pink** network and is stably maintained until accepted by the **orange** network however long this might take. When accepted by the **orange** network the **pink** network can determine the completeness of its flow and can accept the next wavefront transition into its network.

The wavefront accepted by the **orange** network from the **pink** network flows to the enable of the **blue** network and is stably maintained by the **orange** network until accepted by the **blue** network however long this might take. When accepted by the **blue** network the **orange** network can determine the completeness of its flow and can accept the next wavefront transition into its network.

The wavefront accepted by the **blue** network from the **orange** network flows to the enable of a next network and is stably maintained by the **blue** network until accepted by the next network however long this might take. When accepted by the next network the **blue** network can determine the completeness of its flow and can accept the next wavefront transition from the **orange** network into its network.

And so on.

3.9.1.   Pipeline wavefront flow

Each oscillation network of a pipeline network is individually and spontaneously alive continually striving to transition between **D** completeness and completely **N**. The oscillation networks mutually coordinate each other's striving through the closure links. At each closure link an early interaction wavefront transition will wait indefinitely for a corresponding closure enable transition and an early closure enable transition will wait indefinitely for a corresponding interaction wavefront transition. As oscillation networks individually oscillate, alternating wavefronts of transition to **D** completeness and completely **N** spontaneously flow, fully coordinated, from oscillation network to oscillation network through the pipeline network with transition to **D** wavefronts interacting as they flow through the constant networks encompassed within the oscillation networks.

3.9.2.   The closure link

Placing the completeness for a locality of a delivering oscillation network after the enable of the same locality of the receiving oscillation network forms a closure **link**. A closure link for a locality is determined by the expression of the locality which specifies the determination of its

transition to **D** completeness and its enable completeness. Determining the transition to completely **N** is universally the same for all localities. Figure 3.22 shows closure links for several locality structures with their completeness determination networks after their locality enable network. A link is typically on the output of a network and is defined by the expression of the output locality. In Figure 3.25 through Figure 3.30 each locality declaration is shown above the corresponding completeness determination network.
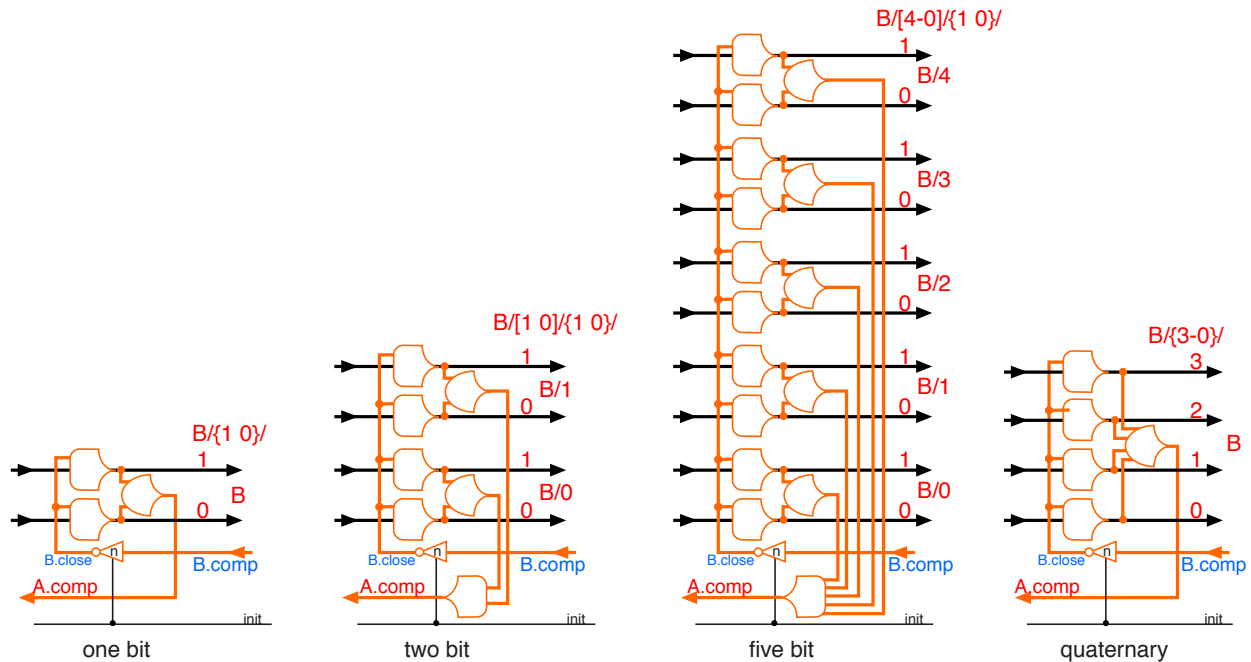
Figure 3.22. Completeness of localities with their structure expressions.

See the localities of **net** Figure 3.24 for one bit locality links. See **qtob/B** Figure 3.25 for a two bit locality link. See **5bitadd** section 3.4.10 for five bit locality links. See **qtob/A** Figure 3.25 for a quaternary locality link.

The closure link is similar to the notion of asynchronous handshakes.

> "in its most general form, asynchronous design removes the global clock in favor of distributed local handshaking to control data transfer and changes of state." [6]

This is a view that defers to clocked Boolean design as the fundamental referent with asynchronous design as a subordinate derivative variation. The present narrative takes the opposite view considering depemdently flowing asynchronous behavior as fundamental and synchronous behavior as derivative (sections 1.1.1, 1.8, 3.4.6 and 5.7.5). Closure represents a more integrated wholeness than does the notion of handshake.

3.9.3.   The closure protocol

- A locality completeness closes with (enables monotonic transition of) all localities on which its completeness is dependent.
- A locality is closed by (its monotonic transition enabled) all the localities which depend on it for their own completeness transition.

a constant network typically has two or more input localities and one output locality which is dependent on all of the input localities. So the completeness of the output closes with all the inputs as shown in Figure 3.20 forming an oscillation network. The inputs, however, are two different localities from two different networks and must be viewed as closed individually. The oscillation network on the left of Figure 3.23 is redrawn to show the localities **A** and **B** closed individually.
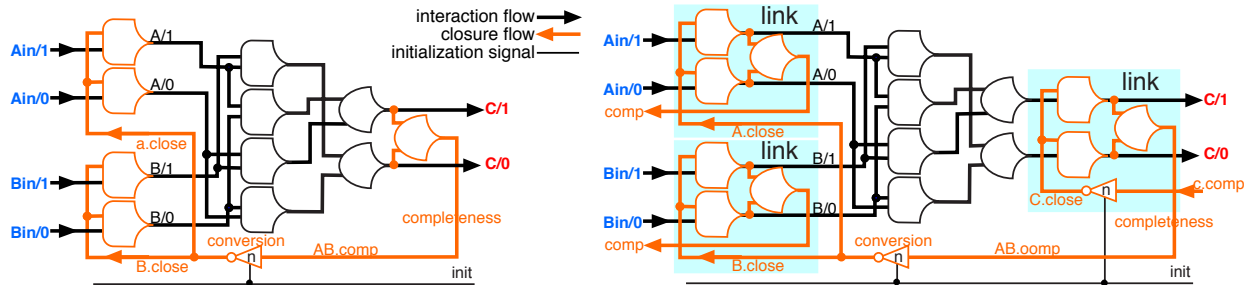
Figure 3.23. Localities A and B are shown as individually closed localities.

Each locality **A** and **B** will determine its own completeness and close with the localities on which it is dependent as shown on the right of Figure 3.23. Locality **C** contributes to the completeness of some locality and will be closed with and enabled by that locality as shown on the right of Figure 3.23. Links appear on the input and output of the oscillation network.

Figure 3.24 illustrates the network at the right of Figure 3.23 in the context of its constant network boundaries shown with dashed lines. The links for the two inputs are now output links in separate networks. Links will be generally associated with network outputs. Closure relations flow beyond the oscillation network expression and necessarily pass through its binding portal closing with links in other networks.
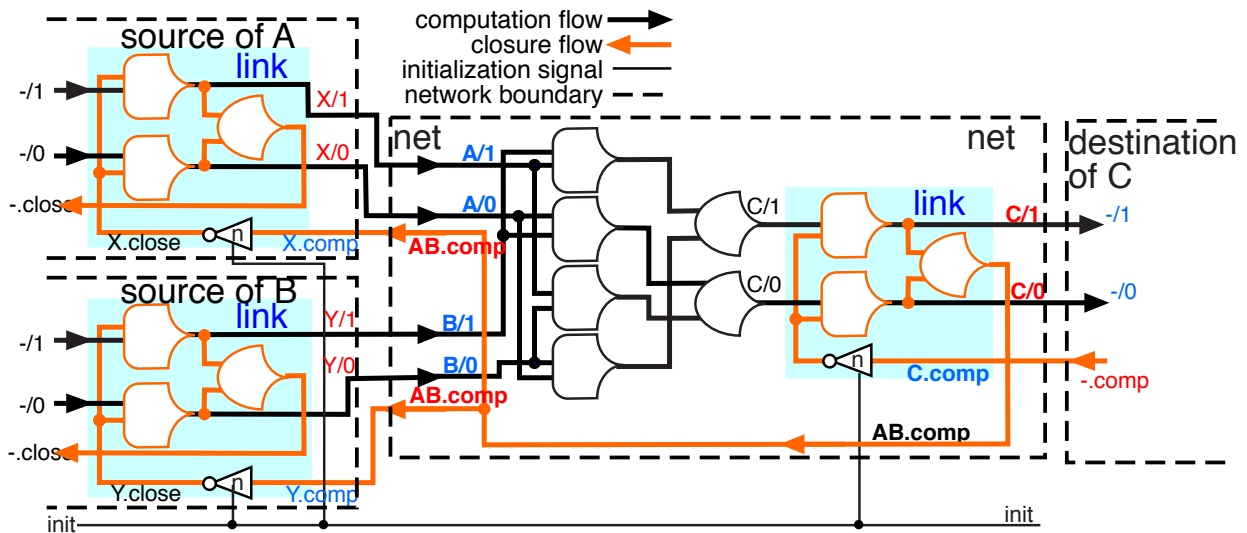
Figure 3.24. Constant network encompassed with explicit closure links.

3.9.4.   Counter flowing networks

Figure 3.24 and Figure 3.21 provide the first glimpses of the counter flowing networks of a pipeline network. A pipeline network composed of oscillation networks forms a structure of

interpenetrating counter-flowing networks. Interaction transition wavefronts flow through the constant networks in the direction of interaction. Closure transitions flow through the closure network counter to the direction of interaction flow. The counter flowing networks intersect through localities with links that coordinate the counter directional flows with common singularly appreciable completeness relations (section 3.9.7).

3.9.4.1. Bubble flow

The closure network transition flow from the output of the pipeline to the pipeline input through the links is referred to as bubble flow. The wavefronts spontaneously flow into bubbles as the bubbles flow in the reverse direction around the wavefronts (Appendix D).

3.9.5. Expressing closure flow and composition

The expression of the network named **net** in Figure 3.24 begins with the expression of the enclosed constant network from Figure 3.11.

The expression of the constant network encompassed by the oscillation network.

bound reference name **binding portal localities**

$$(net\ (A/\{1\ 0\}/\ \ B/\{1\ 0\}/\ =>\ C/\{1\ 0\}/)$$
$$\begin{array}{l} C/\{\ 1<=\{[A/0\ B/1]\ [A/1\ B/0]\} \\ 0<=\{[A/0\ B/0]\ [A/1\ B/1]\}\ \}\ ) \end{array}$$

dependency relations

A link is then added to the output locality **C**. The output **C** is enabled by the conversion of **C.comp** received from outside the network. **AB.close** is generated inside the network from the completeness of **C** and sent through the binding portal to close with the sources of **A** and **B**.

The binding portal must now include the flow of both the constant network and the closure network.

Closure link integrated into the expression of net.

$$(net\ (A/\{1\ 0\}/,AB.comp/\ \ B/\{1\ 0\}/,AB.comp/\ =>\ C/\{1\ 0\}/,C.comp/)$$
$$\begin{array}{l} AB.comp<=?C/[\{\ 1<=\{[A/0\ B/1]\ [A/1\ B/0]\} \\ 0<=\{[A/0\ B/0]\ [A/1\ B/1]\}\ \}\ \sim C.comp]\ \ ) \end{array}$$

AB.close is dependent on the completeness reduction **?** of **C** the completeness of which is dependent on the completenesses of **A** and **B** and the conversion **~** of **C.comp.**

Every interaction locality of the binding portal is paired by a comma with a corresponding closure flowing in the opposite direction: **A/{1 0}/,AB.comp** and **B/{1 0}/,AB.comp** and **C/{1 0}/,C.comp**. Again **blue** is input and **red** is output. The interaction and closure flows through a portal are dependent but are not simultaneous. For instance, in the binding portal **A** is paired with **AB.comp**. In Figure 3.24 an **AB.comp** transition will flow out of the portal and later enable the transition of **X** which will then flow into the portal as a transition of **A**.

3.9.5.1. Half oscillations

There is one complete oscillation network in Figure 3.24 but the expression of the oscillation network occurs over three component constant network expressions none of which contains a complete oscillation network. Each component constant network with a *link* on its output locality forms a *pipeline component network* expressing two *half oscillations*, an input/completeness half oscillation closing with the inputs to the network and an output/enable half oscillation closed by the output of the network.

3.9.6.   Composing half oscillations

Pipeline composition is illustrated by adding links to the output locality of the constant networks **qtob** and **btoq** of section 3.4.9 forming the pipeline component networks of Figure 3.25.
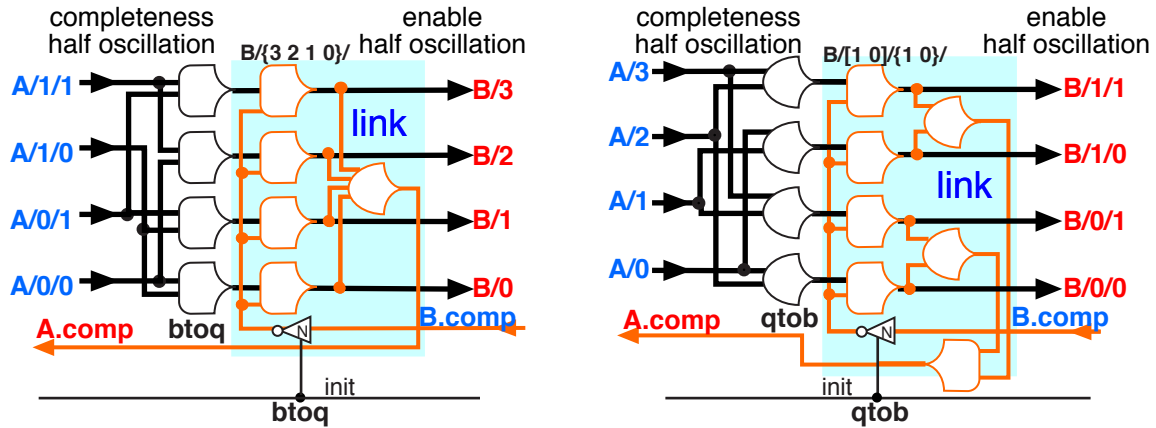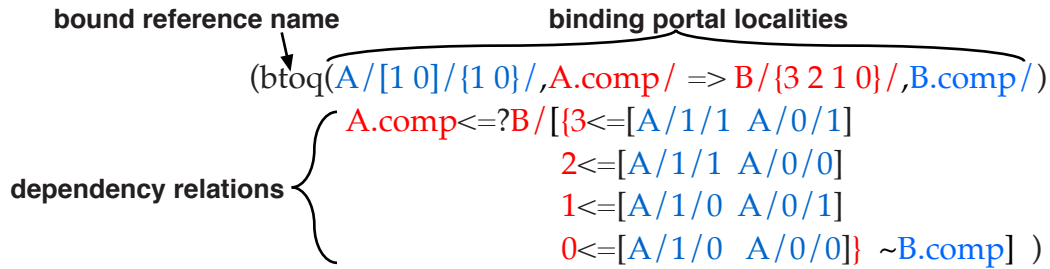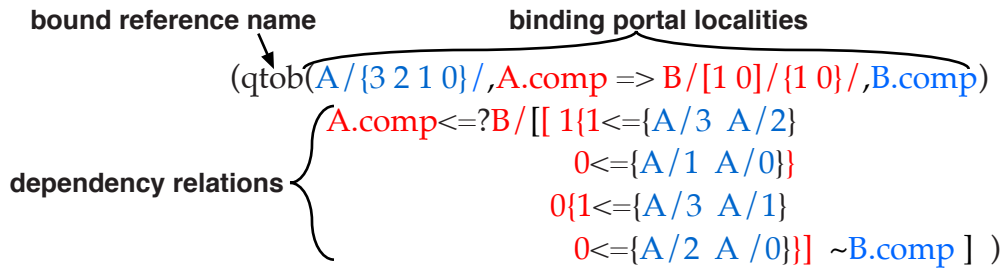

Figure 3.25. Primitive pipeline component networks bounded by half oscillations.

Expression for pipeline component **btoq**.

bound reference name   **binding portal localities**

$$(\text{btoq}(A/[1\ 0]/\{1\ 0\}/,A.\text{comp}/\ =>\ B/\{3\ 2\ 1\ 0\}/,B.\text{comp}/)$$

dependency relations
$$A.\text{comp}<=?B/[\{3<=[A/1/1\ \ A/0/1]$$
$$2<=[A/1/1\ \ A/0/0]$$
$$1<=[A/1/0\ \ A/0/1]$$
$$0<=[A/1/0\ \ A/0/0]\}\ \sim B.\text{comp}]\ )$$

A.comp is dependent on the completeness reduction **?** of **B** the completeness of which is determined by the dependency relations of its components, **{3 2 1 0}** on input locality **A** components and ~**B.comp**. In the binding portal input/presentation locality **A** is paired with closure **A.comp** and output/assertion locality **B** is paired with closure **B.comp**.

Expression for pipeline component **qtob**.

bound reference name   **binding portal localities**

$$(\text{qtob}(A/\{3\ 2\ 1\ 0\}/,A.\text{comp}\ =>\ B/[1\ 0]/\{1\ 0\}/,B.\text{comp})$$

dependency relations
$$A.\text{comp}<=?B/[[\ 1\{1<=\{A/3\ \ A/2\}$$
$$0<=\{A/1\ \ A/0\}\}$$
$$0\{1<=\{A/3\ \ A/1\}$$
$$0<=\{A/2\ \ A/0\}\}]\ \sim B.\text{comp}\ ]\ )$$

A.comp is dependent on the completeness reduction **?** of **B** the completeness of which is determined by the dependency relations of its components, **[1 0]/{1 0}** on input localities **A** components and ~ **B.comp**. In the binding portal input/presentation locality **A** is paired with closure **A.comp** and output/assertion locality **B** is paired with closure **B.comp**.

3.9.6.1.   Associating the half oscillations

The example pipeline network is composed by associating the output/enable half oscillation locality **btoq/B** to the input/completeness half oscillation locality **qtob/A** forming a pipeline network containing one oscillation network and bounded by half oscillations. The constant network is composed as before by associating a binding portal output to a binding portal input (Figure 3.17) but now the composition involves associating the paired closures which simultaneously composes the two counter flowing networks
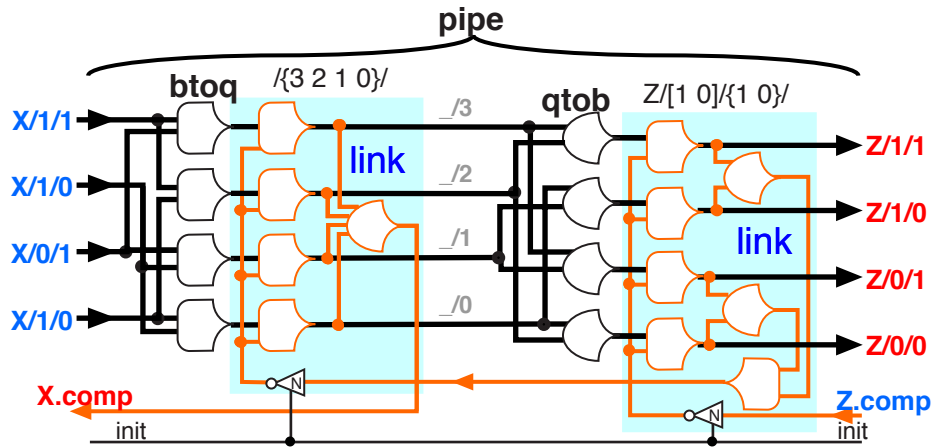
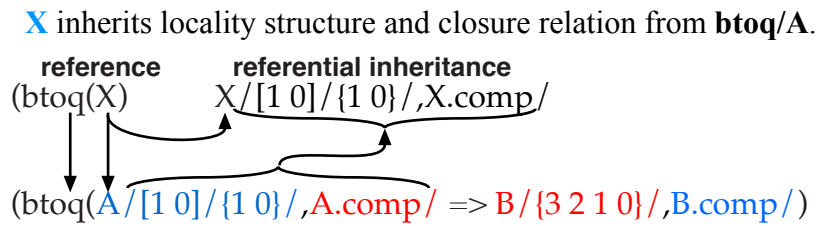**Figure 3.26. linked pipeline segment networks forming a pipeline network.**

3.9.6.2.   Locality inheritance

The expression of composition, Figure 3.26, is simplified by expressing the interaction flow dependency relations in terms of whole locality references and nameless nesting relations.
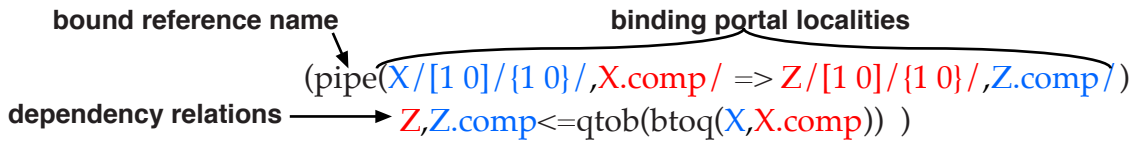
The dependency expression for **pipe**.

bound reference name   **binding portal localities**

$$(\text{pipe}(\overbrace{X => Z})$$

dependency relations ⟶ $Z<=\text{qtob}(\text{btoq}(X))\ )$
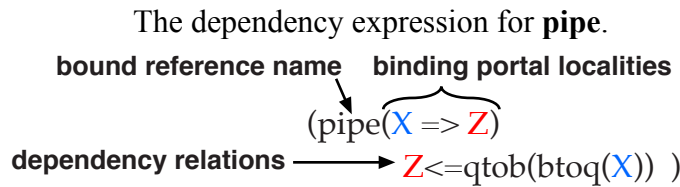
and inheriting closure relations as well as locality structure from the referenced component network expressions (section 3.4.9.3).

Z inherits locality structure and closure relation from **qtob/B**.

reference   **referential inheritance**
$$Z<=\text{qtob}(\ \ )\qquad Z/[1\ 0]/\{1\ 0\}/,Z.\text{comp}/$$

$$(\text{qtob}(A/\{3\ 2\ 1\ 0\}/,A.\text{comp}/ => B/[1\ 0]/\{1\ 0\}/,B.\text{comp}/)$$

X inherits locality structure and closure relation from **btoq/A**.

reference   **referential inheritance**
(btoq(X)   X/[1 0]/{1 0}/,X.comp/

(btoq(A/[1 0]/{1 0}/,A.comp/ => B/{3 2 1 0}/,B.comp/)
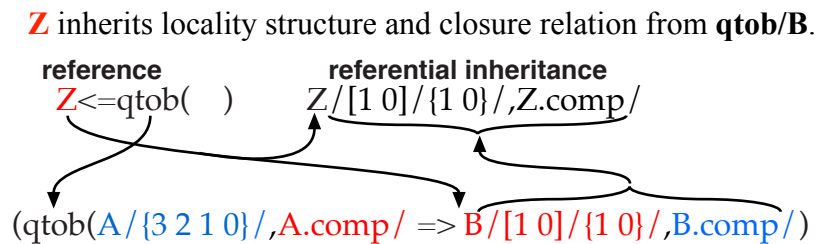
The inheritance expanded expression for **pipe** network of Figure 3.26.

bound reference name   **binding portal localities**

(pipe(X/[1 0]/{1 0}/,X.comp/ => Z/[1 0]/{1 0}/,Z.comp/)
dependency relations ——→ Z,Z.comp<=qtob(btoq(X,X.comp)))

pipe/Z,Z.comp is dependent on **qtob/B,B.close** which is dependent through **btoq** on **pipe/X,X.comp**.

3.9.6.3.   Composing bigger pipelines

Pipeline networks bounded by half oscillations are composed by associating half oscillation binding portal localities to form larger pipeline networks. The expression below connects two **pipe** networks to form **bigpipe** network of Figure 3.27.

The dependency expression for **bigpipe** network of Figure 3.27

bound reference name   **binding portal localities**

(bigpipe(M => N)
dependency relations ——→ N<=(pipe(pipe(M)))
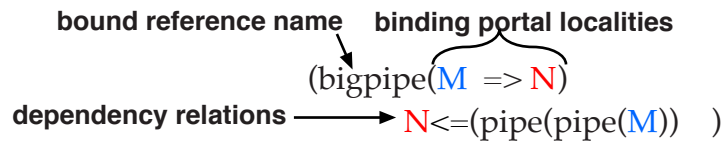
N is dependent on the outer **pipe** which is dependent on the inner **pipe** which is dependent on **M. N** inherits its locality structure and closure flow from **pipe/Z** which inherited its structure and closure flow from **qtob/B. M** inherits its locality structure and closure flow from **pipe/X** which inherited its structure and closure flow from **btoq/A**.

The inheritance expanded expression for **bigpipe** network of Figure 3.27.

bound reference name   **binding portal localities**

(bigpipe(M/[1 0]/{1 0}/,M.comp/ => N/[1 0]/{1 0}/,N.comp/)
dependency relations ——→ N,N.comp<=(pipe(pipe(M,M.comp)))
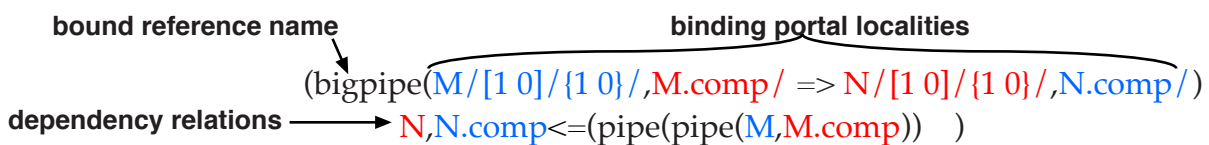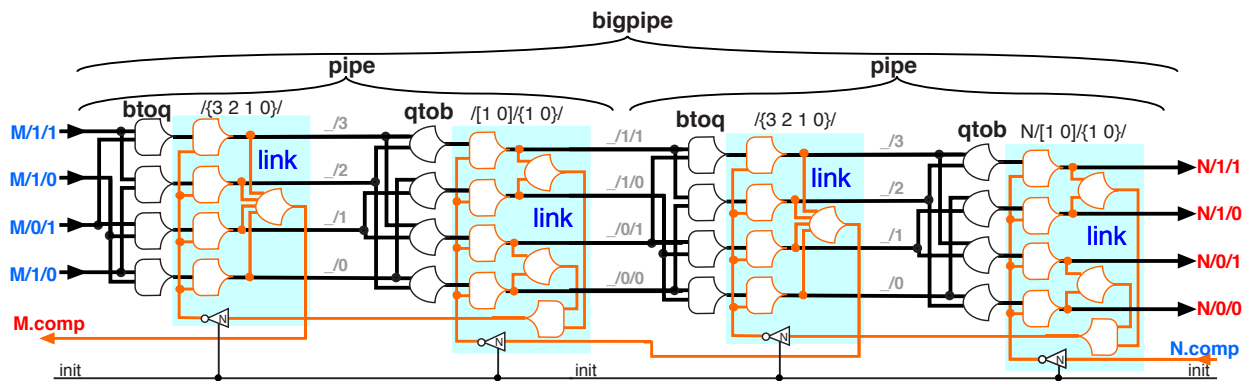
Figure 3.27. Longer pipeline composed of shorter pipelines.

The pipeline **bigpipe** contains three complete oscillation networks and is bounded by half oscillations ready for further composition.

3.9.6.4. Simultaneously flowing instances of interaction

Each transition to **D** completeness presented to an input half oscillation boundary of a pipeline network initiates a wavefront of transition to **D** completeness flowing through the pipeline network. After the **D** wavefront propagates through one or more oscillation networks the input completeness half oscillation boundary can accept a presentation transition to completely **N**. After the transition to completely **N** wavefront propagates through one or more oscillation networks the input completeness half oscillation boundary can accept a next presentation of transition to **D** completeness initiating a next transition to **D** completeness wavefront into the pipeline network. A pipeline network maintains the independence and integrity of multiple transition to **D** completeness wavefronts separated by transition to completely **N** wavefronts simultaneously flowing through different stages of a long pipeline network each representing an isolated and discrete differentness of instance of interaction behavior and instance of interaction time.

The wavefronts remain isolated because a completely **N** wavefront can only flow into a **D** bubble. It cannot flow into a **D** completeness wavefront. Similarly a **D** completeness wavefront can only flow into a **N** bubble. It cannot flow into a completely **N** wavefront. The wavefronts can never overtake each other in a pipeline network (Appendix D).

3.9.6.5. Pipeline network time

Each successive transition to **D** completeness wavefront in a pipeline network is a future in relation to its predecessor transition to **D** completeness wavefronts and a past in relation to its successor transition to **D** completeness wavefronts.

3.9.6.6. The completeness of input criterion

The first transition to **D** completeness wavefront to flow into a pipeline network produces the first transition to **D** completeness wavefront flowing out of the pipeline network the completeness of which which implies the completeness of the first input presentation. The second input transition to **D** completeness wavefront produces the second output transition to **D** completeness wavefront which implies the completeness of the second input presentation and so on. If each oscillation network fulfills the completeness of input criterion then the pipeline network as a whole fulfills the completeness of input criterion.

3.9.7.   Variations of closure structure

The examples in Figure 3.28 illustrates the possible web granularities of counter flowing closure structures for the constant network of Figure 3.18.
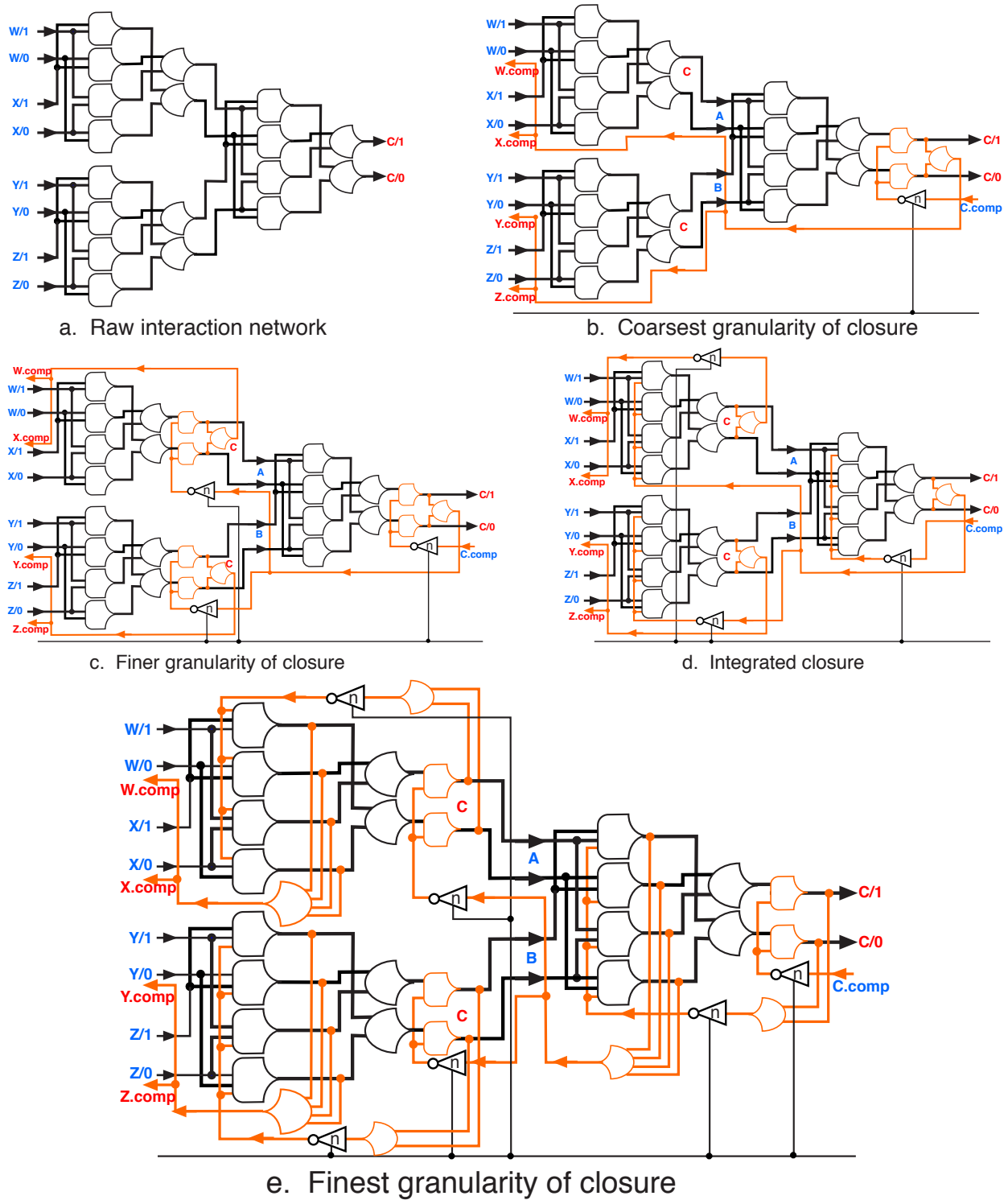

a. Raw interaction network

b. Coarsest granularity of closure

c. Finer granularity of closure

d. Integrated closure

e. Finest granularity of closure

Figure 3.28. The varieties of closure structure for a given constant network.

3.9.7.1.   Figure 3.28a network expression: raw constant network

The raw constant network as described in section 3.4.9.4 can be expressed as a composition of component networks:

$$\textbf{(net(A/\{1\ 0\}/\ \ B/\{1\ 0\}/ => C/\{1\ 0\}/)}$$
$$\textbf{C/\{1<=\{\{[A/0\ \ B/1]\ [A/1\ \ B/0]\}}$$
$$\textbf{0<=\{[A/0\ \ B/0]\ [A/1\ \ B/1]\}\ \}\ \ )}$$

The **bignet** network of Figure 3.28a is expressed as references to **net**.
The dependency expression

$$\textbf{(bigneta(W X Y Z => C)}$$
$$\textbf{C<=net(net(W X) net(Y Z))\ \ )}$$

The inheritance expanded network expression.

$$\textbf{(bigneta(W/\{1\ 0\}/\ X/\{1\ 0\}/\ Y/\{1\ 0\}/\ Z/\{1\ 0\}/ => C/\{1\ 0\}/)}$$
$$\textbf{C<=net(net(W X) net(Y Z))\ \ )}$$

Or the network can be expressed entirely in terms of relations among primitive behaviors with no references and no inheritance. Locality structures have to be explicitly expressed.

$$\textbf{(bigneta(W/\{1\ 0\}/\ \ X/\{1\ 0\}/\ \ Y/\{1\ 0\}/\ \ Z/\{1\ 0\}/\ \ => C/\{1\ 0\}/)}$$
$$\textbf{C/\{1<=\{\ [\ \{[W/0\ X/1]\ [W/1\ X/0]\}\ \{[Y/0\ Z/0]\ [Y/1\ Z/1]\}\ ]}$$
$$\textbf{[\ \{[W/0\ X/0]\ [W/1\ X/1]\}\ \{[Y/0\ Z/1]\ [Y/1\ Z/0]\}\ ]\ \}}$$
$$\textbf{0<=\{\ [\ \{[W/0\ X/1]\ [W/1\ X/0]\}\ \{[Y/0\ Z/1]\ [Y/1\ Z/0]\}\ ]}$$
$$\textbf{[\ \{[W/0\ X/0]\ [W/1\ X/1]\}\ \{[Y/0\ Z/0]\ [Y/1\ Z/1]\}\ ]\ \}\ \}\ \ )}$$

bigneta is a constant network with no closure.

3.9.7.2.   Figure 3.28b network expression: coarsest granularity of closure

There are two approaches to expressing **bignetb** of Figure 3.28b.

1. The link can be syntactically incorporated into the primitive behavior expression of **bigneta**. There are no references to already expressed networks and hence no inheritance so **bignetb** has to explicitly express the closure in detail including the structure of the binding portal.

$$\textbf{(bignetb(W/\{1\ 0\}/,WXYZ.comp/\ \ X/\{1\ 0\}/,WXYZ.comp/\ \ Y/\{1\ 0\}/,WXYZ.comp/\ \ Z/\{1}}$$
$$\textbf{0\}/,WXYZ.comp/ => C/\{1\ 0\}/,C.comp/)}$$
$$\textbf{WXYZ.comp<=?C/[\{1<=\{\ [\ \{[W/0\ X/1]\ [W/1\ X/0]\}\ \{[Y/0\ Z/0]\ [Y/1\ Z/1]\}\ ]}$$
$$\textbf{[\ \{[W/0\ X/0]\ [W/1\ X/1]\}\ \{[Y/0\ Z/1]\ [Y/1\ Z/0]\}\ ]\ \}}$$
$$\textbf{0<=\{\ [\ \{[W/0\ X/1]\ [W/1\ X/0]\}\ \{[Y/0\ Z/1]\ [Y/1\ Z/0]\}\ ]}$$
$$\textbf{[\ \{[W/0\ X/0]\ [W/1\ X/1]\}\ \{[Y/0\ Z/0]\ [Y/1\ Z/1]\}\ ]\ \}\ \}\ \ \sim C.comp\ ]\ )}$$

2. The link can be integrated into the expression of a component network and the component network incorporated by reference. Network **netL** incorporates a closure link to the above network **net**.

(netL(A/{1 0}/,AB.comp/ B/{1 0}/,AB.comp/ => C/{1 0}/,C.comp/)
AB.comp,<=?C/[{1<={{[A/0 B/1] [A/1 B/0]}
0<={[A/0 B/0] [A/1 B/1]} }   ~C.comp])

bignetb references **netL** with a link once and **net** without a link twice.

The dependency expression.

(bignetb(W X Y Z => C)
C<=netL(net(W X) net(Y Z)))

The entire dependency structure including the references to **net** and the binding portal inherits from **netL**.

The inheritance expanded network expression.

(bignetb(W/{1 0}/,W.comp/ X/{1 0}/,X.comp/
Y/{1 0}/,Y.comp/ Z/{1 0}/,Z.comp/ => C/{1 0}/,C.comp/)
C<=[netL(net(W,W.comp X,X.comp) net(Y,Y.comp Z,Z.comp)) ~C.comp])

bignetb forms a pipeline component network of half oscillations.

3.9.7.3.   Figure 3.28c network expression: finer granularity of closure

bignetc of Figure 3.28c composed with three references to **netL**. The expression of **bignetc** inherits its locality structure and closure relations from the references to **netL**.

The dependency expression

(bignetc(W X Y Z => C)
C<=netL(netL(W X) netL(Y Z)))

The inheritance expanded network expression.

(bignetc(W/{1 0}/,W.comp/ X/{1 0}/,X.comp/
Y/{1 0}/,Y.comp/ Z/{1 0}/,Z.comp/ => C/{1 0}/,C.comp/)
C<=[netL(netL(W,W.comp X,X.comp) netL(Y,Y.comp Z,Z.comp)) ~C.comp])

bignetc contains one oscillation network bounded by half oscillations.

3.9.7.4.   Figure 3.28d network expression: integrating the link

The **"all of"** rank of a cross association search can serve as the enable behavior of a link. This can save some primitive behaviors with the tradeoff of increasing the inputs of other primitive behaviors. Whether such a tradeoff is useful depends on the specifics of a network. **bignetd** is composed from component network **net2** that contains the alternatively structured links.

In the expression of **net2** below ~**C.close** is applied to the cross association rank of **"all of"** behaviors that are determining the differentness of locality **C**. The rank of **"all of"** behaviors does double duty as interaction behaviors and as closure enable behaviors.

(net2(A/{1 0}/,AB.comp/ B/{1 0}/,AB.comp/ => C/{1 0}/,C.comp/)
                    (allofrank/{3 2 1 0}/)
                    allofrank/[{3<=[A/1 B/1]
                                    2<=[A/1 B/0]
                                    1<=[A/0 B/1]
                                    0<=[A/0 B/0] } ~C.comp]
                    AB.comp<=?C/{1<={allofrank/1 allofrank/2}
                                        0<={allofrank/0 allofrank/3} })

bignetd composed from three references to **net2**.

The dependency expression

(bignetd(W X Y Z => C)
                        C<=net2(net2(W X) net2(Y Z)))

The inheritance expanded network expression.

(bignetd(W/{1 0}/,W.comp/ X/{1 0}/,X.comp/
                Y/{1 0}/,Y.comp/ Z/{1 0}/,Z.comp/ => C/{1 0}/,C.comp/)
        C<=[net2(net2(W,W.comp X,X.comp) net2(Y,Y.comp Z,Z.comp)) ~C.comp])

bignetd contains one oscillation network bounded by half oscillations.

3.9.7.5.   Figure 3.28e network expression; finest granularity of closure

In Figure 3.28e the completeness and enable behaviors of the link are more finely integrated into the constant network locality by locality forming the longest pipeline version of the network with length of pipeline being characterized by the number of oscillation networks from input presentation to output assertion.

Each component **net3** network contains two links and one complete oscillation network bounded by half oscillations.

(net3(A/{1 0}/,A.comp/ B/{1 0}/,B.comp/ => C/{1 0}/,C.comp/)
(allofrank/{3 2 1 0}/,allofrank.close/)
(A.comp B.comp)<=?allofrank/[{3<=[A/1 B/1]
2<=[A/1 B/0]
1<=[A/0 B/1]
0<=[A/0 B/0] } ~allofrank.close]
allofrank.close<=?[C/{1<={allofrank/1 allofrank/2}
0<={allofrank/0 [allofrank/3} } ~C.comp])

bignete is composed with three references to **net3**:

The dependency expression.

(bignete(W X Y Z => C)
C<=net3(net3(W X) net3(Y Z)))

The inheritance expanded network expression.

(bignete(W/{1 0}/,W.comp/ X/{1 0}/,X.comp/
Y/{1 0}/,Y.comp/ Z/{1 0}/,Z.comp/ => C/{1 0}/,C.comp/)
C<=[net3(net3(W,W.comp X,X.comp) net3(Y,Y.comp Z,Z.comp)) ~C.comp])

bignete contains four oscillation networks two of which are concurrent. The network is bounded by half oscillations.

3.9.7.6.   Webs of singularly appreciable transition completeness
The counter flowing networks form a web of singularity appreciation. In Figure 3.28c the counter flowing networks intersect through singularly appreciable component network outputs. In Figure 3.28e the counter flowing networks intersect through every locality, each an appreciable singularity of transition completeness which is dependent on other localities and has other localities dependent on it, forming a finer granularity of intersection.

3.9.7.7.   Still a constant network
The closure network applied to a constant network does not affect the interaction behavior of the constant network. All five versions of the constant network and counter flowing closure networks in Figure 3.28 deliver the same interaction behavior. The escapement behavior of the component oscillation networks, affects the granularity of instance of interaction time and the throughput performance of the pipeline network.

3.10. INTERLUDE: The pipeline network
A pipeline network is a composition of linked oscillation networks bounded by half oscillations which forms a structure of interpenetrating counter-flowing networks with interaction transition wavefronts flowing through the pipeline network in the direction of interaction and with closure transitions flowing through the closure network counter to the direction of interaction flow. The counter flowing networks intersect through localities with links

that coordinate both counter directional flows with singularly appreciable common transition completeness relations.

The constant network within the pipeline network, regardless of the closure structure encompassing it, remains the determiner of interaction behavior. Its behavior remains constant, always asserting the same output for the same presented input and it continues to fulfill the completeness of input criterion.

3.10.1.   The exposed binding portal and the environment

Even though the component oscillation networks are spontaneously striving to oscillate a pipeline network as a whole remains dependent on the environment beyond its exposed binding portal to present monotonically transitioning input differentnesses, to provide liveness and variability of behavior.

The responsible environment

   The environment still has an imposed requirement for monotonically transitioning the input to the pipeline network. Now an **input completeness half oscillation of the pipeline network determines the appropriate delay between transitions of its input presentation.** The responsibility of the environment is still to honor the closure of the input half oscillation of the pipeline network and if fulfilled the monotonic transitioning requirements for all the component oscillation networks of the pipeline network and their component primitive behaviors are also fulfilled.

From this point on all networks will be pipeline networks with counter flowing interaction and closure networks bounded by half oscillations.

3.10.2.   Pipeline network interaction space

The constant network within the pipeline network remains the interaction space. The encompassment with a closure network does not alter the constant network and its interaction behavior.

3.10.3.   Pipeline network interaction time

The flow of a wavefront of transition to **D** completeness from the input of the pipeline network to its output followed by a wavefront of transition to completely **N** marks one instance of interaction time for the pipeline network as a whole bounded by its binding portal and framed by transitions to completely **N** wavefronts.

There can be multiple wavefronts of transition to **D** completeness each followed by a wavefront of transition to completely **N** simultaneously flowing through stages of a pipeline network (section 3.9.6.4 and Appendix D). The oscillation networks marking instances of interaction time as the wavefronts of transition flow through them all cycle out of phase with each other and there is nothing stable about their phase relations. The internal behavior of a flowing pipeline network is an incoherence of fully coordinated but non-synchronous transitioning. There is no means of sampling an instant of stable analytically meaningful transition behavior across a flowing pipeline network. Only the closure network can appreciate as a singular wholeness the dynamic flow of wavefronts of transition through the dependency relations of the pipeline network.

The only singularly appreciable event differentiating instances of interaction for the pipeline network is the transition of the pipeline network output to **D** completeness followed by its transition to completely **N** marking one instance of interaction for the network as a whole.

3.10.4.   Nothing new.

A pipeline network is still just a network of dependency relations among primitive association behaviors. The pipeline network emerges from the oscillation network which emerges from the completeness of input criterion fulfilling constant network which emerges from the primitive interaction behaviors. Each stage of emergence is just a particular composition of primitive association behaviors defined in section 3.2 which, themselves, emerged from the freely associating persistences and their interacting conditions in the shaking bag of Chapter 2.

3.10.5.   No external metrics

There is no coherent reference frame or external metric of space or of time relative to a pipeline network and trying to impose an external metric onto the pipeline network contributes nothing to either the understanding of or the effective realization of the pipeline network.

3.10.6.   The transcendental view

If one wishes, the closure network of a pipeline network can be removed leaving the raw constant network which can still be characterized as a single step mapping behavior transcending and ignoring its internal structure with its concurrent relations and ignoring the emergent behavior of the pipeline network.

3.11. The autonomous pipeline network: self control

An input/completeness half oscillation, instead of indicating when transitioned input can be accepted, can use its closure to form a completely presented input. A pipeline network can *auto produce* its own input. An output/enable half oscillation instead waiting for downstream closure of the completely formed output wavefront can immediately determine it own completeness of output and close with itself to effectively *auto consume* its own output wavefront,.

Before this point network behavior has been entirely dependent on presented input from an external environment beyond the exposed binding portal of the network for liveness, for wavefront flow. The autonomous pipeline network now transitioning its own input presentation in its own time producing its own wavefronts is autonomously alive, is not dependent on any behavior external to the pipeline network. The pipeline network becomes isolated from the environment behaving independently producing its own wavefronts (liveness) in its own intrinsic time (rate of presentation) in its own intrinsic space (the network) as illustrated in Figure 3.29.
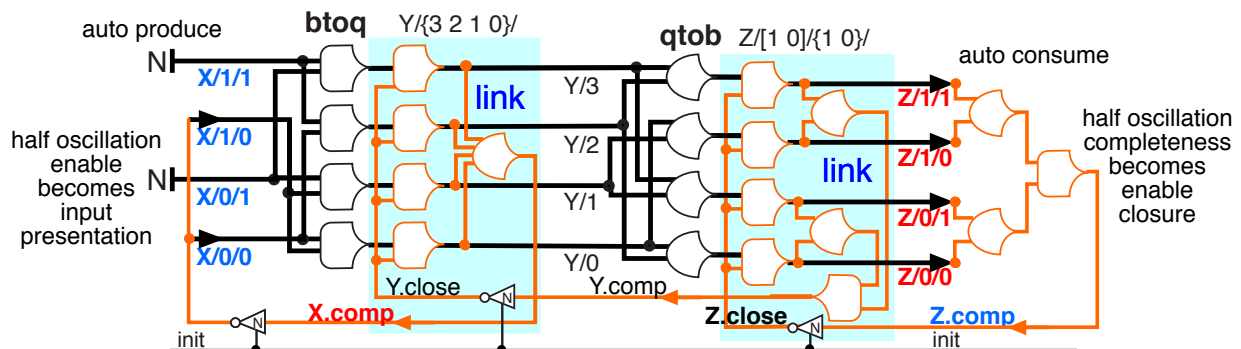

Figure 3.29. Pipeline network with auto produce and auto consume.

The expression of the network, with no activating and guiding reference from outside the network, has no binding portal and no reference name. The network is autonomous, continually alive and entirely on its own.

```
(   (   =>   )
        (Y  X  Z)
        Z<=[qtob(Y)  ~?Z]          /* auto consume */
        Y<=btoq(X/[1{1<=N           /* auto produce  */
                      0<=~?Y }
                   0{1<=N
                      0<=~?Y }] )  )
```

Z is dependent through **qtob** on **Y** and the converted completeness of **Z**. **Y** is dependent through **btoq** on **X**. **X/1/1** is constant **/N**. **X/1/0** is dependent on the converted completeness of **Y**. **X/0/1** is constant **/N**. **X/0/0** is dependent on the converted completeness of **Y**. Localities **X**, **Y** and **Z** inherit their locality structure and closure relations from **btoq** and **qtob**. There are no binding portal localities and there is no reference name. Hence there is no binding locality directional coloring in the expression. The expression cannot be referenced and copied and does not return a referencable result.

The network is continually presented with X/00 and will cycle indefinitely interacting X/00. The network is autonomously behaving and free of the environment but it has lost variability of behavior.

3.12. The embedded pipeline network: the passive environment

An autonomous pipeline network, embedded in an environment, can take full responsibility for its own behavior and still relate to the environment for variability of behavior by sensing the environment at its own rate through a **sampling portal** and imposing output behavior on the environment at its own rate through an **imposition portal**. The environment is still a source of variability of behavior for the network but it is no longer a source of liveness and time for the network. The environment is no longer in control of the network through an exposed binding portal.

The sampling portal and the imposition portal do not form an exposed binding portal. With an exposed binding portal the environment references the network by name and binds external localities to the binding portal localities. The external environment is in control of the referencing and determines the specific bindings. The constant network passively and dependently waits on a presentation to its binding portal from the external environment.

With the sampling portal and imposition portal the embedded network is in complete control of referencing and opportunistically sampling non specific external differentnesses from the relatively continuous flux of the environment and of imposing its output on the environment. The environment is indifferently passive neither waiting on nor dependent on the embedded network.

The network is now in complete control referencing the environment rather than the environment being in complete control referencing the network. The spontaneously behaving differentness of the network is manipulating and appreciating the passive differentness, information, of the external environment.
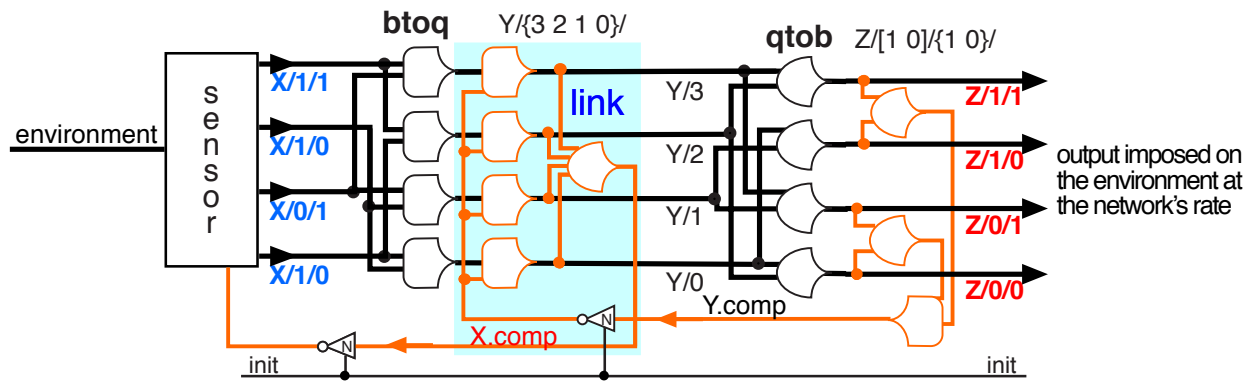
Figure 3.30. A pipeline senses the environment and imposes on the environment.

$$(\quad (\quad => \quad)$$
$$(Y \ \ Z \ \ sensor)$$
$$imposedoutput<=Z<=qtob([Y \ \ \sim?Z])$$
$$Y<=btoq(X<=[sensor \sim?Y] \ ) \ )$$

imposedoutput is dependent on **Z** which is dependent through **qtob** on **Y** and the converted completeness of **Z**. **Y** is dependent through **btoq** on **X** which is dependent on **sensor** and the converted completeness of **Y**.

3.13. INTERLUDE: The autonomous pipeline network

From the pipeline network with its half oscillation boundaries emerges the autonomous network in complete control of its wavefront flow. Closure has finally rendered the constant network independent of its environment. With the oscillation network closure first removed the requirement of the environment to wait an appropriate interval between presentation transitions supplying a time referent to the network. With auto produce and auto consume closure has further removed the requirement of the environment to monotonically transition input presentations supplying liveness to the network.

The constant network is finally in complete control of its own time and liveness on its own behavioral merits. It is not dependent on any extrinsic causative agencies or behaviors such as a mathematician with a pencil or an imposed clock interval or a responsible environment but is its own causative agency continually flowing its own wavefronts. The primitive behaviors passively subordinate to an external environment have collectively contrived to combine their external environments to achieve autonomy and independence.

While the network still relates to an external environment for its variability of behavior the network is in complete control of the relation. Control has migrated from the environment side of a exposed binding portal to the network side of sampling and imposition portals. The exposed binding portal has disappeared. The environment is now passive in relation to a determining network. Quandary 5 of section 3.6 is resolved. The network now expressing its own liveness and time and in complete control of the acquisition of its variability of behavior becomes an active explorer rather than a passive servant

> **The no longer responsible environment**
> The isolated pipeline network now **creates its own monotonically transitioning presentation with the appropriate delay.** There is no longer any responsibility imposed on the environment. The constant network is now in complete control of itself autonomously behaving within an indifferent passive environment.

3.13.1.   A complex persistence

Consider at this point that the passive environment might be a pure condition expression and that the autonomous network might be a complex persistence within it. That the sensor might be sensitive to specific differentness conditions within the environment. That the network enables the sensor and patiently waits for its sensor to encounter a specific condition which initiates a wavefront through the network causing a network behavior that affects the environment. Think proteins in cytoplasm.

3.13.2.   Autonomous pipeline network space and time

The autonomous pipeline network is its own space of interaction. The monotonic transitions between **D** completeness and completely **N** auto produced to the input of the network are the ticks and tocks of its own interaction time. The consequent wavefront of each input flowing through the network is one instance of interaction time delivering its own ticks and tocks to the component oscillation networks within the autonomous pipeline network.

3.13.3.   Still nothing new

An autonomous pipeline network is still just a network of dependency relations among primitive association behaviors (section 3.10.4)

A newly introduced element here is the sensor which is a transducer sensitive to the environment which may or may not be realizable in terms of primitive interaction behaviors.

3.13.4.   No external metrics

There is no coherent reference frame or metric of space or of time relative to an autonomous pipeline network and trying to impose an external metric onto the autonomous pipeline network contributes nothing to either the understanding of or the effective realization of an autonomous pipeline network.

3.13.5.   The transcendental view

The closure structure can still be removed to characterize the encompassed raw constant network as a single step mapping behavior that still always provides the same asserted output for the same presented input but the raw constant network returns to complete dependency on the environment.

3.14. The journey

The last sentence of Chapter 1 declared the pursuit of *an accounting of interaction complete and sufficient in itself with no need of extrinsic support.* The journey began with primitive behaviors made sufficiently expressive and continued to the constant network of dependently related primitive behaviors which remains dependent for its liveness, its referent of time and its source of variability of behavior on presentations of input from an external environment which is in *complete control of the passive network.* The journey continued through the oscillation network, expressing its own referent of time, to the pipeline network with its bounding half oscillations that close on themselves expressing its own source of liveness. The autonomous

network emerged *sufficiently expressive and in complete control of itself* taking from a passively indifferent external environment instead of receiving from a controlling external environment.

3.14.1.   A first principle fulfilled

An autonomous pipeline network is *differentness spontaneously and dependently interacting and changing*(sections 1.8 and 2.5.6).

3.14.2.   A journey not yet complete

The exposed binding portal is gone and the network is in complete control of its own behavior but there remains a dimension of expressivity still to be considered. The common denominator of this chapter is that every example network accepts an input which initiates a wavefront that flows to the output and out of the network performing one isolated instance of interaction in one isolated instance of interaction time. The following wavefront of transition to completely N then erases the instance of interaction from the network isolating successive instances of interaction and allowing the network to be reused with another wavefront. The networks have no behavioral memory and no means of relating different instances of network interaction, different interaction wavefronts, in the context of the network even when the network is in complete control of itself (section 3.4.6.5). The next chapter addresses this expressional inadequacy.

-----

1. David E. Muller, "Asynchronous logics and an application to information processing", Proceedings of Symposium on Applications of Switching Theory in Space Technology, H. Aiken and W. F. Main eds, Stanford University Press 1963. pp. 289-297.
2. Teresa Meng, "Synchronization design for digital systems", (Boston, Kluwer Academic Publishers, 1991) pp 158-163
3. Tom Verhoeff, "Delay Insensitive Codes–An Overview", Distributed Computing (1988) 3, Springer Verlag, pp:1-8
4. DIMS (Delay Insensitive Minterm Synthesis) characterized in terms of C elements ("all of") and OR gates ("one of") and monotonically transitioning dual rail encoding is an asynchronous methodology that closely corresponds to the current discussion. The minterm in DIMS is an exhaustive search that always fulfills the completeness criterion. DIMS was characterized as a mixed logic for practical application that did not quite achieve its efficiency goal. It was not viewed as a coherent model of expression worthy of deeper philosophical consideration and was not pursued as a coherent logic on its own merit. The current discussion is not about efficiency but is about coherency.
   Jens Sparsø, Jørgen Staunstrup, Michael Dantzer-Sørenson, "Design of Delay Insensitive Circuits Using Multi-Ring Structures", Proc. of EURO-DAC'92, Euro- pean Design Automation Conference, 1992, Computer Society Press. pp. 15-20.
5. Alain J. Martin, "The Limitations to Delay-Insensitivity in Asynchronous Circuits", 6th MIT Conference on Advanced Research in VLSI Processes, 1990, pp. 263- 277.
6. Peter A. Beerel, Recep O. Ozdag, Marcos Ferretti, "A Designers Guide to Asynchronous Design", (New York, Cambridge University Press, 2010) p. 1

Chapter 4:
Temporal differentiation

To this point temporal differentiation has manifested as interaction time. One flow of transition to D completeness wavefront into through and out of a constant network followed by a transition to completely N wavefront erasing the interaction from the constant network marked one instance of interaction time in relation to a constant network (sections 3.5.2.2 and 3.4.6.5). A subsequent flow through the same constant network of transition to D completeness wavefront followed by a transition to completely N wavefront marked a different isolated instance of interaction time in relation to the same constant network. Instances of interaction time are differentiated in the context of the network but the constant network with no behavioral memory is not able to account its differentnesses of interaction time. Temporal differentness does not persist to be referenced within the context of the network precluding the constant network from representing a complete and coherent accounting of differentness spontaneously and dependently interacting and changing (section 1.8).

This chapter is about accounting the interaction of temporal differentnesses. Instances of interaction do not flow out of a network but are remembered and accounted flowing within the network which becomes a complete and coherent expression of interaction in contrast to being a fragment of interaction expression (section 3.4.5.5).

4.1.   The ring network: boundless network, endless time

Associating the output/assertion half oscillation of a pipeline network directly to its input/presentation half oscillation forms a ring network. The pipeline network's input and output becomes completely internalized. The network becomes completely isolated. Interacting wavefronts no longer flow out of the network but they also no longer flow into the network. Whatever the network is doing it is completely accounting it.

4.1.1.   The base pipeline

For this example the initial pipeline is the component pipeline networks **qtob** and **btoq**, of Figure 3.25, linked binary to binary with quaternary half oscillation boundaries, Figure 4.1.

The dependency expression.

(baseX=> Z)
     Z<=btoq(qtob(X))   )

The inheritance expanded network expression.

(basepipe(X/{3 2 1 0}/,X.comp/ => Z/{3 2 1 0}/,Z.comp/)
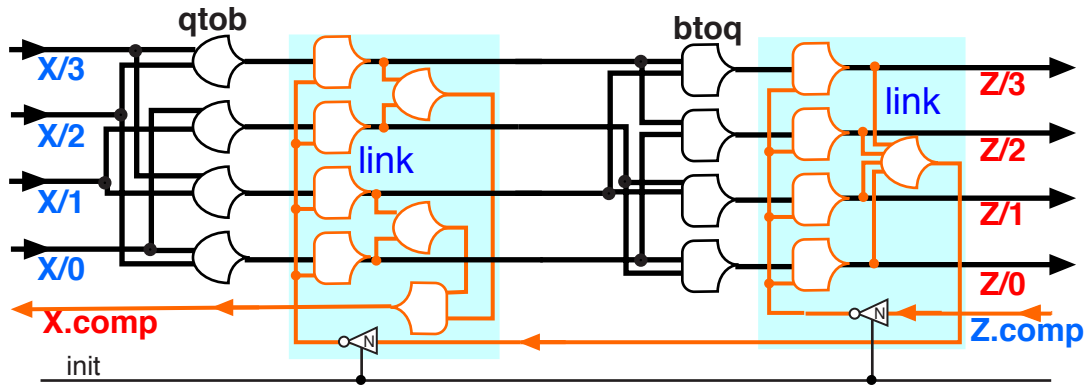   **Z,Z.comp<=btoq(qtob(X,X.comp))   )**

Figure 4.1.   Base pipeline for the ring

4.1.2.   Closing the pipeline

A ring network is a composition of linked oscillation networks which must include at least three oscillation networks to accommodate a transition to **D** completeness wavefront, a transition to completely **N** wavefront and one bubble (Appendix D). If output assertion half oscillation **basepipe/Z** is connected directly to input presentation half oscillation **basepipe/X** the resulting network has only two oscillation networks. So a pipeline component must be inserted between **Z** and **X** to form another oscillation network. The inserted pipeline component performs a shift of quaternary differentness.

$$(Shift(A/\{3\ 2\ 1\ 0\}/,A.comp/ =>$$
$$B/\{3\ 2\ 1\ 0\}/,B.comp/)$$
$$A.comp<=?B/[\{3<=A/2$$
$$2<=A/1$$
$$1<=A/0$$
$$0<=A/3\} \sim B.comp] \ )$$
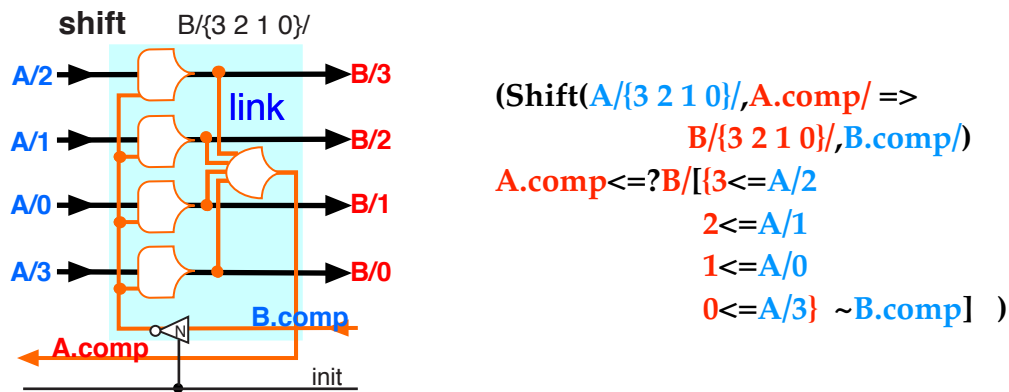
Figure 4.2.   The shift buffer pipeline component.

A.comp is dependent on the completeness of **B** which is dependent on shifted components of **A** which are dependent on ~**B.comp**.

The dependency expression.

$$(\quad (=>)$$
$$(Z:3/\{3\ 2\ 1\ 0\}/)$$
$$Z<= basepipe(shift(Z)\,)\ )$$

Z initialized to **Z/3** is dependent through **basepipe** and through **shift** on **Z**. The dependency of **Z** on **Z** closes the ring. **Z** inherits locality structure and closure relations from **basepipe/Z** and from **shift/A**.

The inheritance expanded network expression.

((=>)
      (Z:3/{3 2 1 0}/,Z.comp/)
      Z,Z.comp<= basepipe(shift(Z,Z.comp))

The expression has no binding portal, no reference name, no boundary localities, nothing flows into or out of the ring, hence there is no binding portal directional coloring in the expression. With no boundary for wavefronts to flow into the ring a **D** completeness wavefront and a completely **N** wavefront must be initialized in the ring. This is expressed with (**Z:3**) specifying that **Z/3** be initialized to **D** (Appendix E).
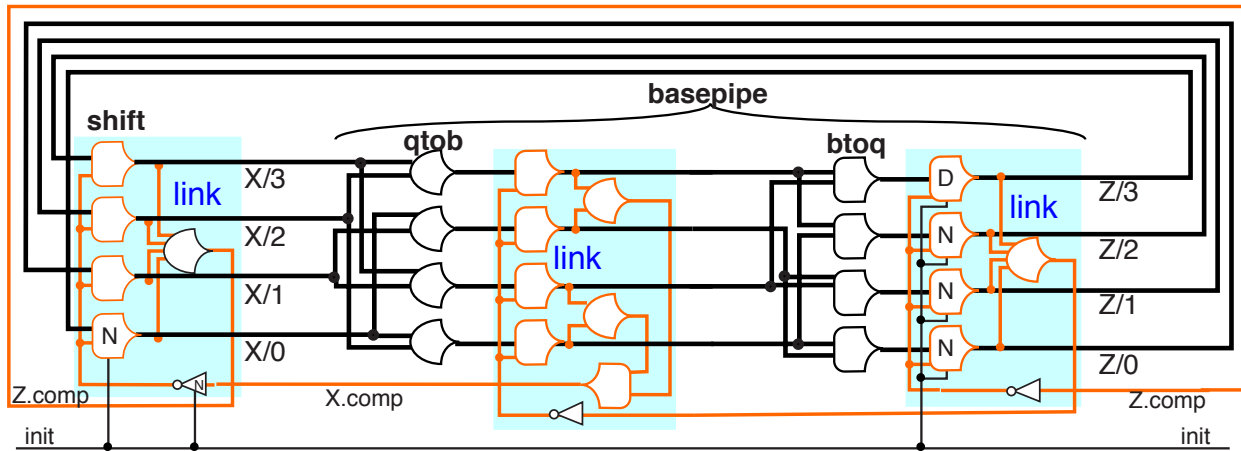
Figure 4.3.   The ring network.

4.2.   INTERLUDE: The ring network

Once instantiated the ring network is a completely self contained network providing its own liveness, time and variability of behavior. The initialized transition to **D** completeness wavefront followed by a transition to completely **N** wavefront autonomously, spontaneously and neverendingly flow around the ring at its own rate in its own space forever or until it fails from internal wear or external insult.

4.2.1.   A new wholeness

The ring network is a wholeness of accounting representing a wholeness of expression as well as a wholeness of network (sections 3.4.5.5 and 3.4.9.8 and 3.5.2.4). The ring network now says everything there is to say about its interaction behavior.

4.2.1.1.   The loss of singular appreciability

The ring network, with no exposed binding portal, no input or output, and with its wavefront continually flowing around the ring network never having flowed into the ring network and never flowing out of the ring network, has no singular referent to characterize a beginning of instance of interaction or instance of interaction time or an end of instance of interaction or instance of interaction time for the ring network as a whole. There is no output assertion

boundary to always assert the same output for the same input presented to the equally nonexistent input presentation boundary.

There is no one transition to completeness that is privileged over any other transition to completeness in the context of the ring network as a whole. The ring network as a whole no longer has a singularly appreciable tick of time.

4.2.1.2.   The incorporated external environment

The ring network with no referent of time itself is now managing the presentation of binding portal input for all of its component oscillation networks and hence determining instance of interaction and instance of interaction time for each component oscillation network. Each instance of interaction wavefront flows out of and is erased from each component oscillation network but each instance of interaction wavefront remains within the ring network flowing to a next interaction. No instance of interaction time wavefront is unaccountably lost from the ring network. The ring network has become an external environment to its component oscillation networks but an external environment fully incorporated into the ring network as a whole. The component oscillation networks previously dependent on the wavefront initiated by the pipeline network input presented from an external environment are now dependent on the eternal wavefront of the ring network.

4.2.1.3.   The master of interaction time

The ring network wavefront flowing around the ring uses, unuses and reuses each component oscillation network over and over. The ring network is managing the interaction time of its component oscillation networks but nothing manages time for the ring network as a whole. The continually flowing ring network wavefront manifests independently flowing, self regulating time within the ring network as a whole. There is no singularly appreciable referent of interaction time for the flowing wavefront but it is in complete control of the flow of interaction time nevertheless.

4.2.2.   Still just a composition of primitive behaviors

A ring network is still just a network of dependency related primitive association behaviors that emerges from the pipeline network (section 3.10.4).

4.2.3.   No metrics

There is no coherent reference frame or metric of space or of time relative to a ring network and trying to impose an external metric onto the ring network contributes nothing to either the understanding of or the effective realization of the ring network.

4.2.4.   The end of the transcendental view: a boundary crossed

The ring network cannot be transcendentally characterized abstractly as a single step mapping behavior apart from its closure structure. Firstly, there is no mapping from a singularly appreciable input to a singularly appreciable output. Secondly, the ring network with its closure structure removed does not work. With the ring network the oscillation network becomes an essential un-transcendable unit of composition.

4.3.   The source ring network: making time

A completely isolated ring network can extend beyond itself by linking the assertion boundary of any component oscillation network of the ring as an output of the ring network. Locality **S** is added and in Figure 4.4 as an assertion output dependent on **Z**.
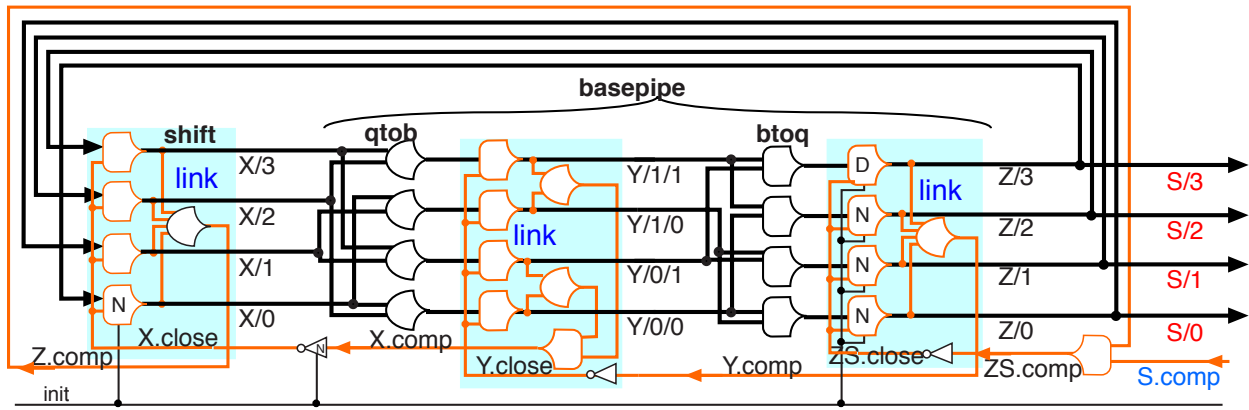
Figure 4.4.   The source ring network.

Since the ring expression cannot be referenced because it has no reference name or binding portal the dependency expression of the ring expression is recapitulated for the source ring. For the source ring there is no presentation locality but there is an assertion locality to be referenced so the source ring expression invites reference with a half binding portal with reference name **source**.

The dependency expression.

(source(=> S)
       (Z:3)
       [Z S]<= basepipe(shift(Z))

S is dependent on **Z** initialized to **Z/3** which is dependent through **basepipe** and through **shift** on **Z**. The dependency of **Z** on **Z** closes the ring. **Z** flows to two destinations **Z** and **S**. **S** and **Z** inherit locality structure and closure relations from **basepipe/Z** and **shift/A**.

The inheritance expanded network expression.

(source(=> S/{3 2 1 0}/,S.comp/)
      (Z:3/{3 2 1 0}/,Z.comp/)
      [Z,Z.comp S,S.comp]<=basepipe(shift(Z,Z.comp))

The source ring can be referenced as:

portal nested	full portal reference	locality nested
(source( ))	**source(=> M)**	**M<=source( )**

through the input half oscillation of a pipeline network.

4.3.1.   Source ring network behavior

The continually flowing wavefront of the source ring network produces successive wavefronts of transition monotonically transitioning between **D** completeness and completely **N** at **S** delivering discrete differentiated instances of interaction time as input to a referencing pipeline network.

The pipeline network is dependent on presentations monotonically transitioning between **D** completeness and completely **N** as a source of liveness, a source of variability of behavior and a source of time from beyond itself honoring its closure. Its beyond, however, is not an uncharacterized external environment but is fully characterized within the source ring network which as a whole is not dependent on anything. As far as the source ring is concerned it is entirely indifferent to the S portal making its continually flowing wavefront occasionally wait.

One might view the ring network "environment" as controlling the pipeline network with input presentations as an external environment does through an exposed binding portal but the ring network, the "environment", is not referencing the receiving pipeline network. The receiving pipeline network is referencing the ring network. The pipeline network initiates and is in complete control of the reference grabbing with its closure an instance of interaction and interaction time from the cornucopia wavefront of the ring network. This is similar to the sampling portal of the network of section 3.12 with its closure grabbing discrete instances of interaction from a passive and continuous external environment.

4.4.   The pipeline ring network: from time to time

A ring network wavefront flow coupled to a pipeline network wavefront flow through at least one shared oscillation network forms a **pipeline ring network** as in Figure 4.6. The example pipeline ring named **pipering** is composed in terms of a pipeline network **pipeline** with **btoq** and **qtob** connected quaternary to quaternary.

The dependency expression

(pipeline(X=> Z)
        Z<=qtoq(btob(X))   )

The inheritance expanded network expression:

(pipeline (X/[1 0]/{1 0}/,X.comp/ => Z/[1 0]/{1 0}/,Z.comp/)
  Z,Z.comp<=qtoq(btob(X,X.comp))   )
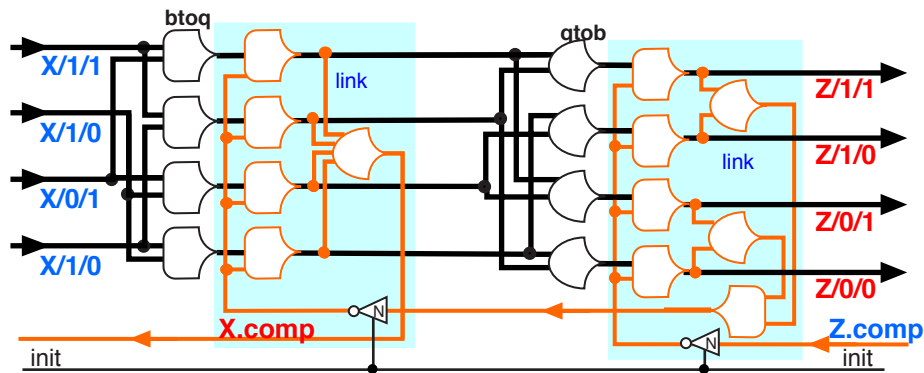
Closing a ring through pipeline requires at least one more oscillation network and the initialization of a **D** completeness wavefront which requires two oscillation networks one initializing the **D** completeness wavefront followed by one initializing a completely **N** wavefront to block the flow of the **D** completeness wavefront during initialization (Appendix E).
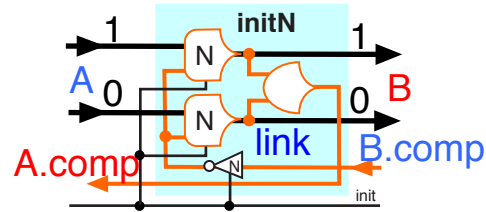
Pipeline component initializing to **1.**   Pipeline component initializing to **N.**
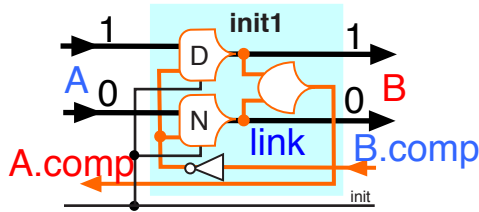
(init1(A/{1 0}/,A.comp/ =>   **(initN(A/{1 0}/, A.comp/ =>**

B:1/{1 0}/,B.comp/)   **B:N/{1 0}/, B.comp/)**

A.comp <=?B<=[A ~ B.comp])   **A.comp <=?B<=[A ~ B.comp])**

The pipeline components are composed into a pipeline segment **pipeseg1** initializing a **D** completeness wavefront to **1** is defined.

Pipeline segment to inialize a **D** completeness wavefront.

(pipeseg1(A/{1 0}/,A.comp/ => B/{1 0}/,B.comp/)

B,B.comp<=initN(init1(A,A.comp)))
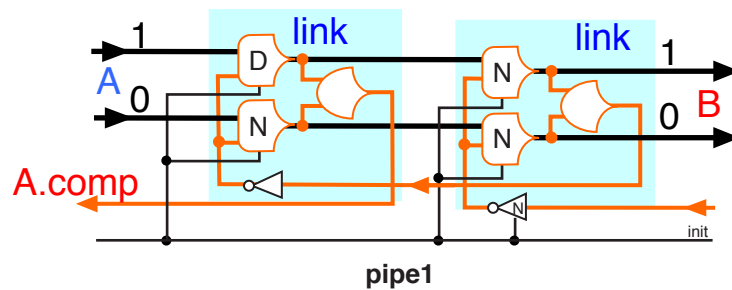
The dependency expression.

(pipering(in => out)

(A)

[A   out]<=pipeline([pipeseg1(A)   in]))

The inheritance expanded network expression.

(pipering(in/{1 0}/,in.comp/ => out/{1 0}/, out.comp/)

(A/{1 0}/,A.comp/)

[A, A.comp   out, out.comp]<=pipeline([pipeseg1(A, A.comp)   in, in.comp]))

To form the pipeline ring the components of the **in** and **out** localities split and associate separately. **in/1** and **out/1** connect to the ring. **in/0** is the network input and **out/0** is the network output. The syntax   [A   **out**]<=pipeline   indicates that the output of **pipeline** associates to **A** and to **out** corresponding to the topmost level of composition of the **pipeline** output locality **Z**.

Locality **Z/1** associates to **A** and **Z/0** associates to **out**. This association relation is illustrated on the left of Figure 4.5.

The syntax   <=pipeline([A   **in**])   indicates that the input of **pipeline/X** associates from **B** and **in** corresponding to the topmost level of composition of the **pipeline** input locality. Locality **B** associates to **X/1** and locality **in** associates to **X/0**. This association relation is illustrated on the right of Figure 4.5.

The **"all of"** relations between the reference and the definition associate then the ordered terms within the **"all of"** relations associate.
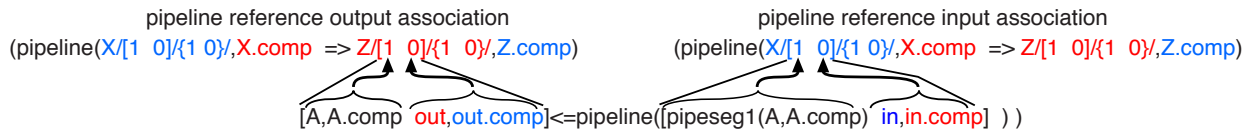
Figure 4.5.   Binding portal associations for pipeline reference.
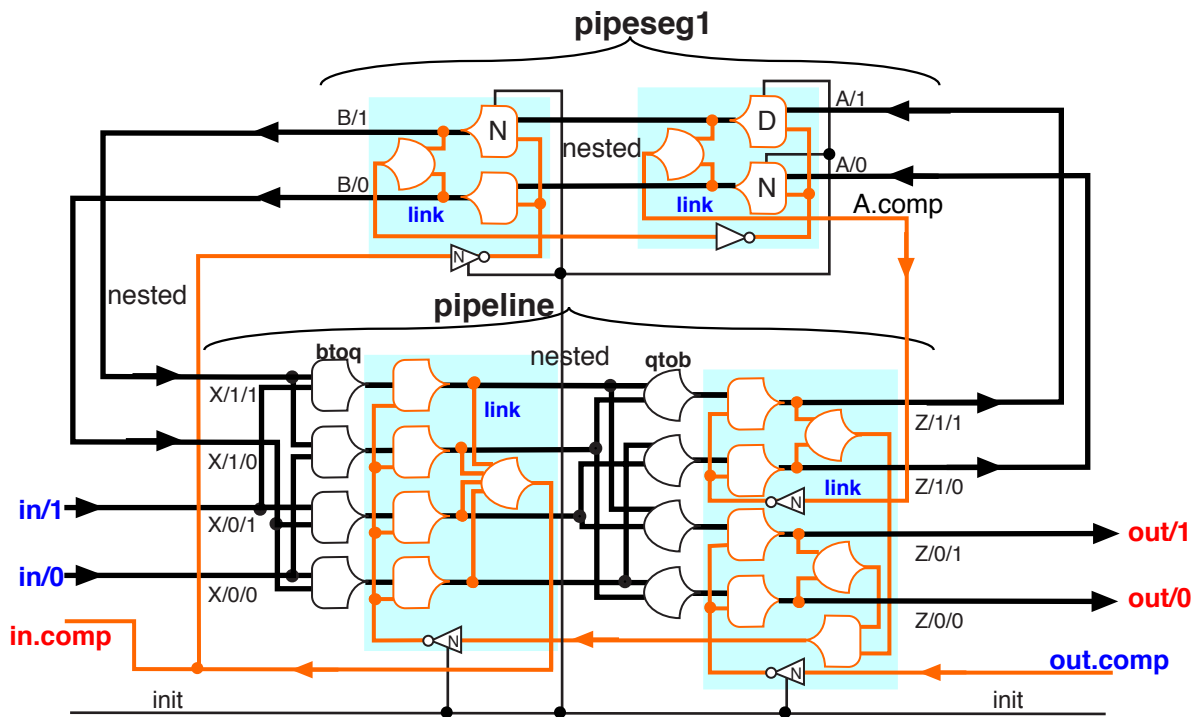
Figure 4.6.   The pipeline ring network.

4.4.1.   Pipeline ring behavior

The pipeline network has an exposed binding portal and just like the pipeline networks of the previous chapter receives presented input and delivers asserted output through its exposed binding portal. Each wavefront begun by the presented input **in** will interact with a wavefront from the ring network as it flows to the output **out** of the pipeline network. There has to be a first **D** completeness - completely **N** wavefront pair initialized in the ring network that will interact with the first transition to **D** completeness - completely **N** wavefront pair presented to the pipeline network. In this case the **D** completeness wavefront is initialized in component network **pipeseg1** and the completely **N** wavefront is initialized in locality **Z/1**. The first wavefront pair entering the pipeline after initialization interacts with the initialized wavefront pair in the ring

network. This first instance of interaction produces the first **D** completeness wavefront to enter the ring network which is a memory of the first pipeline instance of interaction. The following completely **N** wavefront overwrites the **D** completeness wavefront in the pipeline network but the **D** completeness wavefront in the ring is not overwritten and lingers in the siding of the ring. The second **D** completeness wavefront entering the pipeline network interacts with this first pipeline instance of **D** completeness interaction wavefront lingering in the ring network. The third **D** completeness wavefront to enter the pipeline network will interact with the memory of the second pipeline **D** completeness instance of interaction lingering in the ring network. And so on.

The ring network coupled to the pipeline network with its wavefront flowing backward in pipeline space and forward in pipeline time enables a pipeline network to retain an interaction result wavefront and associate it with and interact with a future input wavefront of the same pipeline network. A present instance of interaction wavefront interacts with a past instance of interaction wavefront. The network now has an internal memory establishing an interaction dependency relation across different instances of interaction time. Temporal memory has emerged in relation to the network. Such a relation was not possible in the networks of Chapter 3.

The pipeline ring network is the first example of a network with an exposed binding portal that is not constant. The pipeline ring network as a whole no longer asserts the same completeness of output for the same completeness of presented input.

4.5.   INTERLUDE: A collision of expression regimes

The pipeline ring network represents a collision of expression regimes. The environment beyond the exposed binding portal is not in complete control of the network behavior because of the ring and the ring within the network is not in complete control of the network behavior because of the exposed binding portal. There are two sources of wavefront flow, of liveness and of time. The colliding wavefront sources cooperate and coordinate through a shared oscillation network but cooperation is not an arrow in the quiver of either regime.

A Tth wavefront flows into the pipeline network through the binding portal and a Tth wavefront flows out of the pipeline network. The fact that there is a ring network coupled to the pipeline network associating and interacting different instances of interaction time inside the pipeline network is not visible to the exposed binding portal. The external environment presenting to the exposed binding portal, however, must take into account this invisible influence on the interaction behavior of the pipeline which varies from presentation to presentation.

4.5.1.   The environment expression regime

It is this influence on behavior not visible through the exposed binding portal (side effect) that concerns functional programming which strives to maintain, at all cost, the constancy of the network expression (stay functional) and for the environment presentation through the exposed binding portal to be in complete control of the interaction (referential transparency), i.e. all sources of wavefront flow are presented from the external environment and there are no sources of wavefront flow, no expressions of liveness or of time, from within the network itself such as a memory relating different instances of interaction. The functional view of interaction cannot get out of Chapter 3.

4.5.2.   The network expression regime
The goal of this narrative is to completely characterize interaction in terms of a network of dependency relations among primitive behaviors in complete control of itself, i.e that all sources of wavefront flow are from within the network expression and there are no sources of wavefront flow, no expressions of liveness or of time, presented from outside the network expression.

4.5.3.   The exposed binding portal
The difference between these two expression regimes is the presence of or the absence of a binding portal exposed to an environment unaccounted by the network. If a network possesses a single exposed binding portal then the entire network and all of the internal sources of wavefront flow, the rings, will patiently wait on the presentation from the exposed binding portal, i.e. the external unaccounted environment remains in complete control of the liveness and temporal behavior of the network. With the pipeline ring network the external environment remains in control.

An exposed binding portal is a leak of accountability in the context of the network (section 3.5.2.4). The network cannot encompass a complete accounting of interaction if it remains dependent on an exposed binding portal. The unaccounted environment can be an arbitrarily complex interaction (a human?) just beyond the exposed binding portal. The only way for a network to completely account its interaction is for the network to not have any binding portals exposed to an unaccounted external environment. The network must be in total control of all relations within the network and with any external environment. Assuming a responsible external environment is not an option for the network.

4.5.4.   Still nothing new
The pipeline ring network is still just a network of dependency relations among primitive behaviors.Nothing new has been introduced.

4.5.5.   No metrics
There is no coherent reference frame or metric of space or of time relative to a pipeline ring network as a whole and trying to impose an external metric onto the pipeline ring network contributes nothing to either the understanding of or the effective realization of the pipeline ring network.

4.6.   Removing the exposed binding portal
The exposed biding portal of Figure 4.6 can be removed and its boundary incorporated into the network expression by closing the pipeline into a ring to form a network of coupled rings, by the half oscillations of the pipeline closing on themselves forming an autonomous pipeline network and by linking the pipeline network to ring networks forming a network of coupled and linked rings.

4.6.1.   The coupled ring network
The pipeline component network of the **pipering** network can be closed by associating its output to its input through **pipeseg1** forming a network of two rings coupled through a shared oscillation network as in Figure 4.7. The new ring still has to be at least three oscillation networks and must have an initialized **D** completeness and completely N wavefront pair so the **pipeseg1** pipeline component network is needed again.

((=>) (out:0/{1 0}/,out.comp/)
  out,out.comp<=pipering(pipeseg1(out,out.comp)))

pipeseg1

pipeseg1
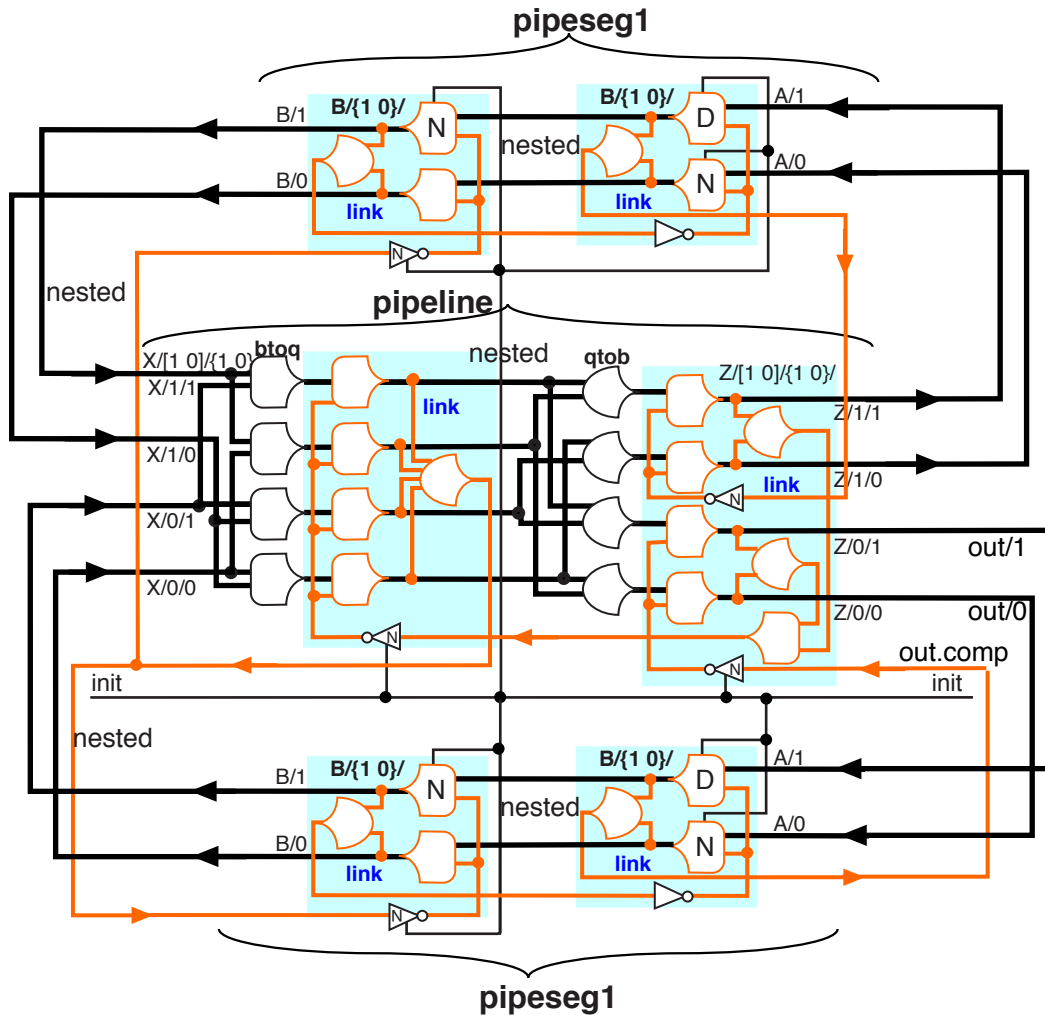
Figure 4.7.   The network of two coupled rings.

Each ring is an internal source of wavefronts and of time which times are coordinated through a shared oscillation network. Neither ring cares that its wavefront might be delayed by the other ring. Each oscillation network including the shared oscillation network mark an instance of interaction time as the ring wavefronts flow through them. But the interaction network as a whole does not mark instances of interaction time. The two wavefronts form one single never ending instance of interaction time.

4.6.2.   The autonomous pipeline network

In Figure 4.8 the exposed binding portal half oscillations of the component pipeline network close on themselves forming an autonomous pipeline network. The pipeline ring network becomes self determined like the network of Figure 3.30.

((=>)
(sensor/{1 0}/,sensor.close   out/{1 0}/,out.close)
        out.close<=?out<=pipering(sensor,sensor.close))
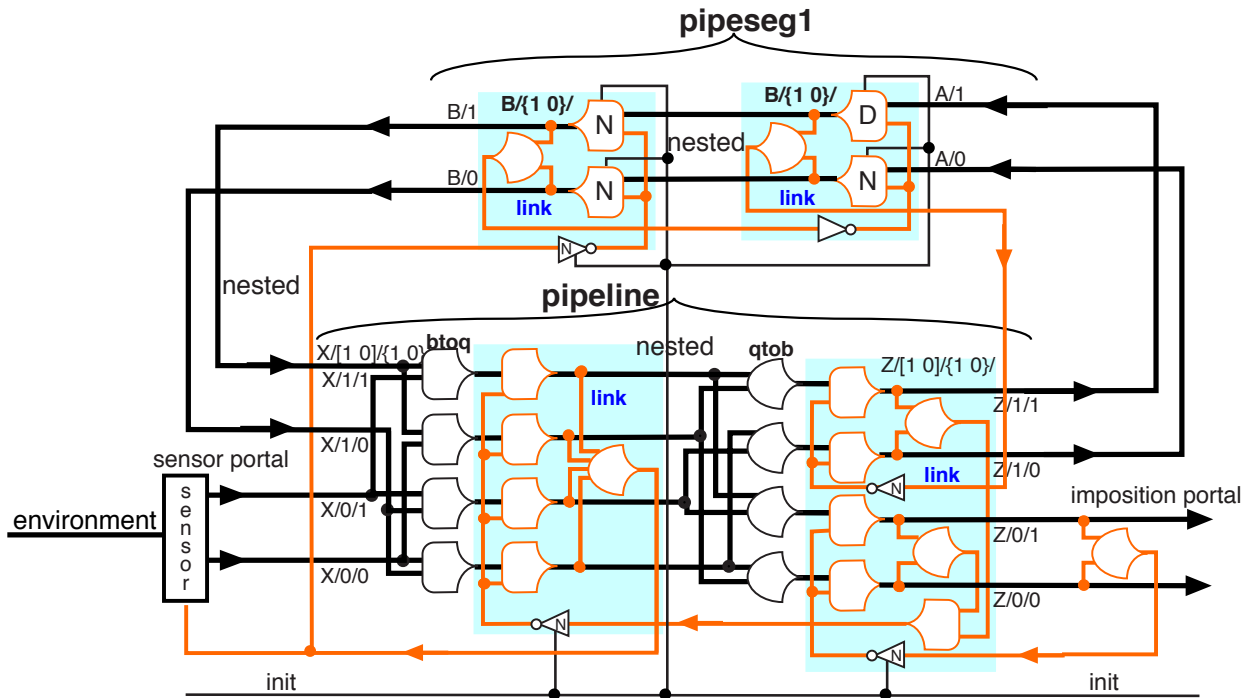
Figure 4.8.   The pipeline network of pipering with auto produce and auto consume.

## 4.6.3.   The network of coupled and linked rings

The binding portal half oscillations of the component pipeline network are linked to rings in Figure 4.9 which supply the input and receive the output of the pipeline. One more pipeline component **buff** that does not initialize is needed to form the three oscillation ring networks.

(buff(A/{1 0}/,A.comp/ => B/{1 0}/,B.comp/)
        A.comp<=?B<=[A ~Bcomp])
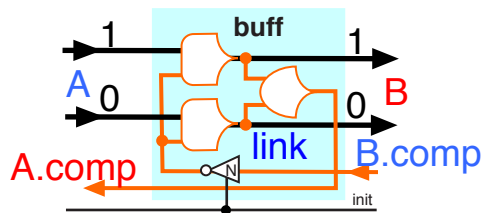
Network **sourcering** is expressed using **buff** and **init1** and **initN** from section 4.4.

> **(sourcering(=> out/{1 0}/,out.comp)**
>   **(Z/{1 0}/,Z.close)**
> **(out,out.comp Z,Z.close)<=initN(init1(buff(Z,Z.close))))**
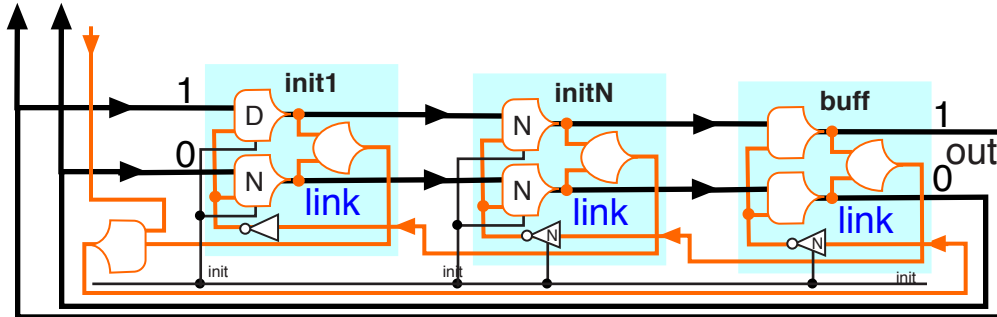
Network **sinkring** is a single rail ring that sinks the output wavefront.

> **(sinkring(in/{1 0},in.comp/ =>   )**
>   **(A/ B/:D C/:N D/)**
>   **A<=?[in   D]   [in.comp B]<=[A ~C]**
>   **C<=[B ~D]   D<=[C ~B]**
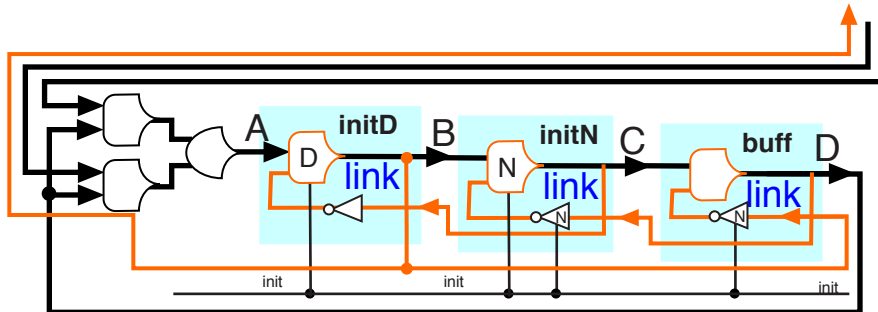
The pipeline ring network with input and output ring caps.

$$(\quad (\quad => \quad)$$
$$\textbf{sinkring(pipering(soucering( ))))}$$
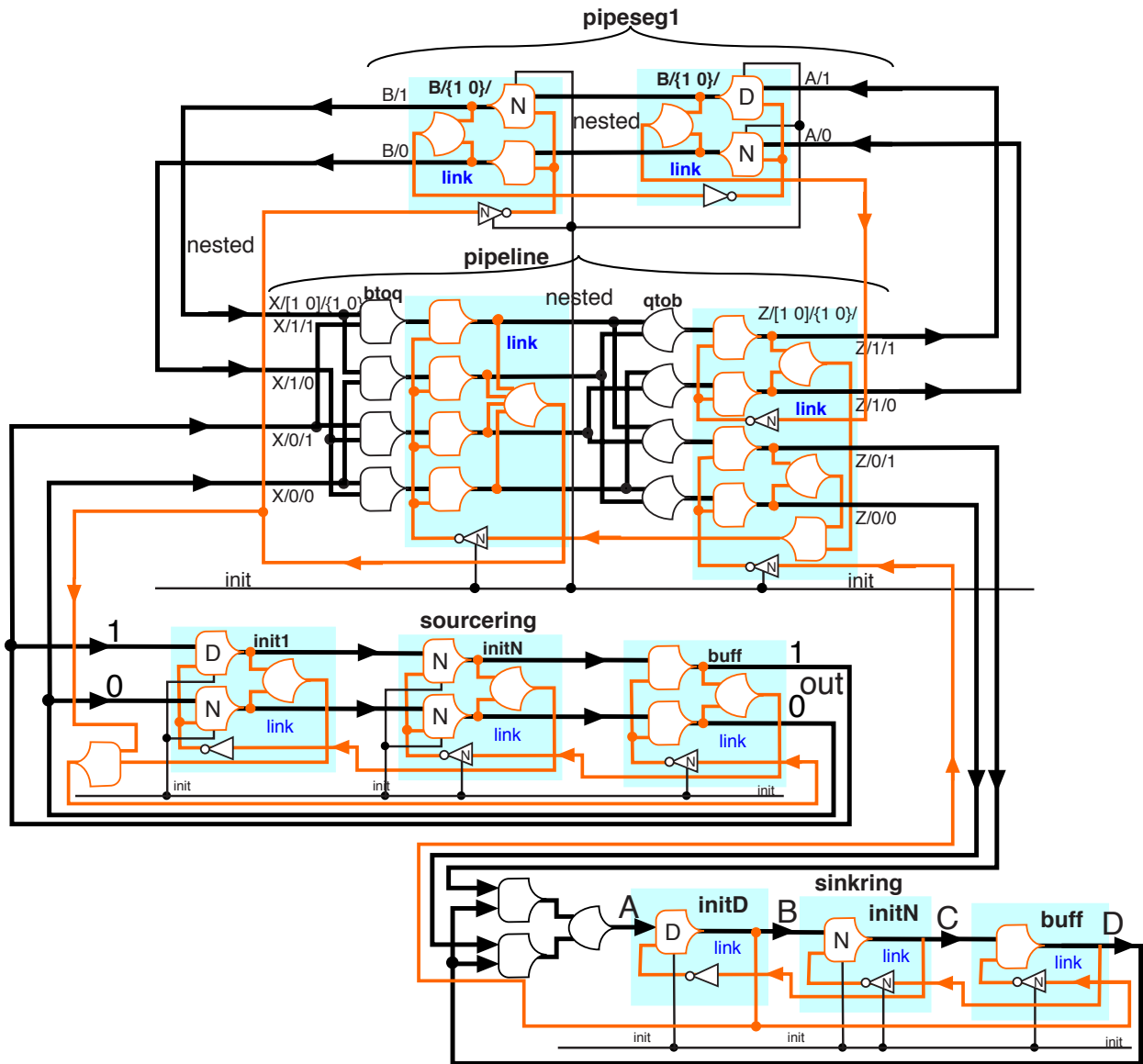
Figure 4.9.   The pipeline network of pipering linked to rings.

The component pipeline network is coupled to a ring through a shared oscillation network and is linked to an input ring and an output ring. All sources of wavefront flow are within the network.

4.7.   INTERLUDE: The self determined network

Rings coupled through shared oscillation networks, rings linked through shared pipelines and imposition portals and sampling portals support indefinitely complex self determined network expressions. In the network of Figure 4.9 there are three different wavefronts in three different rings flowing and coordinating their flow in relation to each other. Consider the possibility of a

trillion mutually coordinating and interacting wavefronts flowing in a network of a trillion linked and coupled rings.

4.7.1.   Nothing in control

With no exposed binding portal there is no singular authority that determines liveness, time and wholeness for the self determined network (sections 3.5.2.4 and 3.4.5.5). There is only the network itself and its intrinsic wavefronts. A self determined network is egalitarian (section 4.2.1). No boundary is more authoritative than any other boundary. Everything is equally subordinate to everything else. Everything is equally dependent on everything else. Everything is equally determinative. There is nothing referentially central. There is nothing referentially global. There is no extrinsic authority. There is no intrinsic authority. There is only wholeness of network and wholeness of expression. There is nothing that references a self determined network. A self determined network does not report to anything else. A self determined network is a complete accounting of interaction within itself free to move through and relate to an external environment and to interact with other self determined networks.

4.7.2.   No appreciable singularity of network behavior

With no exposed binding portal there is no singularly appreciable referent, no tick, of instance of interaction or instance of interaction time in relation to the self determined network as a whole. There is only the incoherent transitioning cacophony of its component oscillation networks (sections 3.5.2.3 and 3.10.3). The only agency capable of effectively appreciating the cacophony of interaction behavior within a self determined network is the counter flowing closure network (section 3.9.7 and 3.10.3) which is also an integral part of the self determined network managing the cacophony of inchoherent behavior in terms of the coherency of dependency completeness relations.

4.7.3.   Dimensions of differentiation

Differentness of condition (Chapter 2) and differentness of association (Chapter 3), interpenetrating, collaborating and mutually extending, can be considered the first two dimensions of differentiation and its interaction (Chapter 5). The differentness of wavefronts of change flowing through the first two dimensions forms a third dimension of temporal differentness of interaction (Chapter 4) which is not persistently referencable as are the first two dimensions.

4.7.4.   The temporal dimension of differentiation

While flowing wavefronts of transition use, unuse and reuse the differentness of each component oscillation network over and over extending their expression of differentness through interaction time there is nothing that similarly reuses and extends the differentness of interaction time over and over. Differentness of interaction time can never be reused because each instance of differentnesses of interaction time is irretrievably erased, irreferencibly unused, by the following transition to completely **N** wavefront.

Interaction time is ephemerally self limiting but it is also indefinitely self extending (sections 3.4.6.5 and 3.5.2 and 3.5.2.1). Transition to **D** completeness wavefronts interact producing subsequent transition to **D** completeness wavefronts which interact producing subsequent transition to **D** completeness wavefronts and so on. **D** completeness wavefronts flowing progressively and indefinitely represent the flow of time for a self determined network. A **D** completeness wavefront trails behind itself **D** completeness bubbles (section 3.9.6.4 and Appendix D). It is the **D** completeness bubbles that the following wavefront of transition to

completely **N** wavefront erases. A transition to completely **N** wavefront cannot overtake and erase a transition to **D** completeness wavefront. Similarly a transition to **D** completeness wavefront can only flow into completely **N** bubbles and cannot overtake and compromise a completely **N** wavefront.

In a self determined network **D** completeness wavefronts form narrow bands of continually progressing transition behavior followed by the transition to completely **N** wavefronts. Each ring network supports one bandlette of **D** completeness wavefront. These narrow bands of **D** completeness transition behavior form the dimension of temporal differentiation within which differentnesses of interaction time can associate and interact. Interaction time is the last but indefinitely extending dimension of differentiation and interaction.

4.8.   The LFSR network: interacting differentnesses of time

The LFSR (Linear Feedback Shift Register) network of Figure 4.10 is a network of 5 rings with 6 independently flowing wavefronts coupled through a structure of shared oscillation networks. Each wavefront represents a different instance of interaction time with 5 instances interacting through a shared pipeline network.

The LFSR network is constructed with pipeline component networks **pipe1** which initializes a wavefront to **1**, **pipe0** which initializes a wavefront to **0**, **pipeN** which initializes a wavefront to completely **N** and a pipeline of **XOR** networks.
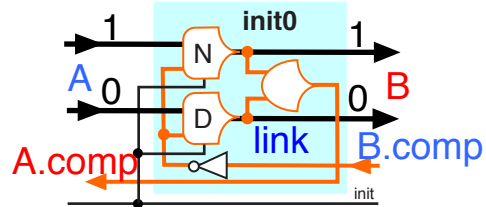
4.8.1.   Pipeline component networks for LFSR

Pipeline component initializing to **1**

(init1(A/{1 0}/,A.comp/ => B:1/{1 0}/,B.comp/)
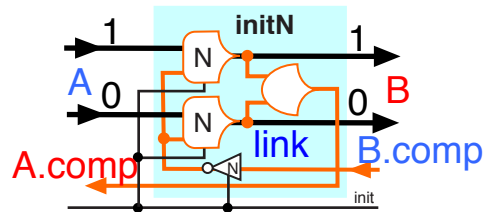   **A.comp <=?B<=[A ~ B.comp])**
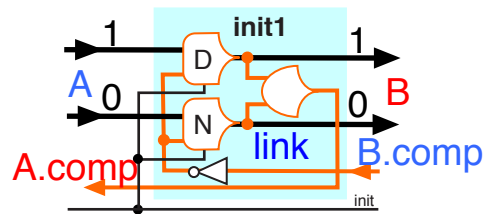
Pipeline component initializing to **0**

(init0(A/{1 0}/, A.comp/ => B:0/{1 0}/, B.comp/)
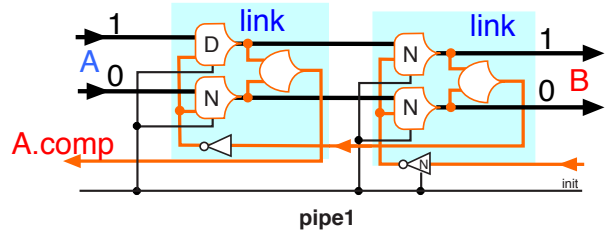   **A.comp <=?B<=[A ~ B.comp])**

Pipeline component initializing to **N**

(initN(A/{1 0}/, A.comp/ => B:N/{1 0}/, B.comp/)
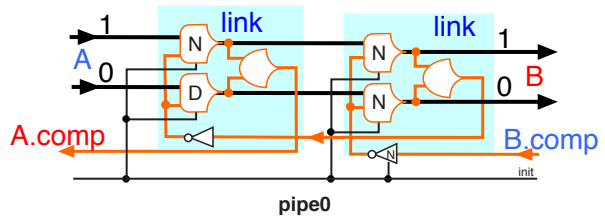   **A.comp <=?B<=[A ~ B.comp])**

4.8.2.   Initializing pipeline segment networks for LFSR

pipe1 pipeline segment

(pipe1(A/{1 0}/, A.comp => B/{1 0}/, B.comp)
  B, B.comp <=initN(init1(A.comp)))
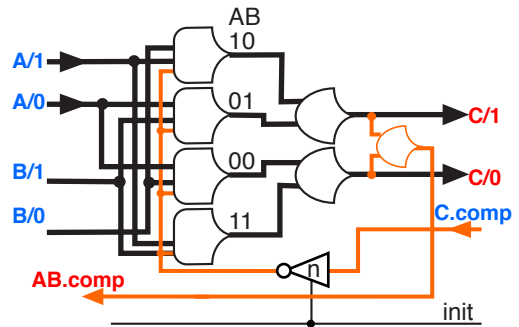
pipe0 pipeline segment

(pipe0(A/{1 0}/, A.comp => B/{1 0}/, B.comp)
  B, B.comp <=initN(init0(A, A.comp)))

4.8.3.   XOR pipeline component network for LFSR

(XOR(A/{1 0}/, AB.comp B/{1 0}/,AB.comp =>
                    C/{1 0}/,C.comp)

AB.comp <=?C/[{1<={[A/0 B/1] [A/1 B/0]}
            0<={[A/0 B/0] [A/1 /B/1]} }
                  ~ C.comp])

4.8.4.   The isolated LFSR network

The dependency expression

((=>)   ((B C D E F O)/{1 0}/,*.close)
      F<=pipe1(E<=pipe0(D<=pipe1(C<=pipe0(B<=pipe0(pipe0(O
          <=XOR(F XOR(XOR(B C) XOR(D E))))))))))

How to represent the two closures?
    F is dependent through **pipe1** on **E** which is dependent through **pipe0** on **D** which is
dependent through **pipe1** on **C** which is dependent through **pipe0** on **B** which is dependent
through **pipe0** and through **pipe0** on **O** which is dependent through **XOR** on **F** and **XOR** which
is dependent through **XOR** on **B** and **C**, and is dependent through **XOR** on **D** and **E**. The
dependence of **O** on **F E D C** and **B** and the dependence of each of them on **O** closes each ring
through the **XOR** network.
    Localities **B**, **C**, **D** and **E** each associates to two places which flow cannot be represented
with syntax relation but must be represented with name correspondence.

The inheritance expanded network expression.

((=>)   (B/{1 0}/,B.comp C/{1 0}/,C.comp D/{1 0}/,D.comp
E/{1 0}/,E.comp F/{1 0}/,F.comp O/{1 0}/,O.comp)
F,F.comp<=pipe1(E,E.comp
<=pipe0(D,D.comp
<=pipe1(C,C.comp
<=pipe0(B,B.comp
<=pipe0(pipe0(O,O.comp
<=XOR(F,F.comp
XOR(XOR(B,B.comp C,C.comp)
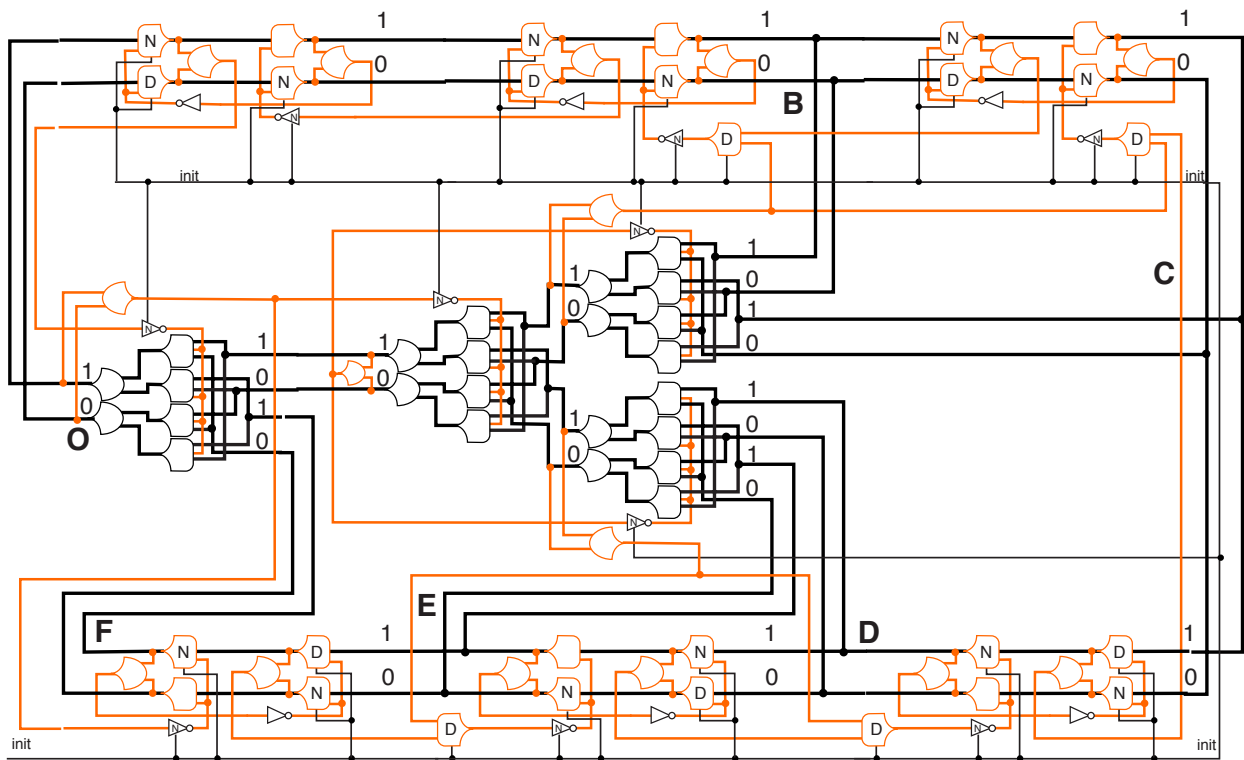**XOR(D,D.comp E,E.comp))))))))))**

Figure 4.10. LFSR example network of multiple coupled rings

Figure 4.11 illustrates the **LFSR** network with 6 different wavefronts continually flowing around the 5 different rings of the network representing 6 differentnesses of time 5 of which interact through the shared **XOR** pipeline.

The 5 rings are color coded in Figure 4.11 to highlight their sharing of the peripheral oscillation networks.
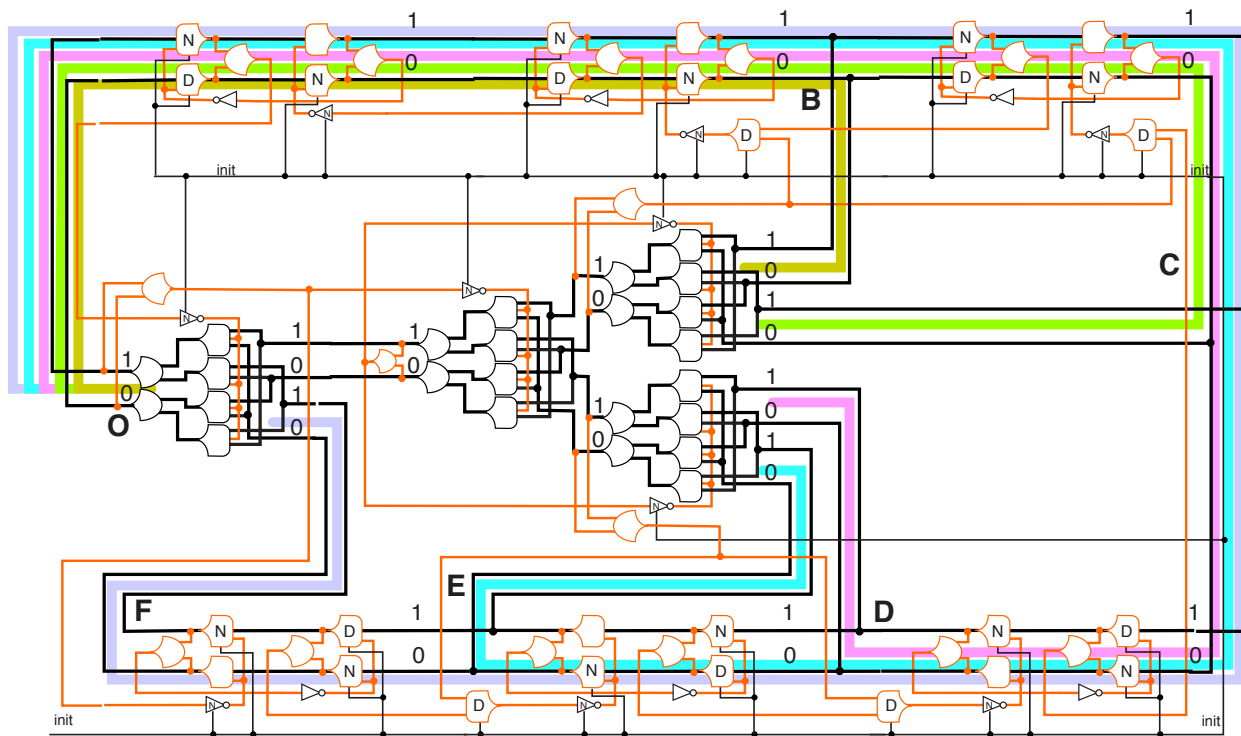

Figure 4.11. Coupled rings and their shared pipeline components.

4.8.5.   The LFSR source network

Locality **O** of the **LFSR** can link to a pipeline network making it a source ring network. An assertion locality **out** dependent on locality **O** is added to the **LFSR** expression as a referenceable output.

The network expression.

(LFSR(=> out/{1 0}/,out.comp/)   (B/{1 0}/,B.comp/ C/{1 0}/,C.comp/
D/{1 0}/,D.comp/ E/{1 0}/,E.comp/ F/{1 0}/,F.comp/ O/{1 0}/,O.comp/)
F<=pipe1(E<=pipe0(D<=pipe1(C<=pipe0(B<=pipe0(pipe0(O
<=[XOR(F XOR(XOR(B C) XOR.(D E))) ~out.comp])))))))
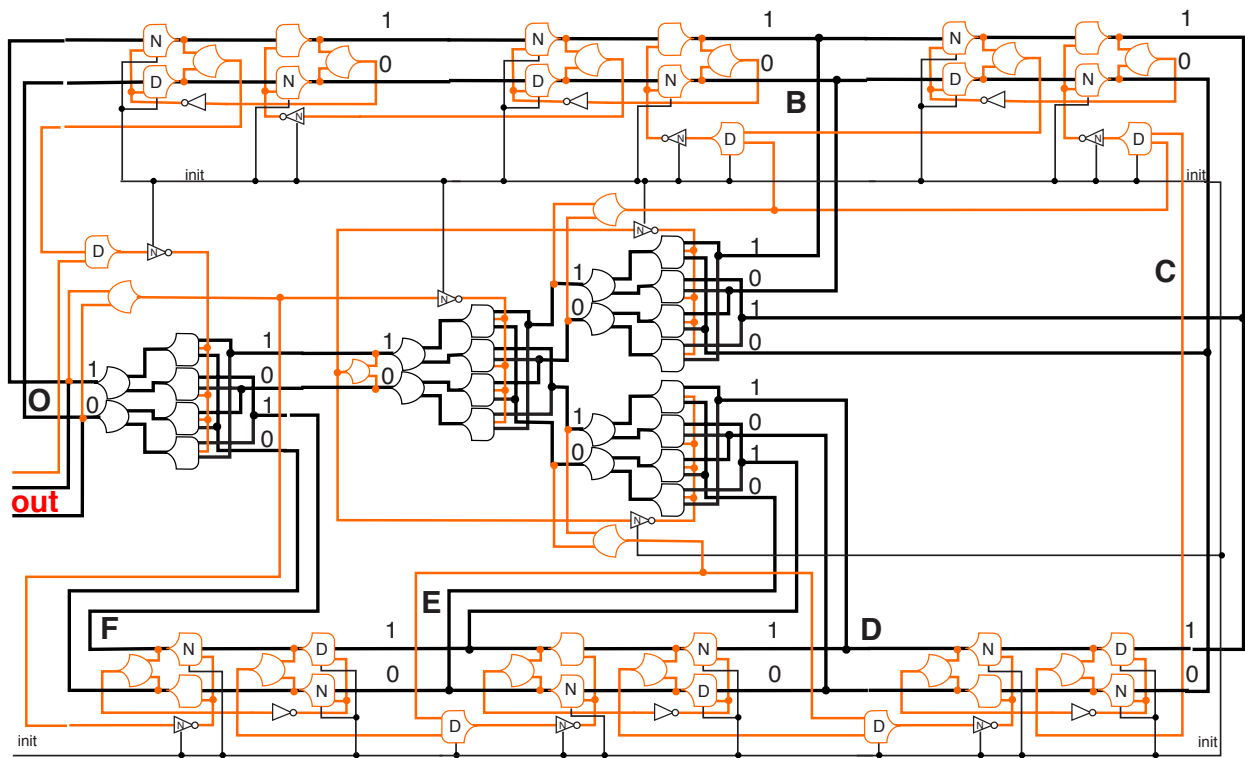**out<=O)**

Figure 4.12. LFSR source network.

The LFSR is referenced as

((LFSR( )   ))   or as   dest<= LFSR( )   or as   LFSR(=> dest)

## 4.9.   The immersed ring network: engaging the environment

The immersed ring network of Figure 4.13 is a network of two coupled rings with an imposition portal to influence the environment and a sampling portal sensitive to the effects of the imposition on the environment forming an interaction network capable of remembering and learning.

4.9.1.   The pipeline component networks for the immersed ring network

(init2(A/{2 1 0}/,A.comp/ =>
B:2/{2 1 0}/,B.comp/)
A.comp <=?B<=[A ~B.comp])
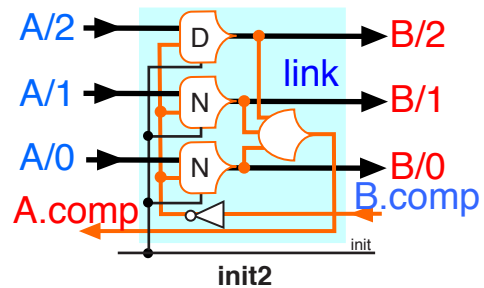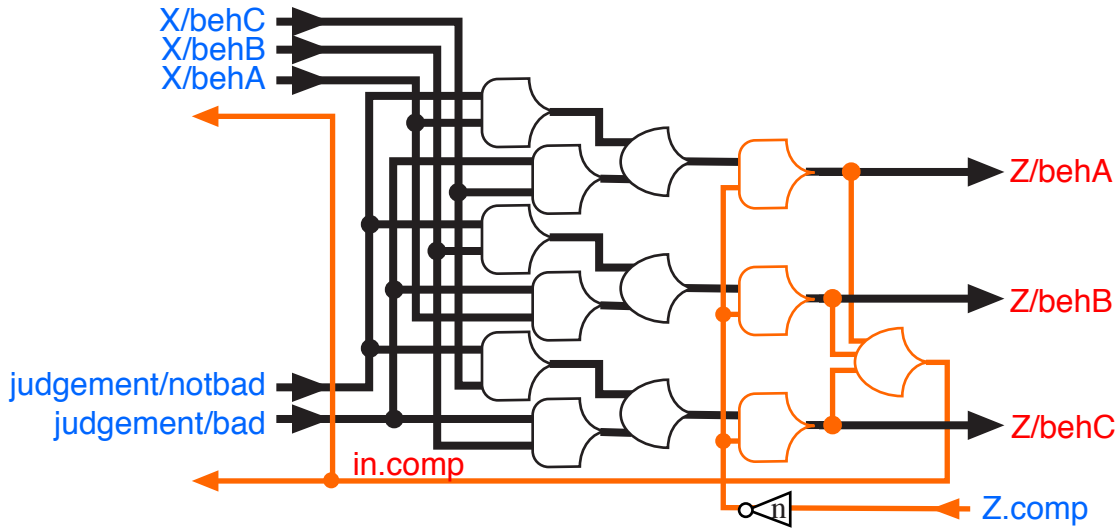
(initN(A/{2 1 0}/, A.comp/ =>
        B:N/{2 1 0}/, B.comp/)
   A.comp <=?B<=[A ~B.comp])
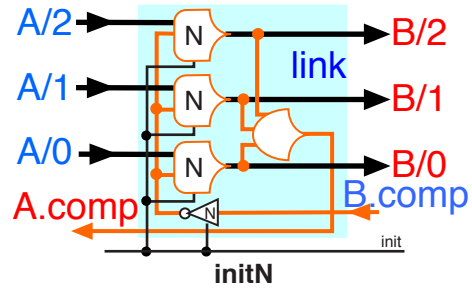
4.9.2.   The behavior interaction pipeline component network

(behmod(X/{behA behB behC}/,in.comp judgement/{bad notbad}/,in.comp =>
                                     Z/{behA behB behC}/,Z.comp)
 in.comp<=?Z/{behA<=[{[X/behC judgement/bad]
                 [X/behA judgement/notbad]} ~Z.comp]
          behB<=[{[X/behA judgement/bad]
                 [X/behB judgement/notbad]} ~Z.comp]
          behC<=[{[X/behB judgement/bad]
                 [X/behC judgement/notbad]} ~Z.comp]
             })

## 4.9.3.   The arbitration pipeline component network

The arbiter (section 3.2.1)

<center>{{Ain Bin => Aout Bout}}</center>

arbitrates the continual liveness flow **comfy** with the occasional sensor flow **danger** into the network as a locality **judgement** of two mutually exclusive differentnesses.

(next(=> judgement/{bad notbad}/,judgement.comp/)
      (carryon/ uhoh/ comfy/ danger/)
      bad.close<=judgement/bad<=[uhoh ~ judgement.comp]
    notbad.close<=judgement/notbad<=[carryon ~ judgement.comp]
{{danger<=[sensor ~bad.close] (comfy<=[~comfy ~notbad.close] => uhoh carryon}})

Arbitrate two independent places
of association into a single locality.
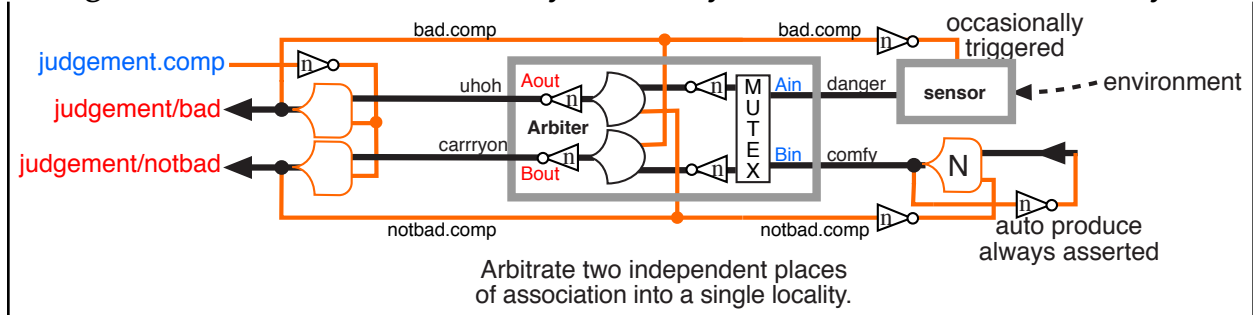
The dependency relations: **judgement** is dependent through the **arbiter** on **uhoh** which is dependent on the **sensor** and on **carryon** which is dependent on the **auto produce**. Locality **judgement** is enabled as a locality but each condition of the locality closes individually with its pre arbiter source.
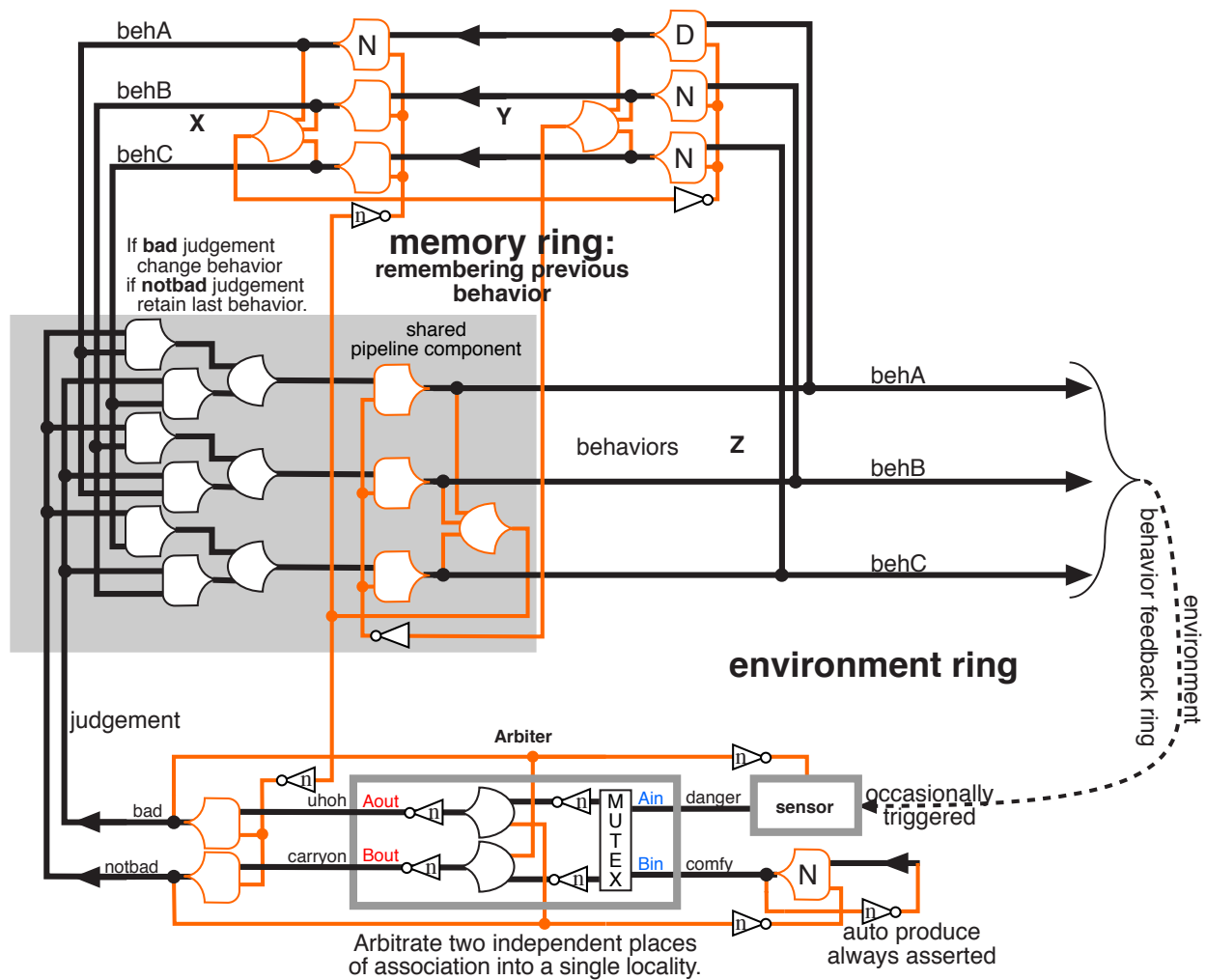
4.9.4.   The immersed ring network

The dependency expression.

((=>) (Z)
      Z<=behmod(init2(initN(Z)) next( )))

The inheritance expanded expression.

((=>) (Z/{2 1 0}/,Z.comp)
      Z, Z.comp <=behmod(init2(initN(Z, Z.comp)) next( )))

Figure 4.13. Linked rings with part of one ring flowing through the environment.

   The upper ring of Figure 4.13 remembers the imposed behavior. The lower ring flowing partially through the environment determines whether the imposed behavior (section 3.12) should change or not. If the sensor's (section 3.12) closure is **D** the sensor will monitor the environment. If the monitoring exceeds some threshold the sensor will transition its output to **D**. When the closure becomes **N** it will transition its output to **N**. Upon which its closure will transition to **D** and the sensor will begin monitoring its input again. A **BAD** response from the sensor might be pain or frustration or danger. In the absence of a **BAD** response from the environment the sensor's output remains at **N** and the network remains alive from the auto produced, **NOTBAD** (section 3.11), continually presenting the same differentness to the environment.

   The sensor **BAD** and the auto produce **NOTBAD** are happening independently and are not coordinated. They might even transition to **D** simultaneously. The arbiter receives the two uncoordinated flows **BAD** and **NOTBAD** and combines them into a single locality of mutually exclusive **judgement** conditions, presented to the network (section 3.2.1 and Appendix C) .The linked rings coupled through the interaction of **judgement** and **X** are continually alive continually imposing behavior on the environment at their intrinsic throughput rate and only occasionally does the environment respond with an indication that behavior should be changed.

If **BAD** continues to be asserted the network will continue to change its behavior until it finds a behavior that no longer asserts **BAD**. When **BAD** ceases the auto produce will play through the arbiter with **NOTBAD** and the network will continue asserting the last behavior that did not elicit **BAD**. If none of the available behaviors unassert **BAD** then the network is in either mortal danger or chronic depression.

The immersed network through its spontaneously interacting differentnesses, is manipulating and appreciating the passive differentnesses of the external environment. While an immersed network can manipulate passive differentness, information, there is nothing manipulating the immersed network which is self contained and self determined. For example, while sequential computers are designed, implemented and manipulated by humans there is no designer and implementor of humans which must arise and behave entirely on their own merits.

4.10. The memory ring network: Stopping time

A wavefront presented to a closure link will wait indefinitely at the link for the closure enabling the wavefront to flow through the link. This behavior of indefinitely waiting for an enable can be used to store a wavefront and retrieve the wavefront. A ring can be configured such that a wavefront in the ring is caused to wait indefinitely at a closure link until an enable wavefront such as a read operation arrives. The memory ring has a binding portal through which write wavefronts and read write directives are presented. When the stored wavefront is enabled out of the pipeline ring it flows through the output of the pipeline ring and also flows around the ring back to the storage closure link to be read again. The memory ring can be written by enabling the wavefront at the wait closure link to be consumed making way for a new wavefront to flow into the pipeline ring to the closure link.

Wavefronts must be stopped and stored in **N-D** pairs so it requires a successive pair of closure links to stop a flowing wavefront. In Figure 4.14 the **D** completeness wavefront initialized to **0** is stored in locality **SD**. The following completely **N** wavefront is stored in locality **SN**.

The stretched behaviors

"all of"   "one of"
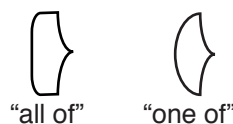
represent behaviors spanning arbitrarily sized localities of differentness.
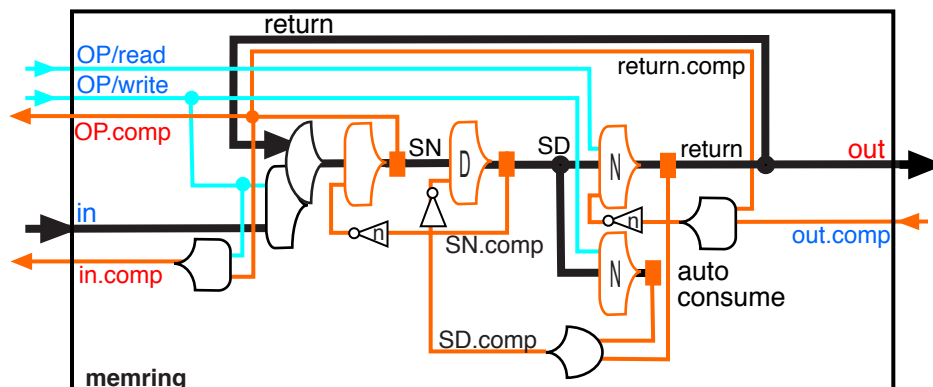

Figure 4.14. The memory ring network.

The memory ring network expression

(memring(in/[31-0]/{1 0}/,in.comp/ OP/{read write}/,OP.comp/ =>
out/[31-0]/{1 0}/,out.comp/)

(return/[31-0]/{1 0}/,return.comp/ SN/[31-0]/{1 0}/,SN.comp/
SD:0/[31-0]/{1 0}/,SD.comp/   )
SD.comp <={?(out, return)<=[SD OP/read ~[out.comp return.comp]]
?[SD OP/write] }
SN.comp <=?SD<=[SN ~~ SD.comp]
in.comp <=[(OP.comp, return.comp)<=?SN
<=[{return [in OP/write]} ~ SN.comp] OP/write])

4.10.1.   memring read behavior

When **OP/read** transitions to **D** and **out.comp** transitions to **N** the stored **D** completeness wavefront is enabled to locality (**return out**) leaving locality **SD** a **D** bubble. The completely **N** wavefront stored in locality **SN** flows into the **D** bubble of locality **SD** leaving locality **SN** a **N** bubble. The **D** completeness wavefront in **return** flows into the **N** bubble of locality **SN** and **OP.comp** transitions to **D**.

When **OP.read** transitions to **N** and **out.comp** transitions to **D** locality (**return out**) becomes a **D** bubble. The completely **N** wavefront in locality **SD** flows into locality (**return out**) leaving locality **SD** a **N** bubble. The **D** completeness wavefront in locality **SN** flows into locality **SD** leaving locality **SN** a **D** bubble. The completely **N** wavefront in locality **return** flows into locality **SN** and **OP.comp** transitions to **N**. **out.comp** transitions to **N** and the **memring** is ready for a next **OP**.

4.10.2.   memring write behavior

When **OP/write** transitions to **D** the **D** completeness wavefront stored in locality **SD** is enabled to the auto consume locality leaving locality **SD** a **D** bubble. The completely **N** wavefront stored in locality **SN** flows into the **D** bubble of locality **SD** leaving locality **SN** a **N** bubble. locality in is enabled into locality **SN** and locality in becomes a **D** bubble. Locality **OP.close** transitions to **D**.

When **OP/write** transitions to **N** the auto consume locality becomes a **D** bubble. The completely **N** wavefront in **SD** is enabled to the auto consume locality leaving locality **SD** a **N** bubble. The **D** completeness wavefront in locality **SN** flows into locality **SD** leaving locality **SN** a **D** bubble. When locality **in** transitions to completely **N** the completely **N** wavefront flows into locality **SN** and **OP.comp** transitions to **N** and the **memring** is ready for a next **OP**.

OP/write and the **in** wavefront are **"all of"** related which must trace back to a common source, such as the decode of an instruction, that establishes the relation. The ring is initialized with a wavefront so there is always a wavefront stored in the memory. It is never empty. A read will always succeed and a write will always succeed.

4.11. A network of addressable memory rings: arranging time

Memory ring networks can be composed into a large addressable wavefront memory. Figure 4.15 expressed as a divergent-convergent network of **"one of"** related branches each branch

containing a memory ring. Each memory ring represents differentness of both place of association and differentness of content. The wavefront stored in the ring is differentness of content. The address of the ring branch references differentness of place of association within the memory network which represents differentness of wavefront time. The memory network is indifferent to the differentness of wavefront content in any of its branch memory rings.
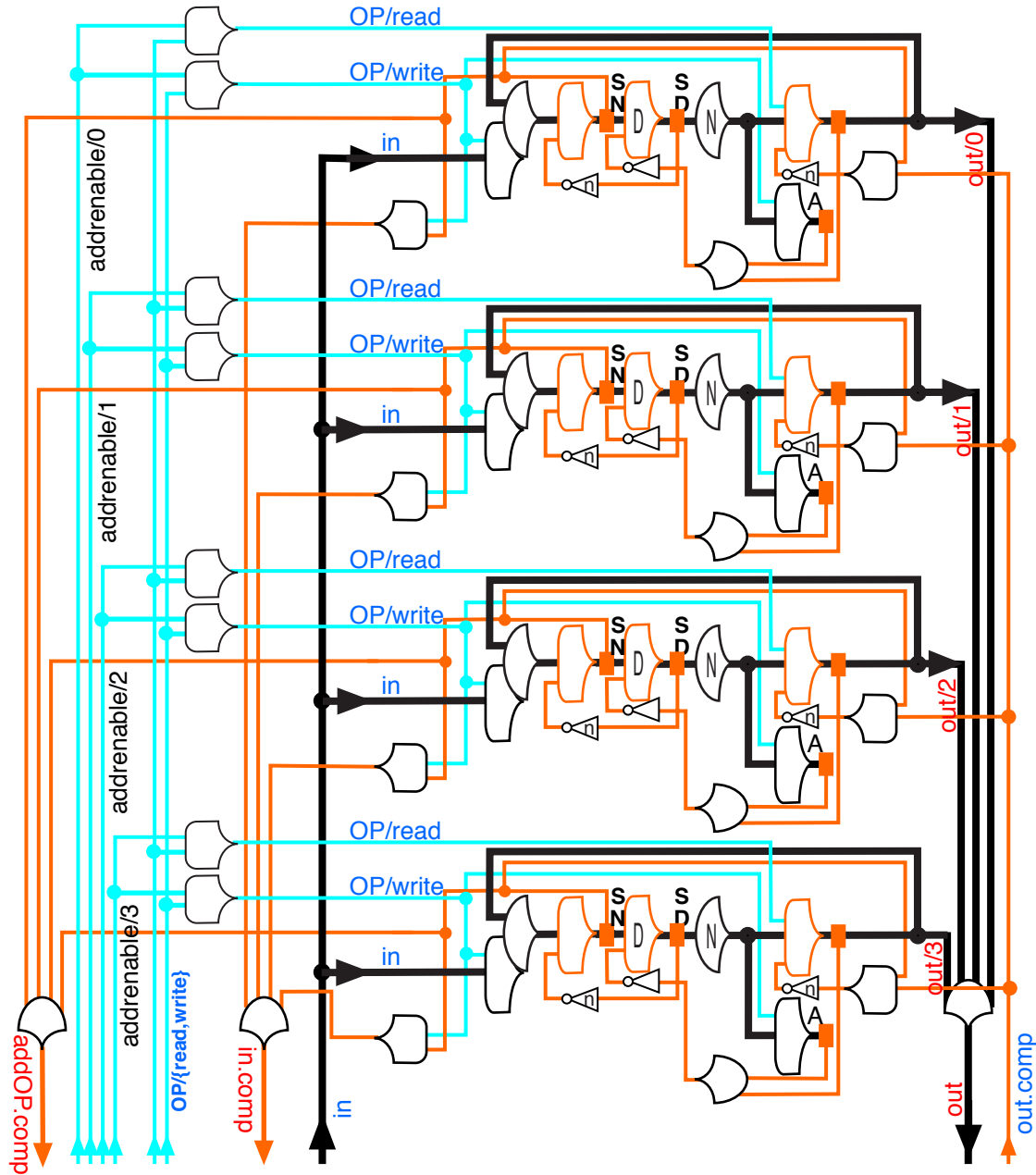
Figure 4.15. Addressable memory of memory rings.

Dependency expression for ram:

Copyright © 2024 by Karl Fant

(ram(in addrenable/{3 0}/ OP => out)
             (out/{3 0})
      readout<= {out/3-0<=memring(in [addrenable/3-0 OP])})

addrenable/{3 0}/ is not inherited but is defined in ram specifying the size of the memory.

Inheritance expanded network expression.

(ram(in/[31-0]/{1 0}/,in.comp addrenable/{3 0}/,addOP.comp
         OP/{read write}/,addOP.comp => out/[31-0]/{1 0}/,out.comp)
         out,out.comp<= {memring(in,in.comp
                              [addrenable/0,addOP.comp OP,addOP.comp])
                    memring(in,in.comp
                              [addrenable/1,addOP.comp OP,addOP.comp])
                    memring(in,in.comp
                              [addrenable/2,addOP.comp OP,addOP.comp])
                    memring(in,in.comp
                              [addrenable/3,addOP.comp OP,addOP.comp])
                    })

   memring being dependent on **addrenable/{3 0}** and on **out/{3 0}** is instantiated four times each receiving the corresponding components of **addrenable/x** and **out**. **in** and **OP** are distributed to all four intances of **memring**. **addrenable** determines which instance of **memring** receives **OP** and performs the specified **OP**.

   Being able to stop a wavefront and to retrieve it on demand allows the rearrangement of wavefront flow through time and enables the possibility of controlling wavefront flow through dynamically constructed dependency networks with wavefront instances of interaction being stopped and saved in the addressable memory then being retrieved as needed. In particular the one behavior at a time sequential interpretation of dependency networks is enabled.

4.12. A most primitive sequence controller

   Any interaction network expression can be decomposed into a sequence of simple behaviors that realize the dependency relations among the interaction network behaviors(section 1.1.1). The difficulty with one at a time sequencing of behaviors is that it forms a one dimensional behavior space in which wavefronts cannot flow directly from behavior to behavior but must flow indirectly through a means of delaying wavefronts and releasing them when needed in the sequence. The addressable memory of Section 4.11 fulfills this requirement enabling the sequential interpretation of dependency networks.

   The other enabler of sequential interpretation is the sequence controller which realizes the one at a time sequence of behaviors by determining each next behavior, retrieving its input wavefronts from the memory flowing them through a specified interaction behavior and returning the result wavefront to memory.

4.12.1.   Reducing an interaction network expression

   An interpreter behavior is referenced as a full portal behavior references (section 3.4.5.1).

behaviorname(localityA localityB => localityC)

An interaction network expression is reduced to an interpretable sequence by transforming dependency relations into referencable interpreter behaviors and by mapping all localities into representations that fit into the wavefront memory. Finally, the behaviors needed to represent the three dimensional network in a one dimensional sequence of behaviors are inserted into the sequence.

The expression from the LFSR example of section 4.8 serves as the reduction example.

$$(\quad (\ => \ ) \quad (B/\{1\ 0\}/ \ C/\{1\ 0\}/ \ D/\{1\ 0\}/ \ E/\{1\ 0\}/ \ F/\{1\ 0\}/ \ O/\{1\ 0\}/)$$

F<=pipe1(E<=pipe0(D<=pipe1(C
       <=pipe0(B<=pipe0(pipe0(O
       <=XOR(F XOR(XOR(B C) XOR(D E))))))))))

An interaction network is expressed in terms of syntax relations and name correspondence relations. The first task is to re express all syntactic nesting relations as name correspondence associations among full portal behavior references. The nameless nested localities of the portal nesting relations are assigned names. The behavior references are related by name correspondence as full portal behavior references referencing the newly assigned names and are ordered to fulfill the dependency precedence relations of the interaction network expression.

4.12.1.1.   The XOR full portal behavior references

XOR(D E => T3)
XOR(B C => T2)
XOR(T2 T3 -> T1)
XOR(F T1 => O)

XOR(D E => T3) and **XOR(B C => T2)** can be in any order but both must be before **XOR(T2 T3 -> T1)** which must be before **XOR(F T1 => O)**.

Some full portal references may need to be further reduced and substituted. Reduction proceeds until all references refer to behaviors realized by the sequence controller referencing localities that fit into the wavefront memory in an ordered sequence that fulfills the dependency relations of the original dependency network expression.

For the purpose of reduction only the interaction dependency relations are considered. The closure network is not regarded. The flow coordination of sequential interpretation is quite different from the closure flow coordination of the interaction network (section 4.12.2.2).

4.12.1.2.   The pipex behavior references

The localities for the locality nested **pipe0** and **pipe1** behaviors are already named except for one which is assigned the name **T0**. The references are re expressed from locality nested references to full portal references in column **A** of Figure 4.16.

The **pipex** behaviors do double duty in the interaction network by initializing localities as well as moving wavefronts between localities. For the sequenced behaviors these duties are separated out in column **B** of Figure 4.16 with separate behaviors. Behavior **initx** initializes wavefronts in a locality and behavior **move** moves wavefronts from locality to locality.

A	B	C
pipex behaviors	**separation**	**integration**

		init0(T0)
	init0(T0)	init0(B)
	init0(B)	init0(C)
pipe1(E => F)	init0(C)	init1(D)
pipe0(D => E)	init1(D)	init0(E)
pipe1(C => D)	init0(E)	init1(F)
pipe0(B => C)	→ init1(F)	→ XOR(D E => T3)
pipe0(T0 => B)	**move(E => F)**	XOR(B C => T2)
pipe0(O => T0)	**move (D => E)**	XOR(T2 T3 => T1)
	move (C => D)	XOR(F T1 => O)
	move (B => C)	**move(E => F)**
	move (T0 => B)	**move(D => E)**
	move (O => T0)	**move(C => D)**
		move(B => C)
		move(T0 => B)
		move (O => T0)

Figure 4.16. Composing the sequence of dependent operations.

These full portal reference behaviors are placed in an order that preserves the dependency relations of the original interaction network expression. The **initx** behaviors can occur in any order among themselves but must all occur before any interaction behaviors in the ordered list of behaviors. In the **LFSR** network the first wavefronts flowing through the **XOR**s will be the initialized wavefronts not the first **move**d wavefronts. So the sequential order in column **C** of Figure 4.16 is to first perform the **initx**s and then perform the **XOR**s then perform the **move**s of the wavefronts. The **move**s of the wavefronts are ordered such that each locality is read and moved before the next behavior overwrites it.

4.12.1.3.   One dimensional complexity

The interaction behaviors of the above list in column **C** of Figure 4.16 (excluding the init behaviors) remain a valid dependency network expression. The behaviors can be shuffled into any order and the name correspondence relations still construct the dependency network minus the inits. For sequential interpretation, however, there are only a few specific orderings of behaviors that will work and there are additional behaviors not derived from the dependency network expression that have to be added for sequential interpretation to work.

An ordered list of behaviors to be interpreted one behavior at a time in sequence forms a one dimensional behavior space that cannot, on its own, characterize the connectivity of a three dimensional interaction network. Because of concurrency and conditionality the next behavior to be realized cannot always be the next behavior in the one dimensional sequence. It is necessary to conditionally and unconditionally jump around in the one dimensional sequence of behaviors. The sequence controller must support the additional behaviors that enable conditional and unconditional jumps within the one dimensional sequence.

Each directive is assigned a unique name so it can be referenced. A name range with an inherent order and easily determined nextness such as numeric is most convenient. In column **A** of Figure 4.17 the assigned names are numeric and an unconditional **jump** is inserted as the last directive closing the ring of dependency. The interpretation begins with the directive named **100**, progresses to directive named **116** then continues with the directive named **106** forming an infinite loop of interpretation emulating the continual behavior and wavefront flow of the **LFSR** ring network.

A		B		C	
100	init0(T0)	100	init0(T0)	100	2(20)
101	init0(B)	101	init0(B)	101	2(11)
102	init0(C)	102	init0(C)	102	2(12)
103	init1(D)	103	init1(D)	103	3(13)
104	init0(E)	104	init0(E)	104	2(14)
105	init1(F)	105	init1(F)	105	3(15)
106	XOR(D E => T3)	106	XOR(D E => T3)	106	4(13 14 => 23)
107	XOR(B C => T2)	107	XOR(B C => T2)	107	4(11 12 => 22)
108	XOR(T2 T3 -> T1)	108	XOR(T2 T3 => T1)	108	4(22 23 => 21)
109	XOR(F T1 => O)	109	XOR(F T1 => O)	109	4(15 21 => 16)
112	move(E => F)	110	wait	110	6
113	move(D => E)	111	move(O => out)	111	5(16 => 24)
114	move(C => D)	112	move(E => F)	112	5(14 => 15)
115	move(B => C)	113	move(D => E)	113	5(13 => 14)
116	move(T0 => B)	114	move(C => D)	114	5(12 => 13)
117	move(O => T0)	115	move(B => C)	115	5(11 => 12)
116	jump (106)	116	move(T0 => B)	116	5(20 => 11)
		117	move(O => T0)	117	5(16 => 20)
		118	jump(106)	118	7(106)

Figure 4.17. Forming and encoding the one dimensional sequence of behaviors.

The **LFSR** source network of Section 4.8.5 can be expressed by inserting a simple interface protocol. In column **B** of Figure 4.17 a **wait** directive is inserted after the last **XOR** directive and a **move** directive is inserted to move the wavefront in locality **O** to locality **out**.

Interpretation begins at **100** and halts at the **wait** directive with the **XOR**s having delivered their output wavefront to locality **O**. An external reference requests (closes with) the wavefront of locality **O**. The wait falls through and the next behavior moves the wavefront in **O** to **out**. The sequence of behaviors continues with the **move** behaviors moving the wavefronts presented to the **XOR**s from locality to locality. The wavefronts then flow through the **XOR**s with their output wavefront flowing to locality **O**. The interpretation then **wait**s for the next request.

4.12.1.4.   Mapping references to memory addresses and to behavior codes

The behavior names are translated into a code that the interpreter can understand to select the appropriate behavior. Each locality name is assigned a numeric memory address name.

numeric locality names		behavior codes
B -> 11		init0 -> 2
C -> 12	T0 -> 20	init1 -> 3
D -> 13	T1 -> 21	XOR -> 4
E -> 14	T2 -> 22	move -> 5
F -> 15	T3 -> 23	wait -> 6
O -> 16	out -> 24	jump -> 7

The ordered list of behaviors with the reassigned names is in column **C** of Figure 4.17. The behavior code and the memory address names of each behavior are contrived to fit into a single unit of wavefront memory that is referenced with one memory address forming an interpretation behavior that a sequence controller can read and understand specifying a flow from wavefront memory through the referenced behavior and back to wavefront memory. The entire expression is now in terms of a sequence of full portal references to numerically coded behaviors realized by the interpreter referencing numeric memory address locality names.

The incoherent transition behavior of a self determined interaction network becomes temporally and spatially regimented behavior in its sequentially interpreted counterpart.

4.12.2.   The sequence controller

The sequence controller of Figure 4.18 is a network of coupled and linked rings that reads each behavior from memory and manages the flow of wavefronts from an addressable memory through specified interaction behaviors whose results flow back to the memory. There is the **program counter ring** network which determines from the current address and from the the type of the previous behavior, branch or not branch, the next address from which to read a behavior. There is the **decoder ring** network which reads and decodes the behavior dispatching the behavior code and the memory addresses to the **ALU/memory ring** network. The **ALU/memory ring** network accesses the input wavefronts from memory, flows them through the specified behavior and places the result wavefront back in memory. There is also the **branch ring** coupled through the **ALU** and the **program counter** which determines the conditional of a branch.

The sequence of behaviors for the interaction network being interpreted are placed in the memory in numeric order. The initial wavefronts of localities are placed in their memory locations. The program counter initializes the next behavior address to the first behavior to be read, in this case **100**. The address is passed to the decoder and the behavior is read, decoded and realized. The sequence controller then cycles through the behaviors. The resulting sequence of behaviors realizes the source interaction network constructing the network's behavior extended through time and memory.

4.12.2.1.   The exposed binding portal

The sequential interpreter is not a completely self determined network that can explore its environment entirely on its own. It has an exposed binding portal in that any program can be loaded into its memory from an external environment. Once a program is loaded, however, it self determinately realizes the program.
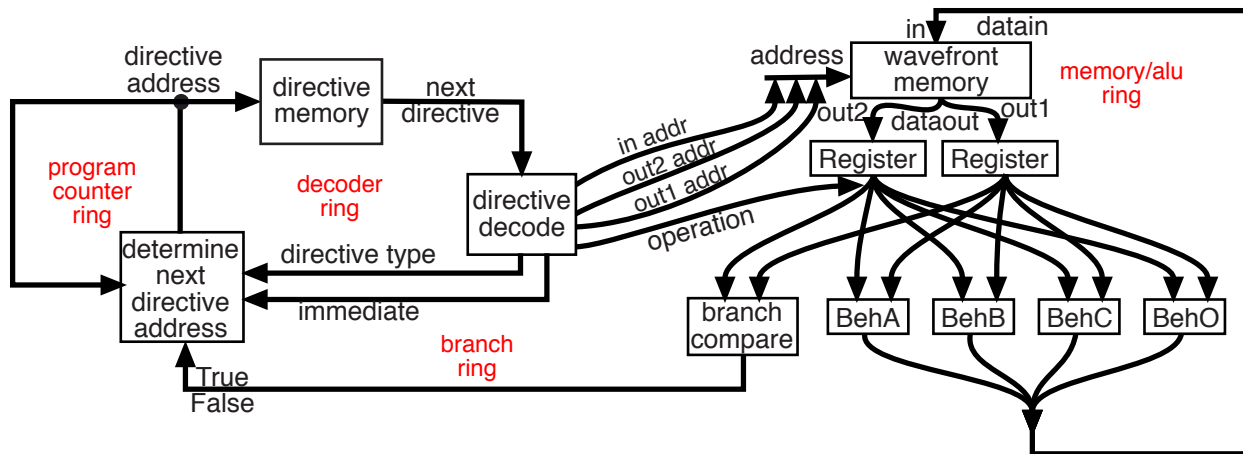
Figure 4.18. The sequential interpreter and its linked rings.

A sequential controller however might be outfitted with imposition and sensor behaviors and loaded with a never ending program that is essentially self determined and capable of independently exploring its environment, a robot.

4.12.2.2.   Sequential flow coordination

Sequential interpretation requires that each directive is completed with its asserted output written to memory before the next sequential behavior is begun. Completeness of output ensures the stable completeness of input presentation for subsequent behaviors. This is different from the interaction network for which the completeness of output of a behavior ensures the completeness of its own input. In both cases interaction behavior is coordinated in terms of completeness of output assertion progressively accumulating to ensure completeness of input presentation.

4.12.2.3.   Scalable generality

The sequence controller and addressable memory form a *sequential interpreter*, an interaction network capable of interpretively realizing the behavior of any other interaction network. Realizing every interaction network as a unique physical network is not practical and does not scale. But being able to express any interaction network directly as a dependency network, translate it to a sequential expression and then to realize the interaction network expression dynamically constructed through time and memory with a simple sequential interpreter that can be indefinitely replicated does scale.

The dynamic and conditionally varying construction of dependency relations through time and memory provides a flexibility and generality of realizability not possible with statically realized networks.

4.12.2.4.   Expressive generality

Why express one of the multitudes of possible sequences when you can express the one unique dependency expression that all the sequences must derive from and which you must understand to express one of the seqeunces and have it mechanically translated, compiled, to an appropriate sequential expression (section 1.1.1).

4.12.2.5.   Again, nothing new or extrinsic

No new primitivity or extrinsic capability has been introduced. The addressable memory and sequence controller are realized entirely in terms of the primitive behaviors of section 3.2.[1] The sufficiently expressive primitive behaviors have finally bootstrapped sequential interpretation.

The derivation rationale of the sequential interpreter did not appeal to a mathematician with pencil and paper nor to the notion of algorithm. There was no appeal to Boolean logic, to timing analysis or to a timing referent such as a clock. Sequential one at a time interpretive stepping was not assumed a priori but emerged as a possibility of universal realizability with the emergence of the addressable memory in which wavefronts could be stored and retrieved.

4.13. A quest fulfilled

The quest of section 1.8 is fulfilled journeying from the pure condition differentiation expression of Chapter 2 through the pure association differentiation expressions of Chapter 3 to the temporal differentiation expressions of Chapter 4 computational interaction with all concurrent relations fully accounted in terms of *differentness spontaneously and dependently interacting and changing* with primitive interaction behaviors (section 3.2) that *reliably express primitive concurrency behavior* and that *realize a most primitive sequential interpreter* bootstrapping sequential interpretation (section 1.8).

4.13.1. Counter example

Sequence control and Boolean logic are considered fundamental to computation. Dependency flow computation presents a counter example expressing deterministic complex computation containing no Boolean logic expression, no sequence control or any other form of explicit control, possessing no stable samplable state and possessing no common behavioral referent such as a clock or a mathematician. Furthermore, sequence control and Boolean logic are derivable from dependency flow computation implying it to be the more fundamental characterization.

4.13.2. Full circle

A self determined interaction network arises able to replicate itself and evolve into a complex self determined interaction network that invents differentness interaction mapping with an exposed binding portal, that becomes a controlling external environment using interaction as a tool to ask questions and receive answers and that becomes fascinated with the limitations of the exposed binding portal.

4.13.3. Still not done

There remains one final stage of the journey, a walk through the spectrum of differentiation relating the dimensions of condition differentiation and association differentiation.

---

1. The author in collaboration with Igor Ilyukhin has implemented a RISC V 32IM as a structure of coupled rings in terms of the primitive interaction behaviors of section 3.2.

Chapter 5:
The spectrum of Differentiation

Any interaction of differentness is represented as a coextensive collaboration between condition differentness and association differentness. The spectrum of differentiation, Figure 5.1, embodies this collaboration with pure condition differentiation at one end of the spectrum and pure association differentiation at the other end with various proportions of collaboration in between.
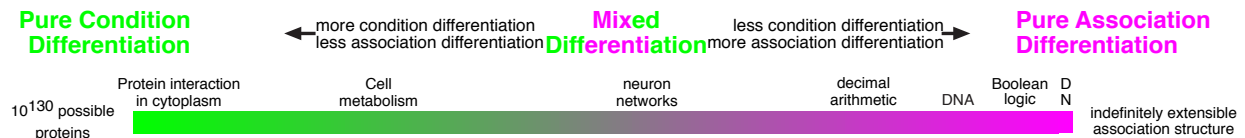

Figure 5.1.   The spectrum of differentiation.

5.1.   The collaboration

Different conditions are different within a same place of association. A same condition is different within different places of association. A range of mutually exclusively different conditions extends the expression of differentness of a same place of association. Different places of association asserting the same range of different conditions extends the expression of differentness of the same range of different conditions.

With decimal place value numbers, for instance there are 10 different numeral conditions **0**, **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8** and **9**. One differentness of place of association in a decimal place value number can represent 10 different numeral conditions extending the range of differentness expression of the one place by ten differentnesses. Four different places of association asserting the same numeral condition, 3333, extends the range of differentness expression of the one numeral condition by four differentnesses. Each numeral condition is different by virtue of its differentness of place. Condition differentness and association differentness collaborate to express a large range of mutually exclusive differentness.

With binary place value numbers each differentness of place of association is extended to two differentnesses **0** and **1**. Each place of association of a binary number can express only **0** or **1** so it takes more places of association to express a comparable decimal number. Decimal number **1023** requires four decimal places while the comparable binary number **1111111111** requires 10 binary places. Two representations of a same differentness with different proportions of condition differentiation and association differentiation.

Twenty six letter conditions asserted one at a time at different places of association relations form words. Words associate to form sentences. Sentences association to form paragraphs and so on. Japanese uses an enormous range of different symbol conditions asserted at different places of association to form its words and sentences. Again different proportions of condition differentness and association differentness to express comparable meaning differentness.

The spectrum of differentiation encompasses this proportionality of collaboration of mutual extension between condition differentness and association differentness in the representation of mutually exclusive differentnesses and their interactions providing a unifying foundation of expressivity among what have been considered to be quite disparate forms of representing differentness and the interaction of differentness.

5.2.   A walk along the spectrum

The collaboration is illustrated by walking one temporal instance of one specific interaction of differentnesses along the spectrum with condition differentiation and association differentiation collaborating in various proportion to represent the one specific interaction.

One differentness from the group of three mutually exclusive differentnesses named **U**, **V**, **W** and one differentness from the group of three mutually exclusive differentnesses named **X**, **Y**, **Z** associate and interact producing one of nine mutually exclusive differentnesses named **A**, **B**, **C**, **D**, **E**, **F**, **J**, **K**, **L**. The example interaction is shown in Figure 5.2 as an interaction mapping table and as a list of interaction dependency relations.

mapping
table

	U	V	W
X	A	B	C
Y	D	E	F
Z	J	K	L

specifies
interaction→
dependencies

[W Z] => L
[V Z] => K
[U Z] => J
[W Y] => F
[V Y] => E
[U Y] => D
[W X] => C
[V X] => B
[U X] => A

Figure 5.2.   Example interaction.

The tradeoffs between the proportion of condition differentiation and association differentiation are considered in terms of the example interaction at various locations along the spectrum characterized in terms of the amount of condition differentiation available. The walk begins at the pure condition end of the spectrum with sufficient condition differentiation to express the example interaction at a single place of common association and continues by constraining condition differentiation until there is only one available differentness condition at the pure association end of the spectrum.

5.3.   With fifteen available differentness conditions

With fifteen available differentness conditions named **A**, **B**, **C**, **D**, **E**, **F**, **J**, **K**, **L**, **U**, **V**, **W**, **X**, **Y**, **Z** the example interaction can be represented with pure condition differentiation (Chapter 2 for a more complex example of interaction expressed purely in terms of condition differentiation) at a single place of common association as illustrated with the bag in Figure 5.3. The bag begins empty. One of conditions **X**, **Y**, **Z** and one of conditions **U**, **V**, **W** enter the bag and interact with the interaction propensities expressed in the mapping table producing one of conditions **A**, **B**, **C**, **D**, **E**, **F**, **J**, **K**, **L**. The result condition exits the bag leaving the bag empty bounding one temporal instance of one specific interaction of differentness. The interaction mapping is the same as Figure 5.2.

	U	V	W
X	A	B	C
Y	D	E	F
Z	J	K	L

```
[W,Z] -> L
[V,Z] -> K
[U,Z] -> J
[W,Y] -> F
[V,Y] -> E
[U,Y] -> D
[W,X] -> C
[V,X] -> B
[U,X] -> A
```
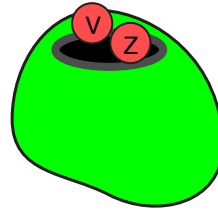
Figure 5.3.   Fifteen differentness conditions, 1 shaking bag.

With only one place of common association and with associations inside the bag being indeterminate there is no differentiation in terms of association. The differentnesses and their interaction is expressed purely in terms of condition differentnesses and their interaction propensities mapping input differentness conditions to output differentness conditions.

The interaction dependency expression as relations of names of conditions:

$$(L<=[W\ Z]$$
$$K<=[V\ Z]$$
$$J<=[U\ Z]$$
$$F<=[W\ Y]$$
$$E<=[V\ Y]$$
$$D<=[U\ Y]$$
$$C<=[W\ X]$$
$$B<=[V\ X]$$
$$A<=[U\ X]\ )$$

There is no referencing or binding in a pure value expression. Conditions are sufficiently differentiating. They just show up and interact.

All possible interaction mappings can be expressed by specifying the interaction propensities in the mapping table. For instance:

mapping table

	U	V	W
X	B	A	D
Y	D	B	A
Z	C	E	B

specifies interaction→ dependencies

```
W Z => B
V Z => E
U Z => C
W Y => A
V Y => B
U Y => D
W X => D
V X => A
U X => B
```

The interaction is specified entirely in terms of condition differentiation.

5.4.   Constrained to nine available differentness conditions

Nine conditions are sufficient to represent the output differentnesses but six of the conditions must also be used to represent the input differentnesses.

mapping
table

	A	B	C
D	A	B	C
E	D	E	F
F	J	K	L

specifies
interaction→
dependencies

C F => L
B F => K
A F => J
C E => F
B E => E
A E => D
C D => C
B D => B
A D => A
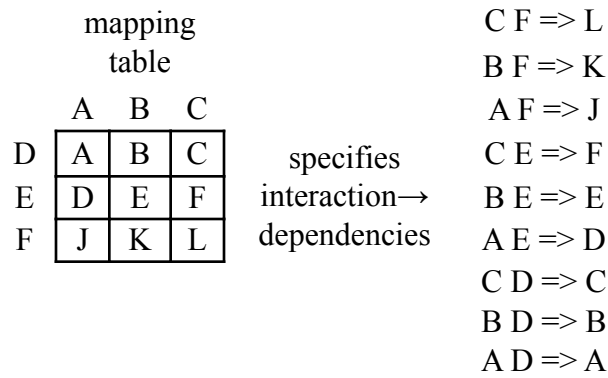
Figure 5.4.   Interaction mapping with nine available differentness conditions.

Input differentness conditions must be differentiated from the identical output differentness conditions. This is represented with an association interaction behavior that isolates input conditions from identical output conditions establishing isolated localities of differentness for its inputs and its output, see Figure 3.4 and section 3.1.3.

An association interaction behavior, as explained in Section 3.1.6, continually receiving input interaction conditions and continually asserting an output interaction condition requires a condition named **N** explicitly representing absence of interaction condition as well as the monotonic transitioning between interaction condition completeness and completely **N** including the ability of the interaction association behavior to recognize transitions of input to interaction condition completeness and to completely **N**. The condition named **N** is included as integral to this discussion and will be explicitly referenced in the interaction behavior mapping tables and in the interaction dependency expressions.

Interaction no longer occurs in terms of freely associating conditions but now occurs inside the association interaction behavior in terms of directly associated conditions. The interaction is characterized by the mapping table on the right of Figure 5.5 which includes the nine differentness conditions plus **N** with the – indicating no transition. The interaction mapping behavior transitions its asserted output to an interaction differentness condition only when both presented inputs are interaction differentness conditions (interaction differentness completeness) and transitioning its asserted output condition to **N** only when both presented inputs are differentness condition **N** (completely **N**) representing emptiness of interaction differentnesses (Section 3.1.6). When the presented input is not complete, i.e. there is one **N** and one interaction differentness condition presented, the interaction behavior does not transition its asserted output condition (section 3.2).
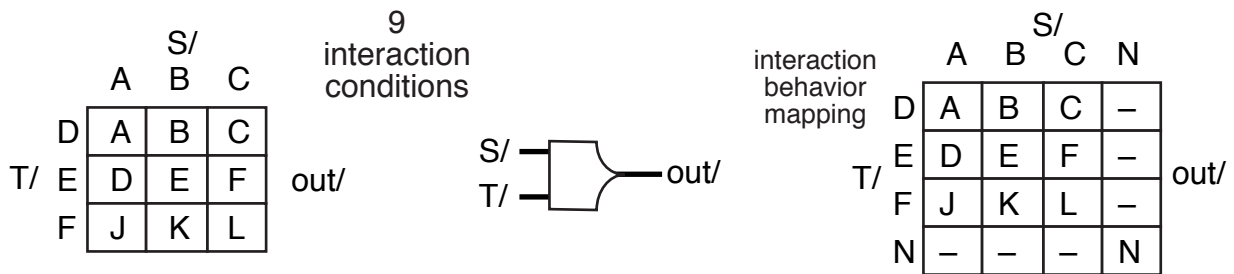
Figure 5.5.   Nine differentness conditions with one association interaction behavior.

The interaction behavior forms a simplest of interaction networks. Each differentness now is expressed as a differentness of place of association combined with a differentness of condition. Locality **S** can assert three interaction differentnesses **S/C**, **S/B**, **S/A** and **S/N**. Locality **T** can assert three interaction differentnesses **T/F**, **T/E**, **T/D** and **T/N**. Locality **out** can assert differentnesses nine interaction differentnesses **out/L**, **out/K**, **out/J**, **out/F**, **out/E**, **out/D**, **out/C**, **out/B**, **out/A** and **out/N**. Remember that **N** is not an interaction differentness. The differentnesses asserted with identical conditions at different places of association are different by virtue of the differentness of place of association. **T/F** and **out/F** are different by virtue of the associational differentness of **T** and **out**.

An interaction begins empty with **S/N T/N** and **out/N.** Inputs **S/** and **T/** transition to interaction conditions and **out/** transitions to the determined interaction condition. **S/** and **T/** transition back to **N**, and **out/** transitions to **N** leaving the interaction behavior empty bounding one temporal instance of one specific interaction of differentness.

The interaction dependency expression as relations of names of conditions and names of places of association:

(9condexamp(S/{C B A N}/ T/{F E D N} => out/{L K J F E D C B A N})
out/{L<=[S/C T/F]
K<=[S/B T/F]
J<=[S/A T/F]
F<=[S/C T/E]
E<=[S/B T/E]
D<=[S/A T/E]
C<=[S/C T/D]
B<=[S/B T/D]
A<=[S/A T/D]
N<=[S/N T/N] })

All possible interaction mappings can be expressed by arranging the output differentnesses in the mapping table. The burden of representing differentiation and interaction is now shared between condition differentiation and association differentiation.

5.5.   Constrained to six available differentness conditions

Six differentness conditions named **A**, **B**, **C**, **D**, **E**, **F** are still sufficient to represent the input differentnesses but six differentness conditions are not sufficient to represent the nine output conditions so the output differentness **out** must now be represented as two associated output conditions asserted by two association interaction behaviors asserting locality **out** with two component localities **Y** and **Z**. Input locality **S/** can assert one of the interaction differentness conditions **A** or **B** or **C**. Input locality **T/** can assert one of the interaction differentness conditions **D** or **E** or **F**. The example interaction mapping is shown in Figure 5.6.

The specific interaction mapping.

specified interaction relations

mapping table S/				specified interaction relations

	A	B	C
D	A,A	B,A	C,A
T/ E	D,A	E,A	F,A
F	B,B	C,B	D,B

out/Y, out/Z

S/A T/D => out/Y/A out/Z/A
S/B T/D => out/Y/B out/Z/A
S/C T/D => out/Y/C out/Z/A
S/A T/E => out/Y/D out/Z/A
S/B T/E => out/Y/E out/Z/A
S/C T/E => out/Y/F out/Z/A
S/A T/F => out/Y/B out/Z/B
S/B T/F => out/Y/C out/Z/B
S/C T/F => out/Y/D out/Z/B
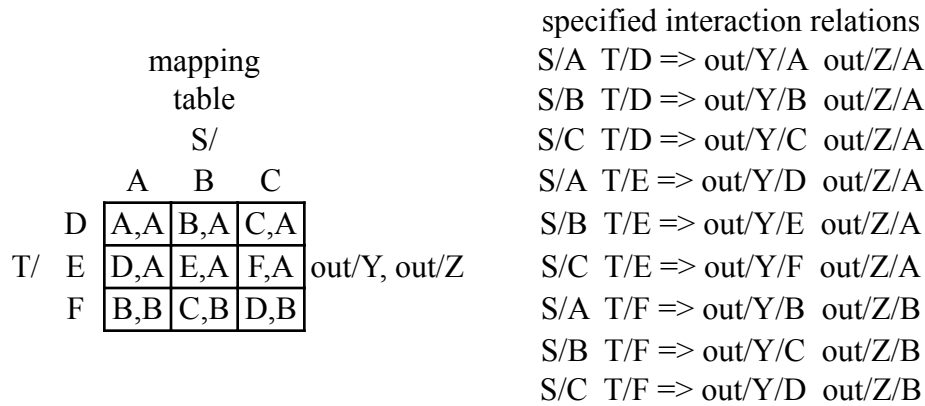
Figure 5.6.   Interaction mapping with six available differentness conditions.

The interaction in is now expressed as a network of two association interaction behaviors asserting the two output conditions. The input conditions asserted by **S** and **T** are fanned out to the two interaction behaviors. The interaction mapping **6cond1** asserts **out/Y** and the interaction mapping **6cond2** asserts **out/Z**. The availability of all possible interaction mappings means that any arbitrary mapping to the output can be represented.

An interaction begins empty with **S/N T/N** and **out/Y/N** and **out/Z/N**. inputs **S/** and **T/** transition to interaction differentness conditions and **out/Y** and **out/Z** transition to the determined interaction differentness conditions. **S/** and **T/** transition back to **N**, and **out/Y** and **out/Z** transition to **N** leaving the interaction network empty bounding one temporal instance of one specific interaction of differentness.
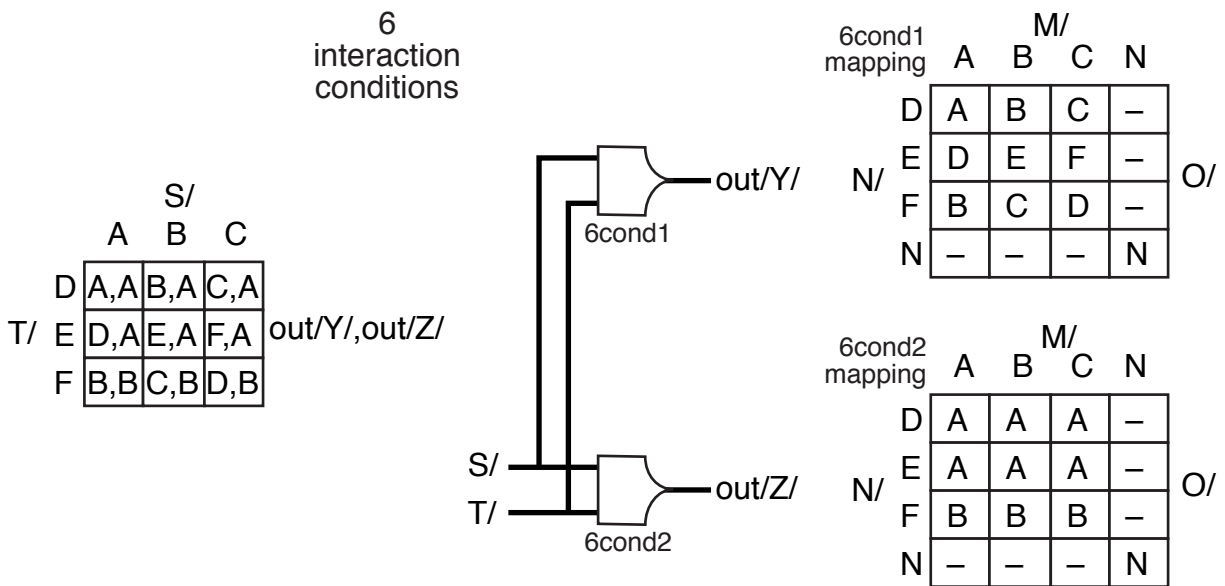
Figure 5.7.   Six differentness conditions with two association interaction behaviors.

The dependency expressions for the component association interaction behaviors and for the network as relations of names of conditions and names of places of association:

(6cond1(M/{C B A N} N/{F E D N} => O/{F E D C B A N})
        O/{A<=[M/A N/A]
          B<={[M/B N/D] [M/A N/F] }
          C<={[M/C N/D] [M/B N/F] }
          D<={[M/A N/E] [M/C N/F] }
          E<=[M/B N/E]
          F<=[M/C N/E]
          N<=[M/N N/N] })

(6cond2(M/{C B A N} N/{F E D N} => O/{F E D C B A N})
        O/{A<={[M/C N/D] [M/B N/D] [M/A N/D]
            [M/C N/E] [M/B N/E] [M/A N/E] }
        B<={[M/C N/F [M/B N/F] [M/A N/F]}
        N<=[M/N N/N] })

(6condexamp(S/{C B A N} T/{F E D N}=> out/[Z Y]/{F E D C B A N})
          6cond1(S T => out/Y)
          6cond2(S T => out/Z)   )

Expressions **6cond1** and **6cond2** express the mappings of the interaction behaviors. The expression **6condexamp** referencing **6cond1** and **6cond2** expresses how the mappings dependently associate as a network to realize the interaction of **S/** and **T/** to **out/[Y Z]/**. The network as a whole becomes the element of interaction that is singularly referencable from interaction to interaction

All possible interaction mappings can be expressed by arranging the output differentnesses in the mapping tables of the interaction behaviors to represent any desired mapping of combined output conditions.

There is less condition differentiation and more association differentiation.

5.6.   Constrained to three available differentness conditions

There are two ways to constrain the expression of condition differentiation. First, the number of available conditions can be limited with the availability of association interaction behaviors capable of representing all the possible interaction mappings. Second, the available interaction mappings can also be limited. These two constraints can be conveniently illustrated and discussed in the context of three available interaction differentness conditions

5.6.1.   First: all possible interaction mappings available

Three interaction differentness conditions named **A**, **B**, **C** are still sufficient to represent each input differentnesses. The nine output differentnesses can still be represented with two association interaction behaviors asserting locality **out** as two component localities **Y** and **Z** which must now be associatively ordered to differentiate **out/Y/A out/Z/B** from **out/Y/B outZ/A**. With six interaction differentness conditions the **Y** and **Z** outputs did not necessarily have to be ordered to distinguish the nine output differentnesses. Input locality **S/** asserts one of the interaction differentness conditions **A** or **B** or **C**. Input locality **T/** asserts one of the interaction differentness conditions **A** or **B** or **C**. The example interaction mapping is shown in Figure 5.8.

mapping
table
S/

	A	B	C
A	A,A	A,B	A,C
T/ B	B,A	B,B	B,C
C	C,A	C,B	C,C

out/Y/, out/Z/

specified interaction relations
S/A T/A => out/Y/A out/Z/A
S/B T/A => out/Y/A out/Z/B
S/C T/A => out/Y/A out/Z/C
S/A T/B => out/Y/B out/Z/A
S/B T/B => out/Y/B out/Z/B
S/C T/B => out/Y/B out/Z/C
S/A T/C => out/Y/C out/Z/A
S/B T/C => out/Y/C out/Z/B
S/C T/C => out/Y/C out/Z/C

Figure 5.8. Interaction mapping with three available differentness conditions.

The interaction in Figure 5.9 is still expressed as a network of two association interaction behaviors asserting the two output conditions. The input conditions asserted by **S** and **T** are fanned out to the two interaction behaviors. The interaction mapping **3cond1** asserts **out/Y** and the interaction mapping **3cond2** asserts **out/Z**. The availability of all possible interaction mappings means that any arbitrary mapping to the output can be represented.

An interaction begins empty with **S/N T/N** and **out/Y/N** and **out/Z/N.** inputs **S/** and **T/** transition to interaction differentness conditions and **out/Y/** and **out/Z/** transition to the determined interaction differentness conditions. **S/** and **T/** transition back to **N**, and **out/Y/** and **out/Z/** transition to **N** leaving the interaction network empty bounding one temporal instance of one specific interaction of differentness.
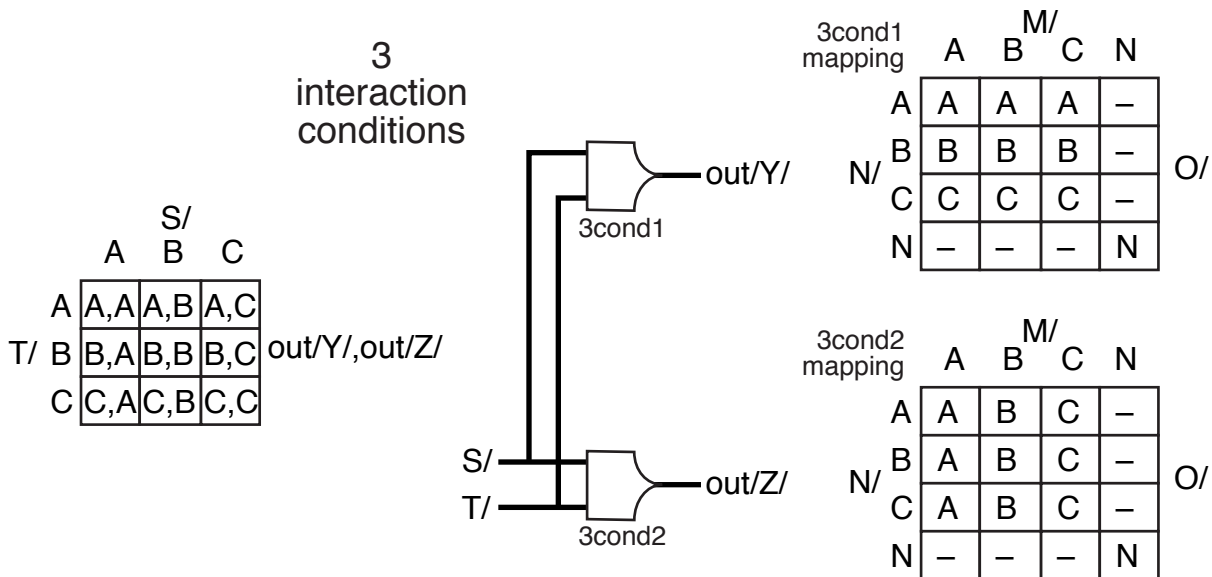
Figure 5.9. Three differentness conditions with two association interaction behaviors.

The dependency expressions for the component association interaction behaviors and for the network as relations of names of conditions and names of places of association.

(3cond1(M/{C B A N} N/{C B A N} => O/{C B A N})
        O/{A<={[M/A N/A] [M/A N/B] [M/A N/C]}
          B<={[M/B N/A] [M/B N/B] [M/B N/C]}
          C<={[M/C N/A] [M/C N/B] [M/C N/C]}
          N<=[M/N N/N] })

(3cond2(M/{C B A N} N/{C B A N} => O/{C B A N})
        O/{A<={[M/A N/A] [M/B N/A] [M/C N/A]}
          B<={[M/A N/B] [M/B N/B] [M/C N/B]}
          C<={[M/A N/C] [M/B N/C] [M/C N/C]}}
          N<=[M/N N/N] })

(3condexamp1(S/{C B A N} T/{C B A N}=> out/[Z Y]/{C B A N})
        3cond1(S T => out/Y)
        3cond2(S T => out/Z)   )

Expressions **3cond1** and **3cond2** express the mappings of the interaction behaviors that realize the interaction mapping. The expression **3condexamp** referencing **3cond1** and **3cond2** expresses how the mappings dependently associate to realize the interaction of **S/** and **T/** to **out/ [Y Z]**/.

Again, all possible interaction mappings can be expressed by arranging the output differentnesses in the mapping tables of the association interaction behaviors to represent any desired mapping to the associated output conditions.

There is less condition differentiation and more association differentiation in that now the association differentness of **S** and **T** are necessary to differentiate the presented inputs and **out/X** and **out/Y** must be associationaly ordered to differentiate **out/Y/A out/Z/B** from **out/Y/B outZ/ A**.

5.6.2.   Second: only five interaction mappings available

Condition differentiation is further constrained by limiting the available association behavior interaction mappings to only five given mappings that cannot be rearranged. The example interaction mapping remains identical to Figure 5.8 but realizing the mapping behavior changes dramatically.

To this point in the walk along the spectrum the ability to map any presented input to any desired output has been supported by the availability of black boxes of all possible association behavior interaction mappings of the available differentness conditions. How the recognition of the presented input and the mapping to assertion of the corresponding output happens within the black box of the association interaction behavior has not been expressed. With the limitation of interaction mappings the internal mechanics of this black box mapping are no longer universally available for all possible mappings but only for the limited mappings. Now the generality of mapping behavior must be explicitly expressed in terms of association relations among the limited available mappings.

5.6.2.1.   Recognition of presented input.

The recognition of presented input differentness is accomplished with two association behavior mappings, Equality and Rotate shown in Figure 5.10.

The **Equality** behavior expresses the recognition of a specific presentation of interaction differentness conditions, **CC**. If the input to **Equality** is **CC** the behavior transitions its output to condition **C** indicating equality success. For any other combination of presented interaction differentness conditions it transitions its output to condition **A** indicating equality failure.

The **Rotate** behavior rotates a presented input interaction differentness condition to a next interaction differentness condition in order. A sequence of rotates will rotate a presented input interaction differentness condition through all of the possible interaction differentness conditions. One of these rotations will output the specific equality interaction differentness condition **C**.

Condition **N** plays directly through the **Rotate** behavior and condition **NN** plays directly through the **Equality** behavior.
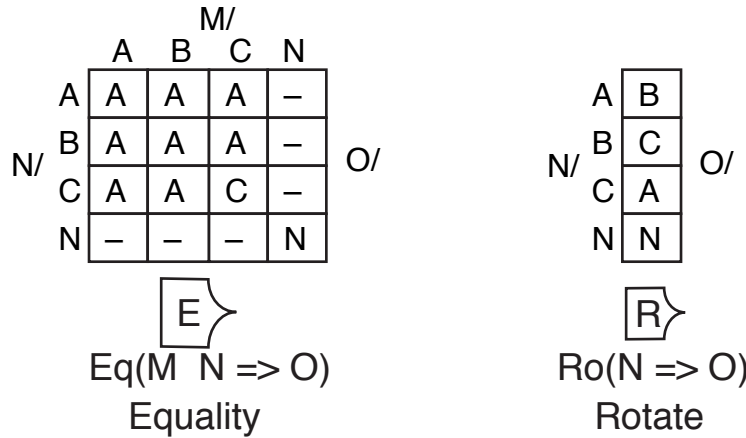
Figure 5.10. The interaction mappings for equality and rotate behaviors.

The interaction mapping expressions.

(Eq(M/{C B A N} N/{C B A N} => O/{C B A N})
        O/{A<={[M/A N/A] [M/A N/B] [M/A N/C]
            [M/B N/A] [M/B N/B] [M/B N/C]
            [M/C N/A] [M/C N/B] [M/C N/C]}
      C<= [M/C N/C]
      N<=[M/N N/N] })

(Ro(N/{C B A N} => O/{C B A N})
        O/{A<= N/C
      B<=N/A]
      C<=N/B]
      N<=N/N)

5.6.2.2.   The cross association recognition network

The presented input condition can be recognized by how many rotations it takes to rotate the input condition to the specific equality condition **C**. Condition **C** after two rotations means that the input condition was **A**. Condition **C** after one rotation means that the input condition was **B**. Condition **C** after no rotation means that the input condition was **C**. For each input each rotated condition will be asserted at a different place of association. One of the places will assert **C**

indicating the presented input condition. The rotation places of association are cross associated. In the cross association of the rotation places there will be one occurrence of **CC** identifying the presented input conditions. One **Equality** behavior in a rank of **Equality** behaviors will recognize the one occurrence of **CC** and transition its output to **C**. The remaining eight **Equality** behaviors will transition their output to **A** indicating recognition failure. Presentation **S/N** and **T/N** is recognized with all nine **Equality** behaviors asssserting **N**.

   The cross association recognition portion of the interaction network is illustrated in Figure 5.11. An interaction begins with an empty network with **S/N** and **T/N** presented as input and all of the association interaction behaviors asserting **N**. S/**A** and T/**B** are presented to the network input. The three conditions are represented in color showing the flow of presented conditions through the rotations, into the rank of equality behaviors and to the assertion of recognition condition **C**. The specific presentation of conditions is recognized at a particular place of association in the network. After the interaction the input presentation transitions to **S/N** and **T/N** and the network empties of interaction conditions.
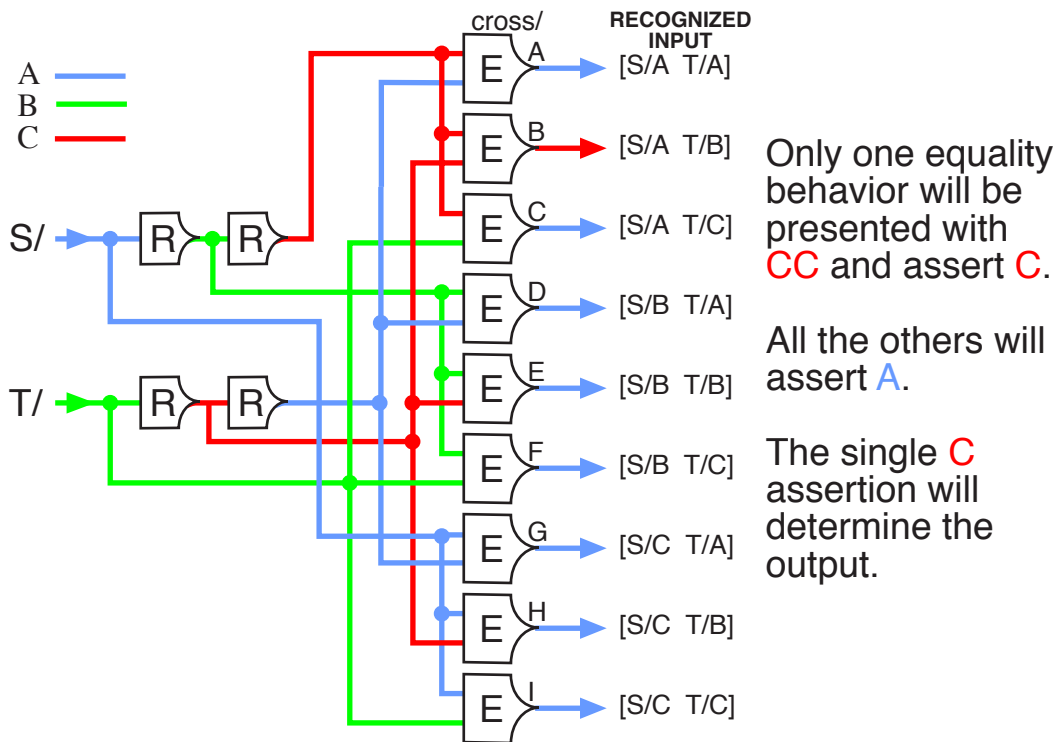
Figure 5.11. Cross association recognition network for example interaction with three differentness conditions and limited behavior mappings.

The cross association recognition portion of the interaction dependency expression.

(3condexamp2(S/{C B A N} T/{C B A N} => OUT/[Y Z]/{C B A N})
        (cross/[A B C D E F G H I]/{C B A N})
/* cross association recognition. Only one transitions to C the other 8 transition to A.*/
/* When both inputs are N all of the equality behaviors will transition to N. */
        cross/[A<=Eq(Ro(Ro(S)) Ro(Ro(T))) /* reocgnize S/A T/A */
              B<=Eq(Ro(Ro(S)) Ro(T))   /* reocgnize S/A T/B */
              C<=Eq(Ro(Ro(S)) T)   /* reocgnize S/A T/C */
              D<=Eq(Ro(S) Ro(Ro(T)))   /* reocgnize S/B T/A */
              E<=Eq(Ro(S) Ro(T))   /* reocgnize S/B T/B */
              F<=Eq(Ro(S) T)   /* reocgnize S/B T/C */
              G<=Eq(S Ro(Ro(T)))   /* reocgnize S/C T/A */
              H<=Eq(S Ro (T))   /* reocgnize S/C T/B */
              I<=Eq(S T)]...   /* reocgnize S/C T/C */......)

5.6.2.3.   Assertion of the mapped output

The rank of **equality** behaviors outputs one **C** and eight **A**s. The task is to have the one **C** condition assert the appropriate output condition by prioritizing the flow of **B**s and **C**s over the flow of **A**s to the output assertion. This is accomplished with two **Assert** behaviors and one **Priority** behavior, shown in Figure 5.12.

If the **AssertB** behavior is presented with condition **C** it will output condition **B** but if presented with condition **A** will output condition **A**.

If the **AssertA** behavior is presented with condition **C** it will output condition **A** and if presented with condition **A** will output condition **A**. If the one **C** condition is converted to **A** there are no **B**s or **C**s converging to the output. There are only **A**s converging and the output will be **A**.

Condition **C** does not need to be transitioned into condition **C** so there is no **AssertC** behavior.

The **priority** behavior will pass the highest priority condition with **A** the lowest priority **B** the middle and **C** the highest priority. A network of priority behaviors will pass the highest priority condition presented to the network.
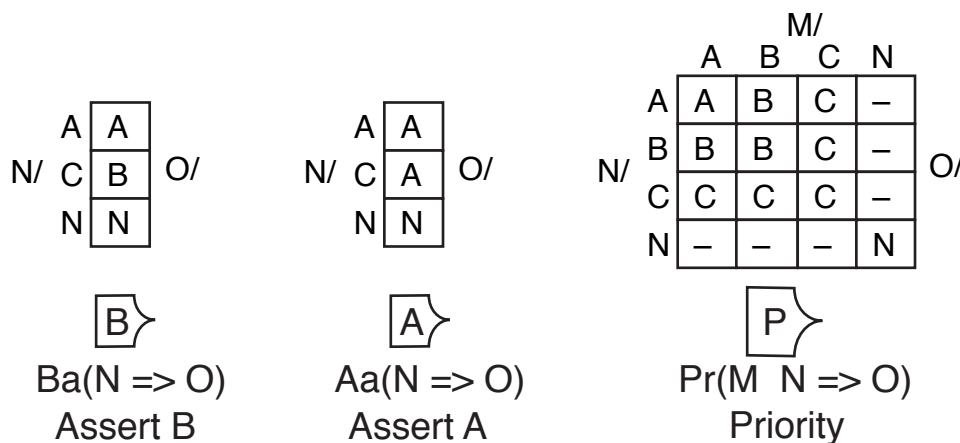
Figure 5.12. Output convergence behaviors.

The interaction mapping expressions:

(Pr(M/{C B A N} N/{C B A N}/ => O/{C B A N})
         O/{A<=[M/A N/A]
           B<={[M/A N/B] [M/B N/A] [M/B N/B]}
           C<={[M/C N/A] [M/C N/B] [M/C N/C] [M/A N/C] [M/B N/C]}
   **N<=[M/N N/N] })**

(Aa(N/{C B A N} => O/{C B A N})
         O/{A<= {N/A N/C}
           N<=N/N })

(Ba(N/{C B AN} => O/{C B A N})
         O/{A<= N/A
           B<=N/C
           N<=N/N })

5.6.2.4. The convergence assertion network

The one condition **C** of the cross association recognition determines the output conditions. Each recognized input that specifies output condition **B** the recognizing **C** condition flows through an **AssertB** behavior. Each recognized input that specifies output condition **A** the recognizing **C** condition flows through an **AssertA** behavior. Each recognized input that specifies output condition **C** is not transitioned because the recognition success condition is already **C**. The result of the assert behaviors is eight **A** conditions and one asserted output condition which may also be condition **A**. If all nine recognition conditions are condition N then **out/Y** will be condition **N**.

In Figure 5.13 all nine input recognitions flowing through locality **cross/** are presented to a priority network that converges the flows to the assertion of **out/Y**. The three condition are represented in color showing the flow of presented conditions through the assertion and priority behaviors to **out/Y**.
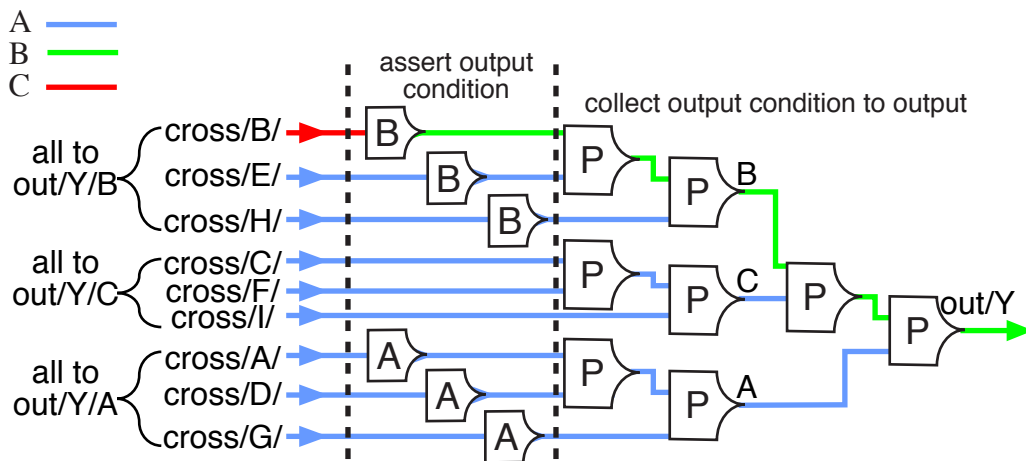
Figure 5.13. Three condition output convergence network to assert out/Y.

Assertion convergence dependency expression:

out/Y<= Pr(Pr(Pr(Pr(Ba(cross/B) Ba(cross/E)) Ba(cross/H))
Pr(Pr(cross/C cross/F) cross/I))
Pr(Pr(Aa(cross/A) Aa(cross/D)) Aa(cross/G)))

5.6.2.5.   The complete network realizing the example interaction.

Figure 5.14 shows the association network expressing the example interaction with three differentness conditions **A**, **B** and **C** and the set of five interaction behaviors **Ro(…)**, **Eq(…)**, **Ba(…)**, **Aa(…)** and **Pr(…)** .

An interaction begins with the network empty of interaction condition with **S/N T/N,** all the internal behaviors asserting **N** and **out/Y/N** and **out/Z/N.** The inputs transition to **S/B** and **T/C**. The three condition are represented in color showing the flow of presented conditions through the rotations, into the rank of equality behaviors and through the convergence network to **out/Y** and **out/Z**. **S/** and **T/** transition to **N**, all internal behaviors transition to N and **out/Y** and **out/Z** transitions to **N** leaving the interaction network empty with completely **N** bounding one temporal instance of one specific interaction of differentness.

The flow through the network is a collaboration between differentness of condition, differentness of interaction behavior and differentness of place of association in the network.
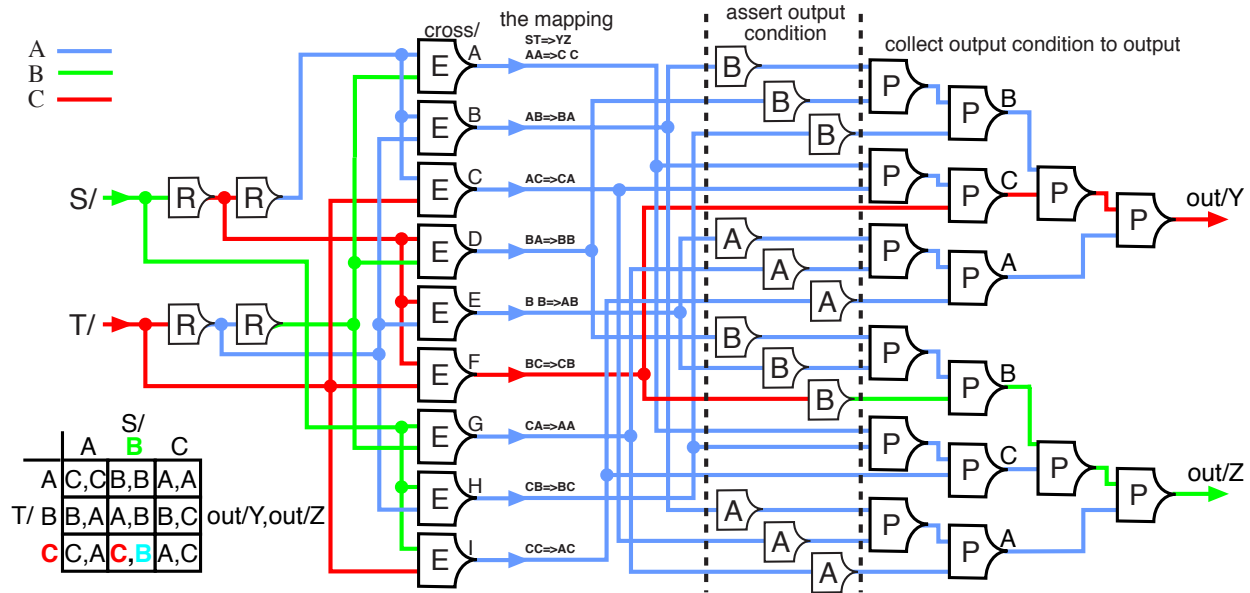
Figure 5.14. Three differentness conditions with five interaction mappings realizing the example interaction with 41 behaviors.

The complete interaction network expression.

(3condexamp2(S/{C B A N} T/{C B A N} => OUT/[Y Z]/{C B A N})
      (cross/{A B C D E F G H I}/{C B A N})
/* cross association recognition. Only one transitions to C the other 8 transition to A.*/
/* When both inputs are N all of the equality behaviors will transition to N. */
            cross/[A<=Eq(Ro(Ro(S)) Ro(Ro(T))) /* reocgnize S/A T/A */
                  B<=Eq(Ro(Ro(S)) Ro(T))   /* reocgnize S/A T/B */
                  C<=Eq(Ro(Ro(S)) T)   /* reocgnize S/A T/C */
                  D<=Eq(Ro(S) Ro(Ro(T)))   /* reocgnize S/B T/A */
                  E<=Eq(Ro(S) Ro(T))   /* reocgnize S/B T/B */
                  F<=Eq(Ro(S) T)   /* reocgnize S/B T/C */
                  G<=Eq(S Ro(Ro(T)))   /* reocgnize S/C T/A */
                  H<=Eq(S Ro (T))   /* reocgnize S/C T/B */
                  I<=Eq(S T)]…   /* reocgnize S/C T/C */…
/* The convergence to out/Y */
      out/Y<= Pr(Pr(Pr(Pr(Ba(cross/B) Ba(cross/E)) Ba(cross/H)) /* assert Y/B */
                  Pr(Pr(cross/C cross/F) cross/I))   /* assert Y/C */
                  Pr(Pr(Aa(cross/A) Aa(cross/D)) Aa(cross/G))) /* assert Y/A */
/* The convergence to out/Z */
      out/Z<= Pr(Pr(Pr(Pr(Ba(cross/D) Ba(cross/E)) Ba(cross/F)) /* assert Z/B */
                  Pr(Pr(cross/A cross/H) cross/I))   /* assert Z/C */
                  Pr(Pr(Aa(cross/B) Aa(cross/C)) Aa(cross/G))) /* assert Z/A */)

All possible interaction mappings are now expressed by restructuring the network in terms of the limited mappings. There is less condition differentiation and considerably more association differentiation.

5.7.   Constrained to two available differentness conditions

Constrained to two available interaction differentness conditions named **A** and **B** the inputs with three differentnesses and the output with nine differentnesses have to be represented with two interaction differentness conditions and four interaction differentness conditions respectively as shown in Figure 5.15.

The interaction mapping.

mapping
table

S/Y, S/Z

|	A,A	A,B	B,A	
A,A	A,A,A,A	A,A,A,B	A,A,B,A	
A,B	A,A,B,B	A,B,A,A	A,B,A,B	out/W, out/X, out/Y/, out/Z/
B,A	A,B,B,A	A,B,B,B	B,A,A,A	

T/Y, T/Z is the row label on the left.

specified interaction relations

```
S/Y/A S/Z/A T/Y/A T/Z/A => out/Y/A out/Z/A out/Y/A out/Z/A
S/Y/A S/Z/B T/Y/A T/Z/A => out/Y/A out/Z/A out/Y/A out/Z/B
S/Y/B S/Z/A T/Y/A T/Z/A => out/Y/A out/Z/A out/Y/B out/Z/A
S/Y/A S/Z/A T/Y/A T/Z/B => out/Y/A out/Z/A out/Y/B out/Z/B
S/Y/A S/Z/B T/Y/A T/Z/B => out/Y/A out/Z/B out/Y/A out/Z/A
S/Y/B S/Z/A T/Y/A T/Z/B => out/Y/A out/Z/B out/Y/A out/Z/B
S/Y/A S/Z/A T/Y/B T/Z/A => out/Y/A out/Z/B out/Y/B out/Z/A
S/Y/A S/Z/B T/Y/B T/Z/A => out/Y/A out/Z/B out/Y/B out/Z/B
S/Y/B S/Z/A T/Y/B T/Z/A => out/Y/B out/Z/A out/Y/A out/Z/A
```

Figure 5.15. Interaction mapping with two available differentness conditions.

5.7.1.   Two differentness condition interaction behavior mappings

With only two interaction differentness conditions **B** becomes the Equality condition and **AssertB** is no longer needed. The limited interaction mappings become the four association interaction behaviors in Figure 5.16.
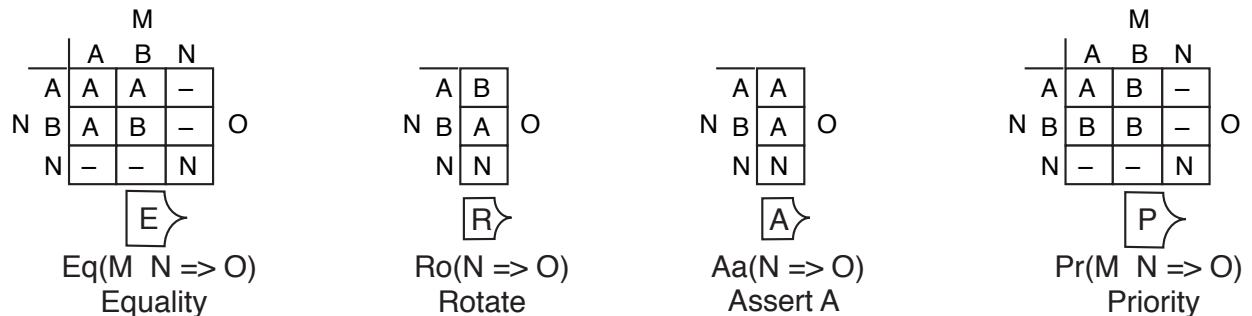
Figure 5.16. Two differentness condition association interaction behaviors.

The component behavior expressions:

(Eq(M/{B A N} N/{B A N} => O/{B A N})

O/{A<=\{[M/A N/A] [M/A N/B] [M/B N/A]}

B<= [M/B N/B]

N<=[M/N N/N] })

(Ro(N/{B A N} => O/{B A N})
    O/{A<= N/B
     B<=N/A
     N<=N/N })

(Aa(N/{B A N} => O/{B A N})
    O/{A<= {N/A N/B}
     N<=N/N })

(Pr(M/{B A N} N/{B A N}/ => O/{B A N})
    O/{A<=[M/A N/A]
     B<= {[M/A N/B] [M/B N/A] [M/B N/B]}
     N<=[M/N N/N] })

5.7.2. Recognition of presented input

The differentness localities **S** and **T** are each represented as two places of association **S/Y/**, **S/Z/** and **T/Y/**, **T/Z/** extending the two condition differentnesses to cover the three input interaction differentnesses. With the input differentnesses represented with two localities two stages of cross association recognition are required. In the left half of Figure 5.17 each input locality is expanded with **rotate** behavior presented to a rank of three **equality** behaviors only one of which will assert **B** and the rest will assert **A**. The two resulting localities of asserted conditions are then cross associated and only one cross association will be the specific recognition presentation of **BB**. One of the rank of **equality** behaviors will recognize the one cross association presenting **BB** asserting its output condition **B** indicating recognition success with the other eight **equality** behaviors assert output condition **A** indicating recognition failure.
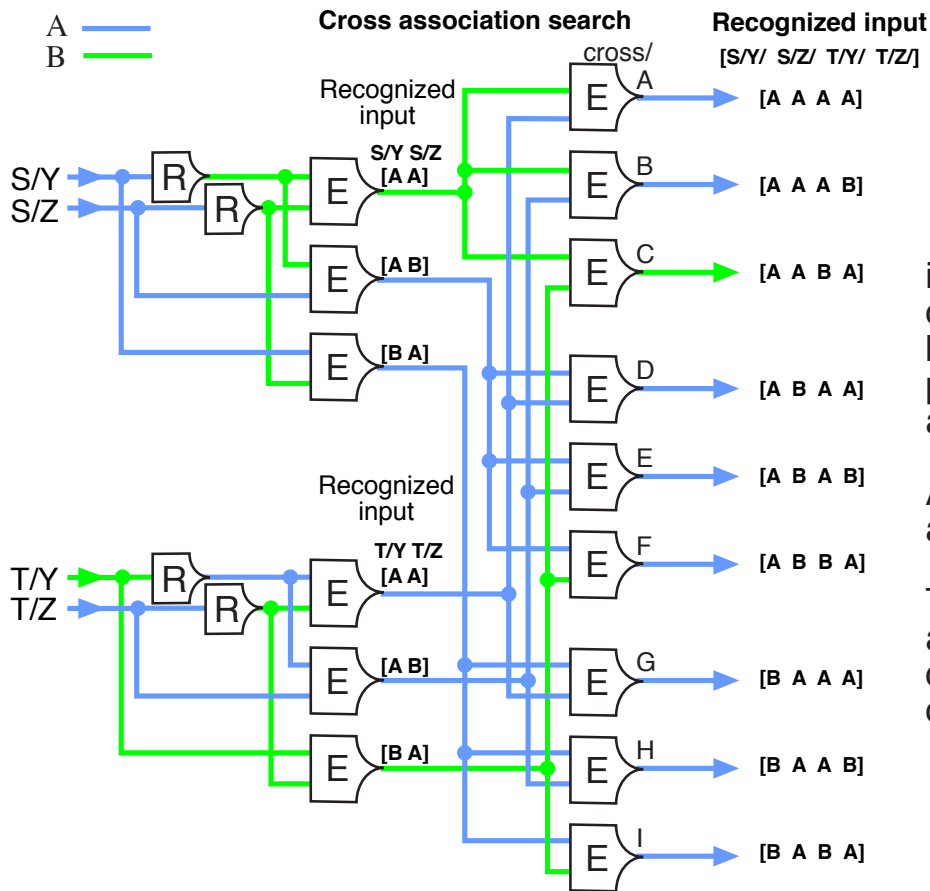
Figure 5.17. Cross association recognition network for example interaction with two differentness conditions.

The recognition portion of the interaction expression:

(2condexamp(S/[Y Z]/{B A N} T/[Y Z]/{B A N} => OUT/[W X Y Z]/{B A N})

    (cross/[A B C D E F G H I]/{B A N} Scross/[A B C]/{B A N}

      Tcross/[A B C]/{B A N})

        Scross/[A<=Eq(Ro(S/Y) Ro(S/Z))   /* reocgnize S/Y/A S/Z/A */

                B<=Eq(Ro(S/Y) S/Z)   /* reocgnize S/Y/A S/Z/B */

                C<=Eq(S/Y Ro(S/Z))]   /* reocgnize S/Y/B S/Z/A */

        Tcross/[A<=Eq(Ro(T/Y) Ro(T/Z))   /* reocgnize T/Y/A T/Z/A */

                B<=Eq(Ro(T/Y) T/Z)   /* reocgnize T/Y/A T/Z/B */

                C<=Eq(T/Y Ro(T/Z))]   /* reocgnize T/Y/B T/Z/A */

/* cross association recognition. Only one transitions to B the other 8 transition to A.*/

/* When the input is all N all of Scross, all of Tcross and all of cross transition to N */

    cross/[A<=E(Scross/A Tcross/A)   /* reocgnize S/Y/A S/Z/A T/Y/A T/Z/A */

      B<=Eq(Scross/A Tcross/B)   /* reocgnize S/Y/A S/Z/A T/Y/A T/Z/B */

      C<=Eq(Scross/A Tcross/C)   /* reocgnize S/Y/A S/Z/A T/Y/B T/Z/A */

      D<=Eq(Scross/B Tcross/A)   /* reocgnize S/Y/A S/Z/B T/Y/A T/Z/A */

      E<=Eq(Scross/B Tcross/B)   /* reocgnize S/Y/A S/Z/B T/Y/A T/Z/B */

      F<=Eq(Scross/B Tcross/C)   /* reocgnize S/Y/A S/Z/B T/Y/B T/Z/A */

      G<=Eq(Scross/C Tcross/A)   /* reocgnize S/Y/B S/Z/A T/Y/A T/Z/A */

      H<=Eq(Scross/C Tcross/B)   /* reocgnize S/Y/B S/Z/A T/Y/A T/Z/B */

      I<=Eq(Scross/C Tcross/C)] /* reocgnize S/Y/B S/Z/A T/Y/B T/Z/A */

  …...)

5.7.3.   Assertion of the mapped output

    The one condition **B** of the recognition determines each output through a **priority** convergence network in which the recognition results are first **priority** collected and then presented to the **AssertA** behavior just before the final **priority** interaction behavior that asserts the output condition.

    The one **B** condition and the eight **A** conditions from the cross association recognition will pass through the convergence network and set **out/Z/A** or **out/Z/B**. When all nine inputs are **N** the output will be **out/Z/N**.
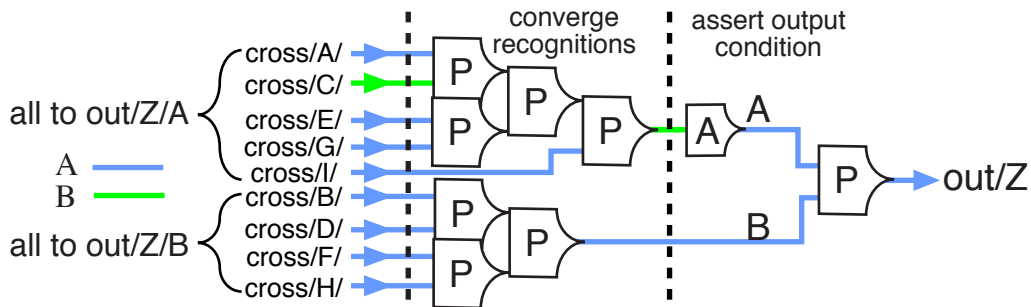

Figure 5.18. Two condition output convergence network to assert out/Z.

Output convergence portion of interaction dependency expression:

out/Z<= Pr(Aa(Pr(Pr(Pr(cross/A cross/C) Pr(cross/E cross/G)) cross/I))
Pr(Pr(cross/B cross/D) Pr(cross/F cross/H)))

5.7.4.   The complete network realizing the example interaction.

An interaction begins as an empty network with **S/Y/N**, **S/Z/N**, **T/Y/N**, **T/Z/N** all internal behaviors asserting **N** and **out/W/N**, **out/X/N**, **out/Y/N**, **out/Z/N.** Inputs **S/Y/A**, **S/Z/A**, **T/Y/B**, **Y/Z/A** transition to interaction differentness conditions and the output localities **out/W/A**, **out/X/B**, **out/Y/B**, **out/Z/A** transitions to the determined interaction differentness conditions. **S/Y**, **S/Z**, **T/Y**, **Y/Z** transition back to **N**, and **out/W**, **out/X**, **out/Y**, **out/Z** transitions to **N** leaving the interaction network empty of interaction differentness conditions bounding one temporal instance of one specific interaction of differentness.
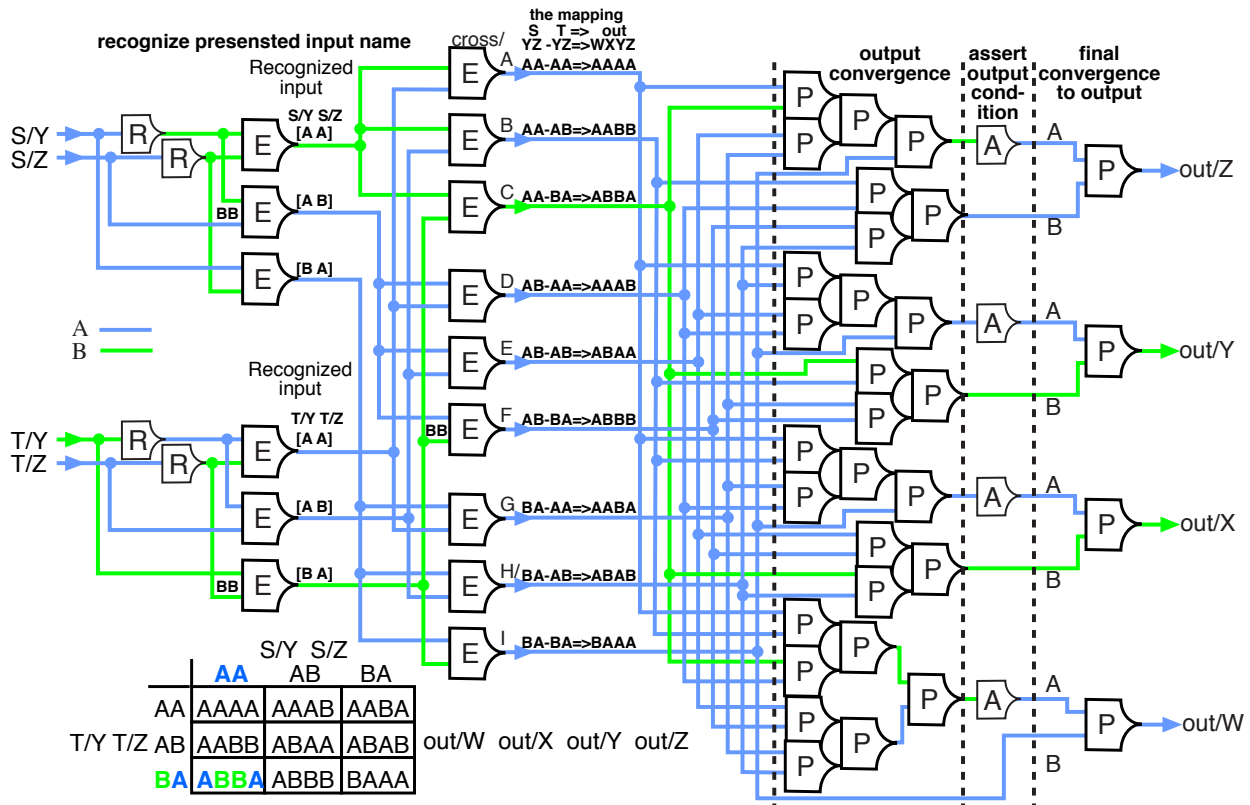
Figure 5.19. Two interaction differentness conditions with four interaction mappings realizing the example interaction with 55 behaviors.

The complete interaction dependency expression:

(2condexamp(S/[Y Z]/{B A N} T/[Y Z]/{B A N} => OUT/[W X Y Z]/{B A N})
　　　(cross/[A B C D E F G H I]/{B A N}/
　　　　　Scross/[A B C]/{B A N} Tcross/[A B C]/{B A N})
　　　　　Scross/[A<=Eq(Ro(S/Y) Ro(S/Z))　　　　/* reocgnize S/Y/A S/Z/A */
　　　　　　　B<=Eq(Ro(S/Y) S/Z)　　　　　　/* reocgnize S/Y/A S/Z/B */
　　　　　　　C<=Eq(S/Y Ro(S/Z))]　　　　　/* reocgnize S/Y/B S/Z/A */
　　　　　Tcross/[A<=Eq(Ro(T/Y) Ro(T/Z))　　　/* reocgnize T/Y/A T/Z/A */
　　　　　　　B<=Eq(Ro(T/Y) T/Z)　　　　　　/* reocgnize T/Y/A T/Z/B */
　　　　　　　C<=Eq(T/Y Ro(T/Z))]　　　　　/* reocgnize T/Y/B T/Z/A */
/* cross association recognition. Only one transitions to B the other 8 transition to A.*/
/* When the input is all N all of Scross, all of Tcross and all of cross transition to N */
　　　cross/[A<=Eq(Scross/A Tcross/A)/* reocgnize S/Y/A S/Z/A T/Y/A T/Z/A */
　　　　　B<=Eq(Scross/A Tcross/B)　/* reocgnize S/Y/A S/Z/A T/Y/A T/Z/B */
　　　　　C<=Eq(Scross/A Tcross/C)　/* reocgnize S/Y/A S/Z/A T/Y/B T/Z/A */
　　　　　D<=Eq(Scross/B Tcross/A)　/* reocgnize S/Y/A S/Z/B T/Y/A T/Z/A */
　　　　　E<=Eq(Scross/B Tcross/B)　/* reocgnize S/Y/A S/Z/B T/Y/A T/Z/B */
　　　　　F<=Eq(Scross/B Tcross/C)　/* reocgnize S/Y/A S/Z/B T/Y/B T/Z/A */
　　　　　G<=Eq(Scross/C Tcross/A)　/* reocgnize S/Y/B S/Z/A T/Y/A T/Z/A */
　　　　　H<=Eq(Scross/C Tcross/B)　/* reocgnize S/Y/B S/Z/A T/Y/A T/Z/B */
　　　　　I<=Eq(Scross/C Tcross/C)]　/* reocgnize S/Y/B S/Z/A T/Y/B T/Z/A */
/* assert out/Z */
　　　out/Z<= Pr(Aa(Pr(Pr(Pr(cross/A cross/C) Pr(cross/E cross/G)) cross/I))
　　　　　　　　　　Pr(Pr(cross/B cross/D) Pr(cross/F cross/H)))
/* assert out/Y */
　　　out/Y<= Pr(Aa(Pr(Pr(Pr(cross/A cross/D) Pr(cross/E cross/H)) cross/I))
　　　　　　　　　　Pr(Pr(cross/B cross/C) Pr(cross/F cross/G)))
/* assert out/X */
　　　out/X<= Pr(Aa(Pr(Pr(Pr(cross/A cross/B) Pr(cross/D cross/G)) cross/I))
　　　　　　　　　　Pr(Pr(cross/C cross/C) Pr(cross/F cross/H)))
/* assert out/W */
　　　out/W<= Pr(Aa(Pr(Pr(Pr(cross/A cross/B) Pr(cross/C cross/D))
　　　　　　　　　　Pr(Pr(cross/F cross/G) Pr(cross/H cross/G))))
　　　　　　　cross/I)　)

There is less condition differentiation and more association differentiation.

### 5.7.5.　Two differentness condition interaction with extrinsic coordination
With an extrinsic coordinating behavior such as a mathematician or timing analysis and a clock the **N** condition is no longer needed and the **AssertA** behavior, which becomes a default, is

no longer needed. Two differentness condition extrinsically coordinated interaction behaviors reduce to Boolean logic as shown in Figure 5.20.
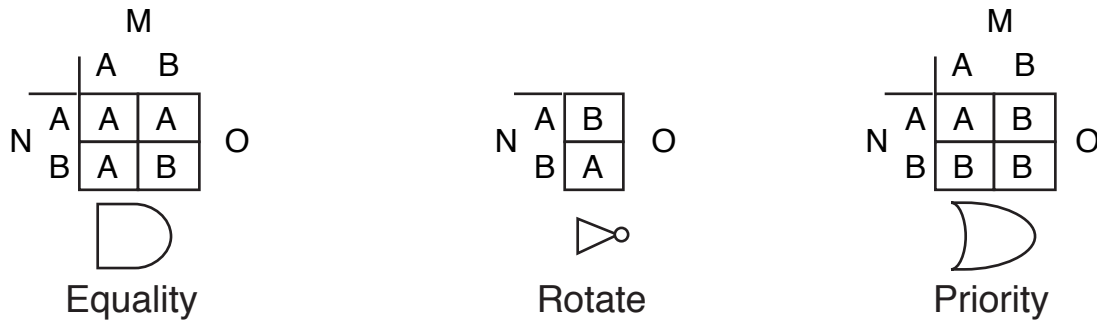
Figure 5.20. Two condition interaction behaviors assuming extrinsic coordination.

An extrinsically coordinated network can make assumptions about behavior such as that after a specific time interval all interactions within the association network have occurred. With only two possibilities and after the interval one possibility was not successful it can be assumed by default that the other possibility was successful. At a specified time or at the discretion of the mathematician a so called truth function will be asserting T indicating its own truth or it will be asserting F indicating by default the alternate truth. The converge network for each output with default behavior reduces to Figure 5.21.
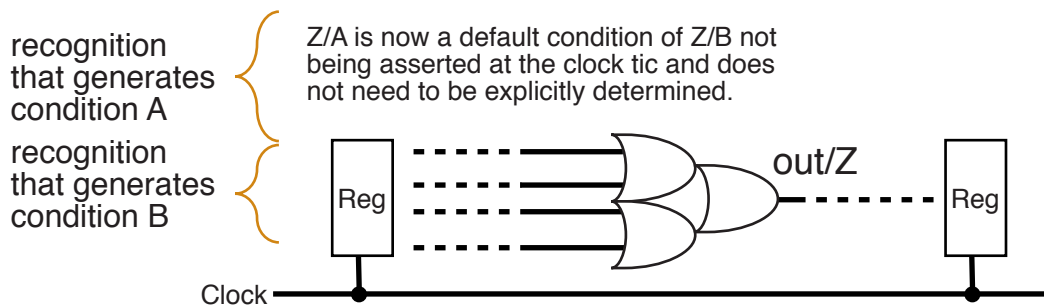
Figure 5.21. Extrinsically coordinated output convergence.

Also, with the truth function comes the "don't care condition. If a truth function can perform its duty by ignoring a related variable then it does not care what the condition of the variable might be. With extrinsic control logical completeness is no longer necessary.

Figure 5.21 seems to be much more efficient than Figure 5.18 but it comes with the conceptual cost of an incomplete logic and a logically extrinsic critical timing requirement and timing referent. Extrinsic coordination is a crutch propping up an insufficient logic.

5.8.   Constrained to one differentness condition: Pure association differentiation

Constrained to one interaction differentness condition named **A** condition interaction differentiation ceases to exist and all interaction differentiation is in terms of differentness of place of association. The mapping is now entirely in terms of names of places of association rather than in terms of names of conditions. The collaboration of condition differentnesses **A** and **N** reduce to serving only to assert or not assert the differentness of a place of association.

mapping
table

	U	V	W
X	A	B	C
Y	D	E	F
Z	J	K	L

specifies
interaction→
relations

W Z => L
V Z => K
U Z => J
W Y => F
V Y => E
U Y => D
W X => C
V X => B
U X => A

There is no longer any need for the **Rotate** behavior or the **Assert** behavior. There are no interactions of different conditions. There are only interactions of two or more **A**s which can be determined with a threshold. The Equality behavior reduces to an "all of" behavior. The priority behavior reduces to a "one of" behavior. Replacing **A** with **D** the behaviors of Figure 5.22 correspond to the "all of" and "one of" behaviors of Section 3.2. For continuity this chapter will continue using **A** as the only differentness condition. These primitive behaviors do not have reference names but are referenced syntactically with [ ] and { }.
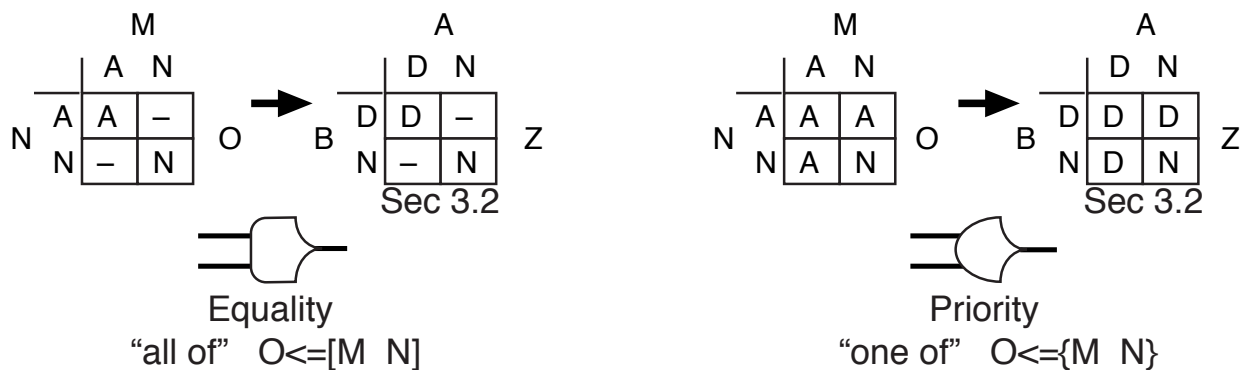
Figure 5.22. One differentness condition association interaction behaviors.

Mutual exclusivity, previously represented by asserting one at a time of two or more differentness conditions at a single place of association is now represented with two or more places of association only one of which, at a time, will transition to **A** with the rest remaining at **N**. In Figure 5.23 the locality named **S** consists of three places of association named **U**, **V** and **W** only one of which, at a time, will transition to **A**. The locality named **T** consists of three places of association named **X**, **Y** and **Z** only one of which, at a time, will transition to **A**. The locality named **out** consists of nine places of association named **A**, **B**, **C**, **D**, **E**, **F**, **G**, **H** and **I** only one of which, at a time, will transition to **A**.

An interaction will begin with the presented input at **N** and with all behaviors asserting **N**. Only one of **U**, **V** or **W** and one of **X**, **Y** or **Z** will transition to **A.** The other inputs will remain at **N**. Only one of the **equality** behaviors will be presented with input completeness **AA** and transition its output to **A**. The other eight **equality** behaviors will be presented with incomplete input and will not transition their outputs which will remain asserting **N**. When the presented inputs transition back to all **N** the one **equality** behavior will transition its output to **N** and all of the **equality** behaviors will be asserting **N** leaving the interaction network empty and bounding

one temporal instance of one specific interaction of differentness. Figure 5.23 illustrates the transition of input differentnesses to **S/U/A** and **T/Y/A**.

Two mutually exclusive differentnesses are mutually inclusively cross associated to produce one mutually exclusive differentness.
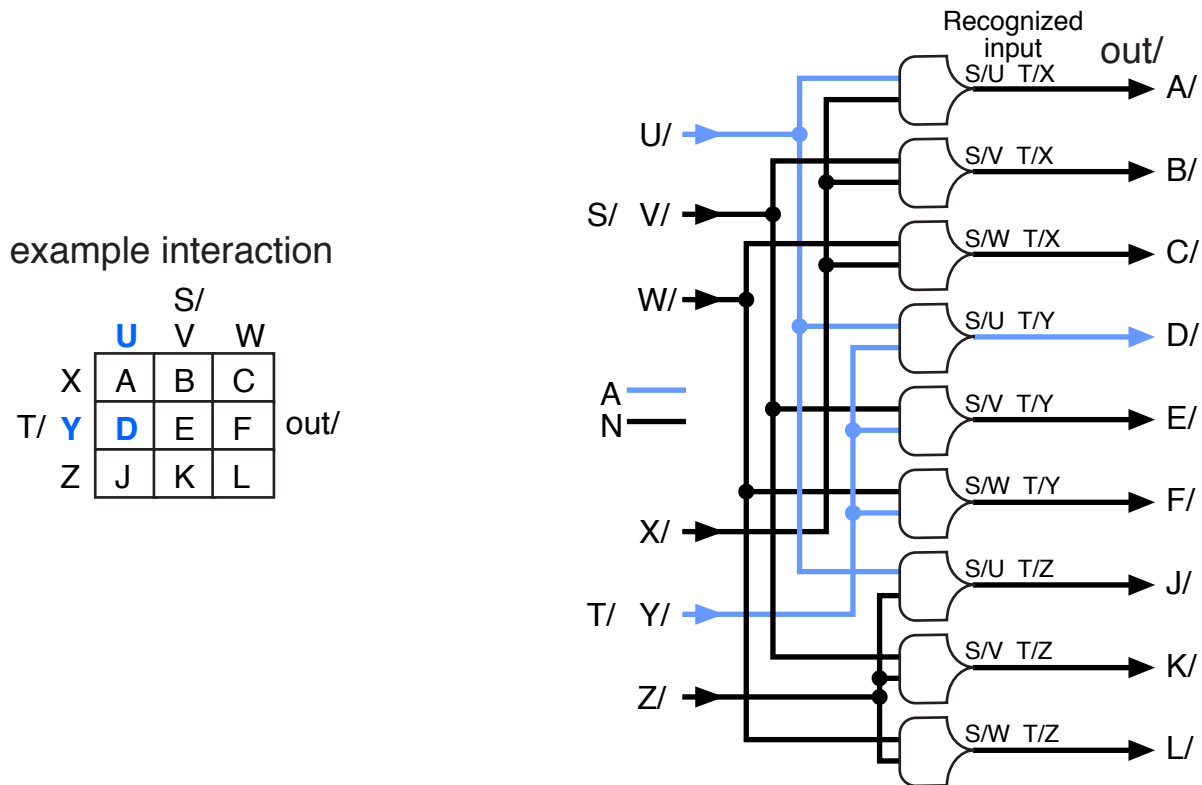

Figure 5.23. One differentness condition with two interaction mappings realizing the example interaction with 9 behaviors.

The complete interaction expression:

(Onecondexamp1 (S/{U,V,W}/{A N}, T/{X,Y,Z}/{A N} -> out/{A,B,C,D,E,F,J,K,L}/{A N});
        out/{A<=[S/U T/X]
            B<=[S/V T/X]
            C<=[S/W T/X]
            D<=[S/U T/XY]
            E<=[S/V T/Y]
            F<=[S/W T/Y]
            G<=[S/U T/Z]
            H<=[S/V T/Z]
            I<=[S/W T/Z] }   )

There is no condition interaction differentiation and all interaction differentiation is association differentiation.

5.8.1.1.   Generality of mapping

The rank of **"all of"** behaviors is the cross association recognition of presented input. The one transition to **A** can realize any mapping to any output locality of fewer than nine differentnesses with convergence through **"one of"** behaviors as illustrated in Figure 5.24. The places of association are renamed with **A**, **B** and **C** to illustrate the differentness of place of association. Differentnesses **S/B/A T/B/A** and **out/B/A** all asserting **/B/A** are all different by virtue of the differentness of the localities of association of **S**, **T** and **out**.

Figure 5.24 illustrates the presentation transition to **S/C/A** and **T/A/A** leading to the assertion of output differentness **out/C/A**.

example interaction
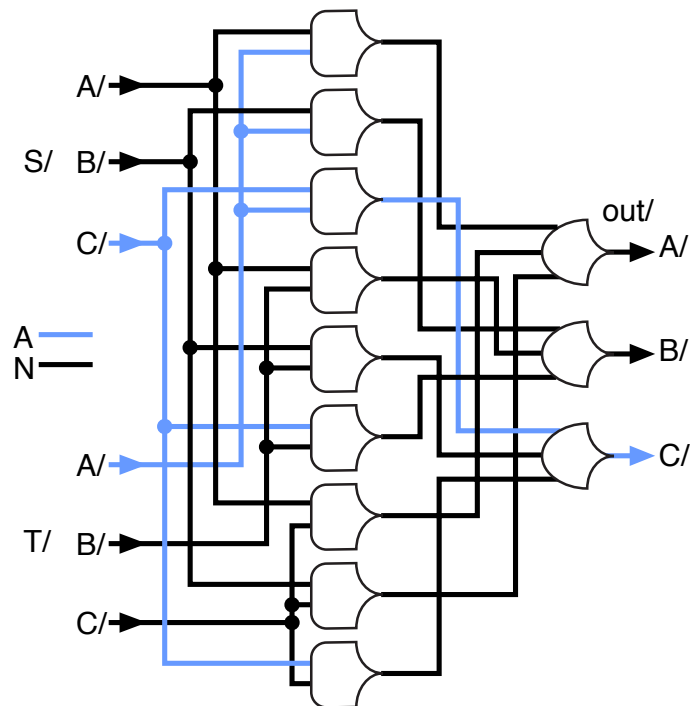
Figure 5.24. General mapping with "all of" and "one of" behaviors.

The complete interaction expression:

(Onecondexamp2 (S/{U V W}/{A N} T/{X Y Z}/{A N} -> out/{A B C}/{A N})
  out/{A<= { [S/U T/X] [S/U T/Z] [S/V T/Z] }
    B<= { [S/V T/X] [S/U T/Y] [S/W T/Y] }
    C<= { [S/W T/X] [S/V T/Y] [S/W T/Z] } }   )

Expressional generality, the ability to map any presented input differentness to any asserted output differentness is now realized entirely in terms of association differentiation. There is no differentiation in terms of differentness of condition.

5.9.   Spectrum summary: The differentness of differentness

Differentnesses and their interaction can be expressed as differences of conditions with specific interaction propensities promiscuously associating at a place of common association (pure condition differentiation) or as differentness of places of association with specific association relations each asserting only two different conditions, **A** and **N** with promiscuous interaction propensities (pure association differentiation) or with varying proportions of

condition differentiation and of association differentiation across a spectrum of differentiation. There must always be a little bit of each domain of differentiation in the expression of any interaction, at least one place of association with pure condition differentiation and at least two conditions with pure association differentiation.

As the spectrum was traversed from the pure condition differentiation end of the spectrum association differentiation took on increasing expressional duties until with pure association differentiation there was no longer any condition differentiation.
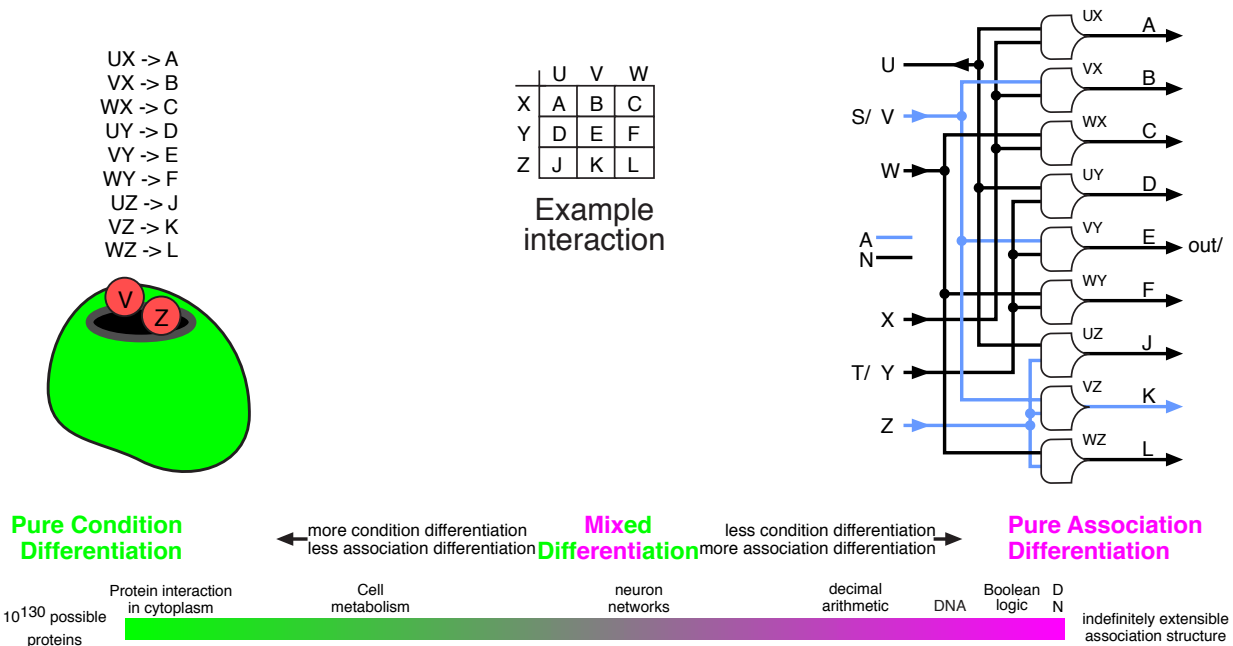
Figure 5.25. The spectrum of differentiation.

In the center of Figure 5.25 is the example interaction. To the right is shown the pure association expression of the interaction. To the left is shown the pure condition expression of the interaction. The pure condition expression and the pure association expression can be viewed as duals in specific respects.

- Each maps directly from the example interaction table.
- There are nine condition interaction propensity relations on the left and nine interaction behaviors on the right.
- There are six input conditions on the left and six input localities on the right.
- There are nine output conditions on the left and nine output localities on the right.
- On the left persistences asserting conditions freely associating in a shaking bag interact according to their asserted condition's interaction propensities.
- On the right a network of statically associated interaction behaviors interact according to their association relations.

The two ends of the spectrum are the expression realms of nature. At the pure condition end is the biological cell with lots of unique conditions, proteins, with specific interaction propensities and relatively little association structure. There are 10^{130} possible protein folding conditions and uncountable possible protein interaction relations. At the pure association end of

the spectrum is the neural network which is the direct association of differentnesses through association interaction behaviors.

The two realms of differentness are coextensive in that a range of differentness of condition reuses and extends the range of differentness expression of a place of association and each place of association reuses and extends the differentness expression of a range of differentness conditions. They collaborate with each extending the expression of differentness of the other to indefinitely extend expression of differentness.

The two realms cooperate in complex ways. A biological cell full of protein conditions contains associating organelles each isolating a pure condition expression. Cells associate to form organs which associate to form an organism in the midst of which is the blood stream, a pure condition expression indiscriminately associating to every cell in the organism.

Humans tend to express inside the spectrum encoding differentness with place-value numbers, place being association differentness and value being condition differentness. There are few conditions, few interaction behaviors and large indefinitely extensible association networks.

One domain of expression can be held constant while the other domain is allowed free rein. Conditions and interaction behaviors can be held constant while supporting arbitrarily complex association networks. Ten different numerals and their arithmetic behaviors support a vast realm of arbitrarily extensible numbers and their interactive association. The association relations can be held constant while condition differentiation is arbitrarily available. All mammals have essentially a constant association structure with the same body design, organ structures and cell structures. The variability among mammals is largely a matter of protein conditions expressed by DNA.

The spectrum of differentiation encompasses and unites disparate expressions of differentness and its interaction that previously appeared to be distinct or only superficially related. Nature and humans interact in fundamentally the same way in terms of differentness and the specific interaction of differentness.

Appendix A: Blinded by Elegance:
The Null Convention Logic Library

The reader may have notice that Null Convention Logic (NCL)[1] and its operator library are not mentioned in the text. There are strategic reasons, conceptual reasons, practical reasons and technical reasons.

A.1.   Null Convention Logic

Null Convention Logic was conceived as a threshold logic with hysteresis behavior that monotonically transitioned between "diff" completeness and completely "null" or "not diff". Because of multi-rail differentness representation complementing is a matter of relabeling rails instead of converting signals so there are no signal conversions in the wavefront path. Wavefront path logic is purely positive. The only signal conversion in NCL is the conversion of the closure of an oscillation.

Threshold logic was academically established and fully characterized. After a thorough review of threshold logic by my colleague Dave Duncan the library of NCL operators was conceived in terms of the threshold functions of four or fewer inputs. It turned out that there was an established mapping between positive Boolean functions of four or fewer inputs and threshold functions of four or fewer inputs. It was considered that this correspondence between positive Boolean functions of four or fewer inputs, threshold functions of four or fewer inputs and the NCL library operators as shown in Figure A.1 was a compelling correspondence of primitivities that would be advantageous.

Source Rationale for the NCL Function Library.

(Muroga, pp 435-437)

Muroga equivalence classes for threshold functions of 4 or fewer variables

NP equivalence classes for positive Boolean functions of 4 or fewer variables

The NCL function library.

		Weights W_i	Threshold T		
1 variable	1	X_1	1	1	buf
2 variables	1	X_1X_2	11	2	th12
	2	$X_1 \lor X_2$	11	1	th22
3 variables	1	$X_1X_2 \lor X_2X_3 \lor X_1X_3$	111	3	th13
	2	$X_1X_2X_3$	111	1	th23
	3	$X_1 \lor X_2 \lor X_3$	111	2	th33
	4	$X_1(X_2 \lor X_3)$	211	3	th23w2
	5	$X_1 \lor X_2X_3$	211	2	th33w2
4 variables	1	$X_1X_2X_3X_4$	1111	4	th14
	2	$X_1 \lor X_2 \lor X_3 \lor X_4$	1111	1	th24
	3	$X_1X_2X_3 \lor X_1X_2X_4$	2211	5	th34
	4	$X_1 \lor X_2 \lor X_3X_4$	2211	2	th44
	5	$X_1X_2X_3 \lor X_1X_2X_4 \lor X_1X_3X_4 \lor X_2X_3X_4$	1111	3	th24w2
	6	$X_1X_2 \lor X_1X_3 \lor X_2X_3 \lor X_1X_4 \lor X_3X_4 \lor X_2X_4$	1111	2	th34w2
	7	$X_1X_2X_3 \lor X_1X_2X_4 \lor X_1X_3X_4$	2111	4	th44w2
	8	$X_1 \lor X_2X_3 \lor X_2X_4 \lor X_3X_4$	2111	2	th34w3
	9	$X_1X_2 \lor X_1X_3X_4 \lor X_2X_3X_4$	2211	4	th44w3
	10	$X_1X_2 \lor X_1X_3 \lor X_2X_3 \lor X_1X_4 \lor X_2X_4$	2211	3	th24w22
	11	$X_1X_2 \lor X_1X_3X_4$	3211	5	th34w22
	12	$X_1 \lor X_2X_3 \lor X_2X_4$	3211	3	th44w22
	13	$X_1X_2 \lor X_1X_3 \lor X_2X_3X_4$	3221	5	th54w22
	14	$X_1X_2 \lor X_1X_3 \lor X_2X_3 \lor X_1X_4$	3221	4	th34w32
	15	$X_1X_2 \lor X_1X_3 \lor X_1X_4 \lor X_2X_3X_4$	2111	3	th54w32
	16	$X_1X_2 \lor X_1X_3 \lor X_1X_4$	3111	4	th44w322
	17	$X_1 \lor X_2X_3X_4$	3111	3	th54w322
	18	$X_1X_2 \lor X_3X_4$			thxor
	19	$X_1X_3 \lor X_2X_3 \lor X_1X_4 \lor X_2X_4$			thand
	20	$X_1X_2 \lor X_2X_3 \lor X_3X_4$			thcomp

Figure A.1. Function classification rationale for the NCL operator library.

There are 28 PN classes of unate or positive boolean functions of four or fewer variables. There are 25 classes of threshold functions of four or fewer variables. Twenty five of the positive Boolean classes are linearly separable and map directly onto the 25 classes of threshold functions of four or fewer variables. The 3 remaining Boolean functions of four or fewer variables are not linearly separable and do not map onto threshold functions but can map onto threshold circuits. Including these three circuits in the library allowed a complete correspondence between the NCL library and all positive Boolean functions of four or fewer variables as shown in Figure A.2.

We considered this correspondence to be elegant, practical and even necessary to relate to existing practice: a useful strategic positioning.

- We could use Boolean equations to design NCL circuits.
- We could relate to a familiar language.
- We could possibly translate clocked Boolean designs directly to NCL designs.
- We could use standard design tools for NCL.

Figure A.2. The 28 NCL library operators with their corresponding positive Boolean functions.

A.2.   The strategic error

Relating so directly to Boolean logic turned out to not be as useful as anticipated. While we could specify an NCL network in terms of Boolean operations the correspondence was not direct. There were a lot of considerations beyond the Boolean equations involved. Also we could not just take clocked Boolean designs and map them to NCL. The two logics behaved very differently.

Furthermore, relating to Boolean logic implied that Boolean logic was somehow more fundamental than NCL. We were trying to establish NCL as a superior form of logic to a population who by training and experience accepted Boolean logic as conceptually fundamental, a minimal necessary and sufficient theoretical standard to which all else reduces. Relating to Boolean logic just reinforced this view. It took many years of experience to realize that relating NCL to Boolean logic was more confusing then clarifying.

While everyone understood Boolean logic, threshold logic was not a current topic of familiarity. Its time had passed without it becoming used or widely understood. We were relating to a dinosaur, an elegant dinosaur but a dinosaur nonetheless.

A.3.   The conceptual error

The logic portrayed by the library of NCL operators is involved and complicated. For all its sense of completeness and closure with the 28 operators the library did not convey a sense of primitivity. Each operator had an assigned threshold and many of the inputs had assigned weights. Networks were difficult to design and optimize because each input had to be connected to a correctly weighted operator input.

The conceptual error was to characterize this complicated set of behaviors as primitive. They seemed primitive. All but three operators were single gate threshold functions. I was aware that the gates had corresponding 1 of M and M of M networks shown in Figure A.3 but the threshold gates seemed like a higher level of primitivity, of atomicity and of efficiency. Even though there was a corresponding network of simpler operations the threshold operator was still conceptually a single gate, not a network, implying primitivity. The fact that it was a logic of threshold functions further implied an established primitivity.

Another confusion was that the operator hysteresis behavior was established with a single feedback signal further implying atomicity and primitivity.

1. A

TH12
2. A + B

TH22
3. AB

TH13
4. A + B + C

TH23
5. AB + BC + AC

TH33
6. ABC

TH23W2
7. A + BC

TH33W2
8. AB + AC

TH14
9. A + B + C + D

TH24
10. AB + AC + AD + BC + BD + CD

TH34
11. ABC + ABD + ACD + BCD

TH44
12. ABCD

TH24W2
13. A + BC + BD + CD

TH34W2
14. AB + AC + AD + BCD
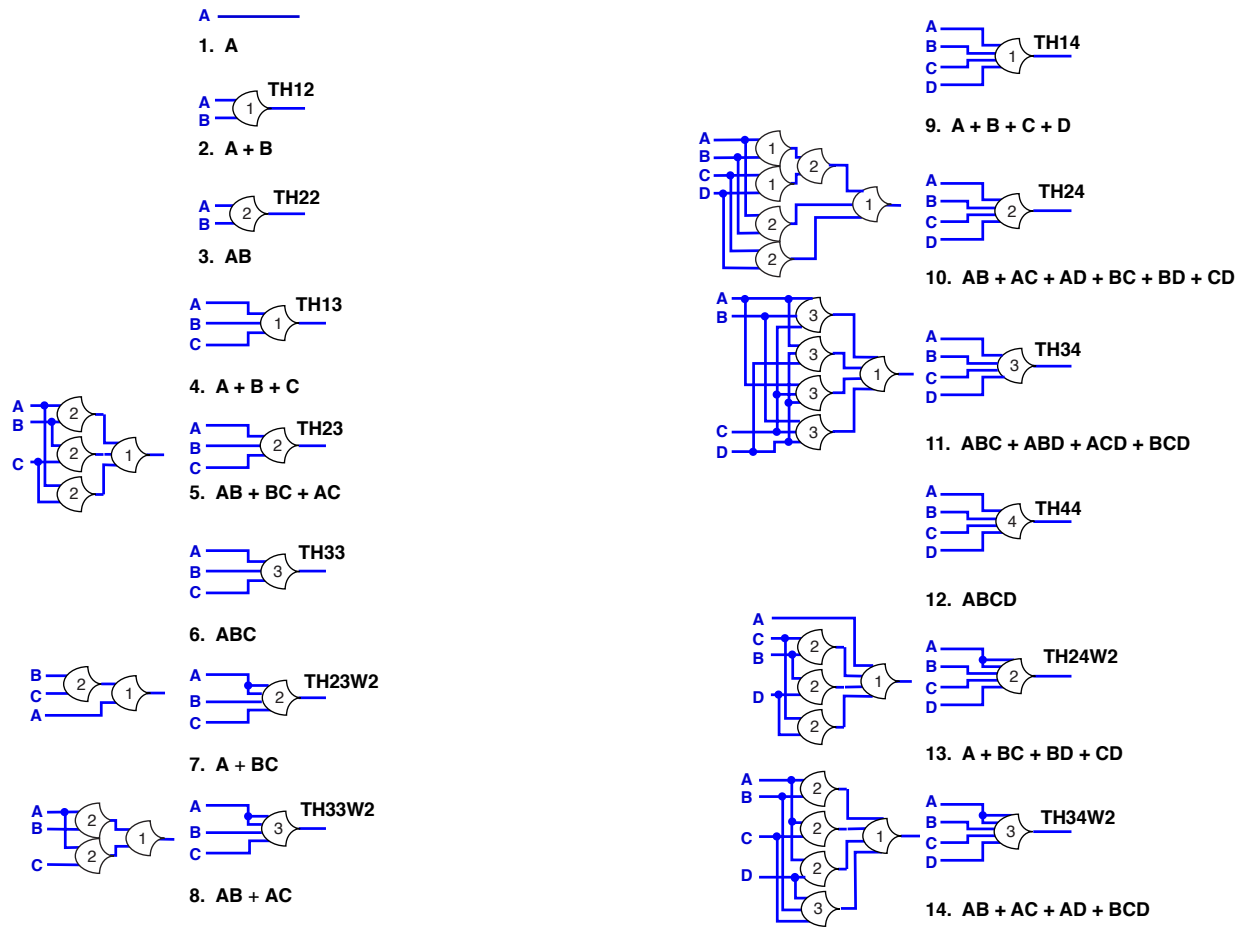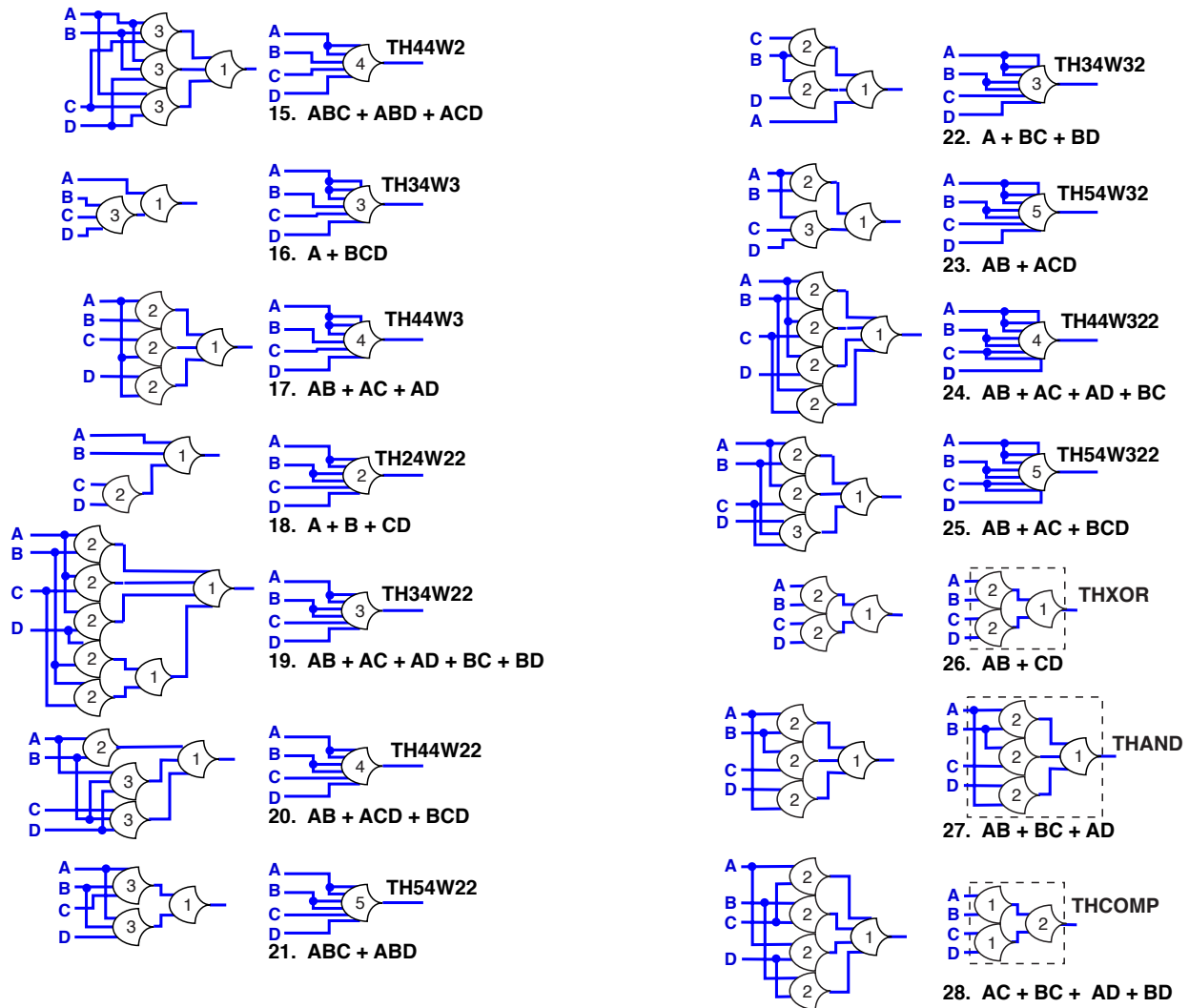
Figure A.3. NCL library with equivalent 1ofM and MofM networks.

The library seemed to embody a superior expressional efficiency and an elegant coherence or completeness. It was not until I started thinking in terms of "all of" and "one of" as primitives that I was able to stand outside the library and detach from Dave's and my creation.

A.4.   The practical error

After studying many ways to implement the library gates including several implementations with actual threshold behavior it was finally decided to implement the gates with transistor switching networks. The decision was driven by what was possible and what was available. While there were fabrication techniques for threshold oriented devices using capacitors and other devices they were not as developed, as available or as efficient as the transistors. It was not so much an error as a constraint of practicality. So the elegant threshold behaviors were implemented with Boolean switching networks which, as it turned out, did not compete well with clocked Boolean switching networks.

Other possibilities of implementing the behaviors with inherent threshold and hysteresis behavior such as quantum dots, magnetic domains, memristors for instance may become practical in the future.

1. A

2. A + B   TH12

3. AB   TH22

4. A + B + C   TH13

5. AB + BC + AC   TH23

6. ABC   TH33

7. A + BC   TH23W2

8. AB + AC   TH33W2

9. A + B + C + D   TH14

10. AB + AC + AD + BC + BD + CD   TH24

11. ABC + ABD + ACD + BCD   TH34

12. ABCD   TH44

13. A + BC + BD + CD   TH24W2

14. AB + AC + AD + BCD   TH34W2
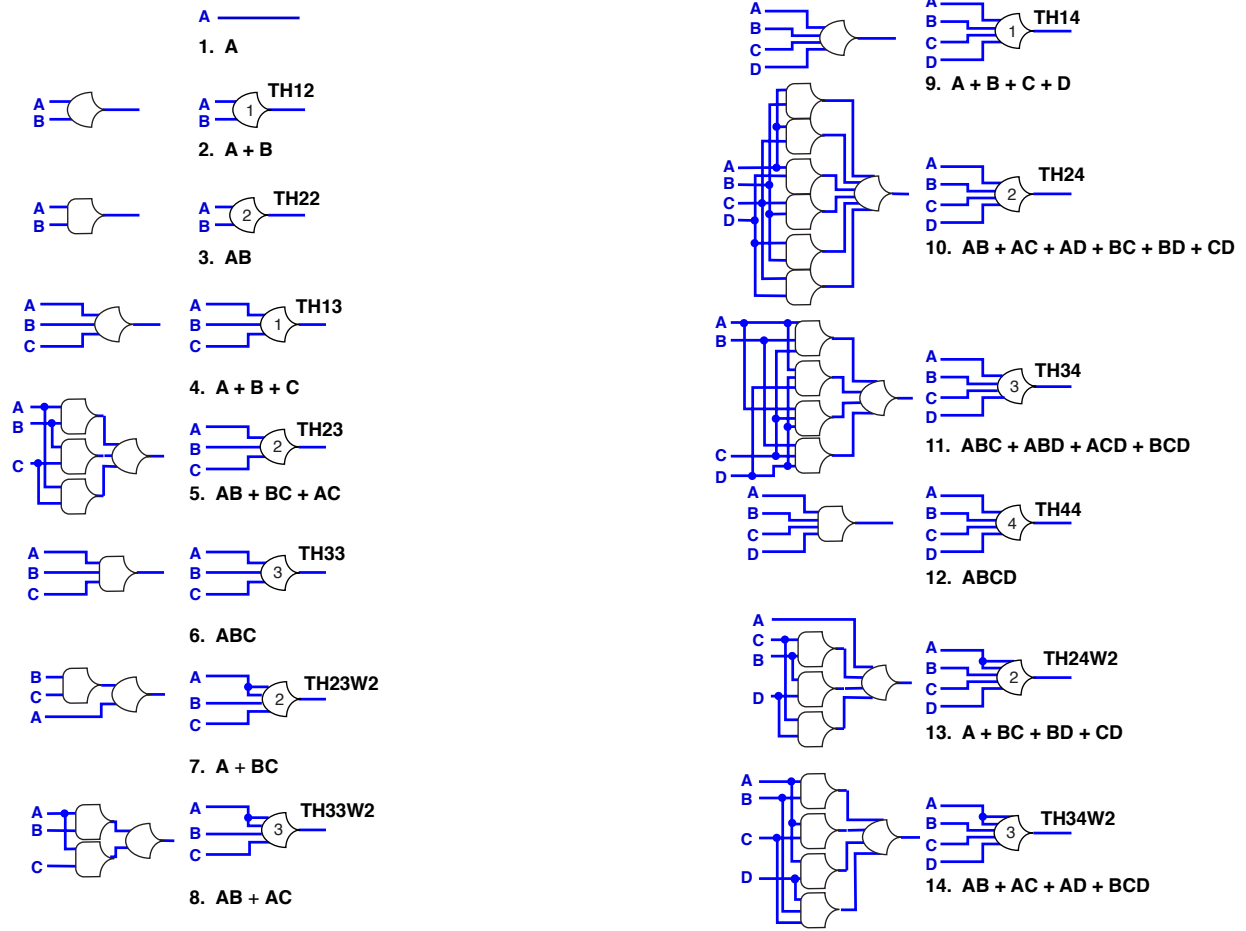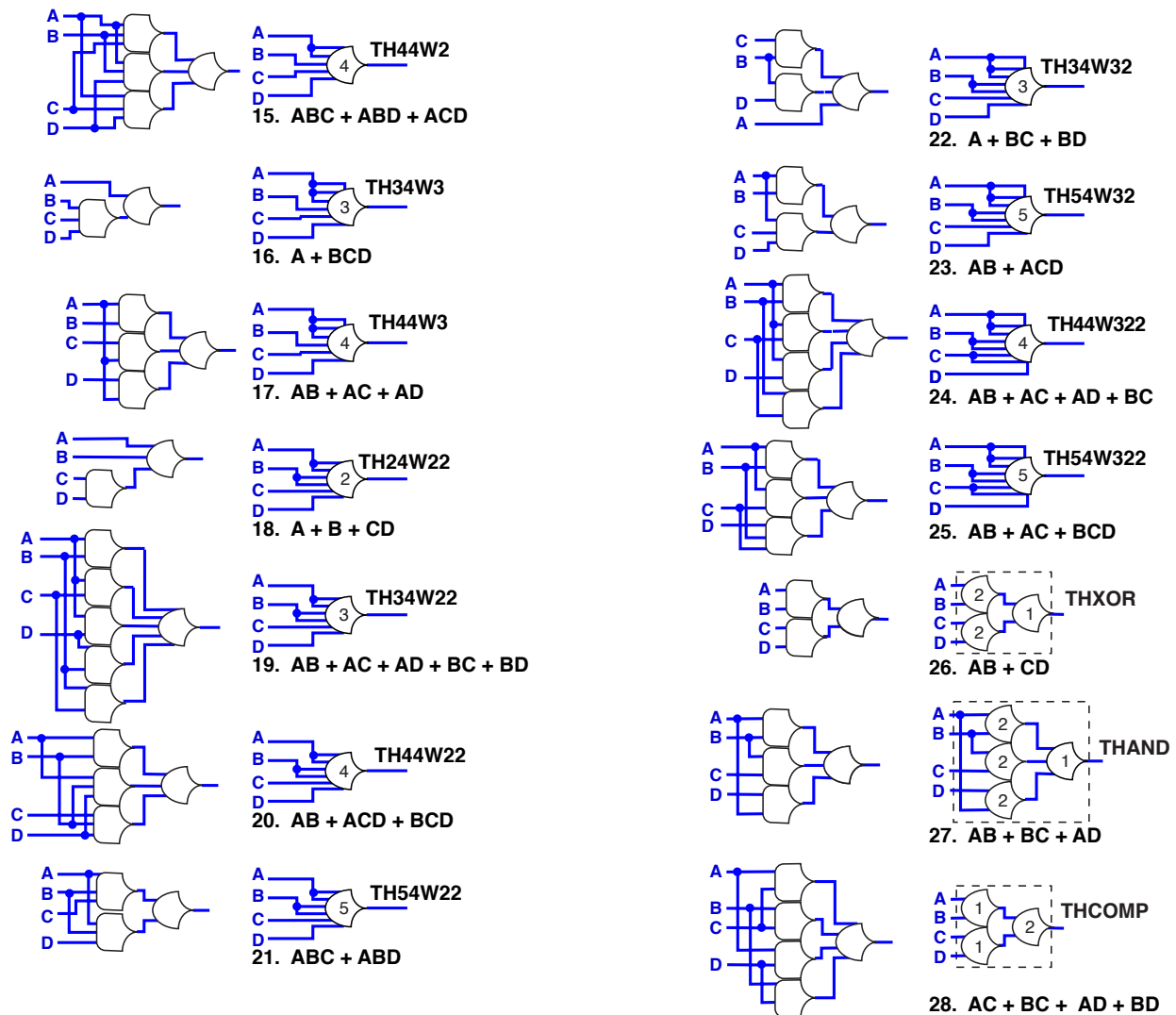
Figure A.4. The NCL library in terms of "one of" and "all of" behaviors.

A.5.   The technical error

The technical error is the completeness behavior of the library operators which can lead to deadlock. In viewing the library operators as threshold functions, including the three operators that actually encapsulated threshold networks, they were considered as primitive with atomic behavior. In particular, completeness relations would apply across all inputs of each operator. When a particular quantity of inputs transitioned to "diff" an operator would transition its output to "diff" and when "all of" the inputs transitioned to "not diff" the operator would transition its output to "not diff". This completeness protocol worked in most circumstances but not always and the occasional failure is the fatal flaw. The difficulty is illustrated with the NCL THXOR operator in relation to its corresponding network.

A.5.1.   The nominal behavior of THXOR

Figure A.5 shows the behavior of THXOR (top) and its associated network (bottom) is valid if [A,B] and [C,D] trace back to a "one of" relationships and are mutually exclusively presented.
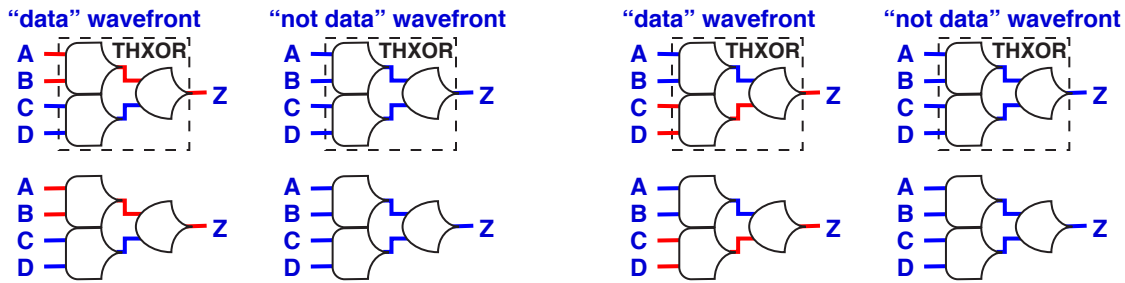
Figure A.5. Nominal behavior of THXOR and its associated network

A.5.2.   The race behavior of THXOR

The inputs of every "one of" behavior must be mutually exclusive, only one transition to "diff" per "diff" wavefront. If [A,B] and [C,D] are not "one of" related they can collide and race to the "one of" behavior.
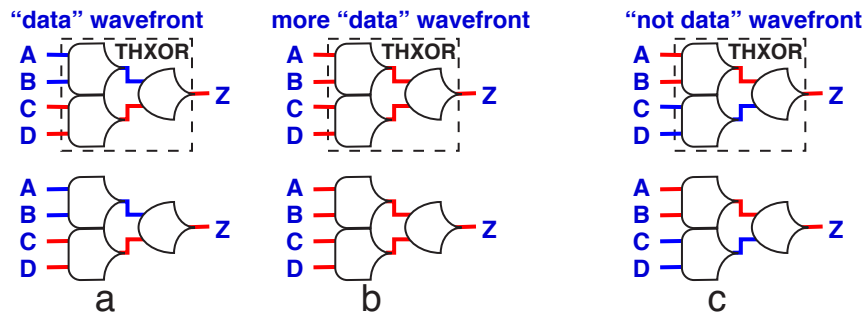
Figure A.6. [A,B] and [C,D] wavefronts collide.

In Figure A.6.a wavefront [C,D] transitions to "diff" first and Z transitions to "diff". In Figure A.6.b wavefront [A,B] transitions to "diff" while [C,D] is still at "diff" but Z is already transitioned to "diff". In Figure A.6.c the [C,D] "not diff" wavefront arrives. Z does not transition to "not diff". [A,B] will never close and the network is deadlocked. The not coordinated wavefronts can also glitch Z which is equally disastrous.

A.5.3.   The deadlock behavior of THXOR

Ensuring that any "one of" combination behavior has only one input transition to "diff" at a time is a basic design rule. The direct means of ensuring that the [A,B] and [C,D] wavefronts are mutually exclusive is for one component of each wavefront to come from the same 1composite locality. For instance, in Figure A.7 A and C are replaced with S/0 and S/1 from

locality S/{1,0};

S/0 will enable B. S/1 will enable D. This is correct in that there will be only one input to the "one of" behavior at a time. The deadlock difficulty with THXOR arises if either B or D transition to "diff" during the "diff" phase of the other. Figure A.7 illustrates D transitioning while [S/0,B] is "diff".

In Figure A.7.a S/0 enables B and Z transitions to "diff". In Figure A.7.b D transitions to "diff" while [S/0,B] are still at "diff". In Figure A.7.c [S/0,B] transitions to "not diff" but the THXOR output Z does not transition to "not diff" because the input to THXOR is not completely "not diff". The protocol for [S/0,B] is not completed and never will be. D is waiting on S/1 which will never occur because S/0 will never be closed and released. THXOR is deadlocked. If B and

Copyright © 2024 by Karl Fant

D transition to "diff" mutually exclusively, which is typical, then the THXOR operator works just fine.

In the corresponding network of Figure A.7.c the completeness is across the individual combination behaviors. D is not preventing the transition of Z so S/0 can complete its oscillation and S/1 can eventually occur. Locality S is sufficient to ensure mutual exclusivity into the "one of" behavior and does not risk deadlock or race.

The network works correctly and the atomic threshold THXOR operator does not.
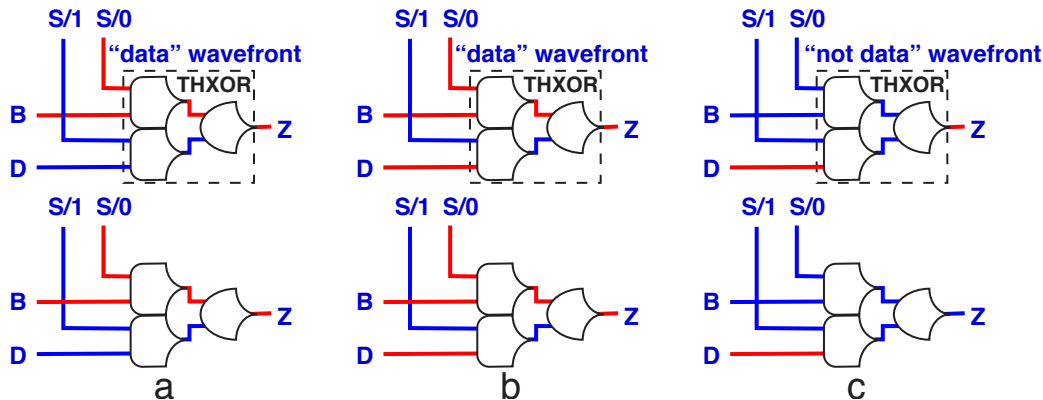
Figure A.7. The MUX protocol and deadlock.

The technical error is in treating the NCL operations as atomic primitives with completeness across all the inputs. This deadlock possibility is present with any NCL threshold operator that includes "one of" relations.

The THXOR difficulty was pointed out to me by others particularly by my colleague Stephen Johnson on numerous occasions. I persisted in viewing the difficulty not as a flaw in the library but as a problem of proper design rules. There is a necessary cooperation between the flow path behavior and the NCL operators. The paths have to monotonically transition and certain mutual exclusivities must be observed. My attitude was that this was just one of the mutual exclusivities that had to be observed. The fact that it rarely occurs contributes to perceiving it as a network design issue rather than a primitive atomic behavior issue. For many years I did not design complex circuits myself but relied on teams of engineers. It was only much later after I lost access to engineering support that I started designing significant NCL circuits myself. Only after I encountered several deadlock situations in my own designs and ended up backing out several threshold gates into networks of simpler gates to solve the completeness issue instead of altering the flow path behavior did I realize that the notion of atomic completeness across all inputs of many of the NCL operators is technically erroneous and that the characterization of the logic as threshold functions was misguided.

The difficulty follows from the dual threshold nature of the NCL operators. Traditional threshold logic is treated as passive logic functions that are coordinated with a clock. There was only one threshold that was either met or not at the clock tic. The next clock tic tested the threshold again. There was no behavior requirement imposed on the threshold function that reflected to the behavior of its input. However the input behaved the threshold gate just monitored the threshold.

With NCL there are two thresholds that have to be alternately matched so the input to the operator must behave properly by monotonically transitioning between completeness conditions. It is this behavior dependence between operator behavior and flow behavior that can be correct

by construction. But this cooperative behavior can also be incorrect by construction as is the case with the NCL operator completeness behavior. The criteria of correct construction can be subtler than expected.

1. Karl M. Fant, Logically Determined Design: Clockless System Design with NULL Convention Logic, (Hoboken, New Jersey, Wiley Interscience, 2005)

Appendix B: Pipeline Performance

B.1.   Basic pipeline network structure and behavior

The first parameter of pipeline performance is the **oscillation network period**. The **oscillation path**, shown in Figure B.1, is characterized in two parts, the **wavefront flow path of the oscillation**, in green, and the **bubble flow path of the oscillation**, in orange[*]. The bubble flow path contains the oscillation closure flow path (section 3.7). The notion of bubble and bubble path will develop over the section. The combined delay of the two paths gives the **oscillation network period**. By convention, the wavefront path includes the enable rank of the input link and the bubble path includes the enable rank of the output link.
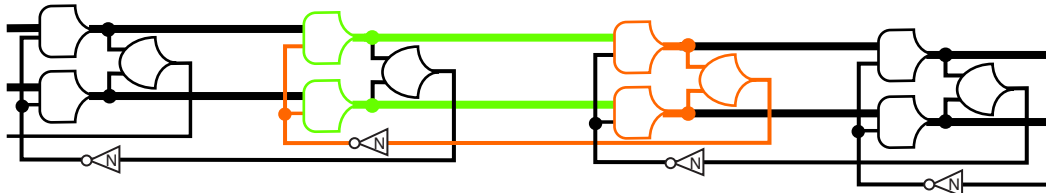
Figure B.1. Oscillation network oscillation/flow path.

Catenating the wavefront flow paths of the linked oscillations, Figure B.2, gives the **wavefront flow path of the pipeline** and adding the delays of the wavefront flow paths gives the **wavefront path delay** or **forward latency** of the pipeline.
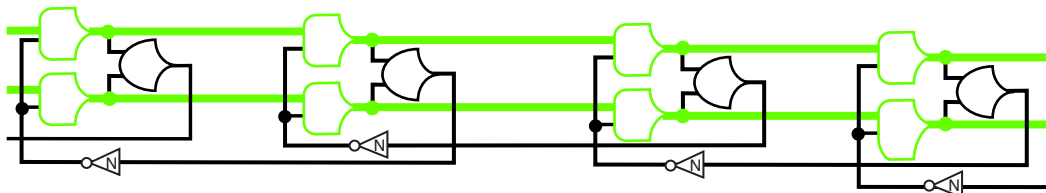
Figure B.2. Wavefront flow path, the forward latency of pipeline.

Catenating the bubble flow paths of the linked oscillations, Figure B.3, gives the **bubble flow path of the pipeline** and adding the delays of the bubble flow paths gives the **bubble path delay** or **reverse latency** of the pipeline.
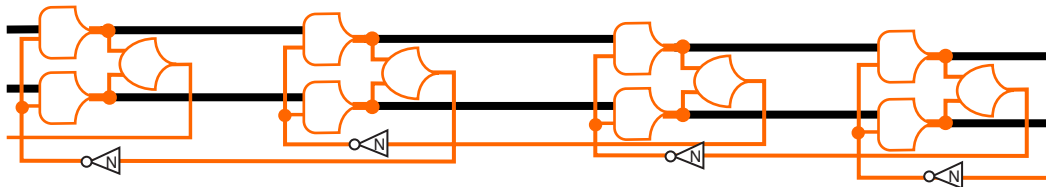
Figure B.3. Bubble flow path, the reverse latency of pipeline.

B.2.   Primitive component performance

While delays in relation to some external time referent have no relevance to the correct behavior of a network, its performance in relation to an external time referent might be important to the network relating to behaviors external to the network. Its performance in relation to an

---

[*]. Bubble, an unusually apt asynchronous design term, is an emptiness into which wavefronts can flow.

external time referent can be determined in terms of subnetwork delays in relation to the external time referent.

Composing two half oscillations makes a complete oscillation. Each half oscillation contributes a portion of delay to each path of the complete oscillation. The half oscillation wavefront path delays are added to get the wavefront path delay for the new oscillation. The half oscillation bubble path delays are added to get the bubble path delay for the new oscillation for the new oscillation. Adding the wavefront path delay and the bubble path delay gives the period of the new oscillation.

Figure B.4 shows the input half oscillation and the output half oscillation of the quad to dual network component and the delays associated with the wavefront path and bubble path of each half oscillation. To keep the delay discussion simple (connections) have no delay and primitive search behaviors have delay based on their inputs.

1 input 20 ps
2 inputs 30 ps
3 inputs 40 ps
4 inputs 50 ps
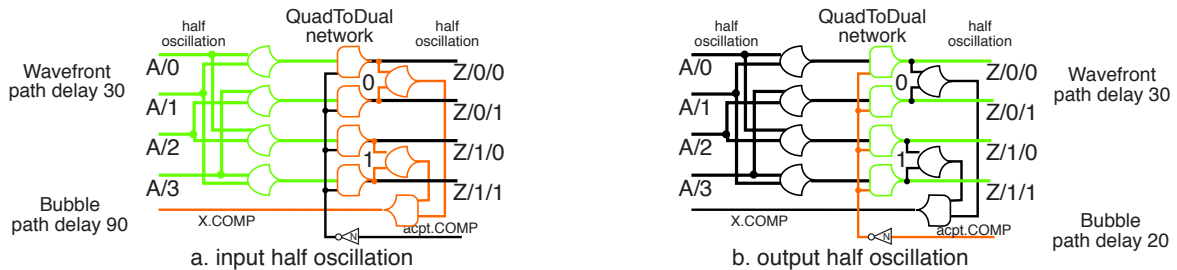
Figure B.4. Half oscillations of quad to dual primitive network.

Figure B.5 shows the delays associated with the half oscillations of the dual to quad network component.
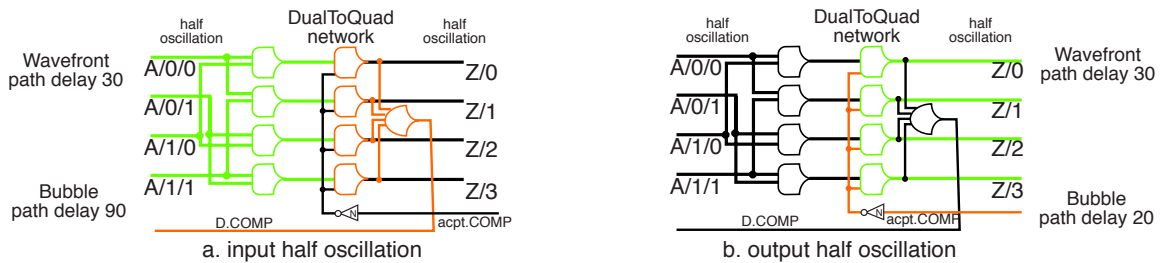
Figure B.5. Half oscillations of dual to quad primitive network.

B.3.   Network composition

Networks are composed by associating the locality name of an output half oscillation to the corresponding locality name of an input half oscillation forming a full oscillation. The performance of the full oscillation is the combination of the performance of the half oscillations.
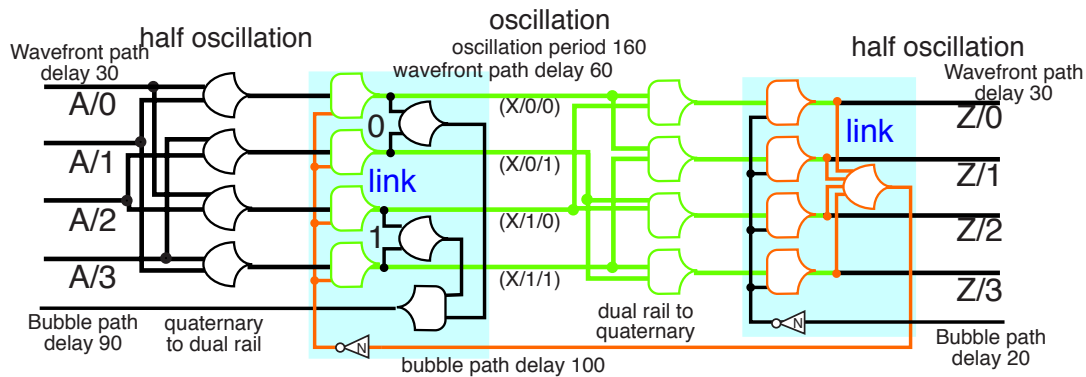
Figure B.6. The composed components form a pipeline network with one full oscillation.

network conversion (A -> Z):
locality (A, Z)/{3,2,1,0}/;
locality X/[1,0]/{1,0}/;
   QuadToDual (A -> X)
   DualToQuad (X -> Z)
endnetwork

B.3.1.   Composition performance

The key factor of pipeline performance is the oscillation period. The performance of a boundary half oscillation is deferred until it is composed. For the composed network of Figure B.6 the oscillation period of the composed oscillation in network conversion is 160 ps so the maximum throughput of the pipeline is one "diff" wavefront every 320 ps. The pipeline will sustain any throughput presented to its input slower than or equal to 320 ps per "diff" wavefront. The initialization delay, the sum of the wavefront propagation delays, for network conversion is 120 ps. The initialization delay for the composed oscillation is 160 ps which in this case determines the initialization delay 160 ps plus margin, for the network. Usually the initialization delay is determined by the network wavefront propagation delay which is typically far longer than any individual oscillation period.

B.3.2.   Further composition

Since the output and the input boundaries of network conversion are compatible half oscillations, two conversion networks can be connected forming a larger longpipe network. Both component networks propagate initialization so the composed network propagates initialization. Both networks propagate completeness behavior so the composed network accepts and propagates completeness behavior. All orphans remain isolated. The composed network, validated and bounded by half oscillations, is ready for further composition.
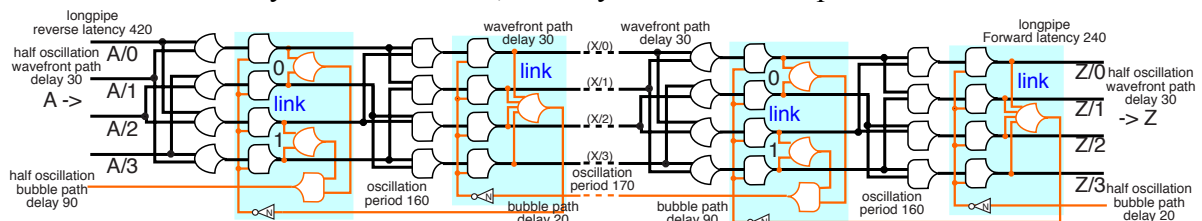
Figure B.7. Bigger network from smaller networks.

network longpipe (A -> Z):
locality (X, A, Z)/{3,2,1,0}/;
   conversion (A -> X)

conversion (X -> Z)
endnetwork

B.3.3.   Longpipe performance

Performance parameters are accumulated as composition progresses. In Figure B.7 the longest oscillation period is the new formed oscillation at 170 ps. The network will sustain a throughput of 340 ps per "diff" wavefront. The forward latency is the sum of the wavefront path delays, 240 ps. The reverse latency is the sum of the bubble path delays, 420 ps. The wavefront path initialization delay is 240 ps. The slowest oscillation initialization delay is 170 ps So the initialization delay for the new network is 240 ps plus margin.

B.3.4.   Pipeline performance

The input and output of a pipeline are half oscillations which, for purposes of performance analysis, are assumed to be sufficient flow in that when the input closure enables an input wavefront a wavefront is present and flows into the pipeline and when an output wavefront is presented its closure is present and the wavefront flows out of the pipeline. A pipeline network is a unit of composition that flows with a characteristic throughput which equals the period of its slowest oscillation and with a characteristic latency which is the delay of its wavefront path.

B.3.5.   Performance in an environment

If the environment responds more slowly than the pipeline's characteristic throughput then the pipeline will conform to the throughput of the environment. If the environment can conform to the throughput of the pipeline then the system will flow at the characteristic throughput rate of the pipeline. If the environment demands a faster throughput than the pipeline can support then the environment/pipeline system will fail.

Appendix C: Wavefront Arbitration

Uncoordinated wavefront flows that share a common resource must be arbitrated into mutually exclusive coordinated flow behavior with only one wavefront proceeding at a time. Wavefronts flow in "not diff"/"diff" wavefront pairs. An arbiter behavior will allow only one wavefront pair to flow at a time.

C.1.   The MUTEX

The mutual exclusion behavior of an arbiter is provided by a MUTEX. Figure C.1 Shows a CMOS implementation of a MUTEX, an SR flipflop with a filter across the two outputs to minimize metastable oscillation and facilitate resolution of the metastability.
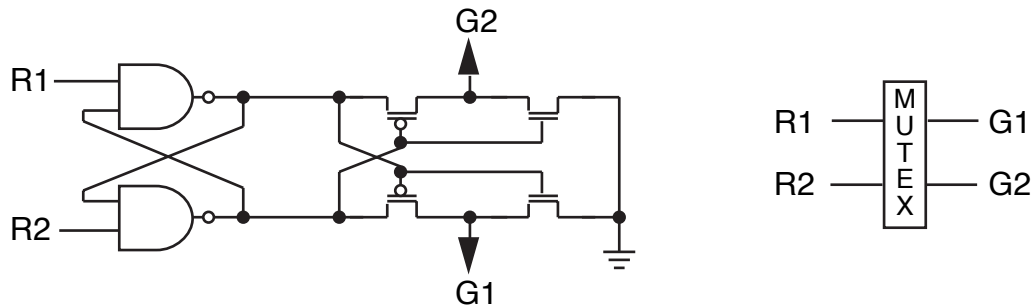
Figure C.1.MUTEX circuit and symbol.

A MUTEX is a primitive mutual exclusion behavior. that will assert grants G1 or G2 mutually exclusively in time. They are asserted, respectively, in response to the requests R1 or R2 which can be asserted at any time. If asserted simultaneously the MUTEX will chose one request to grant and block the other request until the granted request is released. When the granted request is released its grant will be removed and the waiting request will be immediately granted. If one request arrives first it is granted and the second arriving request waits until the first request is released upon which the waiting request is immediately granted. If a request is released before it is granted then it is not granted and has no effect. So there is an assumption that the behaviors asserting R1 and R2 are sensitive to their grants, will wait appropriately and that the assertion of the grant will ultimately cause the release of the request.

C.2.   The arbiter

A wavefront arbiter must allow a "not diff"/"diff" wavefront pair. A MUTEX request occurs with the arrival of a "diff" wavefront and the transition of request to "diff". With the arrival of the "not diff" wavefront the request will transition to "not diff" and the MUTEX will immediately allow a waiting grant. This does not allow for the passage and completeness of the "not diff" wavefront before allowing the next grant. There must be an additional mechanism in addition to the MUTEX that waits for the "not diff" wavefront to completely propagate before granting the waiting request. Figure C.2 shows the arbiter core with the extra circuitry to wait on the "not diff" wavefront.[1] The arbiter core fits into an oscillation and arbitrates wavefront flow through the oscillation.
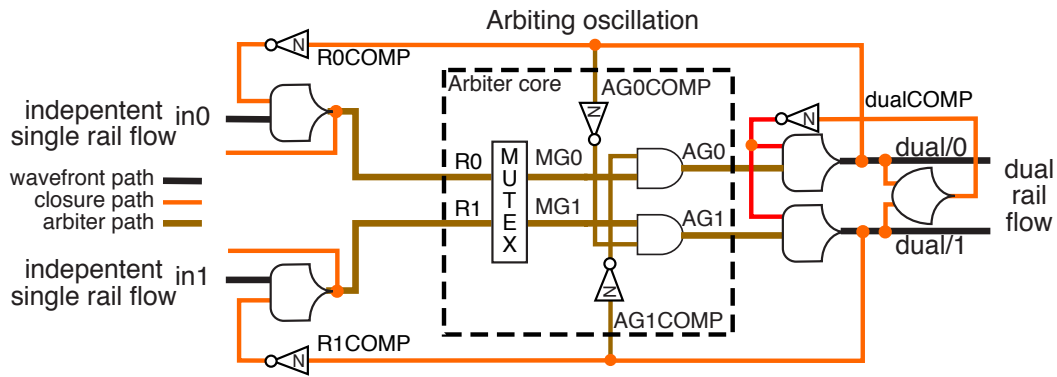
Figure C.2. Arbitrating oscillation.

In the following example an arbitrating oscillation receives two uncoordinated single rail flows and produces a single coordinated dual rail flow. In Figure C.3 the arbiter oscillation begins in an initial configuration awaiting wavefront flow.
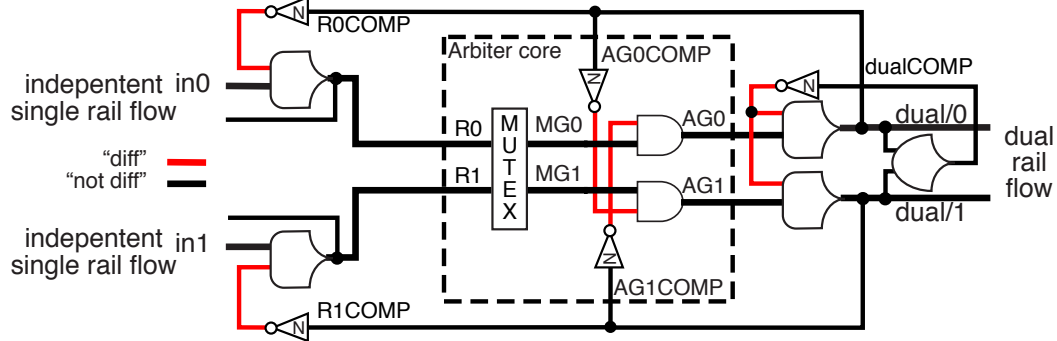
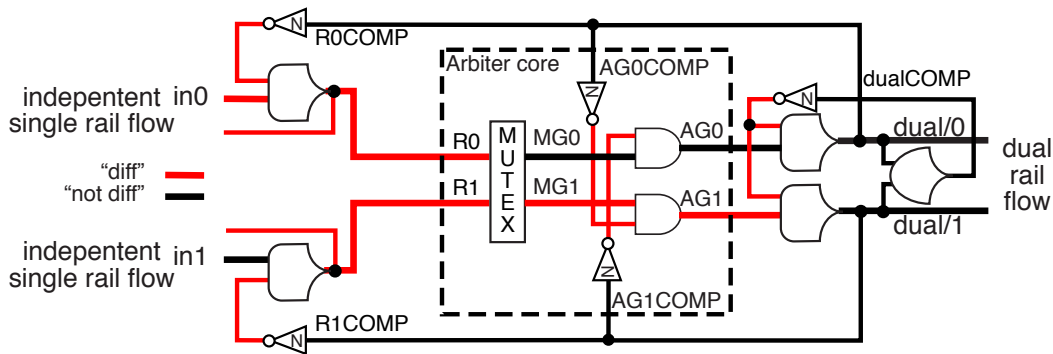**Figure C.3. Arbiter oscillation awaiting wavefront flow.**

"diff" wavefronts arrive simultaneously at R0 and R1.
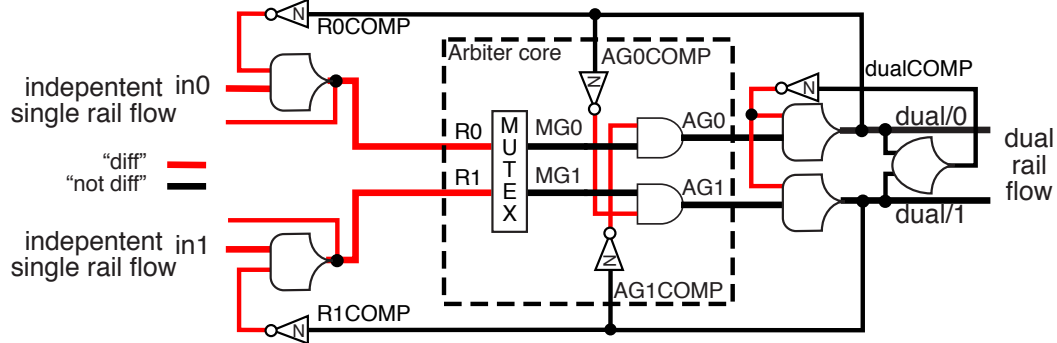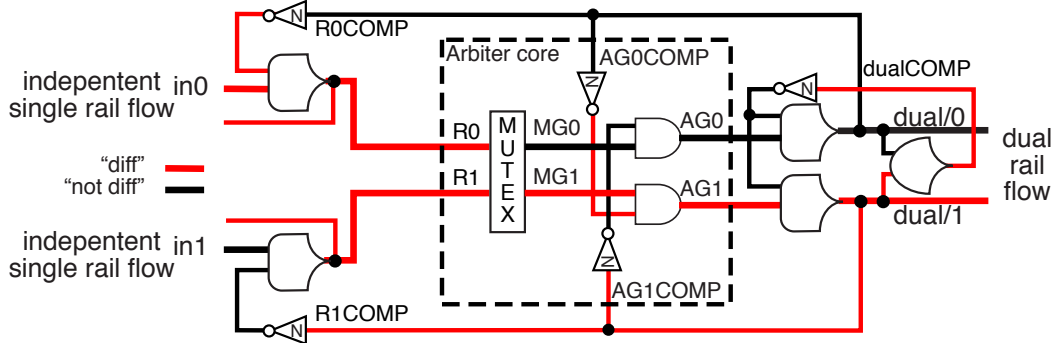

Figure C.4. Request "diff" wavefronts arrive simultaneously at MUTEX.

In Figure C.4 two "diff" wavefronts arrive simultaneously at the MUTEX.

Figure C.5. The MUTEX grants R1 and blocks R0.

In Figure C.5 request R1 is granted with MUTEX Grant MG1 and Arbiter Grant AG1. R0 is blocked by the MUTEX.

Figure C.6. The R1 flow closes with in1 enabling the "not diff" wavefront.

In Figure C.6 R1 flows through the link and closes with in1 enabling the "not diff" wavefront flow. The closure also closed with the arbiter core through AG1COMP disabling AG0.
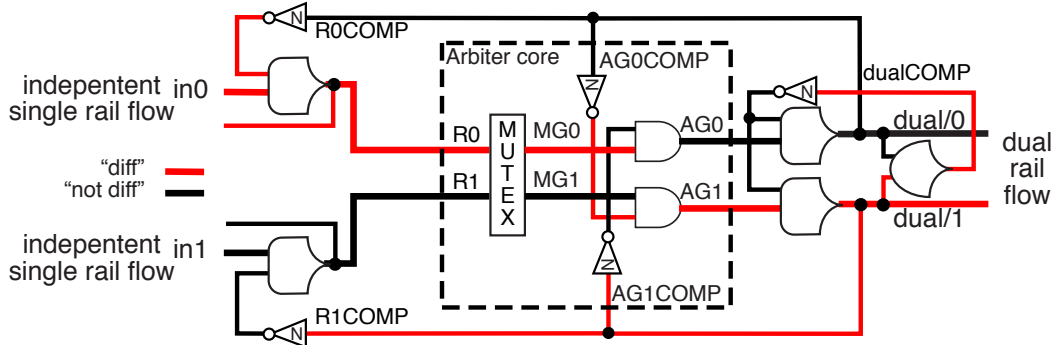
Figure C.7. R1 is released and R0 is immediately granted by the MUTEX.

In Figure C.7 the R1 "not diff" wavefront flows to the MUTEX releasing the R1 request. The MUTEX immediately grants the R0 request with MG0 which is blocked at AG0. **There is a local time relation here in that the AG1COMP flow must arrive at AG0 before the "not diff" wavefront arrives at R1 and releases the request.**
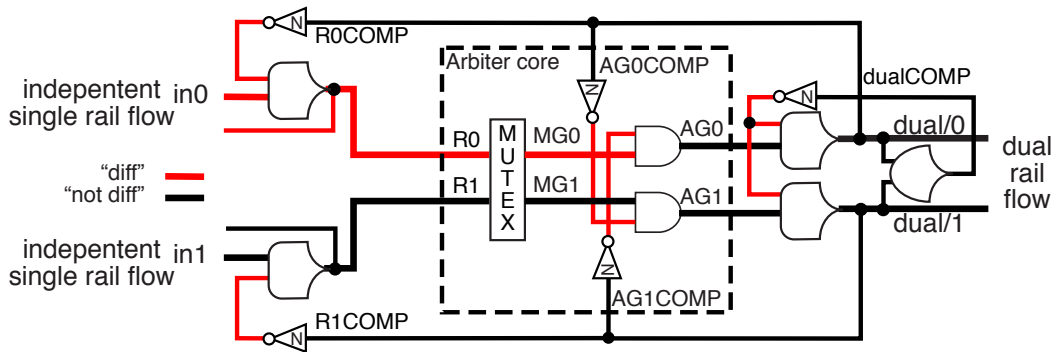
Figure C.8.The "not diff" wavefront closure released the block of AG0.

In Figure C.8 the R1 "not diff" wavefront flow closure flows through AG1COMP and releases the AG0 block.
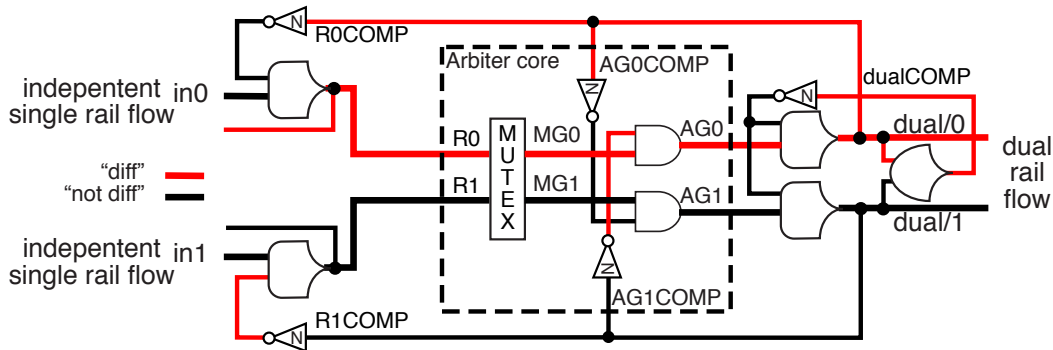
Figure C.9.The R0 "diff" wave flows.

In Figure C.9 the R0 "diff" wavefront flows through AG0 and the link. The "diff" closure flowing through AG0COMP blocks AG1 and enables the "not diff" wavefront at in0.
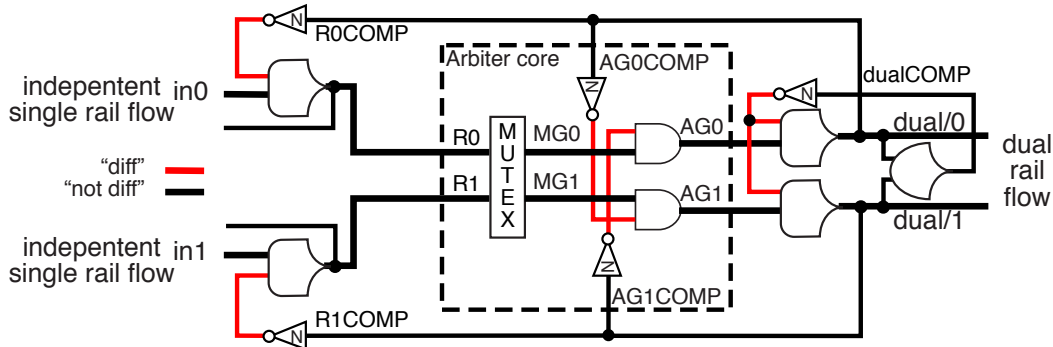
Figure C.10.The R0 "not diff" wavefront flows.

In Figure C.10 the R0 "not diff" wavefront flows through the link closing with in0, releasing AG1 and the arbiter oscillation returns to its initial configuration awaiting the next "diff" wavefront.

C.3.   Arbitrating composite wavefronts

Figure C.11 shows an arbitrating oscillator arbitrating uncoordinated dual-rail flow paths through a "one of" behavior into a single dual-rail flow path. Large composite wavefronts can be

arbitrated. The completeness of the dual-rail wavefront is used as the request. The grant enables the flow of the wavefront through the grant link.
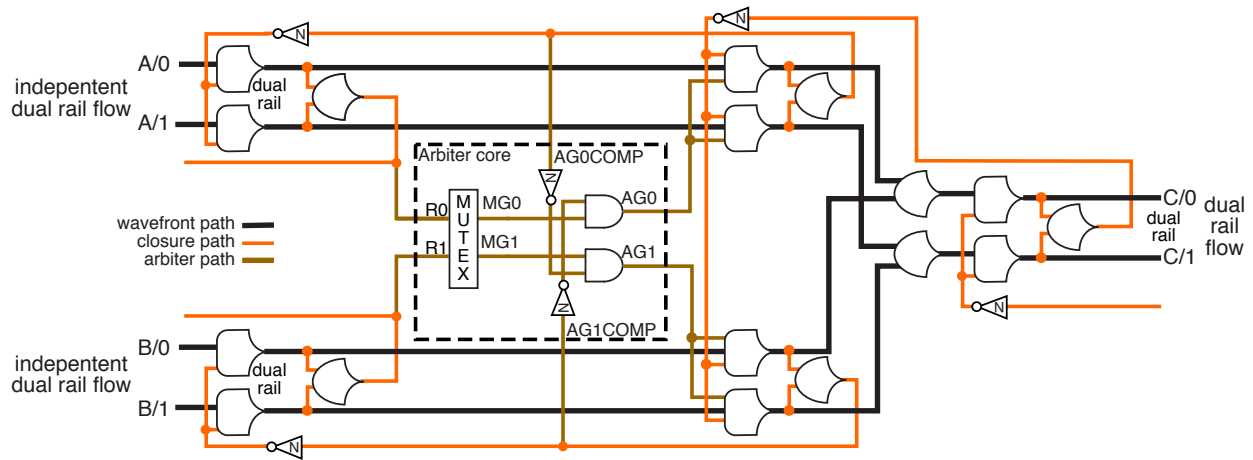
Figure C.11.Two dual rail paths arbitrated into "one of" behavior.

Figure C.12 shows large composite independent wavefronts arbitrated into a single composite wavefront flow. Any one rail portion of the completeness of a wavefront can be used as the request. The grant is used to enable the flow of the wavefront through the grant link. The completeness of flow is managed between the grant link and the requesting link. The completeness of the grant link is reduces to a single rail closure which presents to AGxCOMP and closes with the requesting link.
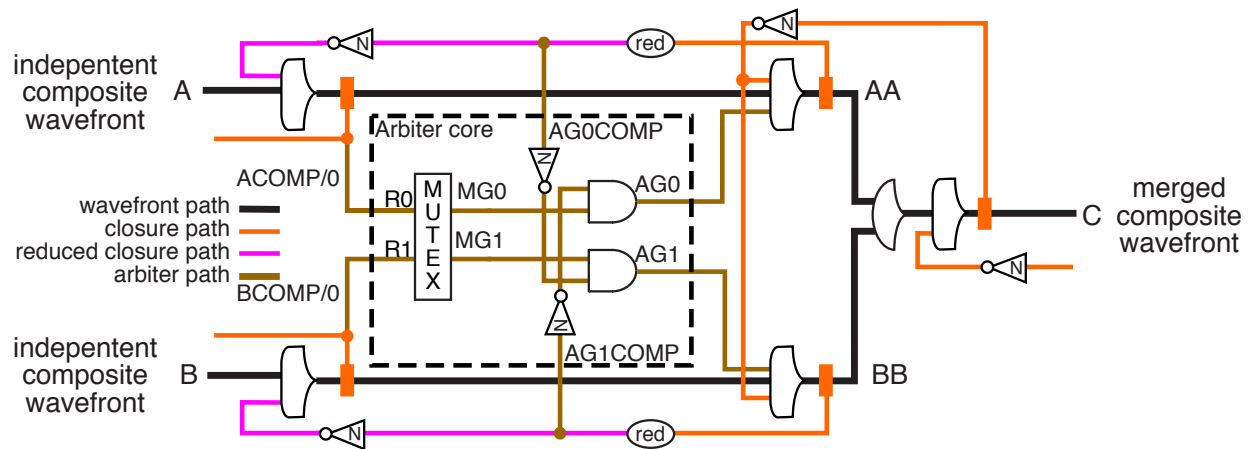
Figure C.12.Composite merge.

---

1. The arbiter core is derived from a standard asynchronous circuit called a tree arbiter the development of which is recounted in Teresa Meng, *Synchronization Design for Digital Systems, (Norwell, Massachusetts, Kluwer Academic Publishers, 1991). pp 158-163.*

Appendix D: Wavefronts and bubbles

Wavefronts flow forward into bubbles. Bubbles flow backward around wavefronts. This difficult to intuit counterflow behavior of wavefronts and bubbles is the fundamental dynamic flow behavior of a network of linked oscillations. Wavefronts and bubbles interact at a link. The link symbol of Figure D.1 is used in the following discussion.
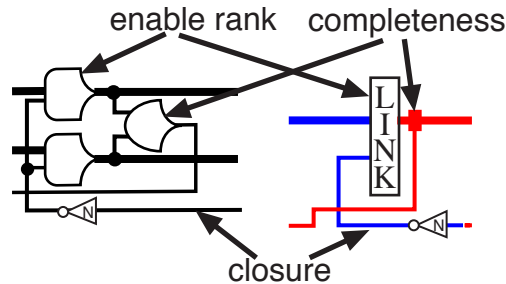
**Figure D.1.The link symbol**

A link presents a **D** wavefront when asserting **D** and its closure is enabling **D**.
A link presents a **D** bubble when asserting **D** and its closure is enabling **N**.
A link presents a **N** wavefront when asserting **N** and its closure is enabling **N**.
A link presents a **N** bubble when asserting **N** and its closure is enabling **D**.
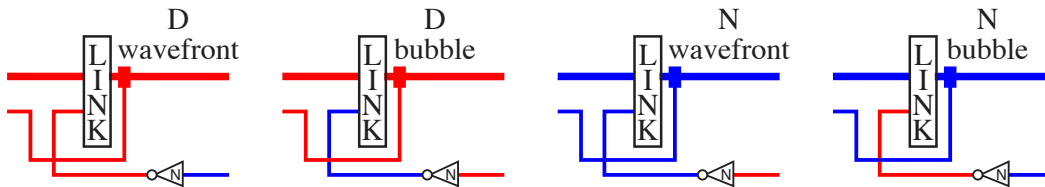
A **D** wavefront cannot flow into a **N** wavefront.
A **D** wavefront can flow into a **N** bubble.
A **D** wavefront should never encounter another **D** wavefront.
A **N** wavefront cannot flow into a **D** wavefront.
A **N** wavefront can flow into a **D** bubble.
A **N** wavefront should never encounter another **N** wavefront.
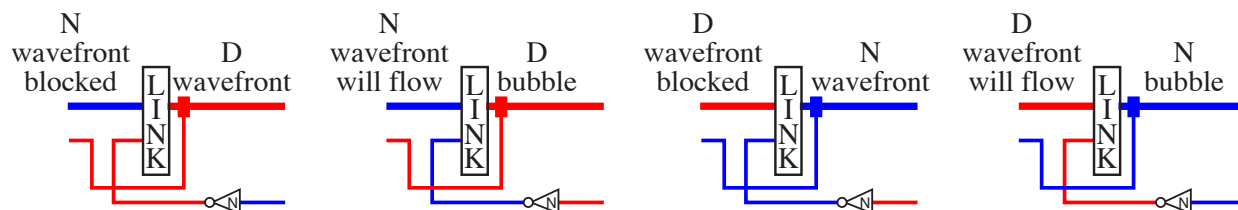
D.1.　Pipeline flow behavior

A **D** wavefront will flow into a **N** bubble. The **N** bubble flows around the **D** wavefront becoming a **D** bubble. A **N** wavefront will flow into a **D** bubble. The **D** bubble flows around the **N** wavefront becoming a **N** bubble. Another way to say it is that **D** Wavefronts flow through an empty background of **N** bubbles trailing **D** bubbles followed by **N** wavefronts flowing through **D** bubbles trailing **N** bubbles restoring the background empty condition of **N** bubbles through which a successor **D** wavefront can flow.

A pipeline network is initialized by an initialization signal to an empty background condition of **N** bubbles. The initialization signal must be held long enough for the initializing **N** wavefront to propagate

through the entire pipeline. The initialization signal is the only aspect of network design that is necessarily associated with an external time referent.
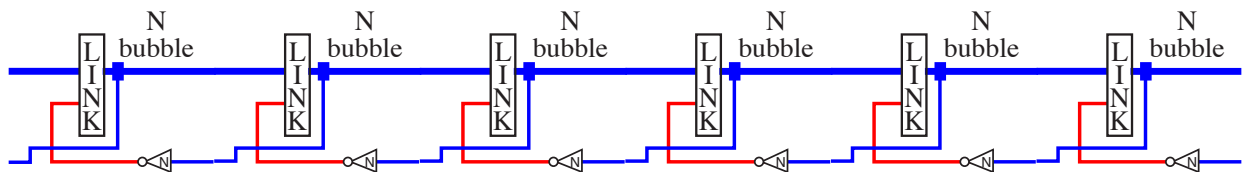
D.1.1.   Pipeline initialization:
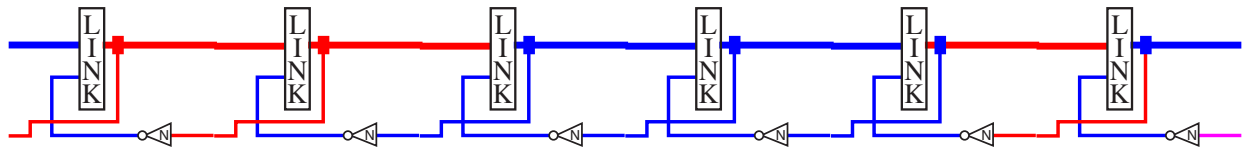
Figure D.2.   Pipeline in random power up state with an initialization signal forcing the converters to assert N.
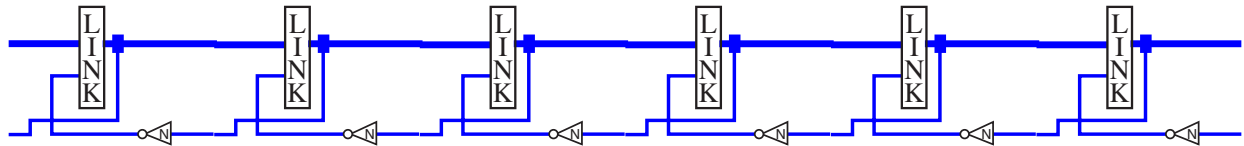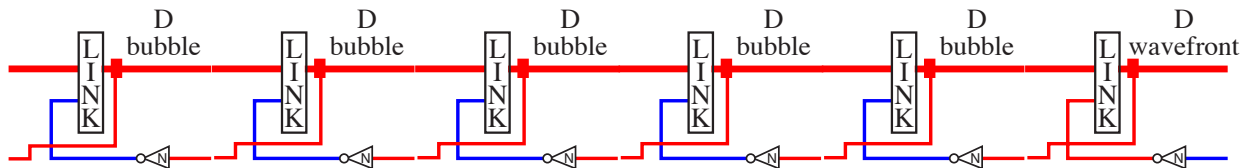

Figure D.3.   A N wavefront presented at the input flows through entire pipeline.

Figure D.4.   Init is released and pipeline becomes filled with N bubbles waiting for a D wavefront.

D.1.2.   Wavefront and bubble flow behavior

In Figure D.5 a **D** wavefront is presented to the input and flows into the **N** bubbles through the entire pipeline leaving behind **D** bubbles waiting for an **N** wavefront. It is not accepted at the output and becomes stalled at the output as a **D** wavefront, .


Figure D.5.   A D wavefront flows through N bubbles.

In Figure D.6 a following **N** wavefront is presented to the input and flows into the **D** bubbles leaving behind **N** bubbles. It cannot flow into the **D** wavefront and is blocked, .
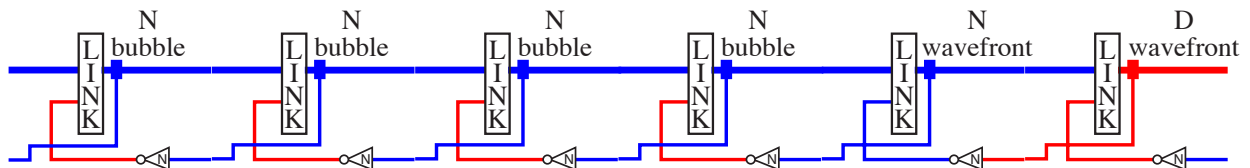

Figure D.6.   A N wavefront flows through D bubbles.

In Figure D.7 successive wavefronts are presented to the input and flow into the pipeline being successively blocked and filling the pipeline with wavefronts waiting for the first **D** wavefront to be accepted.
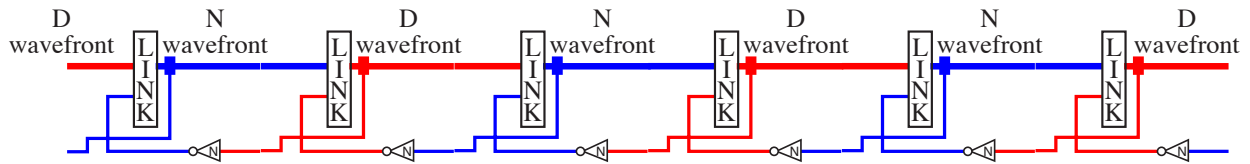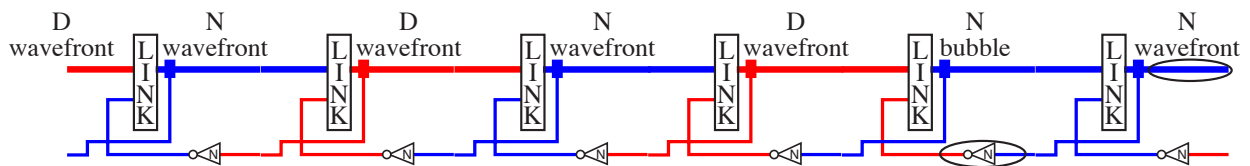

Figure D.7.   Pipeline filled with blocked wavefronts.

In Figure D.8 the first **D** wavefront is accepted and a **D** bubble presents at the output of the pipeline.
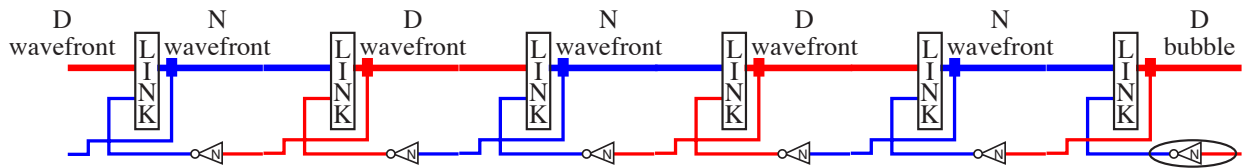
Figure D.8.   Output D wavefront is accepted and D bubble enters pipeline.

In Figure D.9 the **N** wavefront flows into the **D** bubble leaving behind an **N** bubble.

Figure D.9.   N wavefront flows into D bubble.

In Figure D.10 the **D** wavefront flows into the **N** bubble leaving behind a **D** bubble.
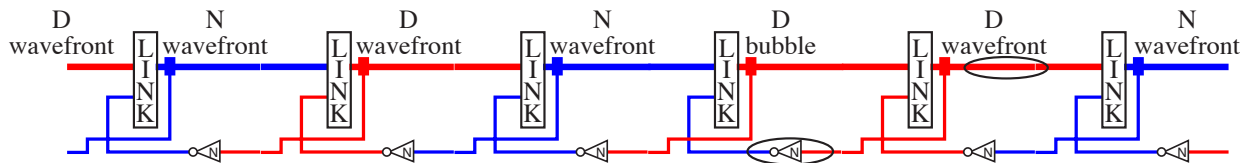
Figure D.10.   D wavefront flows into N bubble.

In Figure D.11 the **N** wavefront flows into the **D** bubble leaving behind a **N** bubble. The rightmost **N** wavefront flows out of the pipeline leaving behind an **N** bubble.
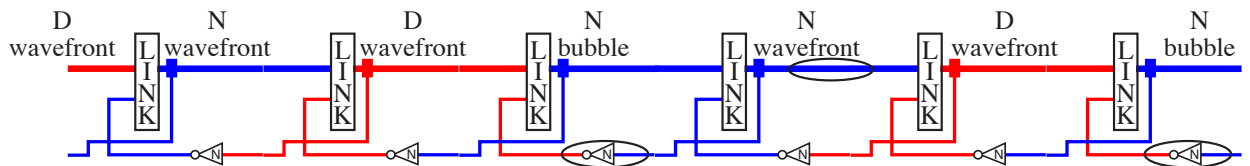

Figure D.11.   N wavefront flows into D bubble.

In Figure D.12 the **D** wavefronts flows into both **N** bubbles leaving behind a **D** bubbles ... And so on.
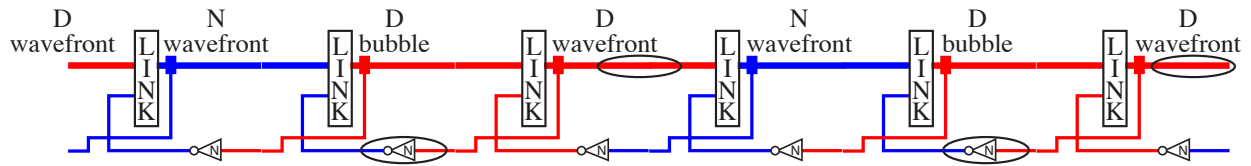
Figure D.12.   D wavefront flows into N bubble.

The bubble, alternating between **N** bubble and **D** bubble, flows backward through the waiting wavefronts in the pipeline. Each wavefront progresses one oscillation with each bubble entering the pipeline and flowing through the wavefronts. As each wavefront flows forward into a bubble a bubble flows backward around the wavefront. In the absence of bubbles wavefronts do not flow. In the absence of wavefronts bubbles do not flow.

Appendix E: Initializing a D wavefront in a pipeline

All interaction networks need to be initialized at inception. A pipeline network of linked oscillations initializes with all conversions forced to **N** and with completely **N** presented to the network inputs which flows through all the links and through each oscillation network to initialize the entire pipeline network to **N** or empty. The init signal, which has to be held long enough for the **N** conditions presented to the input of the network to propagate through the entire network, is the only signal in an interaction network with a specific and extrinsic timing requirement. When the init signal is released, all conversions transition to **D** filling the network with **N** bubbles (Appendix D) enabling the first transition to **D** wavefront to flow into and through the network.

E.1.   Initializing a D wavefront

Occasionally it is necessary to initialize a **D** wavefront in a pipeline network otherwise initialized to all **N**. An individual closure can be initialized to **D** or **N**. Initialization is specified with **D** or **N** in the primitive behavior graphic symbol which is also connected to the init signal. Initializing a **D** wavefront in a pipeline involves two consecutive oscillation networks as shown in Figure E.1. The initializing oscillation network initializes a **D** completeness wavefront across the closure flow path. The oscillation networks receiving the initialized wavefront must block each initialized **D** with an initialized **N** to present a **N** initialization wavefront to the rest of the pipeline. The closures of the two initializing oscillation networks use uninitialized converters. When the init signal is released the **D** wavefront begins spontaneously flowing through the pipeline.
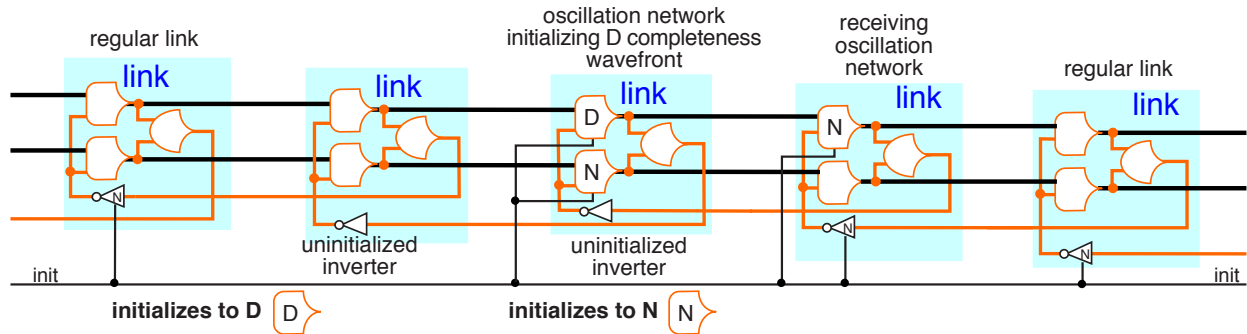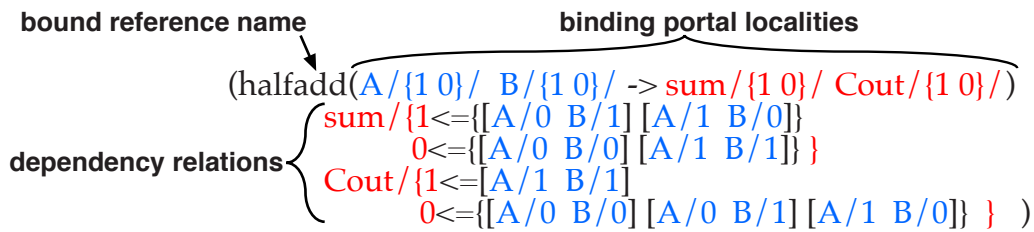

Figure E.1.Initializing a D wavefront in a pipeline network.

Cover the "allof" to D issue the shared closure
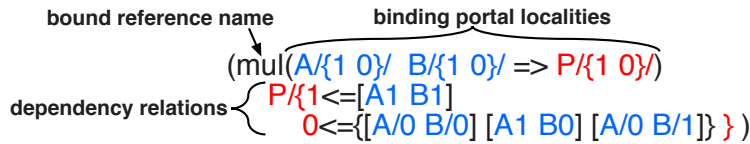
Copyright © 2024 by Karl Fant

Appendix F: 5 bit Multiply interaction network

The **5bitmul** requires two component networks not yet defined **halfadd** and **mul**. The **halfadd** expression below specifies the **halfadd** interaction network in the left of Figure F.1.

bound reference name   **binding portal localities**

$$(\text{halfadd}(A/\{1\ 0\}/\ \ B/\{1\ 0\}/\ ->\ \text{sum}/\{1\ 0\}/\ \text{Cout}/\{1\ 0\}/\ )$$

dependency relations
$$\text{sum}/\{1<=\{[A/0\ B/1]\ [A/1\ B/0]\}$$
$$0<=\{[A/0\ B/0]\ [A/1\ B/1]\}\ \}$$
$$\text{Cout}/\{1<=[A/1\ B/1]$$
$$0<=\{[A/0\ B/0]\ [A/0\ B/1]\ [A/1\ B/0]\}\ \}\ \ )$$

sum is dependent on the dependency relations of its components, **{1 0}** on input localities **A** and **B**. **Cout** is dependent on the dependency relations of its components, **{1 0}** on input localities **A** and **B**

The **mul** expression below specifies the **mul** interaction network in the right of Figure F.1.

bound reference name   **binding portal localities**

$$(\text{mul}(A/\{1\ 0\}/\ \ B/\{1\ 0\}/\ =>\ P/\{1\ 0\}/)$$

dependency relations
$$P/\{1<=[A1\ B1]$$
$$0<=\{[A/0\ B/0]\ [A1\ B0]\ [A/0\ B/1]\}\ \}\ )$$

P is dependent on the dependency relations of its components, **{1 0}** on input locality **A** and **B**.
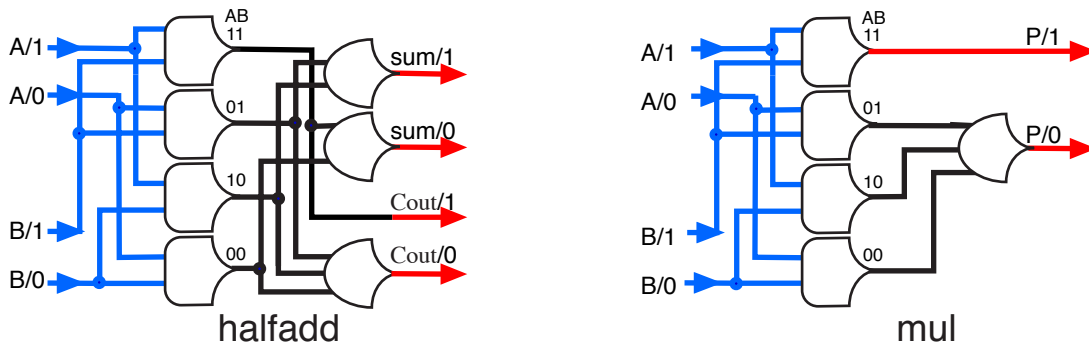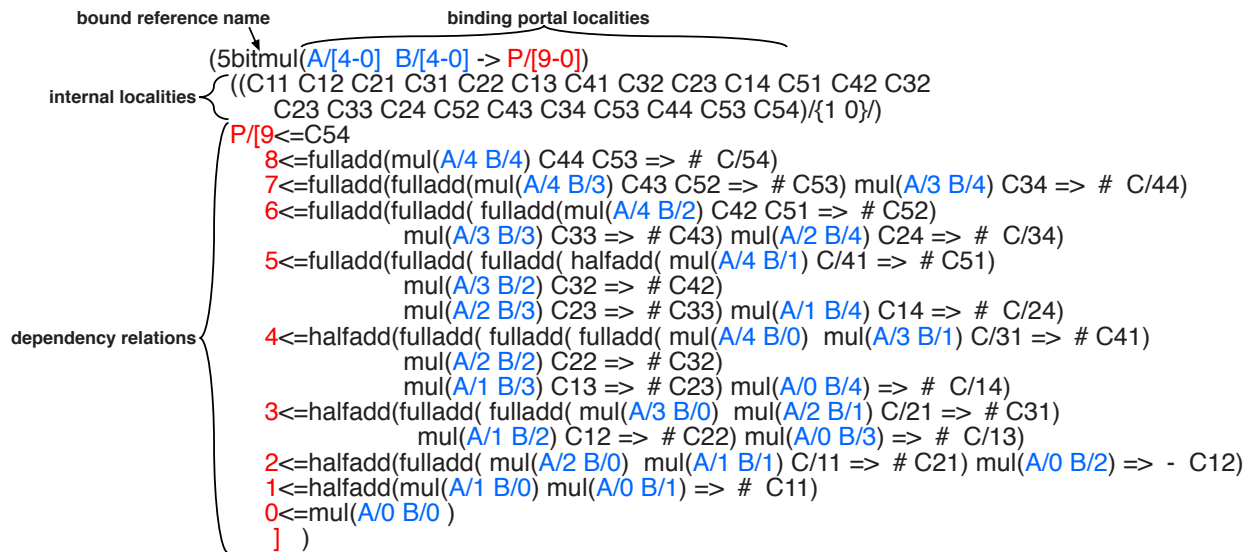

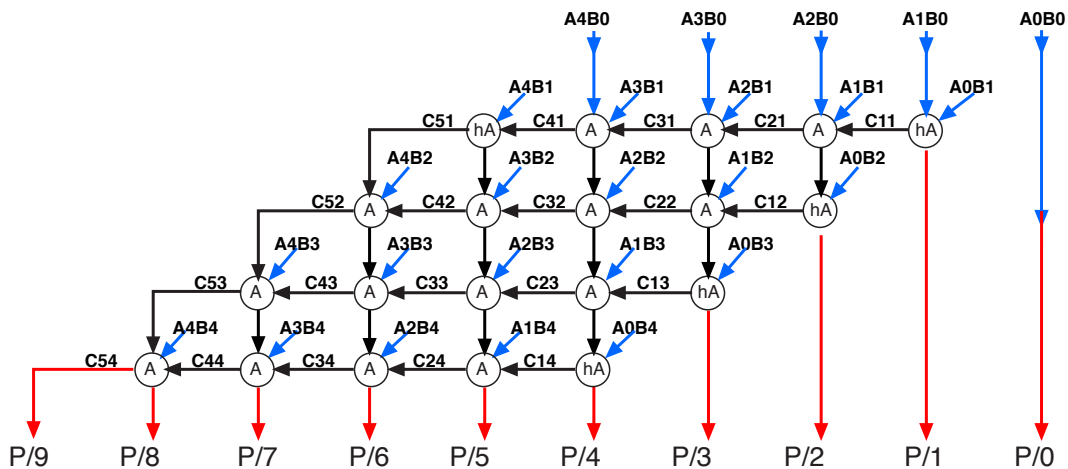**Figure F.1. halfadd and mul component networks**

The **5bitmul** references component networks **fulladd**, **mul** and **halfadd** all of which are constant and fulfill the completeness criterion. The **5bitmul** expression below specifies the 5 bit multiply interaction network of Figure F.2.

bound reference name   binding portal localities

internal localities

dependency relations

(5bitmul(A/[4-0] B/[4-0] -> P/[9-0])
((C11 C12 C21 C31 C22 C13 C41 C32 C23 C14 C51 C42 C32
    C23 C33 C24 C52 C43 C34 C53 C44 C53 C54)/{1 0}/)
P/[9<=C54
    8<=fulladd(mul(A/4 B/4) C44 C53 => # C/54)
    7<=fulladd(fulladd(mul(A/4 B/3) C43 C52 => # C53) mul(A/3 B/4) C34 => # C/44)
    6<=fulladd(fulladd(fulladd(mul(A/4 B/2) C42 C51 => # C52)
            mul(A/3 B/3) C33 => # C43) mul(A/2 B/4) C24 => # C/34)
    5<=fulladd(fulladd(fulladd(halfadd(mul(A/4 B/1) C/41 => # C51)
            mul(A/3 B/2) C32 => # C42)
            mul(A/2 B/3) C23 => # C33) mul(A/1 B/4) C14 => # C/24)
    4<=halfadd(fulladd(fulladd(fulladd(mul(A/4 B/0) mul(A/3 B/1) C/31 => # C41)
            mul(A/2 B/2) C22 => # C32)
            mul(A/1 B/3) C13 => # C23) mul(A/0 B/4) => # C/14)
    3<=halfadd(fulladd(fulladd(mul(A/3 B/0) mul(A/2 B/1) C/21 => # C31)
            mul(A/1 B/2) C12 => # C22) mul(A/0 B/3) => # C/13)
    2<=halfadd(fulladd(mul(A/2 B/0) mul(A/1 B/1) C/11 => # C21) mul(A/0 B/2) => - C12)
    1<=halfadd(mul(A/1 B/0) mul(A/0 B/1) => # C11)
    0<=mul(A/0 B/0)
    ])

5bitmul inherits locality structure in the same way as the 5bitadd above.

(5bitmul(A[4-0]/{1 0}/ B[4-0]/{1 0}/ => P[9-0]/{1 0}/) (…) …)

P is dependent on the dependency relations of its components, **[9-0]** on input localities **A** and **B** and the internal **Cxx** localities. The detailed relations are too involved to verbalize.

**Figure F.2. 5 bit multiply interaction network.**

Each **fulladd**, **halfadd** and **mul** component network in the **5bitmul** network is constant and fulfills the completeness criterion so **5btmul** as a whole is constant and fulfills the completeness criterion. If any part of **A** or **B** is not completely formed a component network input will not completely form, the component network will not transition its output and a part of **P** will not transition. If **P** transitions to completeness it means that the inputs **A** and **B** have transitioned to completeness.

Appendix G: The orphan delay risk

With the emergence of completeness closure and the oscillation network of section 3.7 the orphan risk can be characterized. The delay risk of the orphan is in the **N** wavefront not fully propagating before the next transition to **D** wavefront arrives at an orphan branch. The next transition to **D** wavefront cannot arrive at an orphan branch until the transition to **D** wavefront is allowed into the network by the closure. Consequently, successive **D** wavefronts cannot flow faster than the oscillation period of the network. The delay risk of the orphan can be evaluated by determining:

1. that all orphan branches are isolated to a single branch path within the interaction network and do not propagate through a primitive interaction behavior,
2. That the shortest orphan branch delay contributes to the *shortest oscillation period* of the network so the delay risk is how much longer the longest orphan path delay is than the shortest orphan branch delay,
3. that the *orphan delay risk*, the shortest orphan branch delay subtracted from the longest orphan branch delay, is considerably less than the shortest period of the oscillation network encompassing the orphans ensuring that the longest orphan branch delay will complete its transition to **N** before the next wavefront of transition to **D** can arrive at the orphan branch.

Orphan branches are local and are a component of the oscillation period. A too long orphan branch delay can be contrived but, assuming a local uniformity of manufacturing variations over branches and behaviors, all local component delays will vary similarly and the orphan branch delays will always be proportionally shorter than the oscillation period. In Figure G.1 the longest green path has to propagate faster than the shortest purple path.
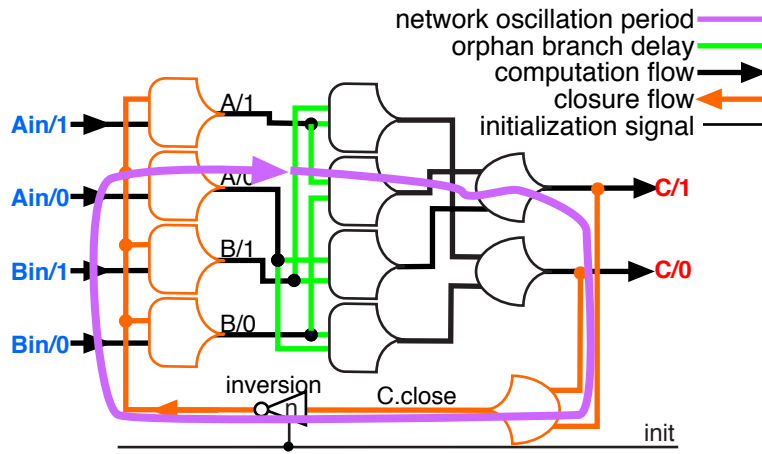
Figure G.1. Orphan risk: shortest oscillation period to longest orphan branch delay.

Orphan delays are timing relations relative to local delays within the network in contrast to timing relations critically relative to a global external time referent or to some indeterminate environmental delay (isochronic fork). It is generally sufficient to ensure that all orphan branches do not propagate through a primitive behavior to mitigate the orphan delay risk.