

# The NCL Orphan

An effective flow path causes a logically observable transition at a completeness boundary

An orphan path is an ineffective branch of an effective flow path whose transition behavior is not logically observable.

So the transition behavior of an orphan must be characterized ("observed") as a temporal relation with the effective observable flow path.

Due to the glitch free monotonic behavior of NCL signals the observable flow path extends around the oscillation back to the branch node.

# The Orphan Illustrated

The values trinary/1 and binary/1 flow to and through the expression

The green paths are the ineffective orphan branches that will not cause a transition at the completeness boundary.

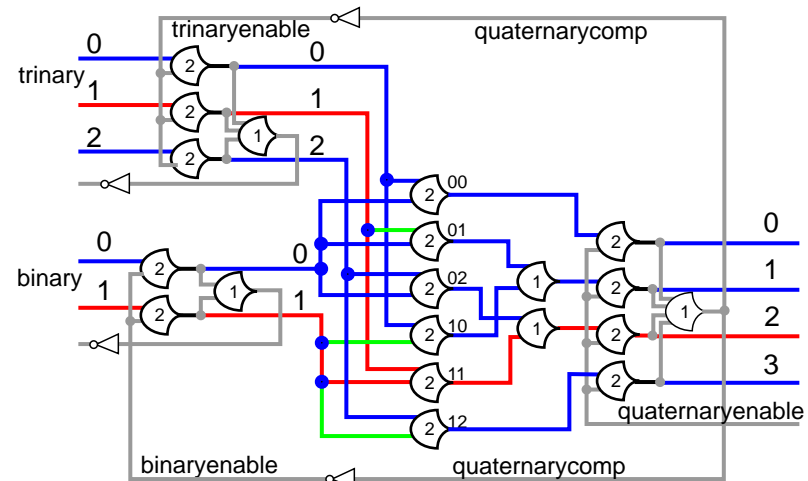
When the 2 output path transitions to data it is determined that the entire red path has transitioned to data.

When the 2 output path transitions to null it is determined that the entire red path has transitioned to null.

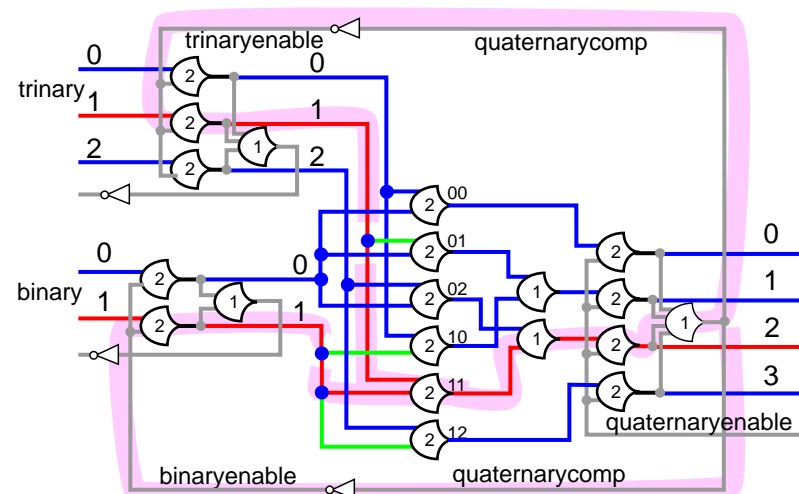
But it is not determined that the green orphan paths have transitioned to null.

The transition behavior of the green paths must be characterized with the timing assumption that they become null before the next data wavefront arrives.

An orphan path must propagate faster than a transition period of the oscillator enclosing the effective flow path.



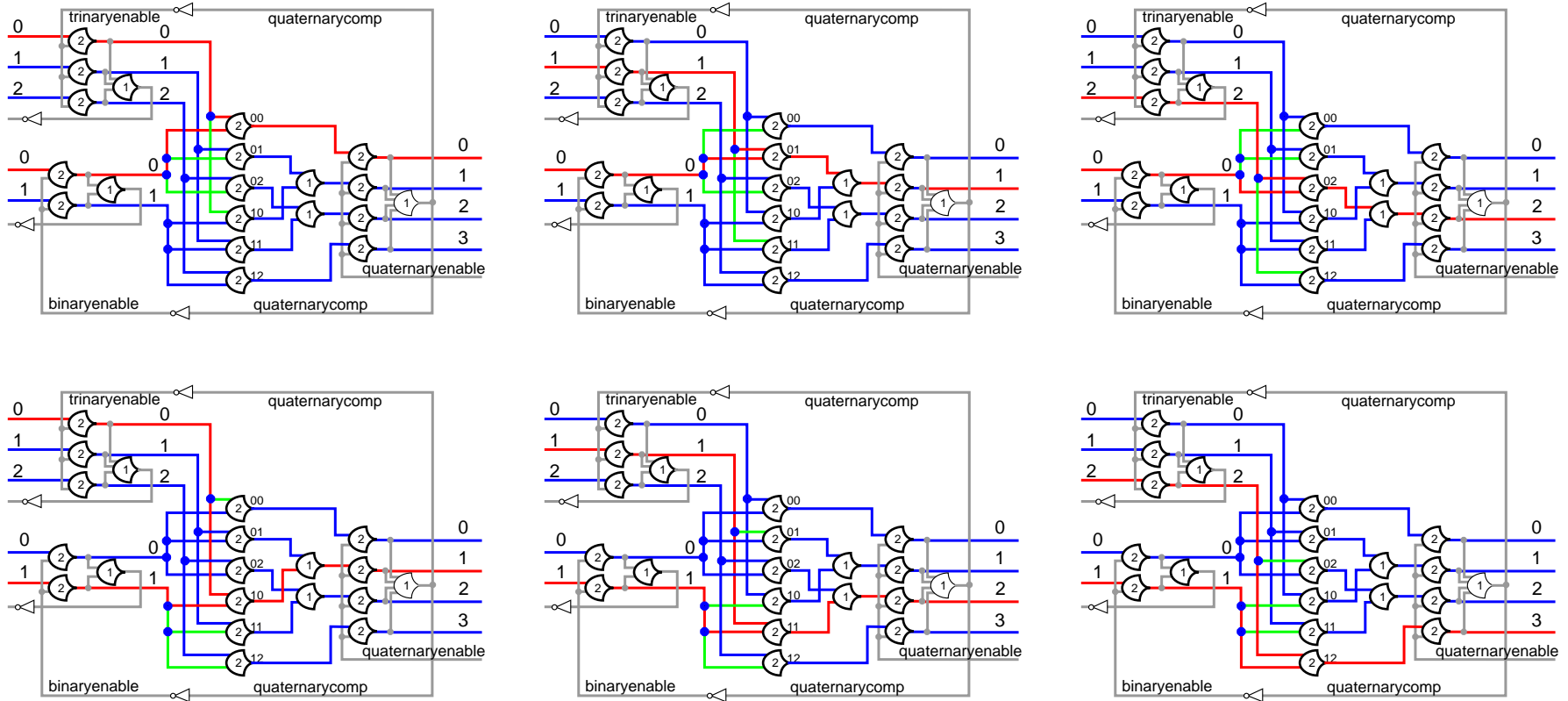
The green paths are the ineffective orphan paths not observable at the output



The green paths must propagate strictly faster than the pink highlighted paths

# The Dynamic View of the Orphan

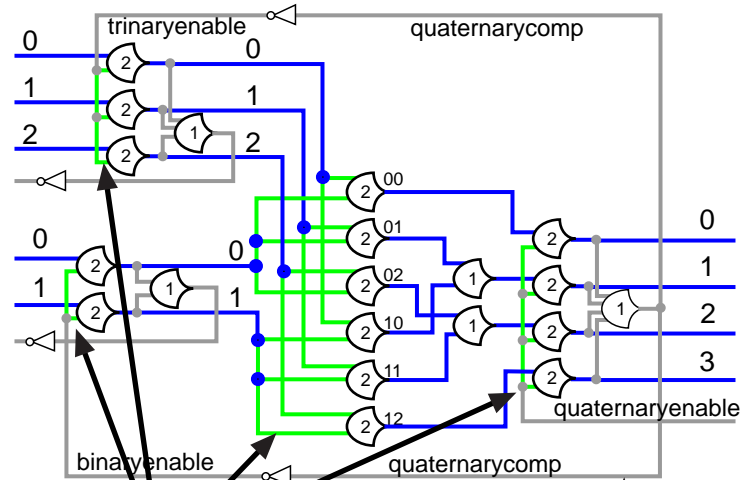
For every possible input there is an effective flow path with ineffective orphan branches



Every orphan branch is an effective flow path for at least one input

# The Static View of the Orphan

All possible orphan paths

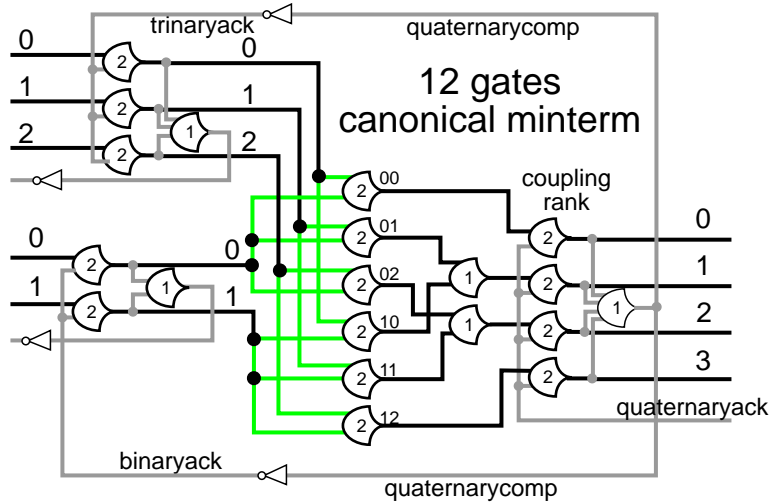


The green paths must propagate within one period of the enclosing oscillation

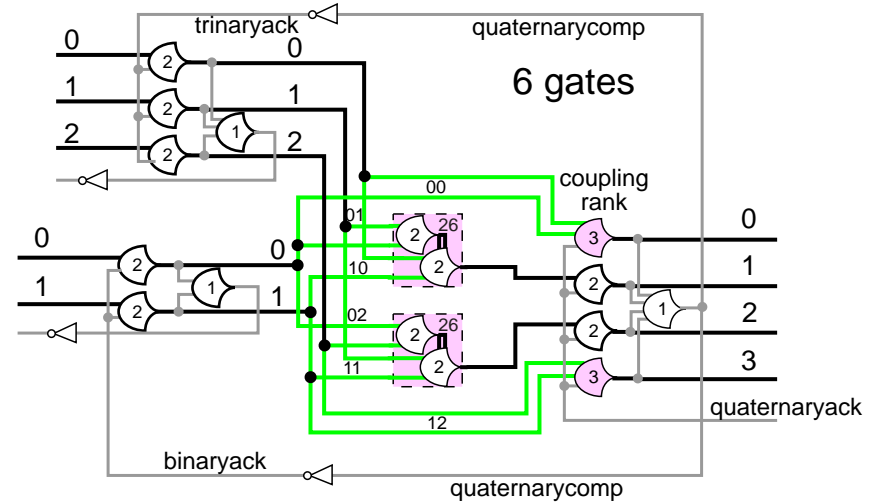
This should be easily realizable by ensuring that all orphan paths are isolated to a single path segment which must transition faster than a path composed of several path segments and logical behaviors.

# Orphan Isolation Survives Optimization

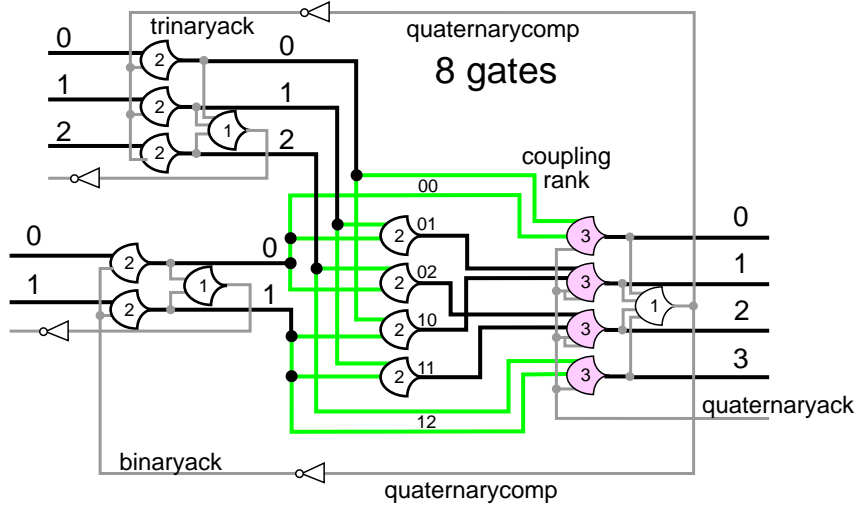
The orphans are all isolated behind the minterm ranks



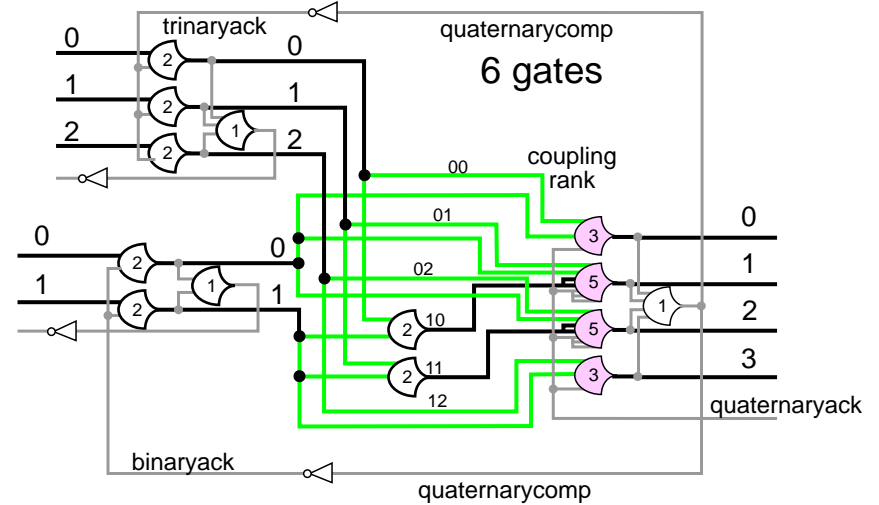
The orphans are still isolated



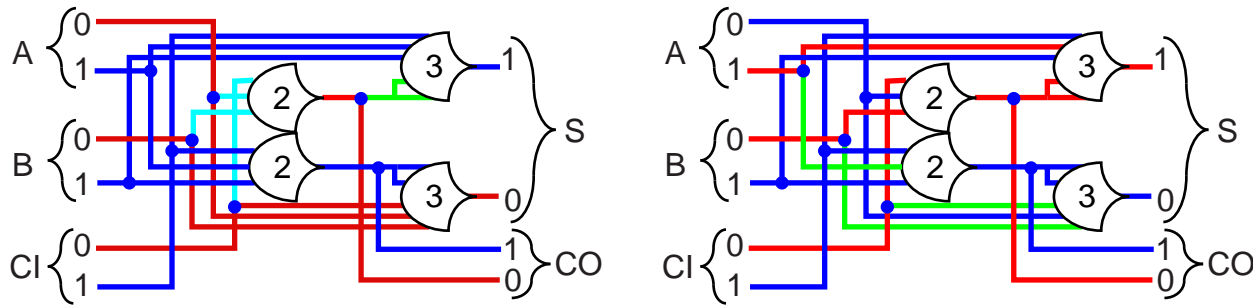
The orphans are still isolated



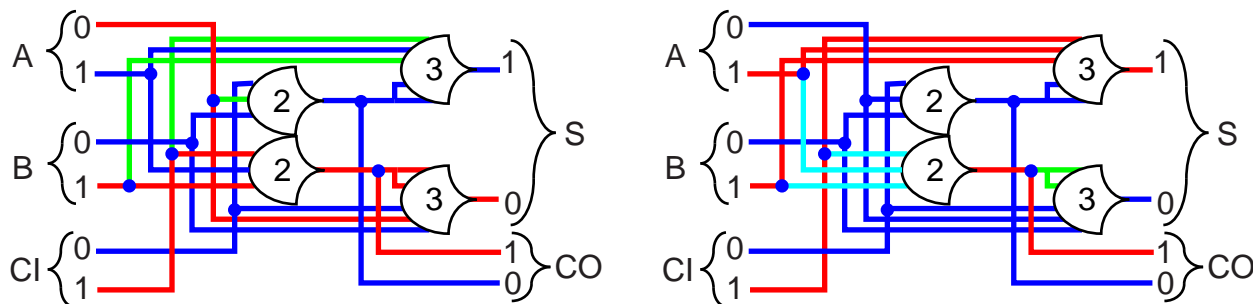
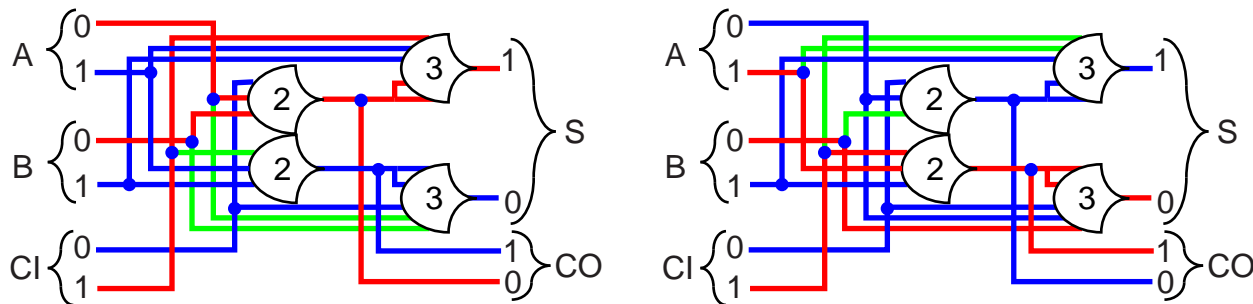
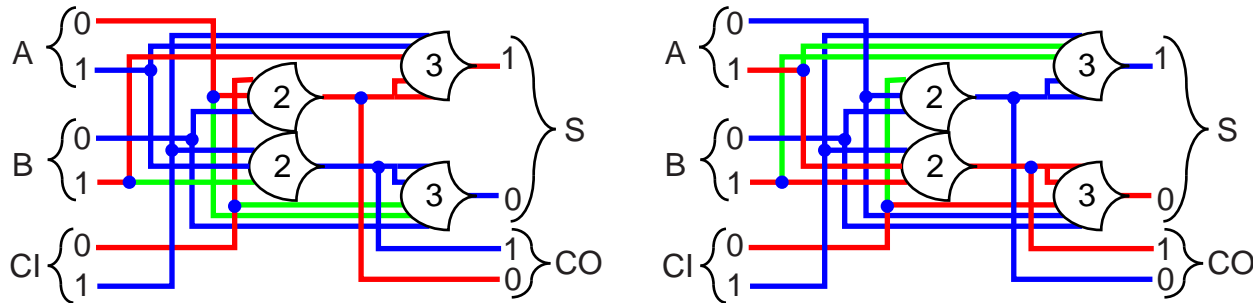
The orphans are still isolated



# Full Adder and the Excess Orphan Path



Dynamic Orphan Paths



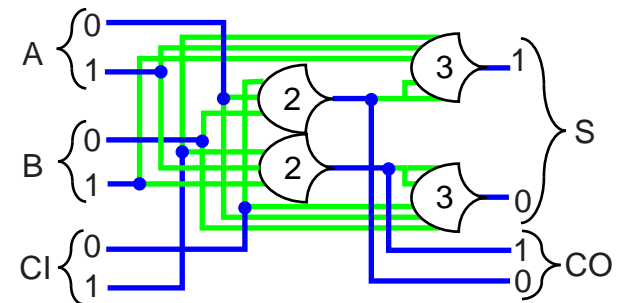
## The excess orphan path

There are three inputs to the 2 of 3 behaviors. When three inputs present the two early arrivals will transition the output of the behavior and the last arrival is ineffective and is an excess orphan.

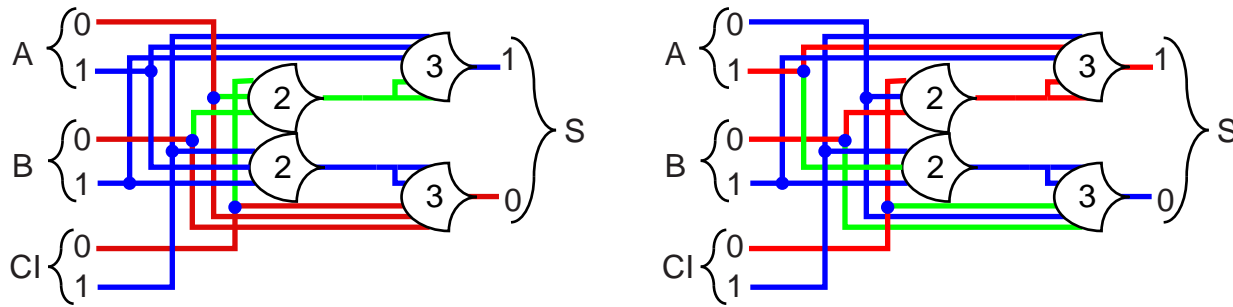
An excess orphan is a branch of an effective path and has the same delay relation to the effective path as any other orphan has to its effective path.

- NULL path — blue
- Effective DATA path — red
- Absolute orphan path — green
- Excess orphan path — cyan

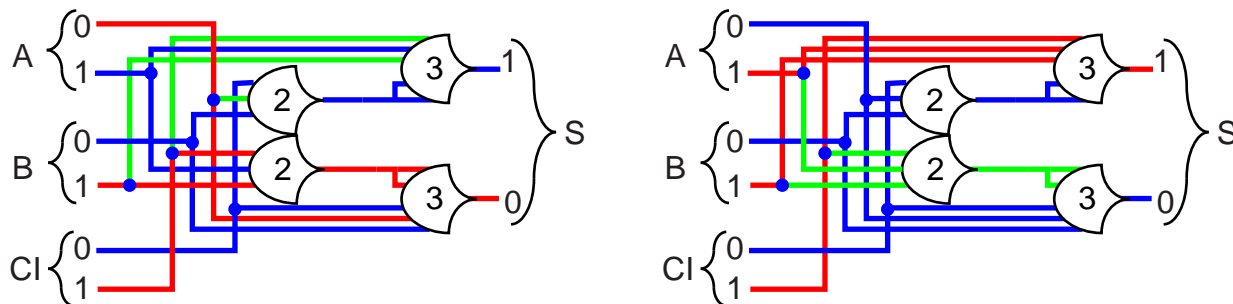
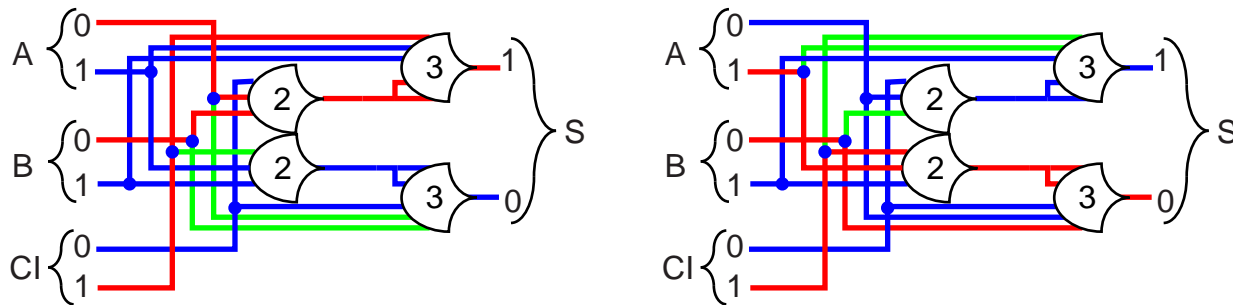
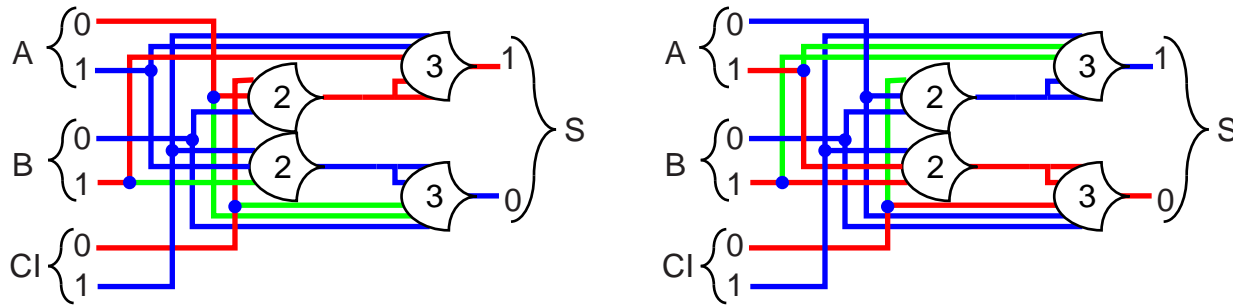
## Static Orphan Paths



# Orphan Paths Containing Behaviors



Dynamic Orphan Paths



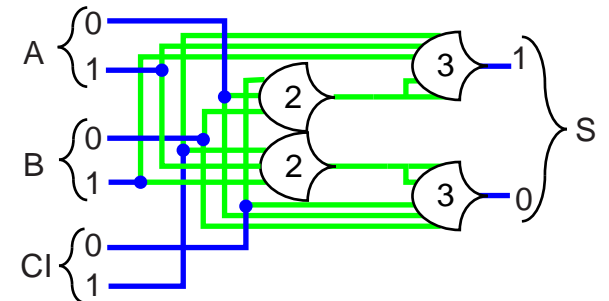
The longer orphan path

The longer an orphan path becomes the more critical its delay relation to the oscillation period. The basic strategy to keep orphan paths short is to insure that an orphan path does not contain a behavior (isolate the orphans to wire only paths).

The fulladder for 0,0 and 1,1 inputs contains an orphan path with a behavior.

- NULL path —
- Effective DATA path —
- Absolute orphan path —
- Excess orphan path —

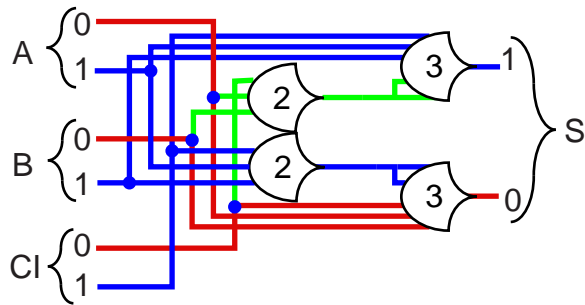
Static Orphan Paths



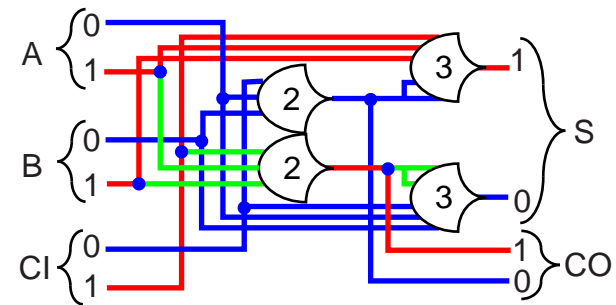
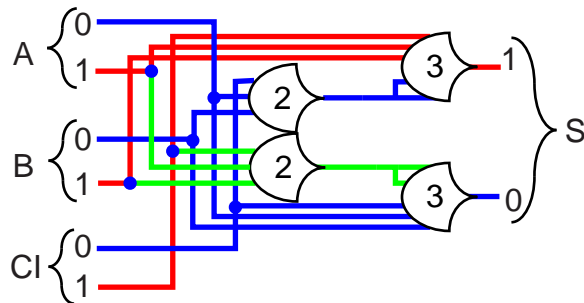
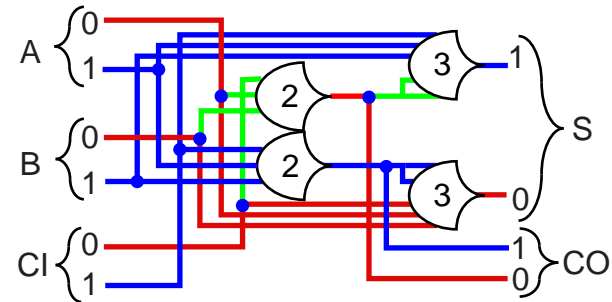
# Explicitly Isolating the Long Orphan Path

An orphan path containing a behavior can be isolated by observing the path after the behavior. The completeness of the carryout of the fulladder observes the orphan path after the behavior so the orphan is now from the observation branch and isolates the orphan path.

An orphan path includes the 2 of 3 behavior



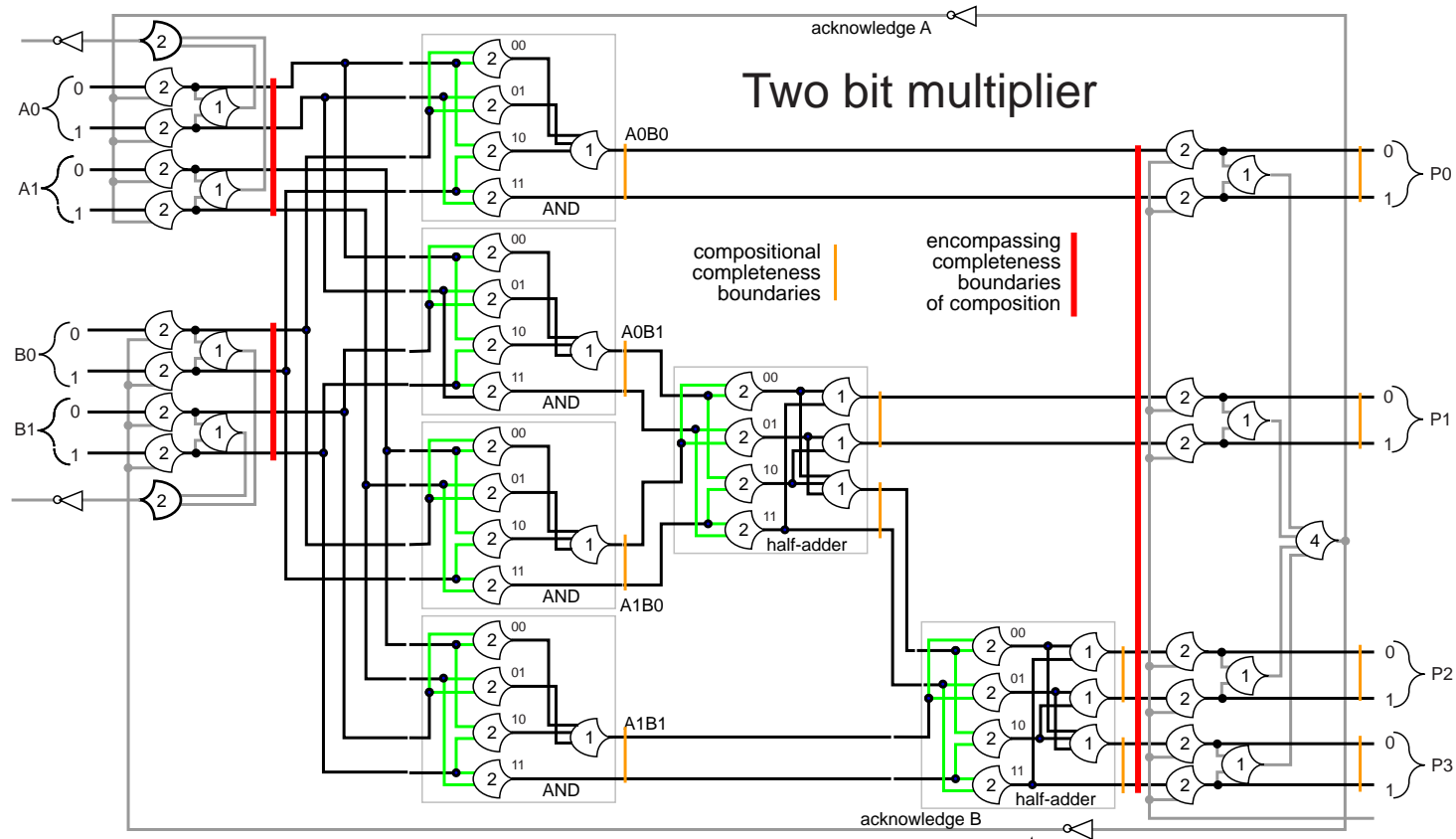
The path after the behavior is observed and the orphan path is fragmented and isolated





# Isolated Orphans in a Larger Composition

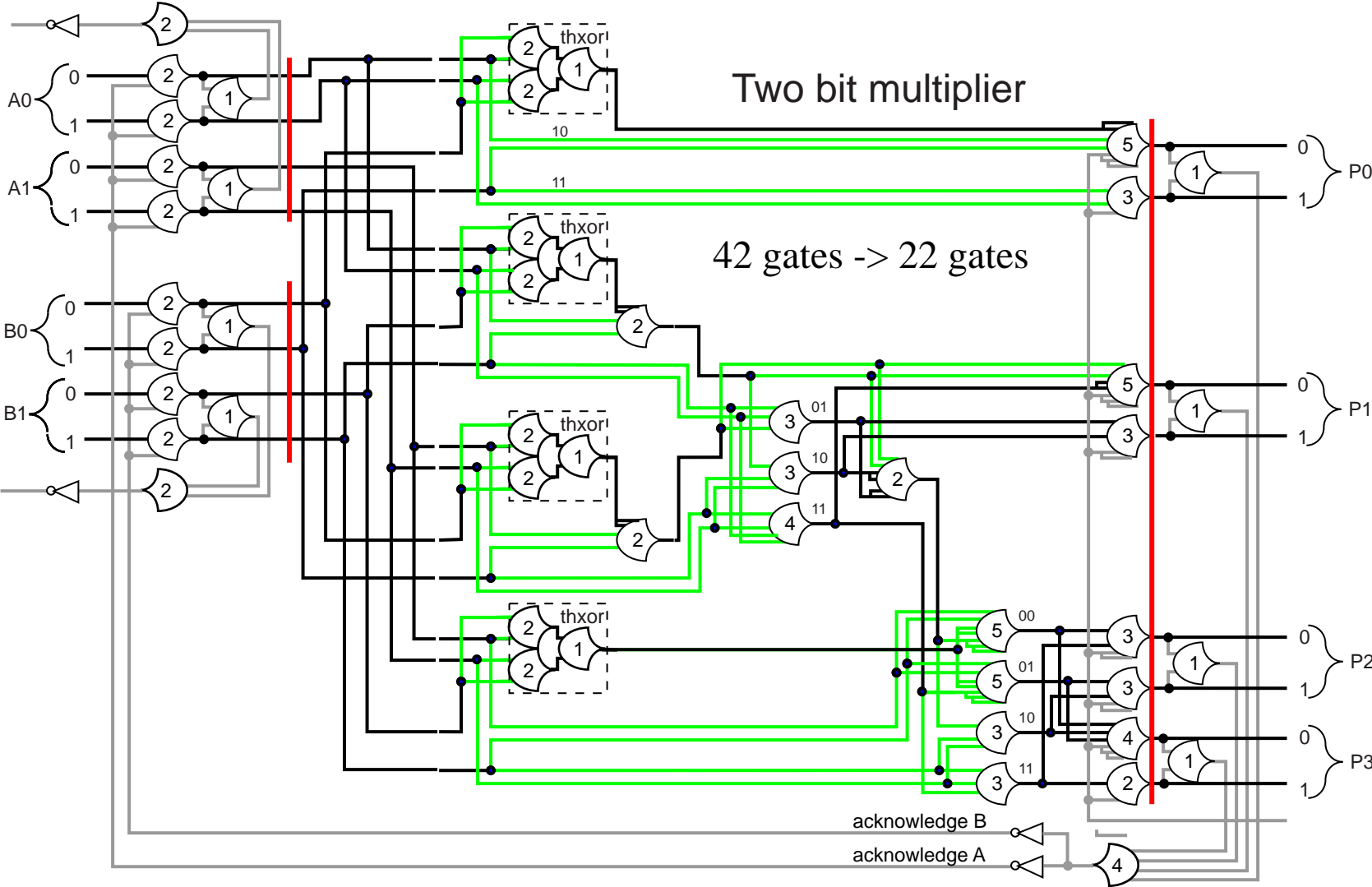
The orphans are all isolated behind the minterm ranks



The green paths must transition within one transition period of the enclosing oscillation

# Isolated Orphans Remain Isolated with Logical Optimization

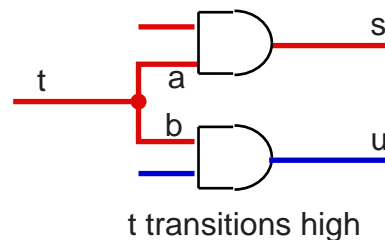
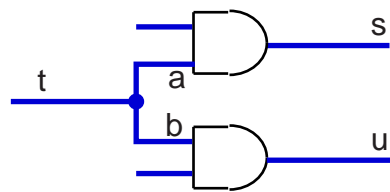
The orphans are still isolated to a single wire branch



# The Isochronic Fork

“In an isochronic fork, when a transition on one output is acknowledged, and thus completed, the transitions on all outputs are acknowledged, and thus completed.”

Alain J. Martin, “The Limitations to Delay-Insensitivity in Asynchronous Circuits”, 6th MIT Conference on Advanced Research in VLSI Processes, 1990, pp. 263-277.



s is a successor transition to branch a and directly acknowledges the transition of branch a and t

u does not transition and thus branch b is not directly acknowledged but is indirectly acknowledged by s. So b must have the same delay as a within one gate delay.

The observable flow path is only to the end of one of the forks which is observed only by the first gate that transitions on that fork. The other forks relate to this observable flow path.

Since they design in terms of Boolean logic, with gates that glitch, and with inversion which precludes strictly monotonic behavior, they cannot make observability assumptions about signal behavior through the extended circuit beyond this first gate as can be made with NCL circuits. So they are stuck with the more critical and difficult to manage timing assumption of the isochronic fork.