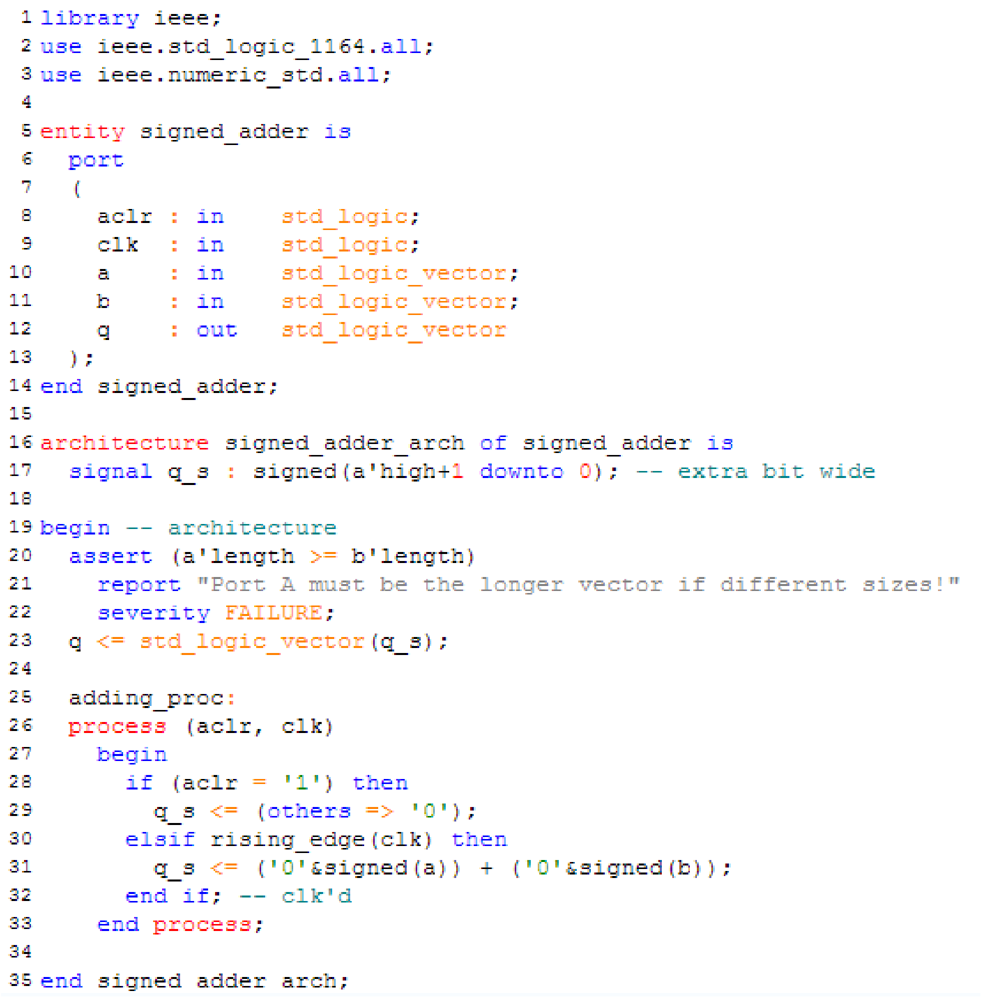
Sec 01: Logisim

Why Logisim?

- Most real-world hardware design is done using a text-based hardware description language – VHDL, AHDL, etc.
  - Schematics can be "compiled" into a text description
  - Can use a simulator to test the circuit
  - Other back-end tools optimize, perform layout and wire routing, floorplan, etc.
  - Final spec is either downloaded onto a programmable device, or etched into silicon
- We will be using Logisim for all hardware design
  - interactive, graphical schematic editor
  - educational use mainly (makes it user-friendly)

VHDL Example: Signed Adder



```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity signed_adder is
  port
  (
    aclr : in    std_logic;
    clk  : in    std_logic;
    a    : in    std_logic_vector;
    b    : in    std_logic_vector;
    q    : out   std_logic_vector
  );
end signed_adder;

architecture signed_adder_arch of signed_adder is
  signal q_s : signed(a'high+1 downto 0); -- extra bit wide

begin -- architecture
  assert (a'length >= b'length)
    report "Port A must be the longer vector if different sizes!"
    severity FAILURE;
  q <= std_logic_vector(q_s);

  adding_proc:
  process (aclr, clk)
    begin
      if (aclr = '1') then
        q_s <= (others => '0');
      elsif rising_edge(clk) then
        q_s <= ('0'&signed(a)) + ('0'&signed(b));
      end if; -- clk'd
    end process;

end signed adder arch;
```

Why Logisim?

- Most real-world hardware design is done using a text-based hardware description language – VHDL, AHDL, etc.
  - Schematics can be "compiled" into a text description
  - Can use a simulator to test the circuit
  - Other back-end tools optimize, perform layout and wire routing, floorplan, etc.
  - Final spec is either downloaded onto a programmable device, or etched into silicon
- We will be using Logisim for all hardware design
  - interactive, graphical schematic editor
  - educational use mainly (makes it user-friendly)

What are we going to talk about?

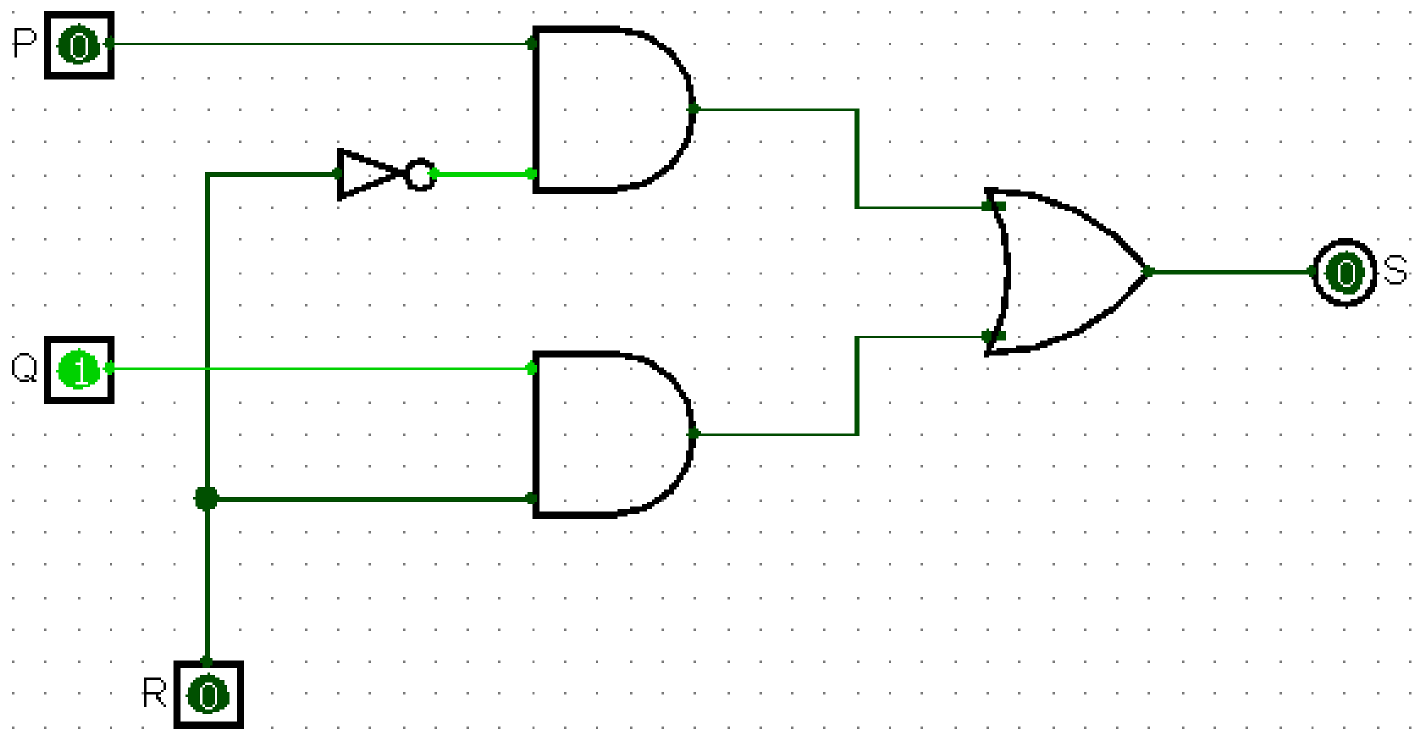
- Using Logisim:
  - pins and subcircuits
  - probes for debugging
  - bundles/splitters (with labels!)
  - logging
  - test vectors

- Example Circuits: 1-bit & 32-bit 2:1 Mux
- Using Subcircuits: 2:1 Mux and Controller
- Logging & Test Vectors

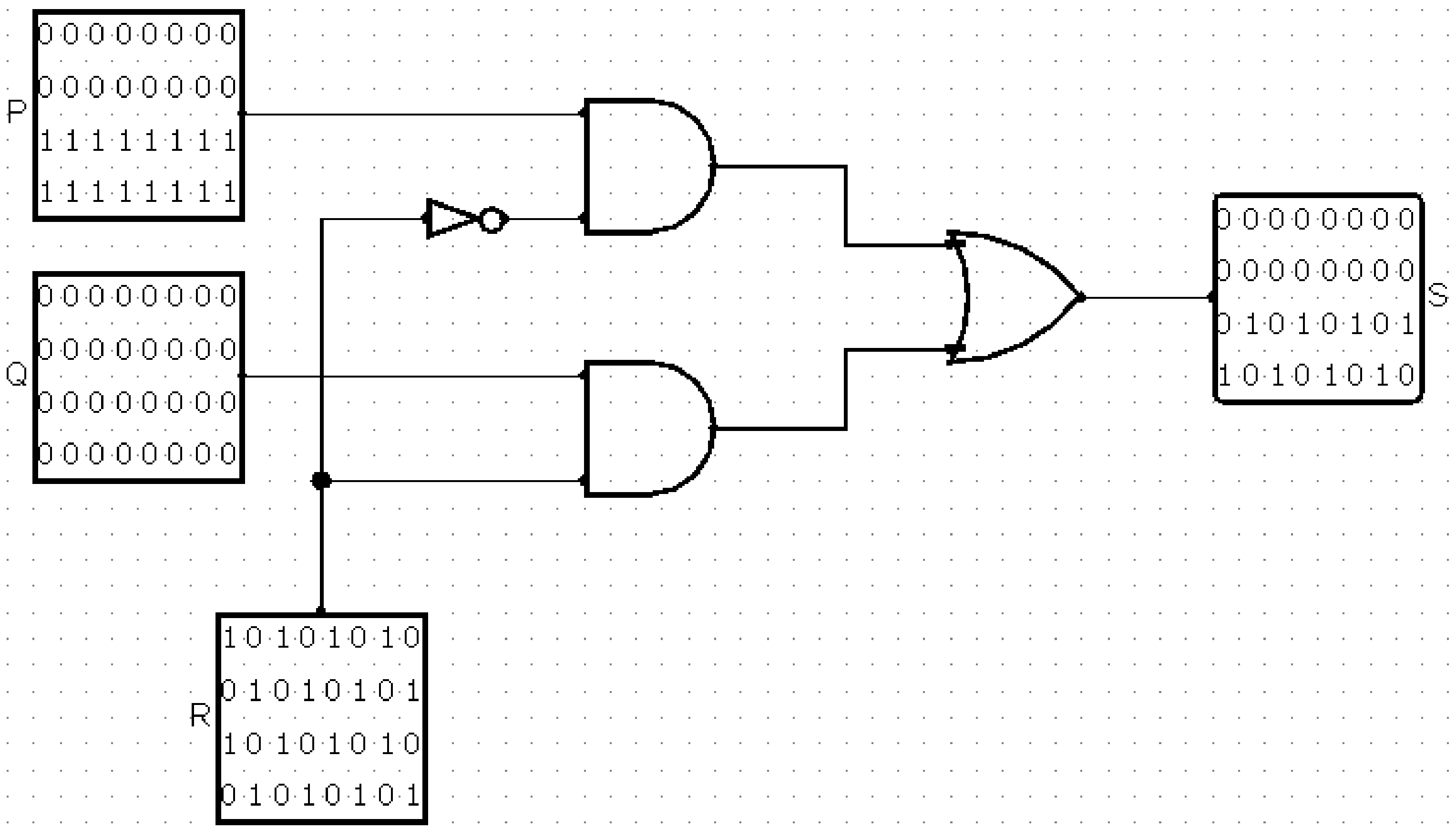
- Logisim Don'ts

Example Circuit: 1-bit 2:1 Mux

S = P if R == 0
S = Q if R == 1

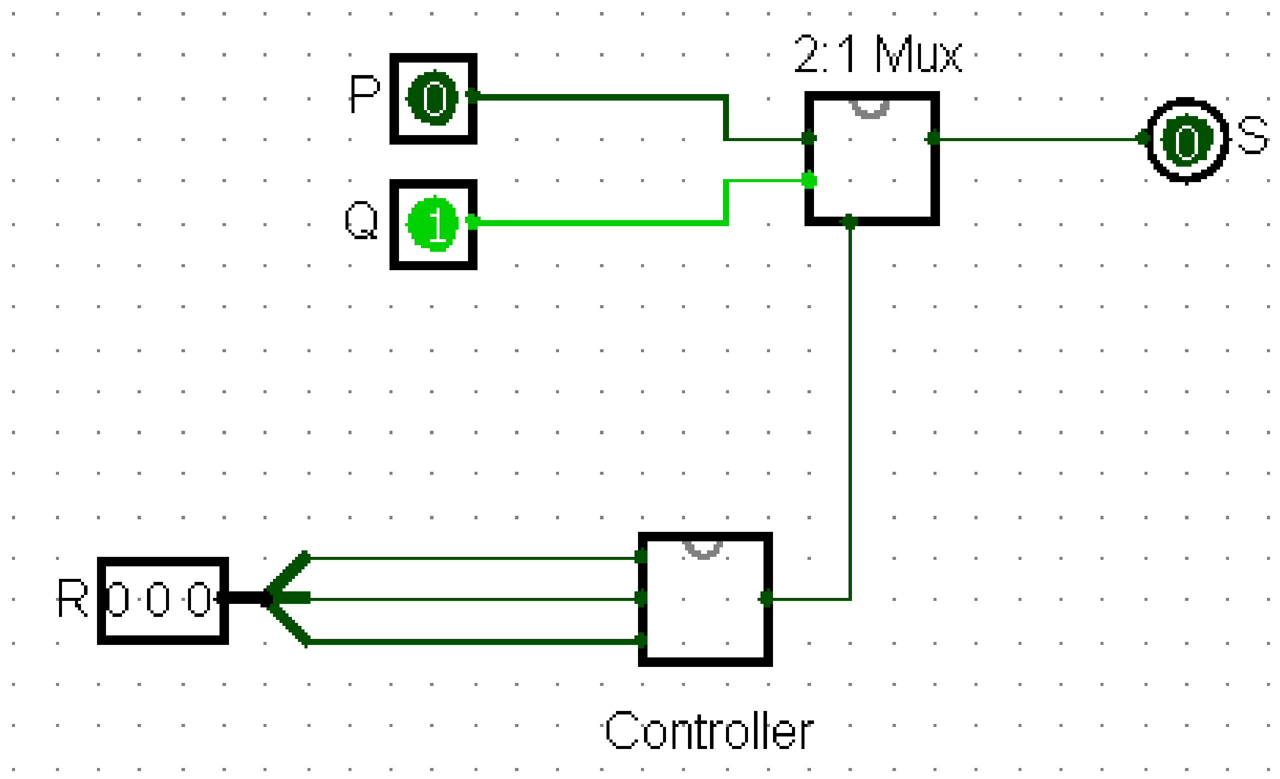


Example Circuit: 32-bit 2:1 Mux



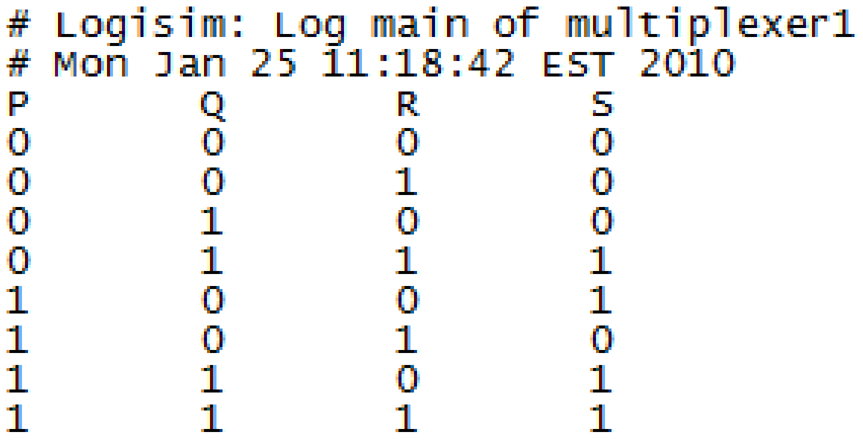
Using Subcircuits: 2:1 Mux and Controller

S = Q if R == 010
S = P otherwise



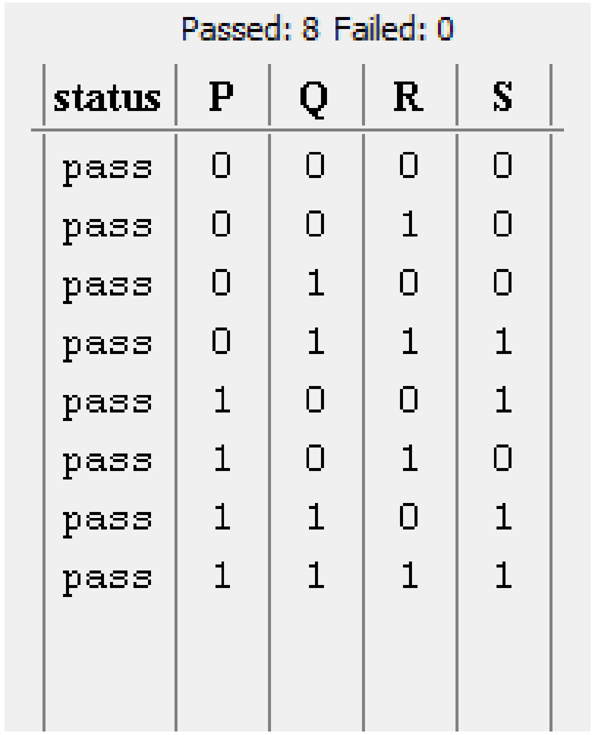
Logging & Test Vectors

Log File



```
# Logisim: Log main of multiplexer1
# Mon Jan 25 11:18:42 EST 2010
P       Q       R       S
0       0       0       0
0       0       1       0
0       1       0       0
0       1       1       1
1       0       0       1
1       0       1       0
1       1       0       1
1       1       1       1
```

Test Vector Truth Table



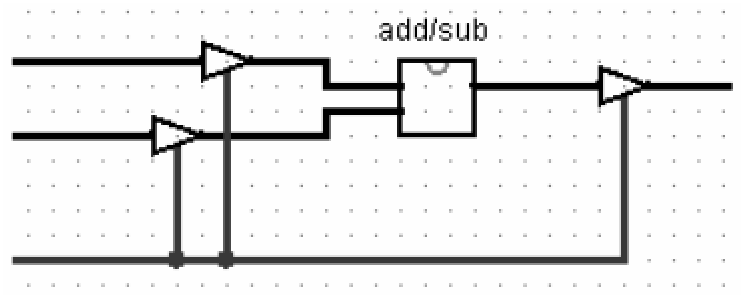
Passed: 8 Failed: 0

status	P	Q	R	S
pass	0	0	0	0
pass	0	0	1	0
pass	0	1	0	0
pass	0	1	1	1
pass	1	0	0	1
pass	1	0	1	0
pass	1	1	0	1
pass	1	1	1	1

Logisim Don'ts

No Controlled Buffers

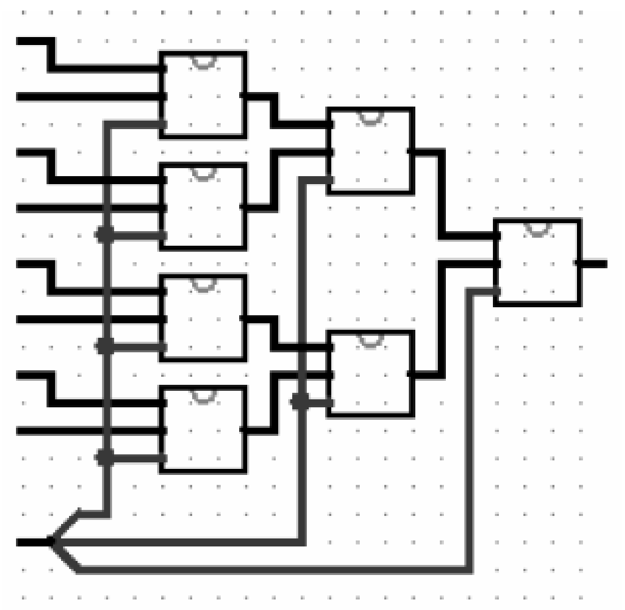
- Leave wires floating
  - Works in Logisim
  - Breaks in real life
- Use a multiplexor instead



Logisim Don'ts

Don't Build Your Own

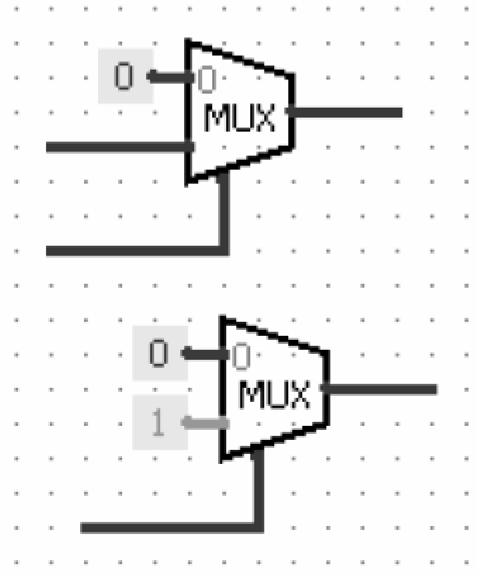
- Building your own wastes time and is confusing to grade
- Almost every component customizable
  - Number of inputs
  - Bit depth


Building an 8-way
Mux the hard way

Logisim Don'ts

Don't Use Constants

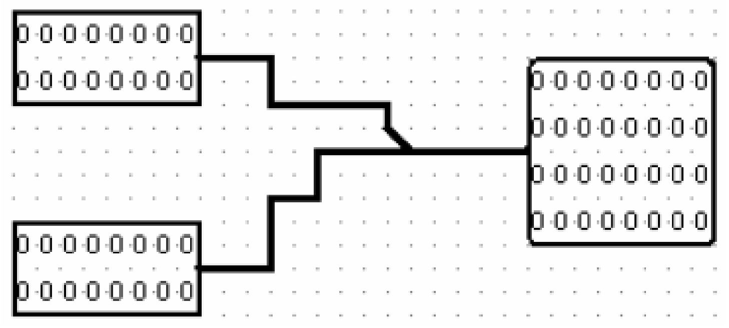
- Constants are almost never necessary
  - Exception is supplying value to extra input
- Try to optimize away before using
  - Think truth tables

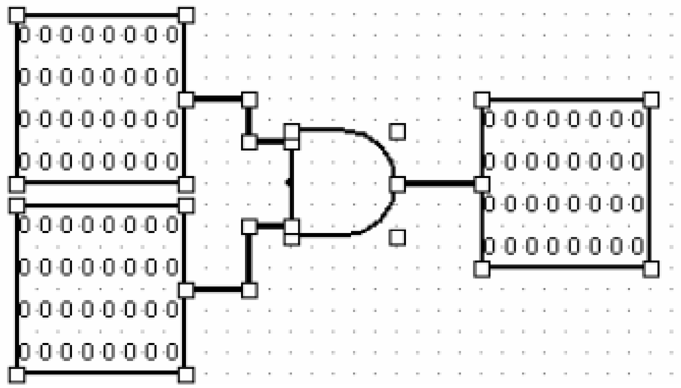


Logisim Don'ts

Don't Make Trivial Sub-circuits

- Try to build logical sub-circuits
- Problems
  - All sub-circuits look the same
  - Wastes time specifying inputs and outputs of small circuits
  - Big hierarchy harder to understand

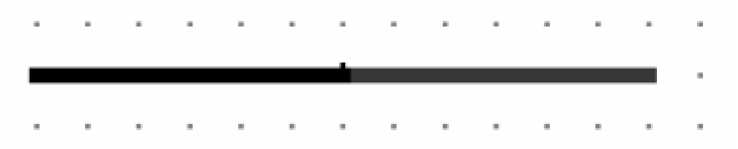




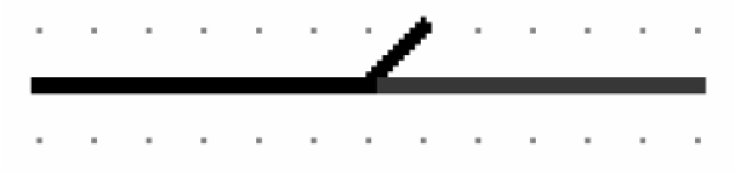
Logisim Don’ts

Don’t Use Invisible Splitters

- Please…
- Its really hard to see them when we grade



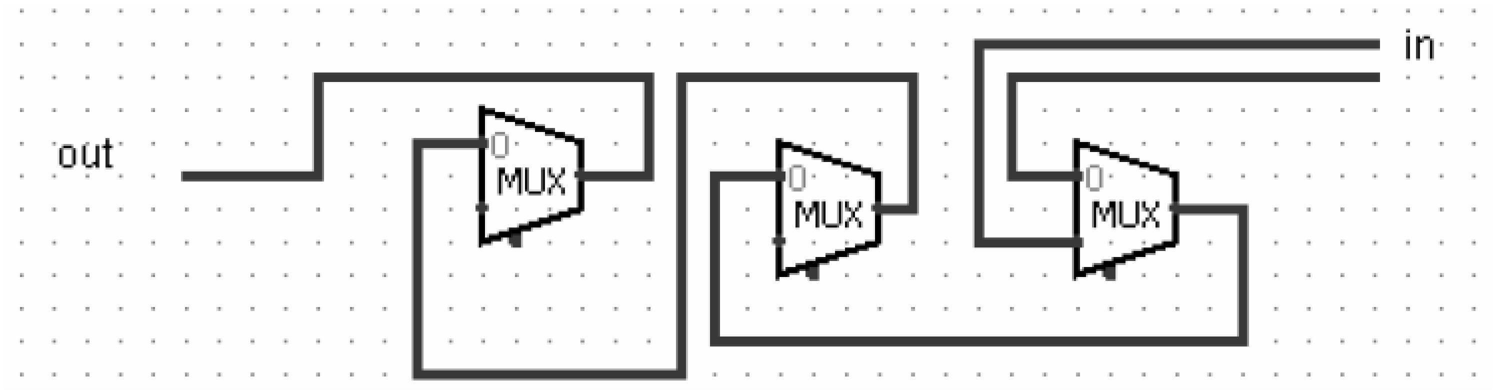
Bad

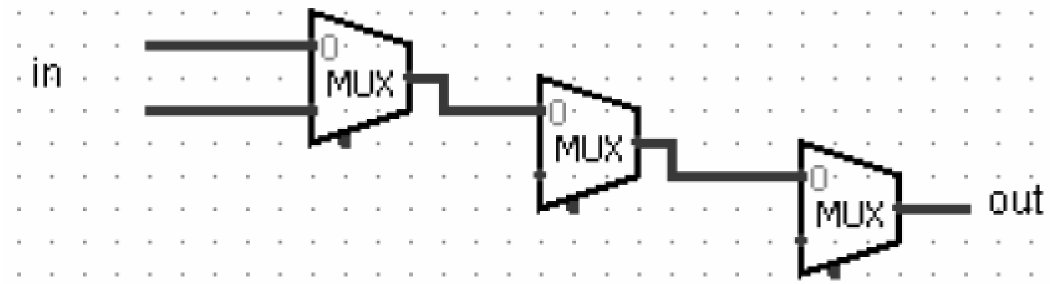


Good

Logisim Don'ts

Don't Work Right to Left





Some more information

- MIPS Assignments:
  - 32-bit ALU
  - 32-bit Processor
  - 32-bit Pipelined Processor

- Webpage:
  http://www.cs.cornell.edu/courses/cs3410/2010sp/