# Simple Cost and Performance Model

## Motivation

When designing things need to estimate cost and performance.

When running synthesis need to know if results are good or bad.

## Simple Models Goals

Choose between design alternatives.
*Plan A is better than Plan B based on the simple model.*

Characterize a particular design approach.
*Based on the simple model the cost of a module of size $n$ is $\propto n^2 \log n$.*

The models will be used throughout the semester.

## Simple Model Non-Goals

The simple model . . .
. . . **is not suitable for** approximating post-synthesis cost and performance.

## *Practice Problems*

Problems based on the material in these slides.

2019 Homework 3. (Analysis of a shift/add module and a recursive multiply module.)

2016 Final Exam Problem 2b and Problem 4 (greedy and fcfs fit).

2017 Final Exam Problem 3. (Two variations on a module.)

# The Simple Cost Model

*Simple Model Base Costs*

**2-input AND gate:** $1\,\mathrm{u_c}$**. (One unit of cost.)**

**2-input OR gate:** $1\,\mathrm{u_c}$**.**

**NOT gate:** $0\,\mathrm{u_c}$**. (Zero cost.)**

## *Simple Model Derived Costs—Basic Gates*

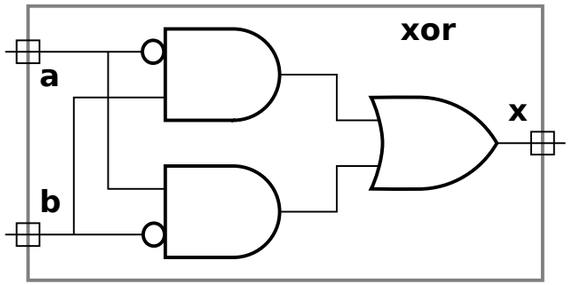
Based on equivalent circuit using gates above.

Cost of *n-input OR gate:* $(n-1)\,\mathrm{u_c}$.

Cost of *n-input AND gate:* $(n-1)\,\mathrm{u_c}$.

Cost of a *2-input XOR gate:* $3\,\mathrm{u_c}$.

XOR:



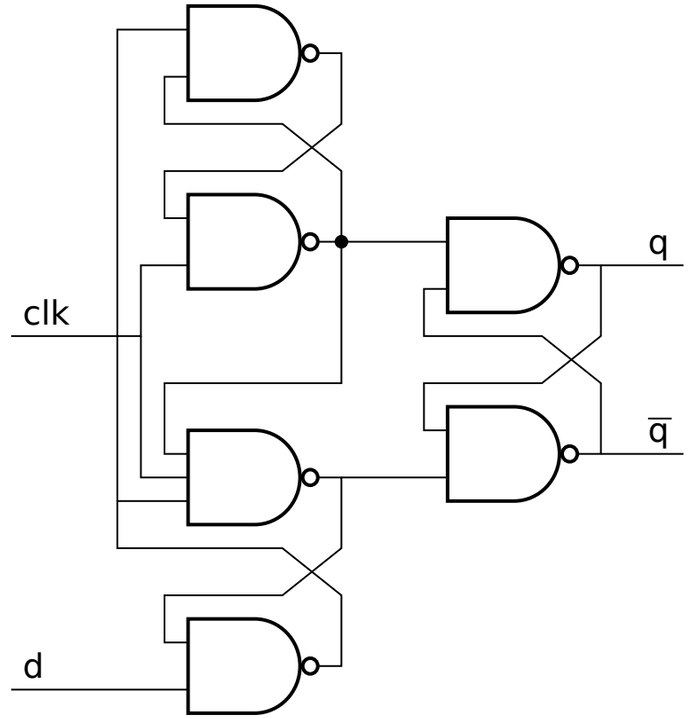
## *Simple Model Derived Costs—Flip Flops*

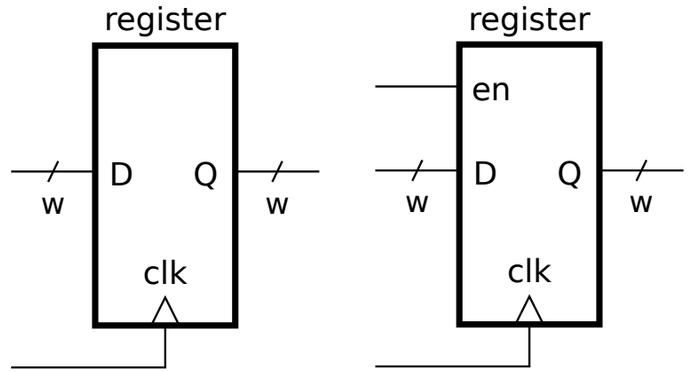
Cost of an *edge-triggered flip-flop* is $7\,\mathrm{u_c}$.

Cost of an *edge-triggered enable flip-flop* is $10\,\mathrm{u_c}$.

Cost of a *w-bit edge-triggered register* is $7w\,\mathrm{u_c}$.

Cost of a *w-bit edge-triggered register with enable* is $10w\,\mathrm{u_c}$.



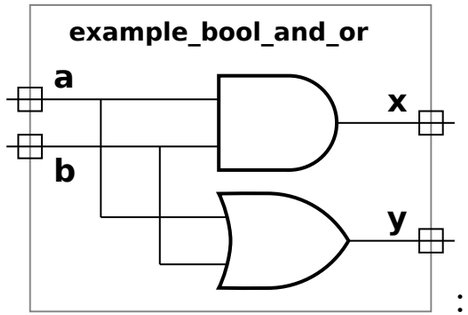



## Examples

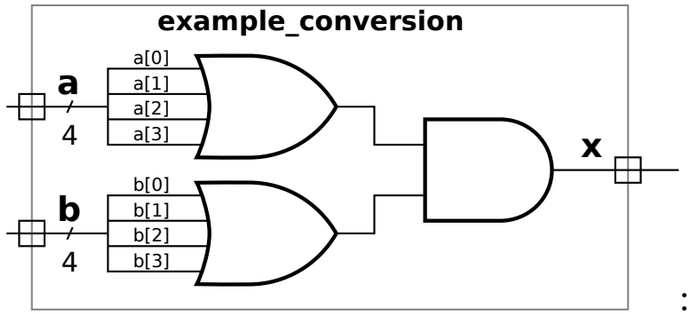
A 2-input AND gate: cost is $1\,\mathrm{u_c}$.

A 10-input OR gate: cost is $9\,\mathrm{u_c}$.

A 1-input OR gate: cost is $0\,\mathrm{u_c}$. Yes, it's free!




: cost is $1 + 1 = 2\,\mathrm{u_c}$.




: cost is $3 + 3 + 1 = 7\,\mathrm{u_c}$.

# The Simple Performance Model

## *The Simple Performance Model Base Delays*

**2-input AND gate:** $1\,\mathrm{u_t}$ **(One time unit.)**

**2-input OR gate:** $1\,\mathrm{u_t}$**.**

**NOT gate:** $0\,\mathrm{u_t}$**.**

## *The Simple Performance Model Derived Delays*

Based on equivalent circuit using gates above.

Delay of a *2-input XOR gate:* $2\,\mathrm{u_t}$.

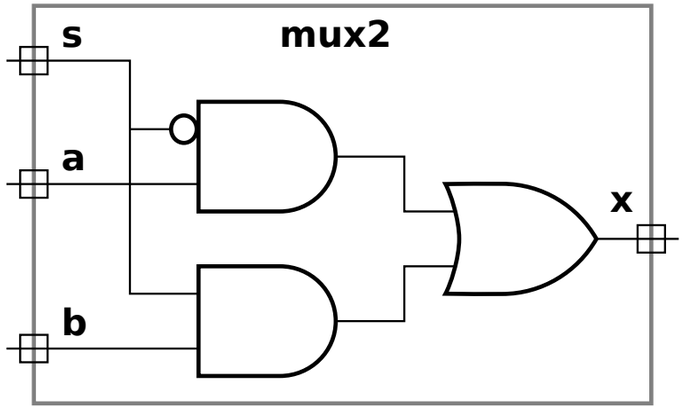
Delay of *n-input OR gate:* $\lceil \lg n \rceil\,\mathrm{u_t}$.

Delay of *n-input AND gate:* $\lceil \lg n \rceil\,\mathrm{u_t}$.

Delay of an *edge-triggered flip-flop* is $6\,\mathrm{u_t}$.

# Multiplexors

*A 2-input, 1-bit mux*:




Cost, $3\,\mathrm{u_c}$. Delay, $2\,\mathrm{u_t}$.

*A 2-input, $w$-bit mux*:

This is equivalent to $w$ copies of the mux above.

Cost, $3w\,\mathrm{u_c}$. **Delay,** $2\,\mathrm{u_t}$.

## *An n-input, w-bit mux, tree implementation:*

Constructed from 2-input multiplexors.

Illustration is for $n = 8$.

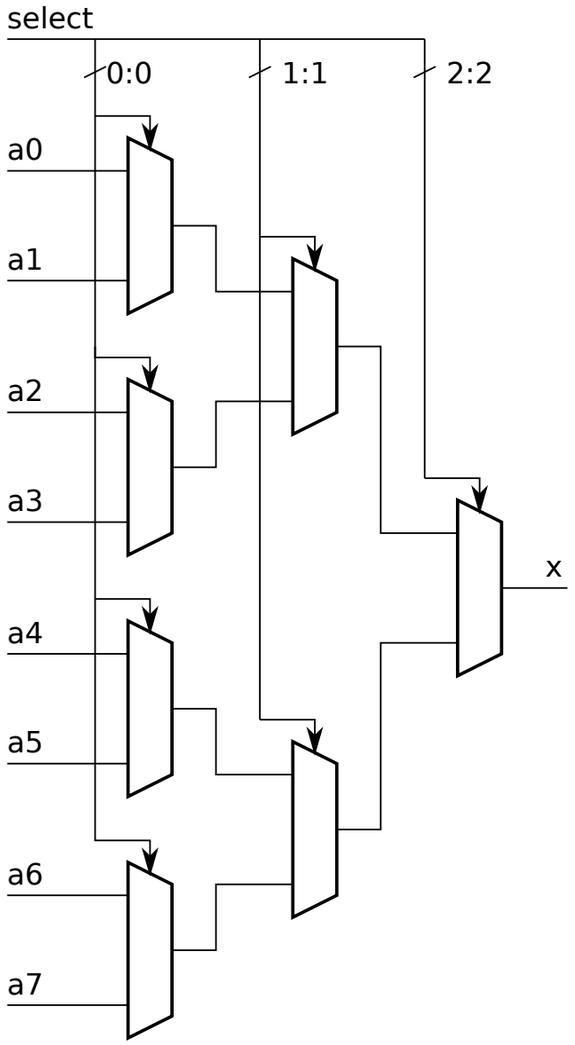
The path from the selected input ...
... is through $\lceil \lg n \rceil$ 2-input muxen ...
... through 3 for illustrated size, $n = 8$.

The number of muxen connected to select bit $i$...
... is $n/2^{i+1}$ for $0 \geq i < \lceil \lg n \rceil$...
... for illustrated size 2 muxen connect to bit 1.

Cost Computation

Total number of 2-input muxen ...
... $\sum_{i=0}^{\lg n-1} n/2^{i+1} = n - 1$ ...
... for illustrated mux, 7 2-input muxen.

Total cost: $3w(n-1)$. Total Delay: $2\lceil \lg n \rceil$.



## *Equality Comparison* (`a == b`)

Output is `1` iff $w$-bit inputs are equal.

Assume $w$ is a power of 2.

Cost Computation

XOR Gates: $3w$.

Reduction tree of AND gates: $\sum_{i=1}^{\lg w} w/2^i = w - 1$.

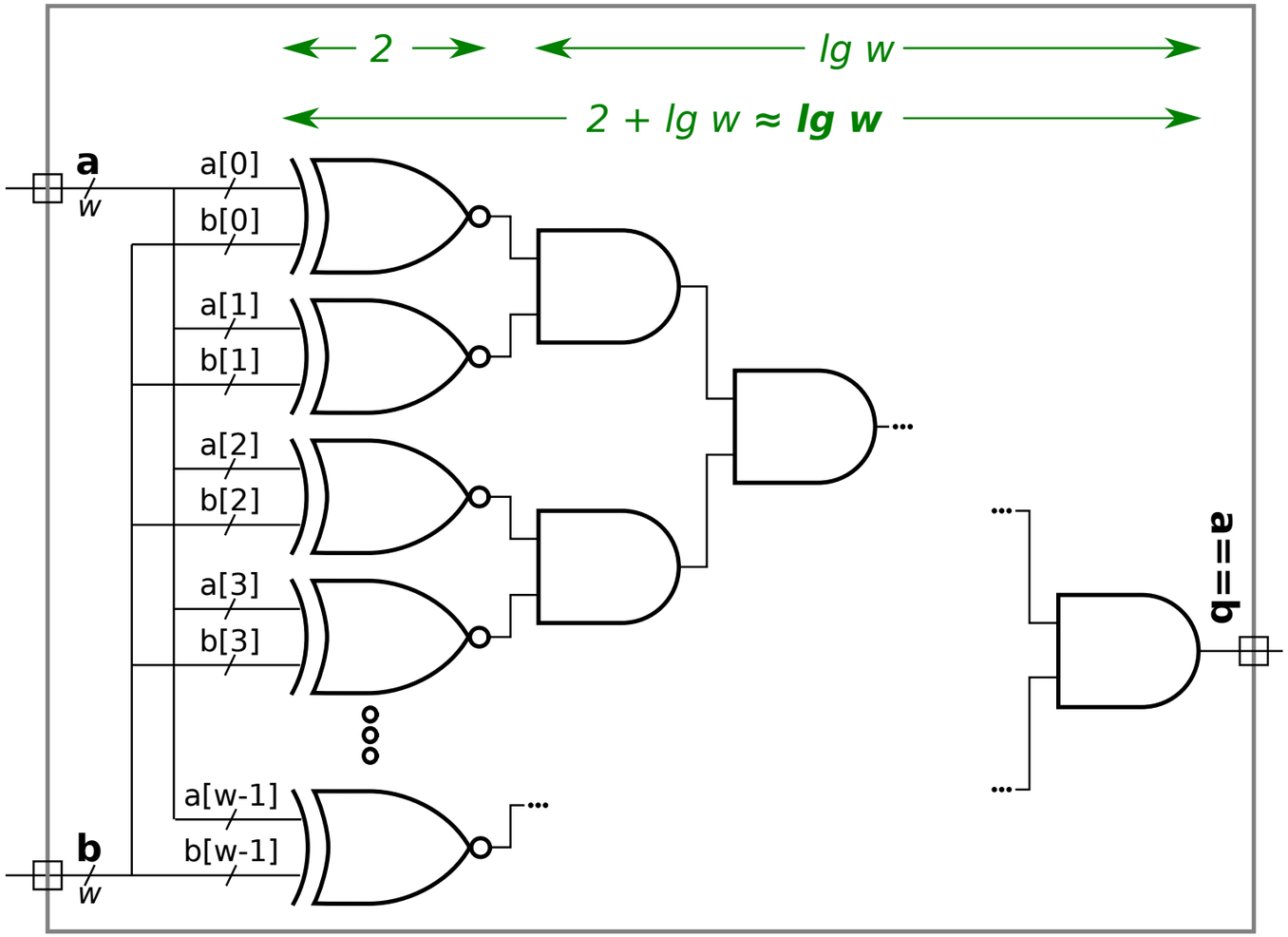
Total Cost: $4w - 1 \approx 4w$.

Delay Computation

XOR Gates: 2.

Reduction Tree: $\lg w$.

Total Delay: $2 + \lg w \approx \lg w$.








smod-11

# Binary Full Adder Constructions

## Prerequisite

If necessary, review material on binary half and full adders and ripple adders.

For example, Brown and Vranesic 3rd Edition Section 3.2.

You should either remember the circuits for BFAs and BHAs ...
... or be able to effortlessly derive them using a truth table.

## Material in this Section

Binary Full Adder (BFA)

Ripple Adder

Magnitude (Greater Than, Less Than, etc.)

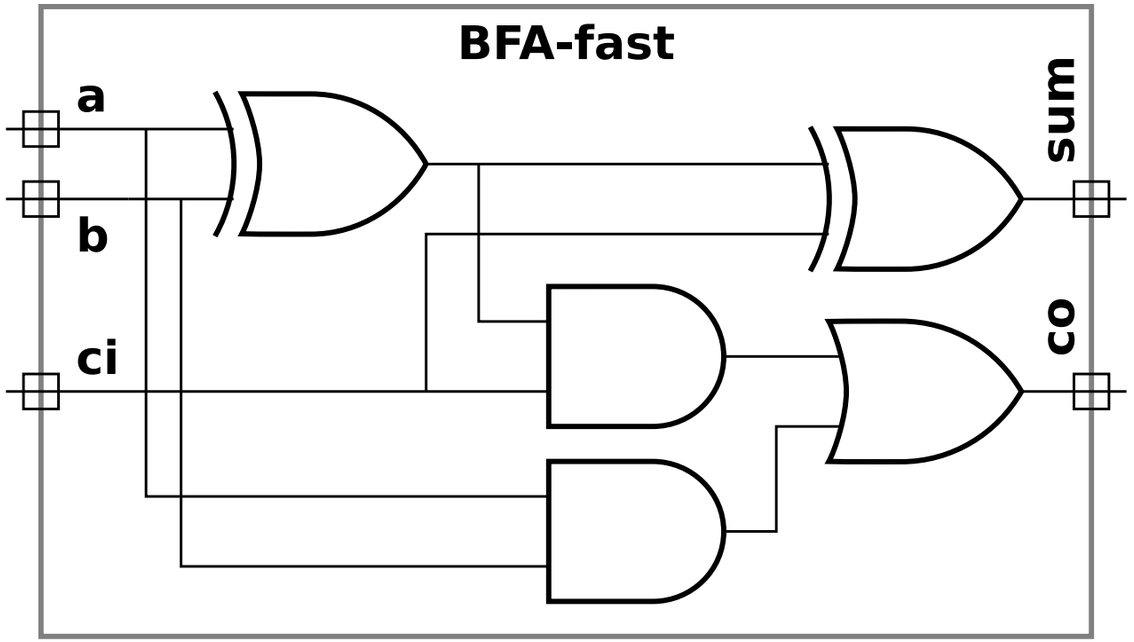
Cascaded Ripple Units

# Binary Full Adder Implementation

*Fast BFA*

Cost Computation

Two XOR, 3 AND: $2 \times 3 + 3 = 9\,\mathrm{u_c}$








smod-14

## *Fast BFA*

Delay: Any input to any output.

See Path I and...
... purple labels in diagram.

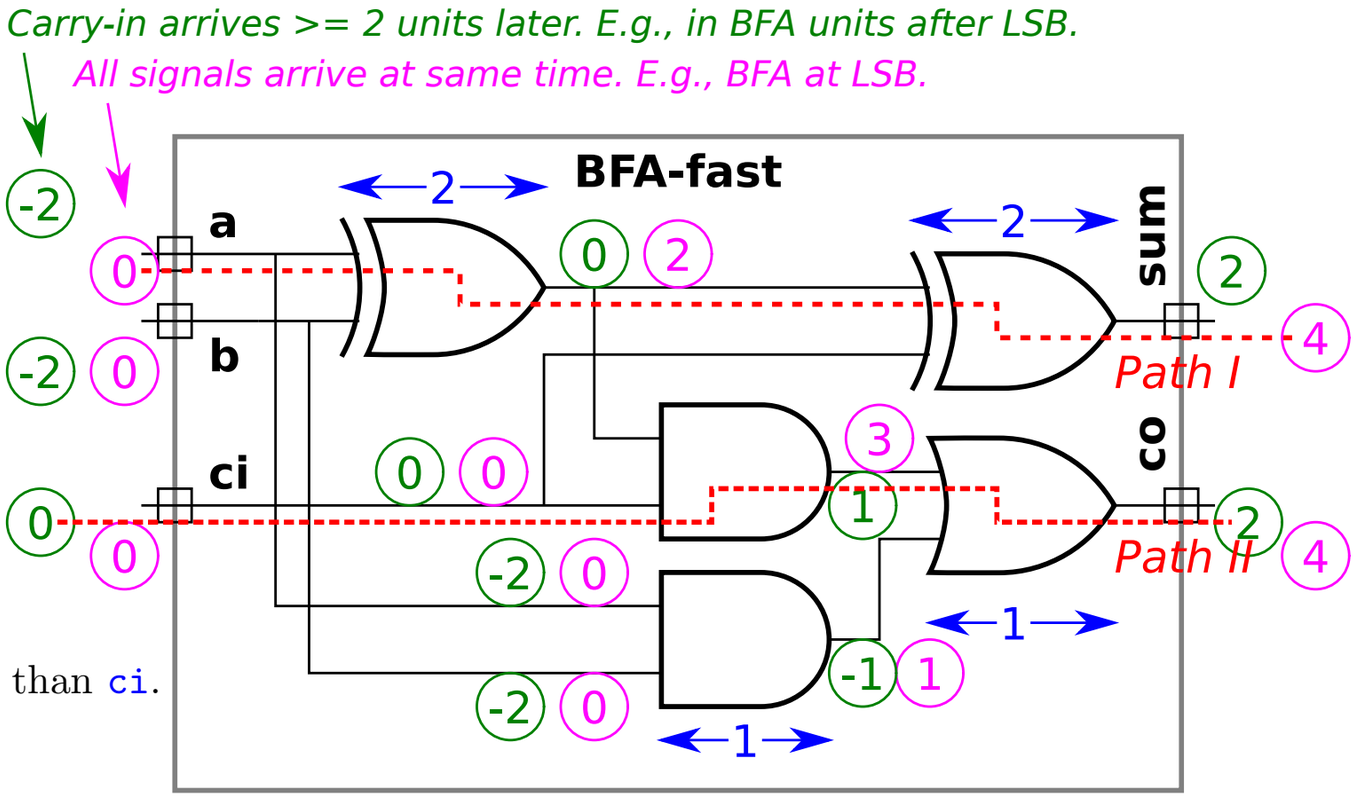
Delay: $4\,u_t$

Delay: `ci` to `co`.

This delay is useful when `a` and `b` arrive earlier than `ci`.

See Path II and green labels in diagram.

Delay: $2\,u_t$

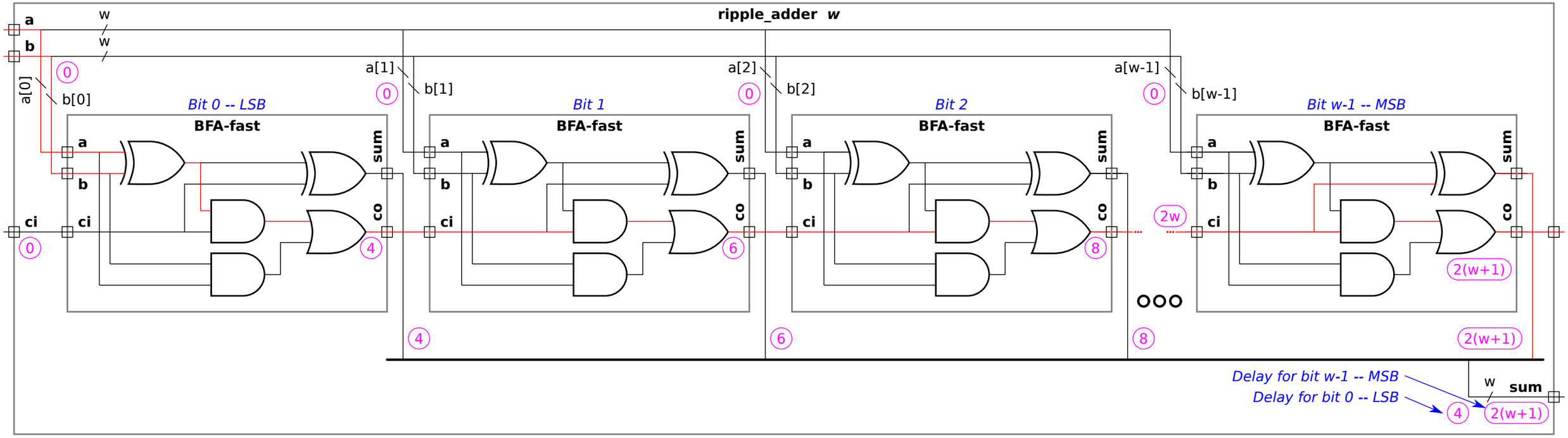







smod-15

## *w-Bit Ripple Adder*




Cost Computation: Cost of $w$ BFAs: $9w\,\mathrm{u_c}$

Delay Computation

See critical path (in red) in diagram.

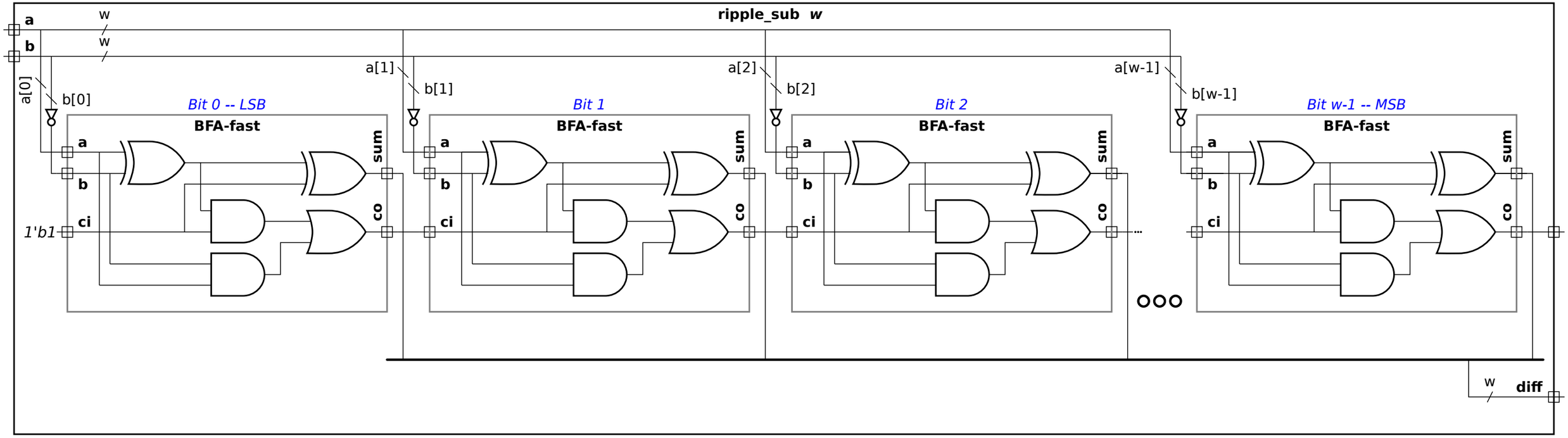
Delay: $2(w+1) \approx 2w$.





smod-16

## *w-Bit Ripple Subtractor*




Cost and delay are slightly less due to constant carry in.





smod-17

## *Integer Magnitude Comparison*

For comparisons like $a < b$.

Implementation:

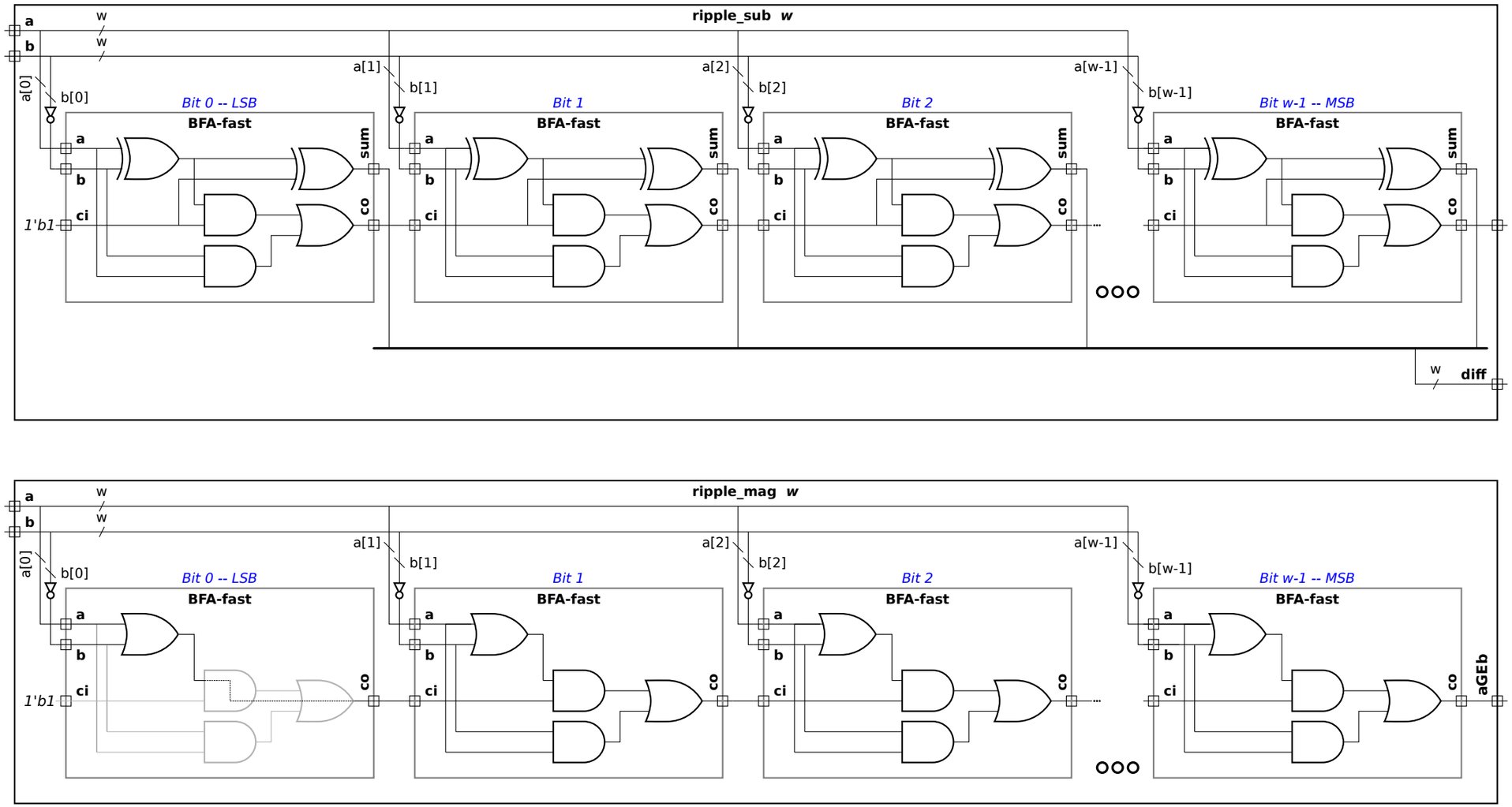
Compute $a - b$ and check whether result negative.

If carry out of MSB is `0` then $a - b < 0$ and so $a < b$ is true.

Omit sum hardware in BFA, and replace remaining XOR with an OR.

*See the illustration on the next page.*

# Integer Magnitude Comparison Construction








smod-19

# Cost and Delay of Integer Magnitude Comparison

## Cost Computation

Each modified BFA now costs $4\,\mathrm{u_c}$.

Total cost: $4w\,\mathrm{u_c}$

## Delay Computation

Delay is 3 for first bit, 2 for remaining bits.

Total delay: $[2w+1]\,\mathrm{u_t} \approx 2w\,\mathrm{u_t}$.

## *Cascaded Ripple Units*

For computations using ripple units. . .
. . . such as $a + b + e$, and $(a + b) < e$, etc.

Cost Computation

Cost is sum of costs of each ripple unit.

For example, $a + b + c$ is two ripple adders, cost is $18w\,\mathrm{u_c}$ . . .
. . . $(a + b) < e$ is a ripple adder plus a magnitude comparison: $[9w + 4w]\,\mathrm{u_c} = 13w\,\mathrm{u_c}$.

Delay Computation:

Consider $(a + b) + e$.

Naïve analysis: wait for $a + b$ to finish, then start $+e$.

But, LSB of $a + b$ available after only $4\,\mathrm{u_t}$ ...
... bit $i$ is available after $(4 + 2i)\,\mathrm{u_t}$ ...
... so the $+e$ computation can start after $4\,\mathrm{u_t}$.

Delay for two ripple units is $[4 + 2(w + 1)]\,\mathrm{u_t}$.

Delay for bit $i$ at output of $n$ ripple units is $[4(n - 1) + 2(i + 2)]\,\mathrm{u_t}$.

Delay for $n$ ripple units is $[4(n - 1) + 2(w + 1)]\,\mathrm{u_t}$.

# Cost and Performance with Constant Inputs

## *Constant Inputs*

Input values which never change.

Cost and delay are radically different when an input never changes.

In Verilog, this might be an elaboration-time constant . . .
. . . or other expressions that never change.

# Multiplexor Constant-Input Optimizations

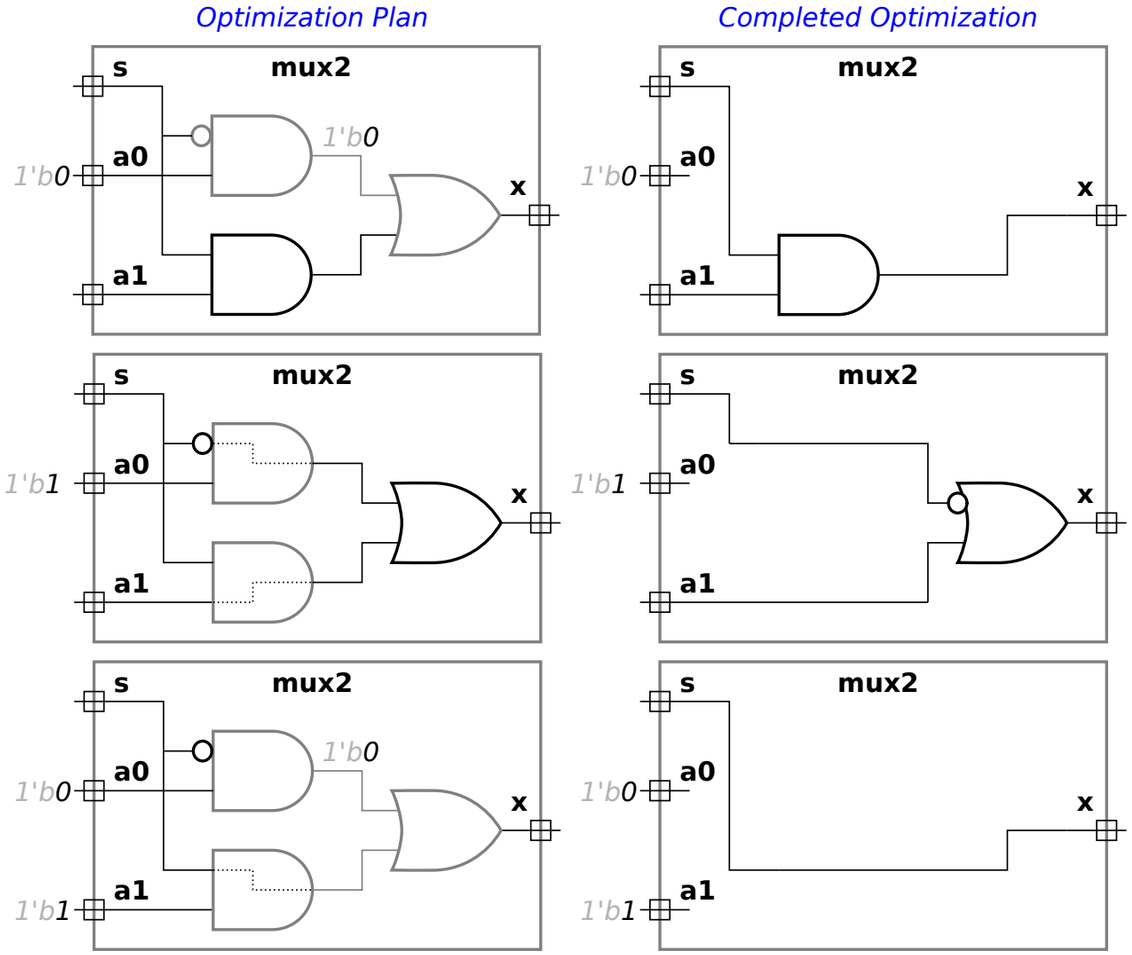
## Sample Mux Optimizations

Costs:

From top to bottom: $1\,u_c$, $1\,u_c$, $0\,u_c$.

Delays:

From top to bottom: $1\,u_t$, $1\,u_t$, $0\,u_t$.

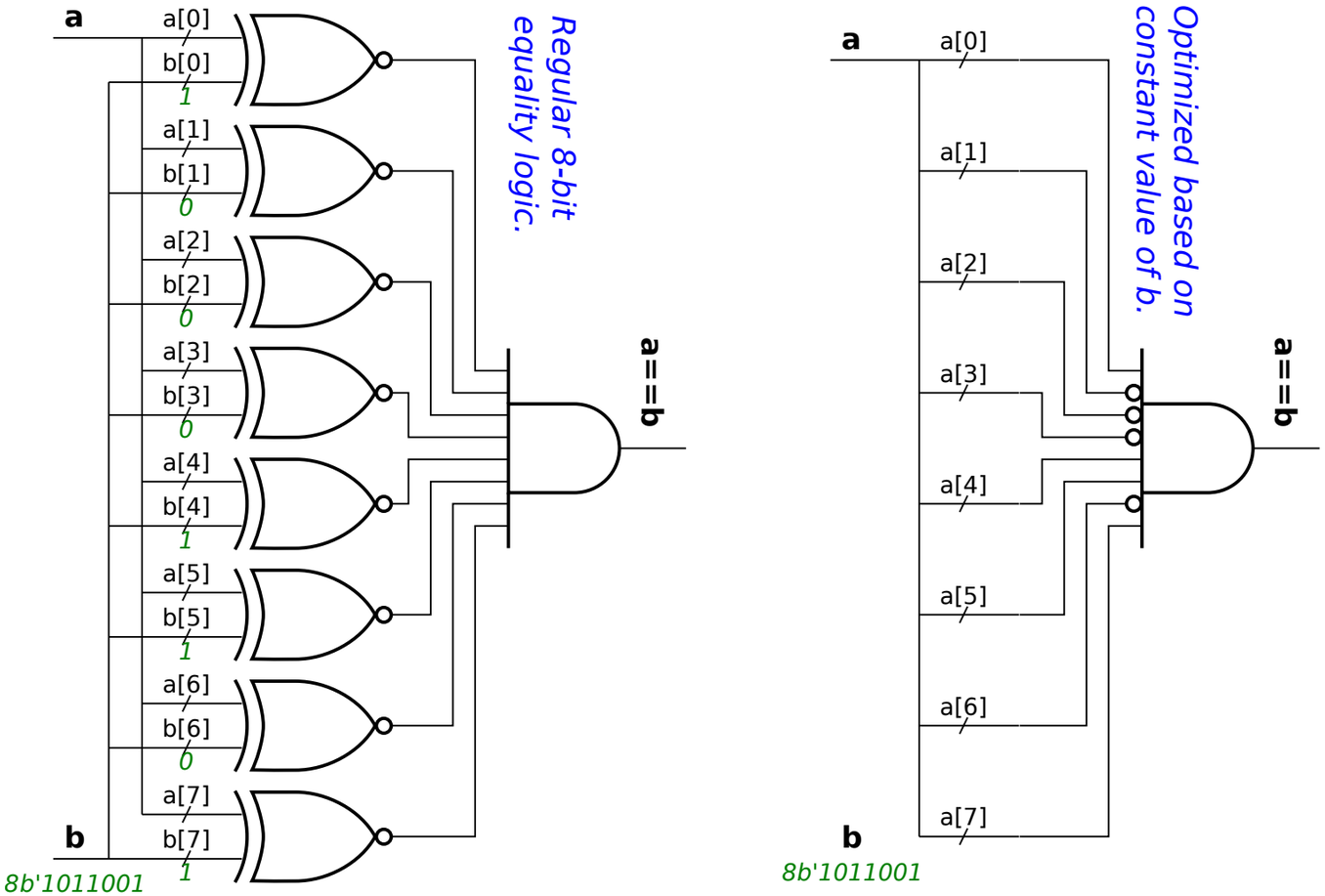


# Comparison Unit Constant-Input Optimization

Consider `a == b` where . . .
. . . `a` is an input . . .
. . . and `b` is a constant, `8b'1011001`.

Cost for $w$-bit comparison to constant: $\lceil w-1 \rceil \, \mathrm{u_c}$.

Delay for $w$-bit comparison to constant: $\lceil \lg w \rceil \, \mathrm{u_t}$.








smod-25