

# Presentation Slides

## Chapter 3

### The Structure of Logically Determined Systems

#### Logically Determined Design: Clockless System Design With NULL Convention Logic

by Karl Fant

John Wiley & Sons, Inc.

Introduce cycles and cycles structures

Diagrams by permission of John Wiley & Sons, Inc.

# The Fundamental Structure

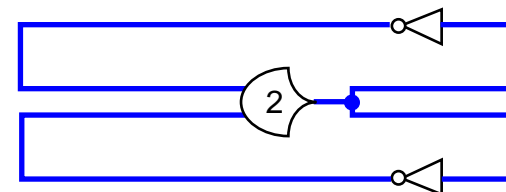
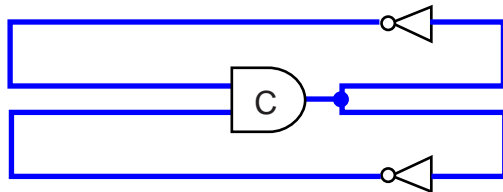
## Ring oscillators coupled through shared completeness paths.

A ring oscillator (a closed circuit with an odd number of inversions) is spontaneously alive continually propagating a monotonically transitioning binary signal around its ring.

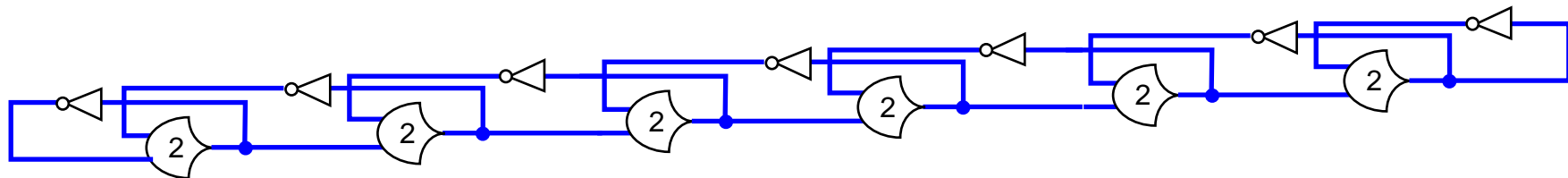


ring oscillator

Ring oscillators can be synchronized through a shared completeness path that demands complete transition of all inputs before outputting a transition that will continue around each oscillator (The shared completeness path implements the completeness criterion). The C-element and the 2 of 2 operator synchronize transitions in both directions.

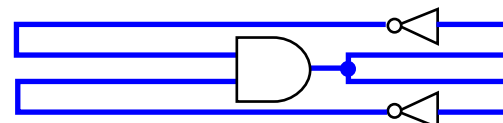
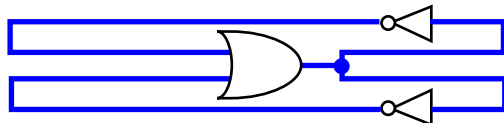


A linear progression of coupled ring oscillators forming a pipeline



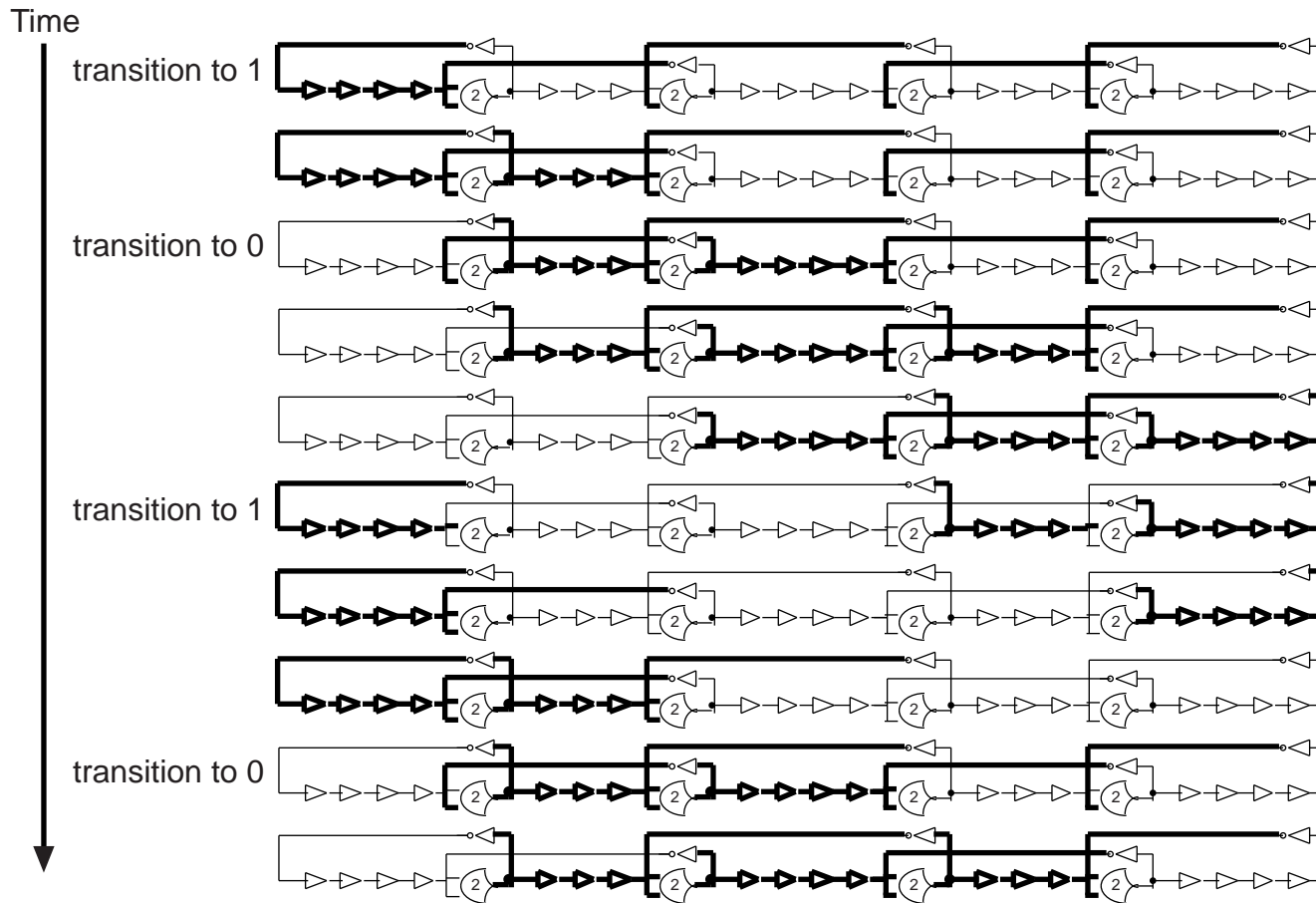
---

Boolean operators synchronize transition behavior in only one direction and do not work.



# The Fundamental Behavior

Spontaneously flowing wavefronts of transition.



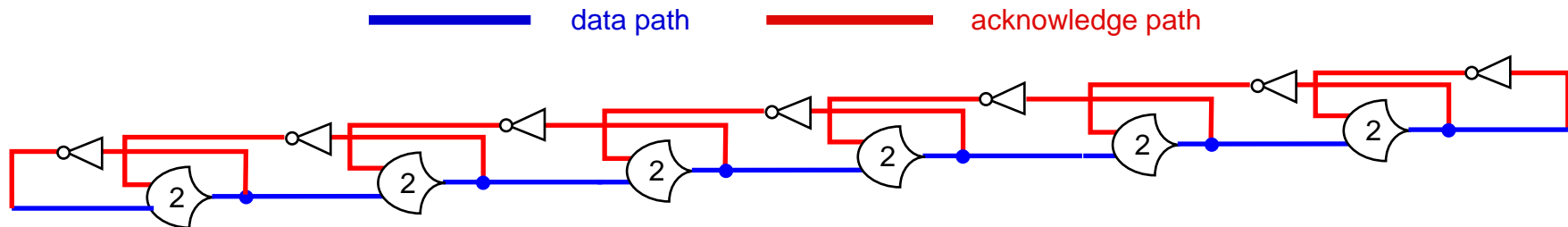
# Convenient Conventions

The portion of each oscillator over which transition flows from oscillator to oscillator we will call the data path

The portion of each oscillator which flows counter to this progression and includes the inversion we will call the acknowledge path.

Of the two transition states we will call one NULL and the other DATA.

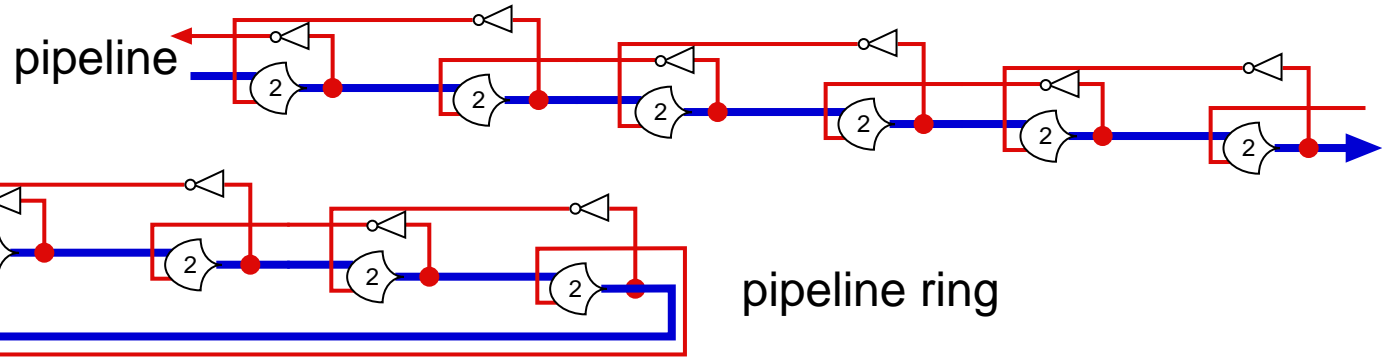
Each ring oscillator, now viewed in terms of a data path and an acknowledge path alternately transitioning between NULL and DATA will be called a cycle.



## Rules of NCL cycle composition

1. Every signal path in a system must be part of a cycle.
2. Every cycle must have an odd number of inversions. It is convenient to assume that every cycle has a single inverter in the acknowledge path.
3. Shared cycle paths must be completeness paths.
4. All signals presented to a completeness path must be in the same monotonic transition phase.

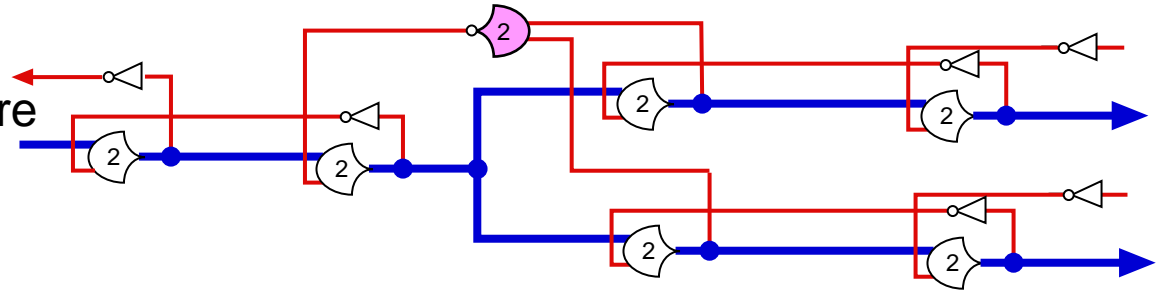
# Cycle Structures



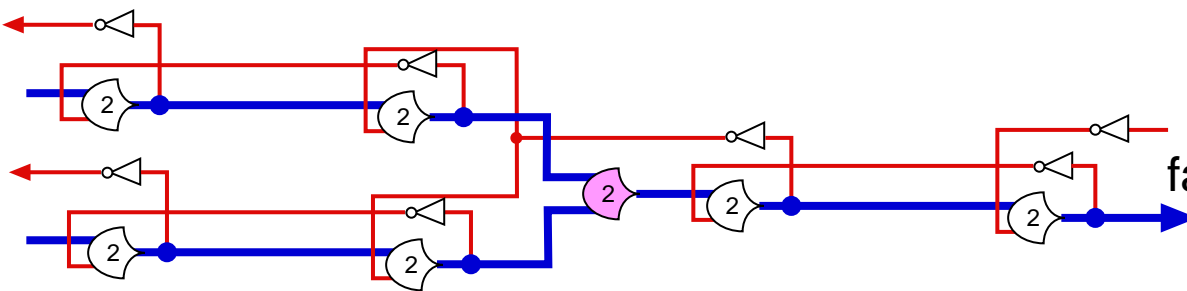
Pipelines can be further composed into more complex structures through shared completeness paths

 shared completeness path

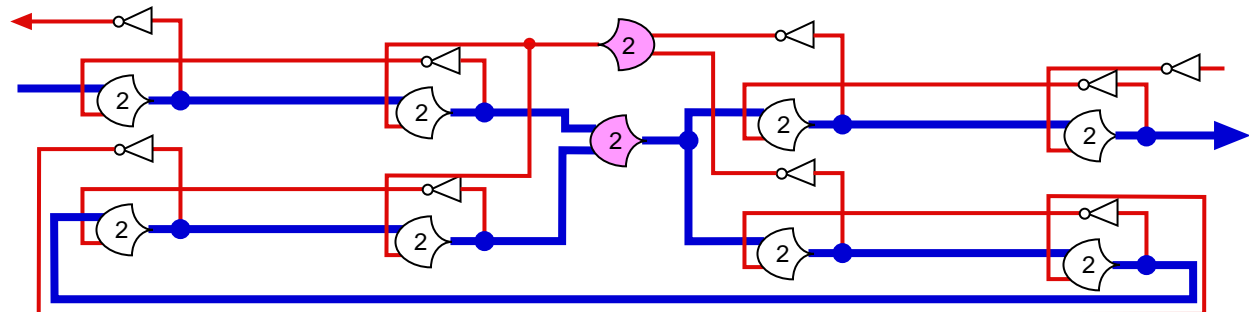
fan-out pipeline structure



fan-in pipeline structure



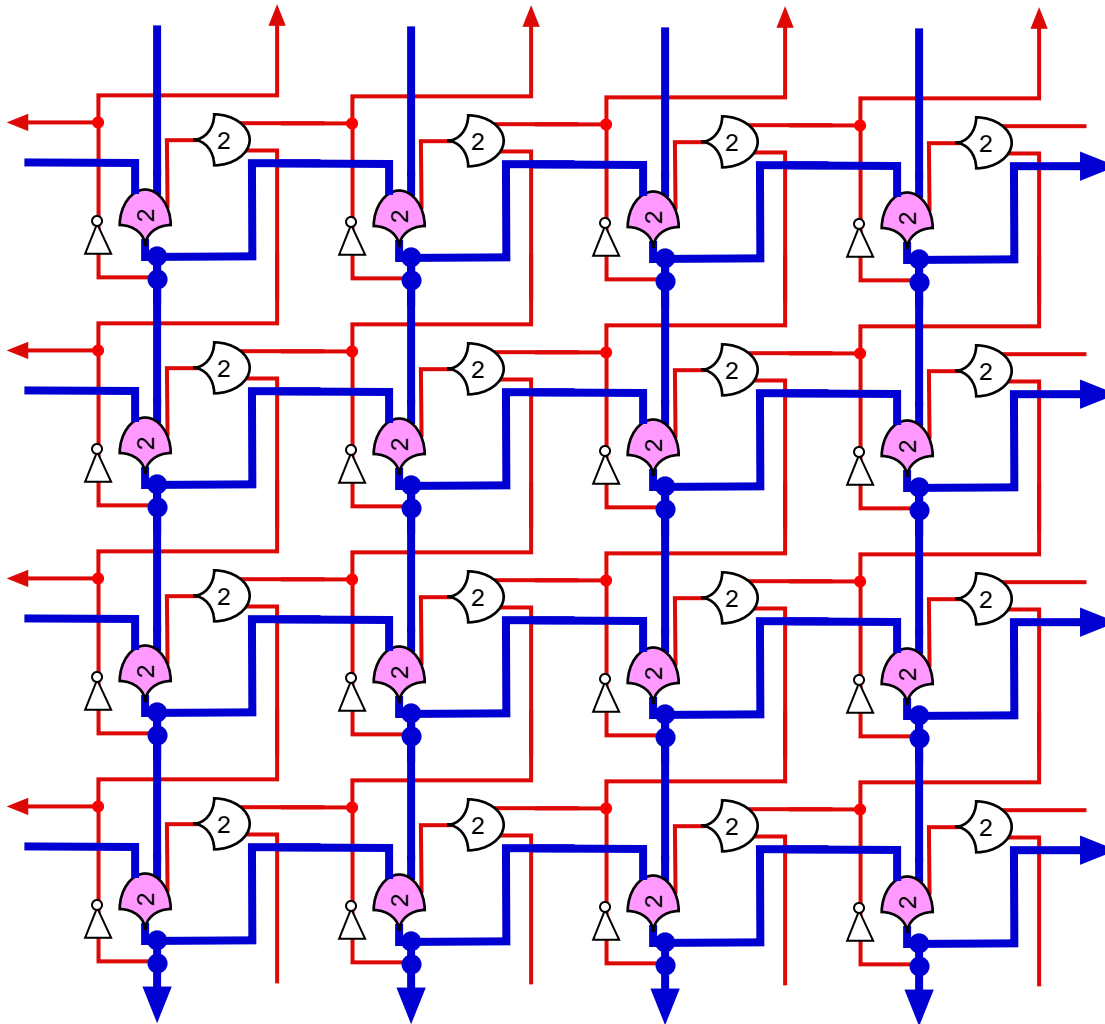
fani-in/fan-out structure coupling a ring to a pipeline



# A 2 DIMENSIONAL ARRAY OF CYCLES

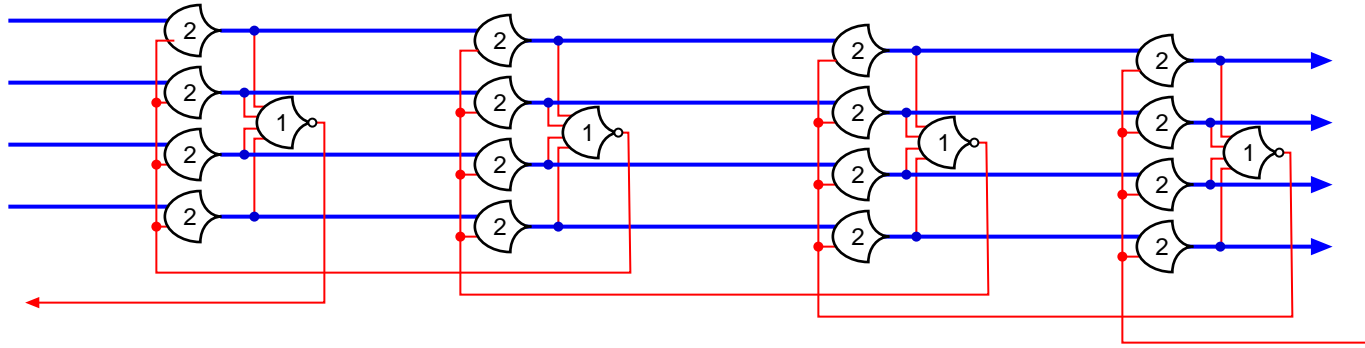
Each pink operator represents a shared completeness path which may be any combinational expression. Wavefronts are presented to the top and left edges. Resolution flows diagonally from top left to bottom right.

This is the basic model for 2 dimensional pipelining.



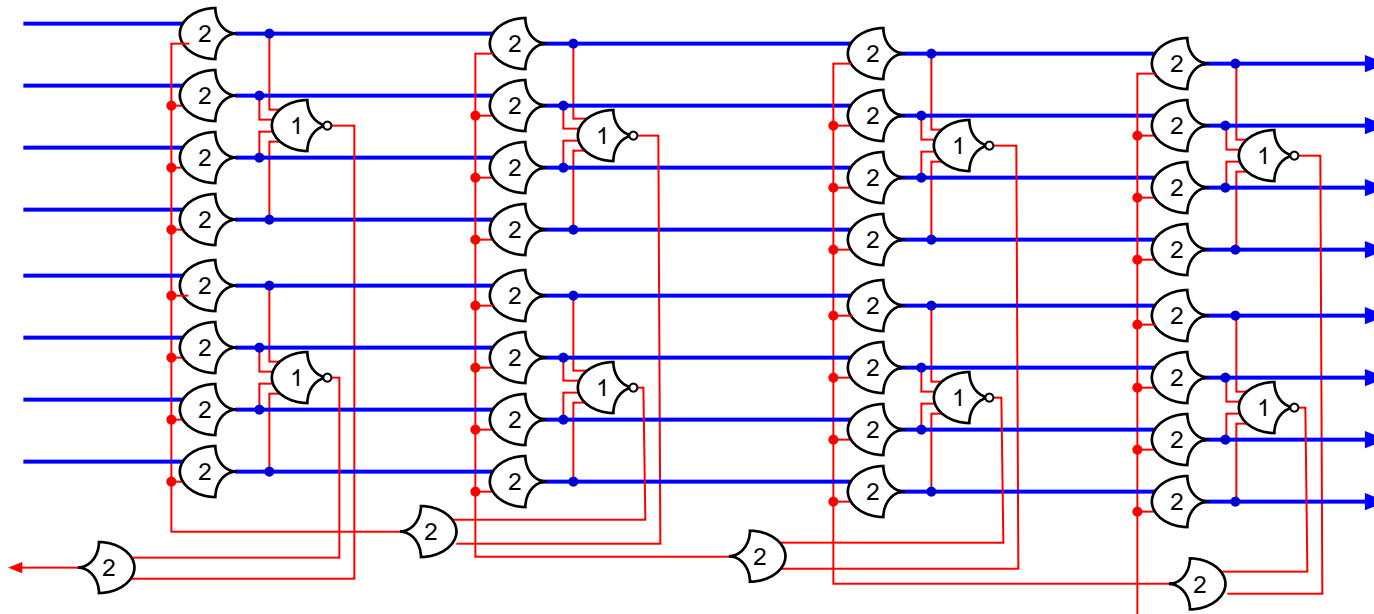
# Multi-value Variable Cycles

Several data paths, only one of which will transition at a time will form a data path of a single cycle, a single oscillating unit that is a multi value variable.



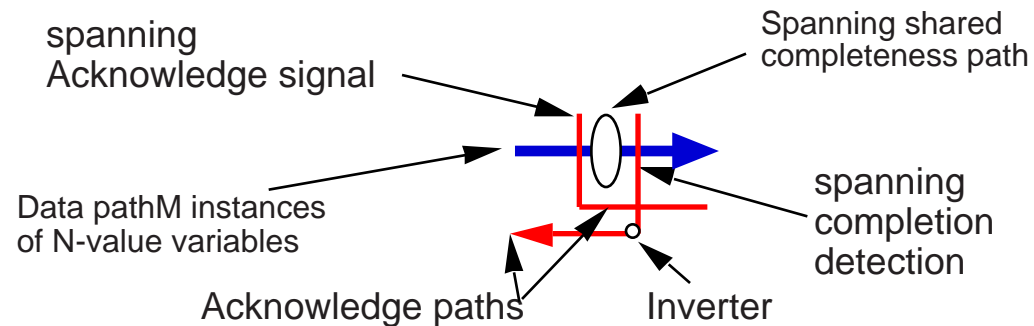
# Multi Variable Data Path

Enforcing a completeness spanning many multi-value cycles forms a multi variable data path.

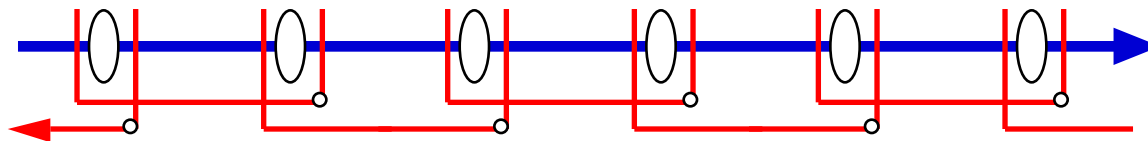


# Abstracting the Data Path

## One graphic completeness stage



## Graphic pipeline of unspecified data path width with 6 completeness stages



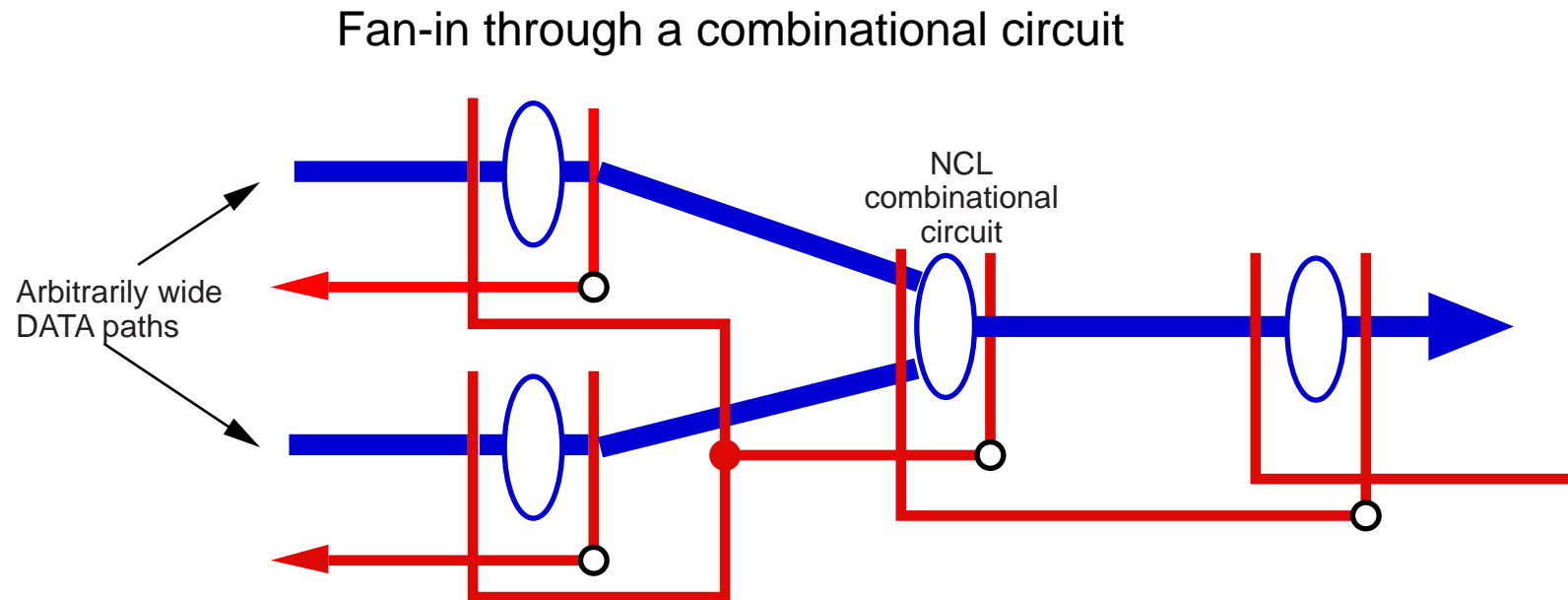
## Composition in terms of completeness stages

1. Each completeness stage must acknowledge every completeness stage that contributed to its data path.
2. Each completeness stage must be acknowledged by every completeness stage to which it contributed a data path.



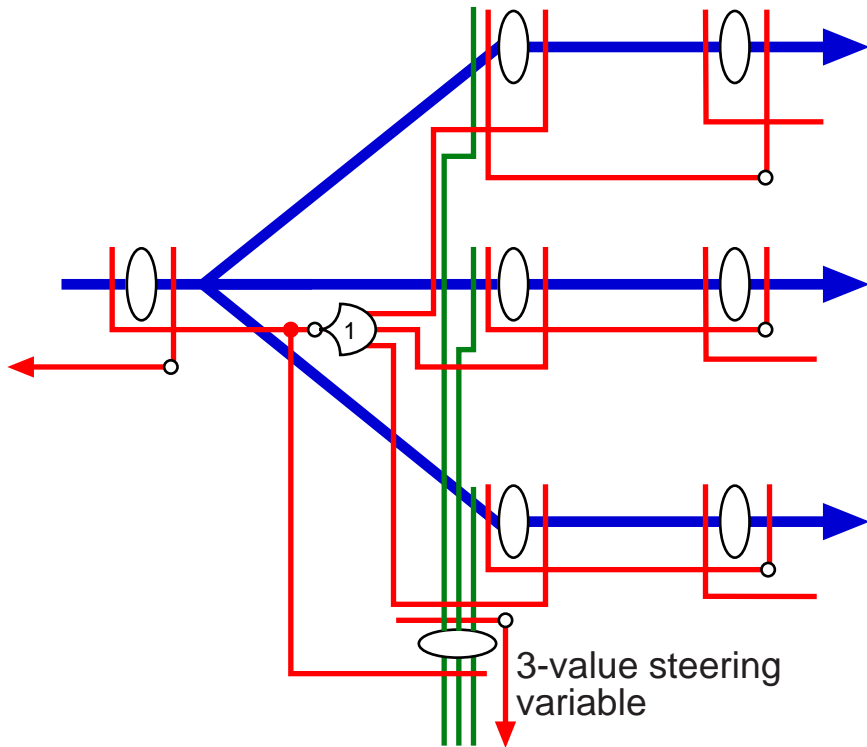
# An NCL Combinational Circuit is a Shared Completeness Path

NCL combinational circuits enforce the completeness of input criterion and the circuit itself is a shared completeness path

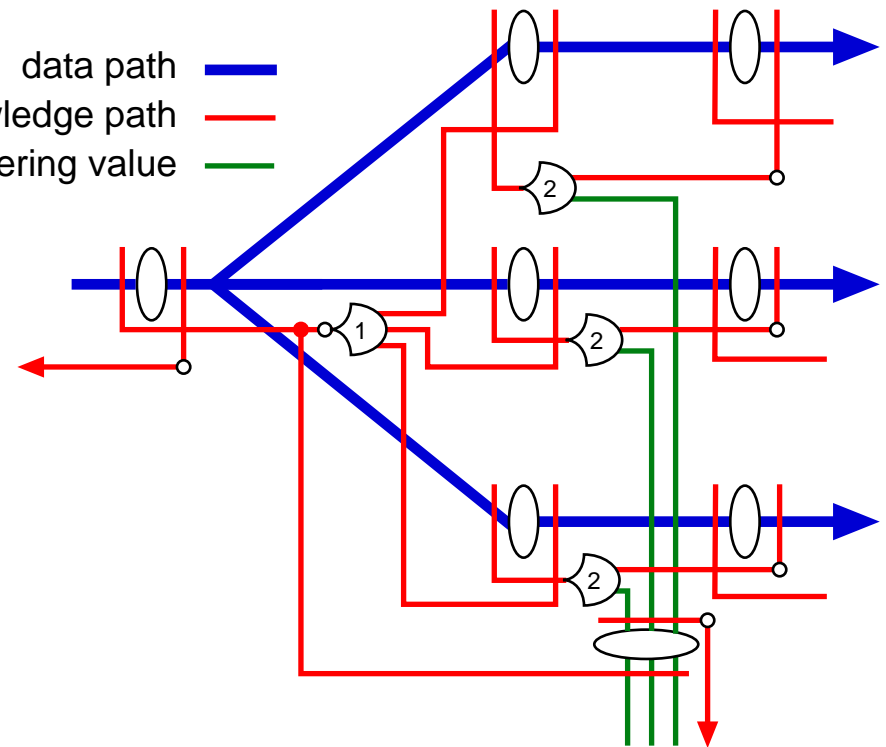


# Wavefront Fan-out Steering Structures

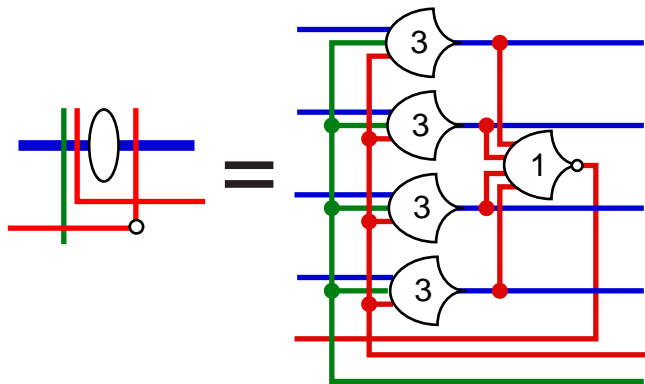
Each Nth steering variable will enable the Nth input data wavefront through one of A, B or C.



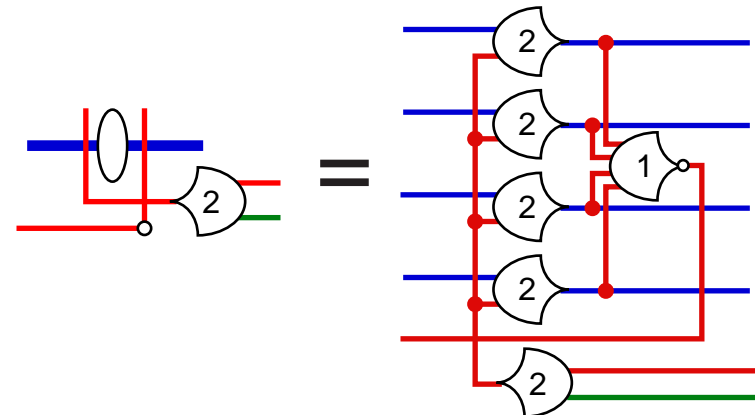
steering variable spanning the data path



steering variable combined with acknowledge path



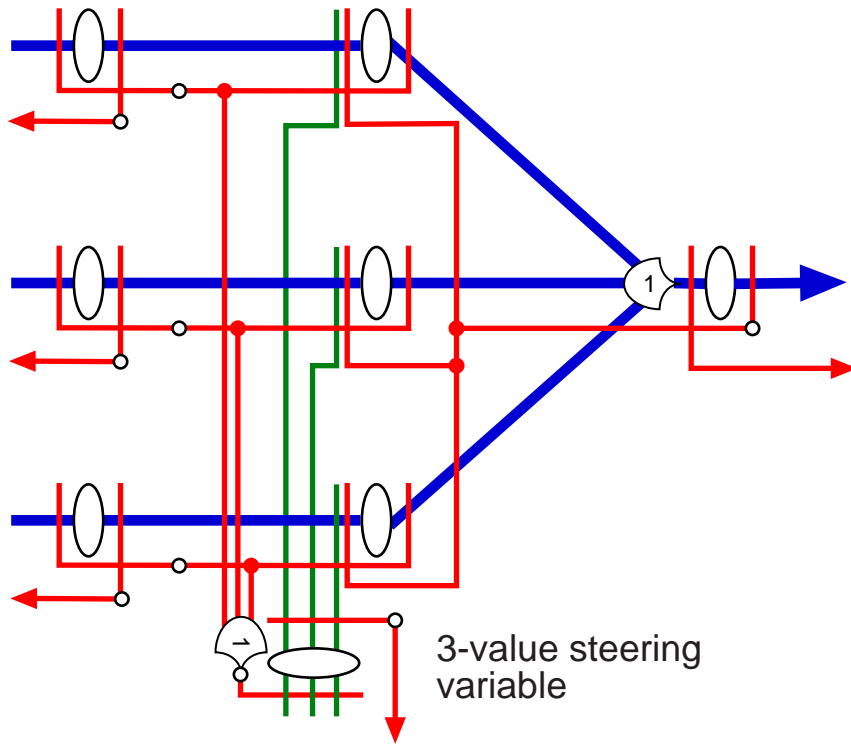
steering value presented to regulator



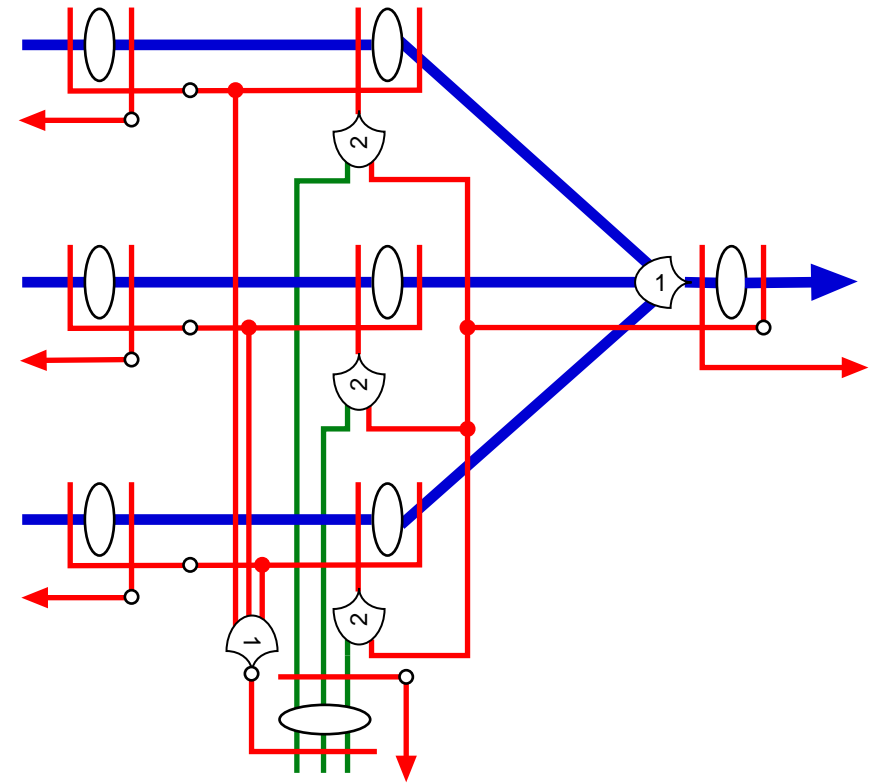
steering value precombined with acknowledge

# Wavefront Fan-in Steering Structures

Each Nth steering variable will enable the Nth data wavefront from one of A, B or C through the output.



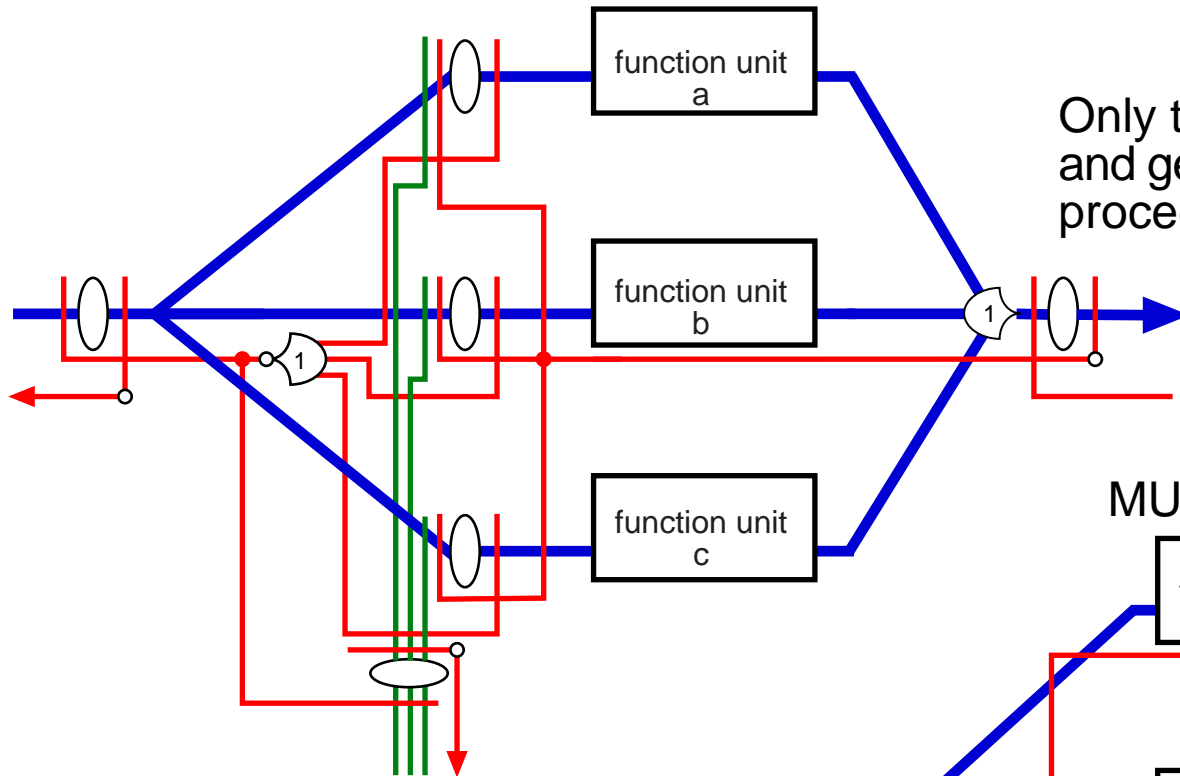
steering variable spanning the data path



steering variable combined with acknowledge path

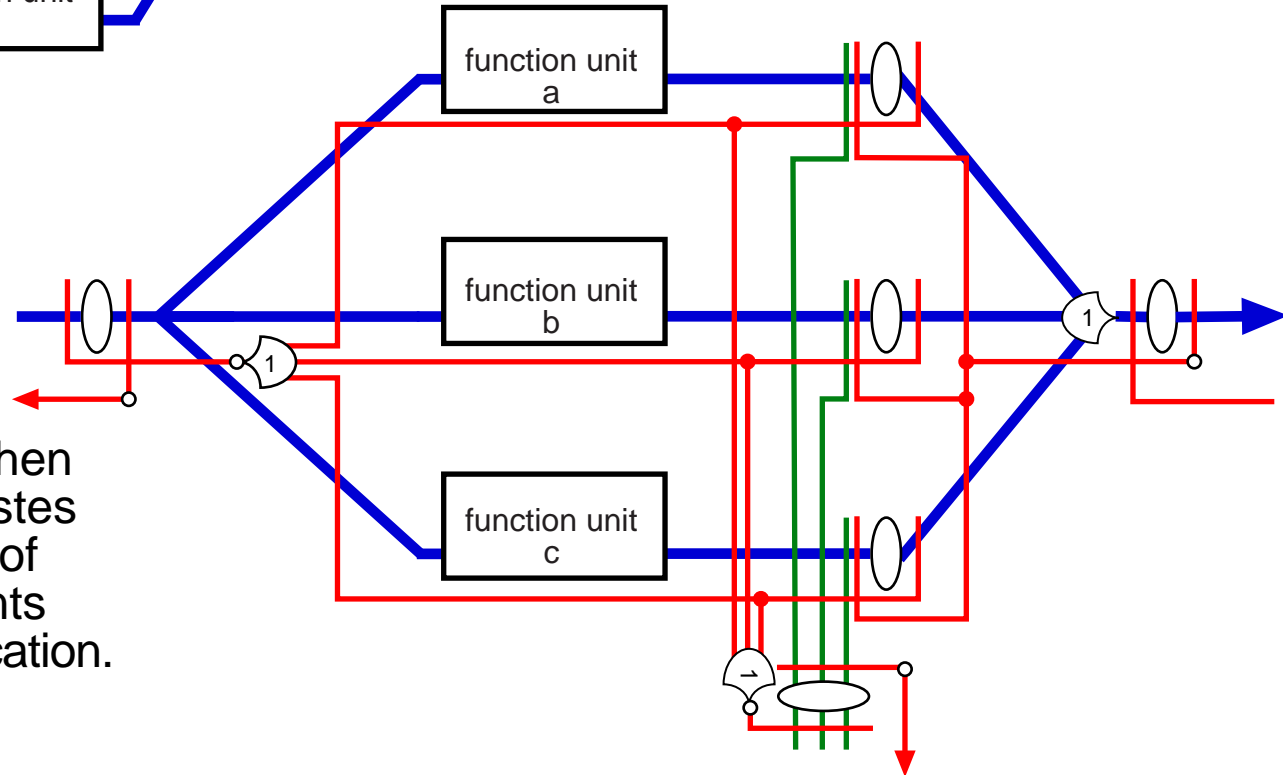
# DEMUX Oriented Fan-out/Fan-in

DEMUX oriented: presteer data flow paths.



Only the selected function occurs and generates a result wavefront that proceeds through the output.

MUX oriented: post select results.



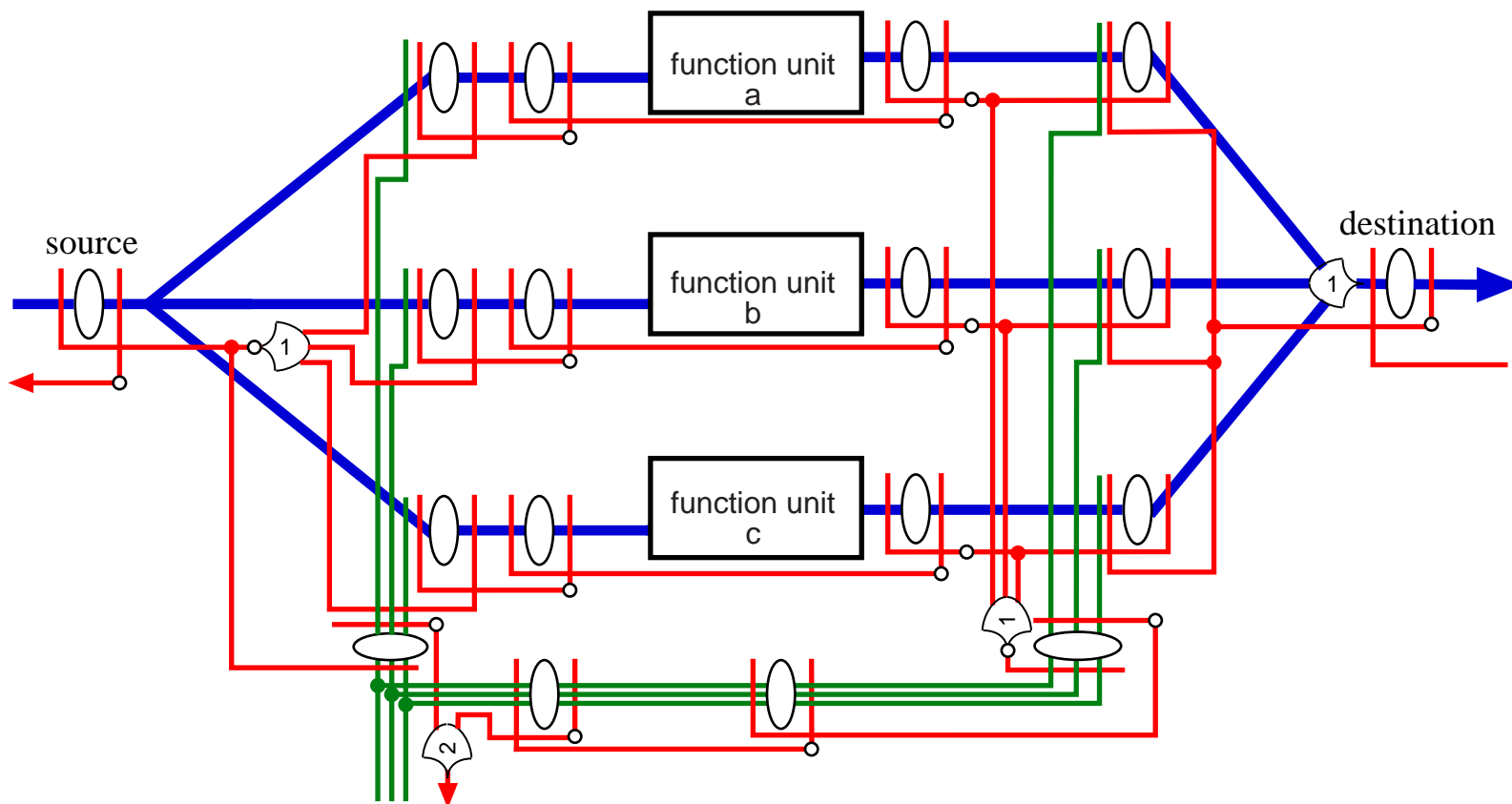
All three functions occur and then the result is selected. This wastes behavior and raises the issue of discarding unwanted wavefronts which is an unnecessary complication.

# Pipelined Fan-out/Fan-in

The steering variable is pipelined along with the result data flow and selects the fan-in in the same way it selected the fan-out at the input. The Nth steering variable steers the Nth input data into the structure and steers the Nth result out of the structure

Results are generated at the output in the same order that the input was presented.

The behavior of the structure is completely independent of the relative speeds of the function pipelines.



# Why Initialize?

Each operation in a system must begin in a stable cooperation state with its neighbors.

Every operator in the system must be set to a known stable output state to establish this initial state of cooperation.

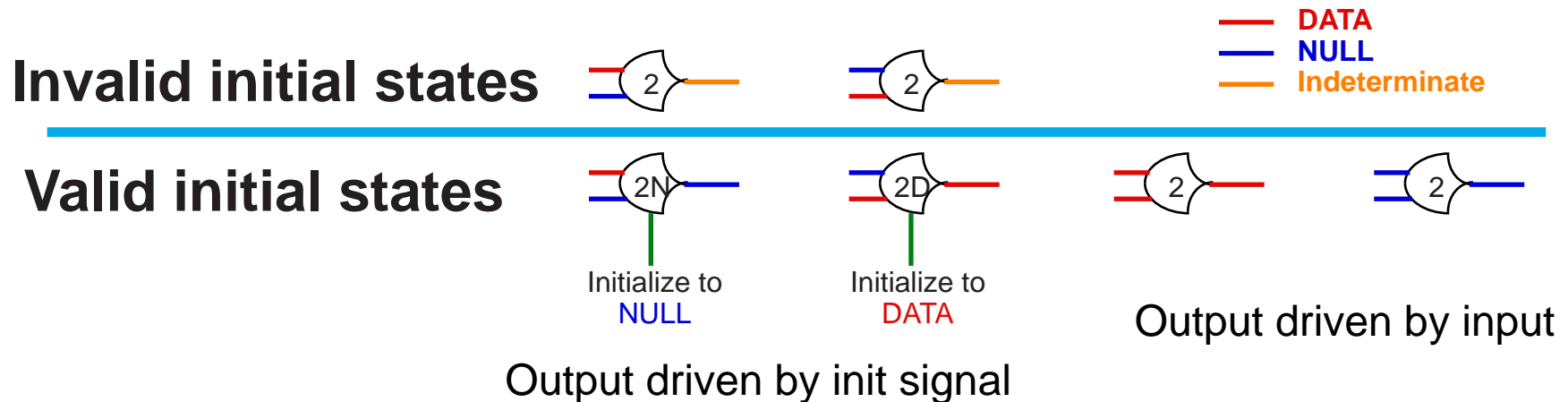
This can be done by flowing initial state with a minimum of explicitly initialized gates.

This initialization procedure is not delay insensitive and must be timed to insure the completeness of initialization.

# Initializing Operators

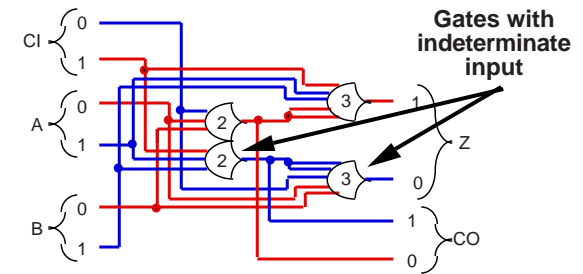
Any NCL operator can be forced to the desired output by presenting it with the appropriate unambiguous input. **DATA,DATA** and **NULL,NULL** or with an explicit initialize signal.

For 2 of 2 operators presented with **DATA,NULL** or **NULL,DATA** the output of the operator could be either **DATA** or **NULL** depending on its history. The problem is that a newly instantiated operator has no history and does not know what output to assert and could even become metastable. In this circumstance the operator must be explicitly forced with a initialize signal to assert the desired output.



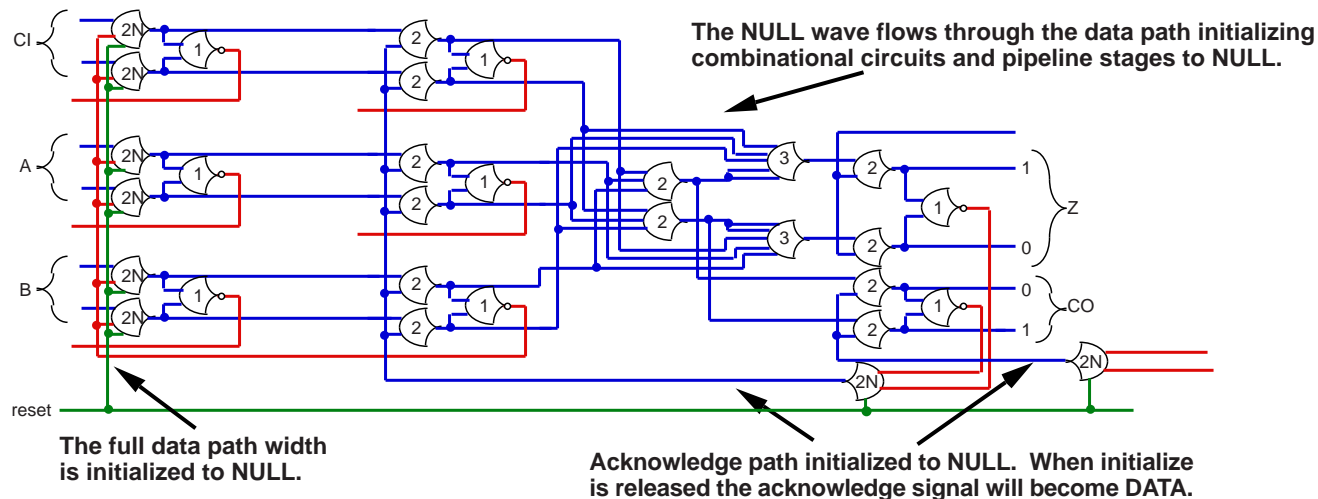
# Initializing Combinational Circuits

Initialization cannot be done with legal DATA input presented to a circuit. Each orphan path in the circuit will present an non-determining input to an operator.



A circuit can be forced to a stable initial state by having an initialize signal on every operator, by presenting an illegal DATA input (all DATA) or by presenting a legal all NULL input. It is undesirable to have an initialize signal on every operator and it is undesirable to begin circuit behavior with illegal DATA states. All NULL input is the only viable choice.

There can be a full width initialization to NULL at the beginning of a data path, then the NULL wavefront can flow through the entire data path.



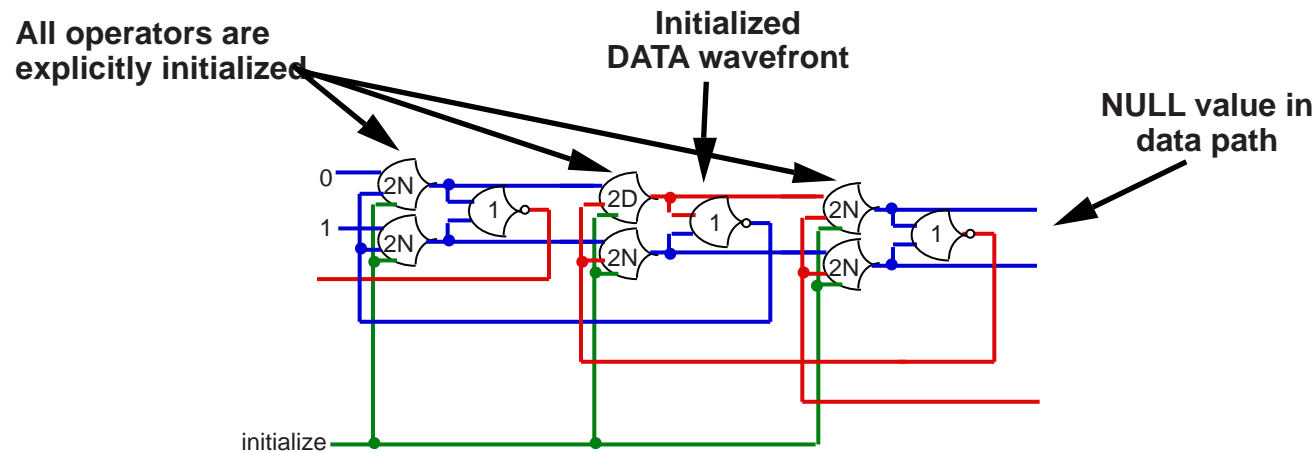
The initialize state has to be maintained long enough for NULL to propagate over all wires in the circuit. There is no way to logically test for completion of initialization. It must be timed



# Initializing a DATA Wavefront

Initializing a DATA wavefront puts a DATA wavefront into the data path that is otherwise being initialized to NULL. The altered phase relationships must be properly considered. For instance, a DATA wavefront must not be presented to a combinational circuit.

By surrounding the initialized DATA wavefront with explicit initialization to NULL we isolate the phase differences and present the greater circuit with NULL values in the data path.



This page intentionally blank

This page intentionally blank

This page intentionally blank