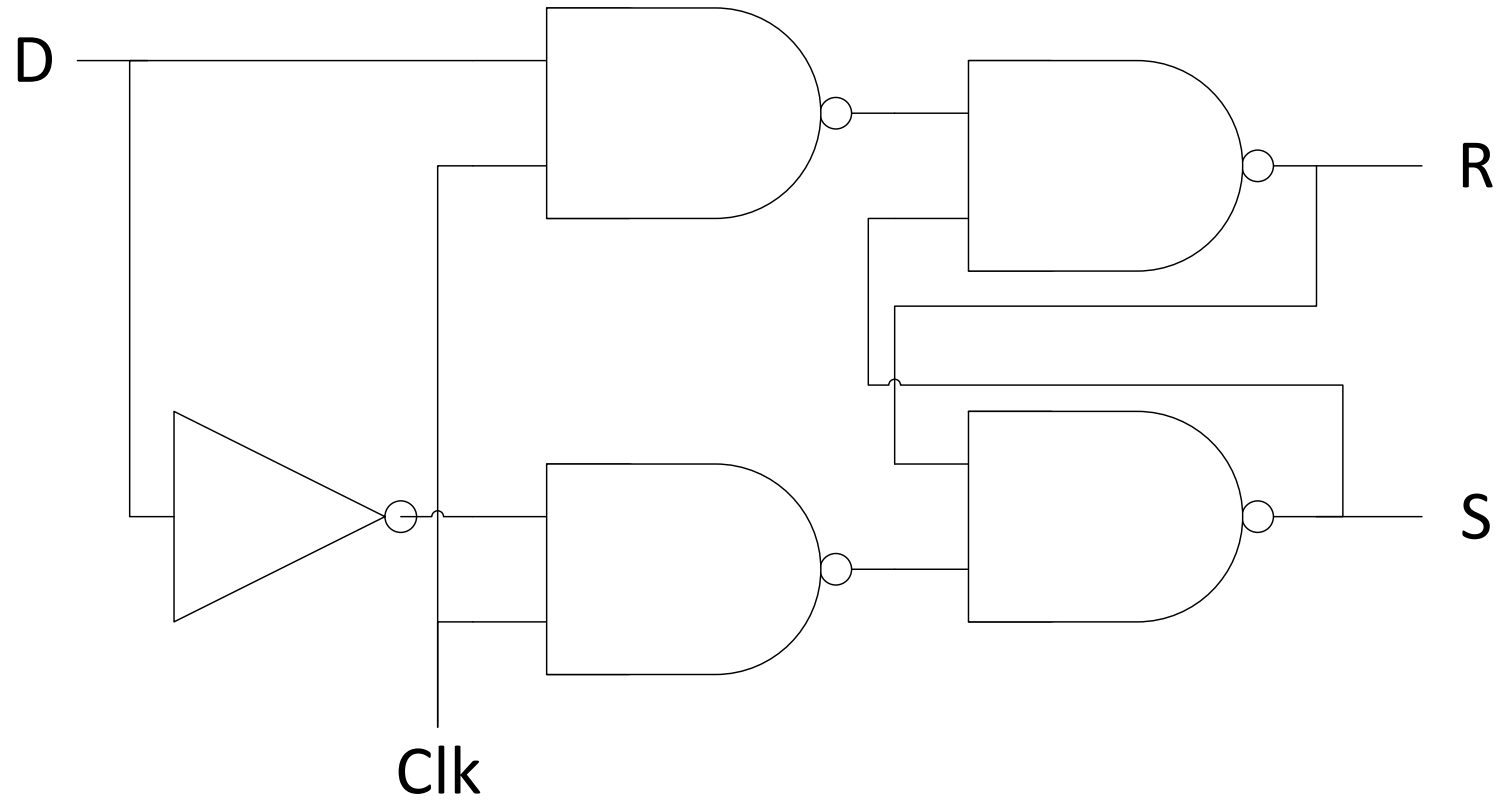


Boolean Logic Continued

Prof. James L. Frankel
Harvard University

Version of 7:15 PM 15-Feb-2016
Copyright © 2016 James L. Frankel. All rights reserved.

D Latch



D	Clk	R	S
X	0	$\sim S_0 = R_0$	$\sim R_0 = S_0$
D	1	D	$\sim D$

D Latch Observations

- Because of the inversion caused by the NAND gates to D and $\sim D$, the state of R is the same as D and the state S is the complement of D
 - In D Latches, we often refer to R as Q
 - And, we refer to S as $\sim Q$

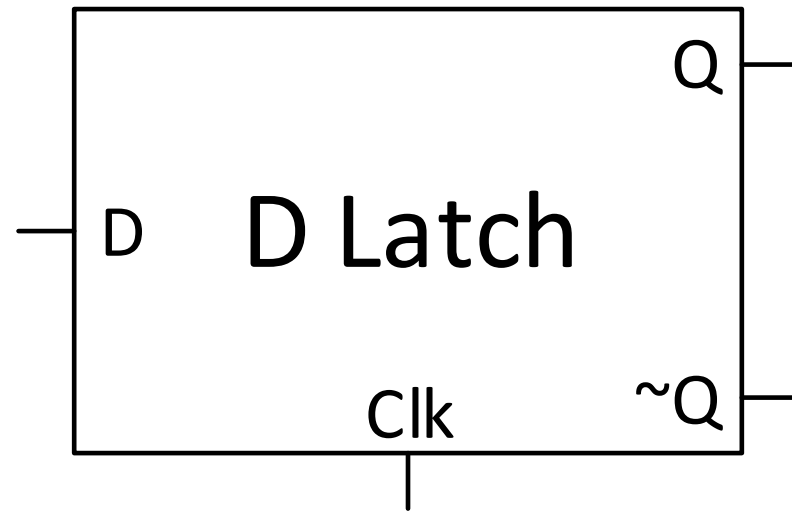
Timing Considerations

- Time is required for an output of a gate to reflect its state after inputs change
- For any one family of logic gates, this time is simplified and referred to as a **gate delay**
 - That is, one gate delay is the time that any primitive gate takes to produce a stable output after inputs change
 - The symbol tau, τ , is used to refer to the gate delay

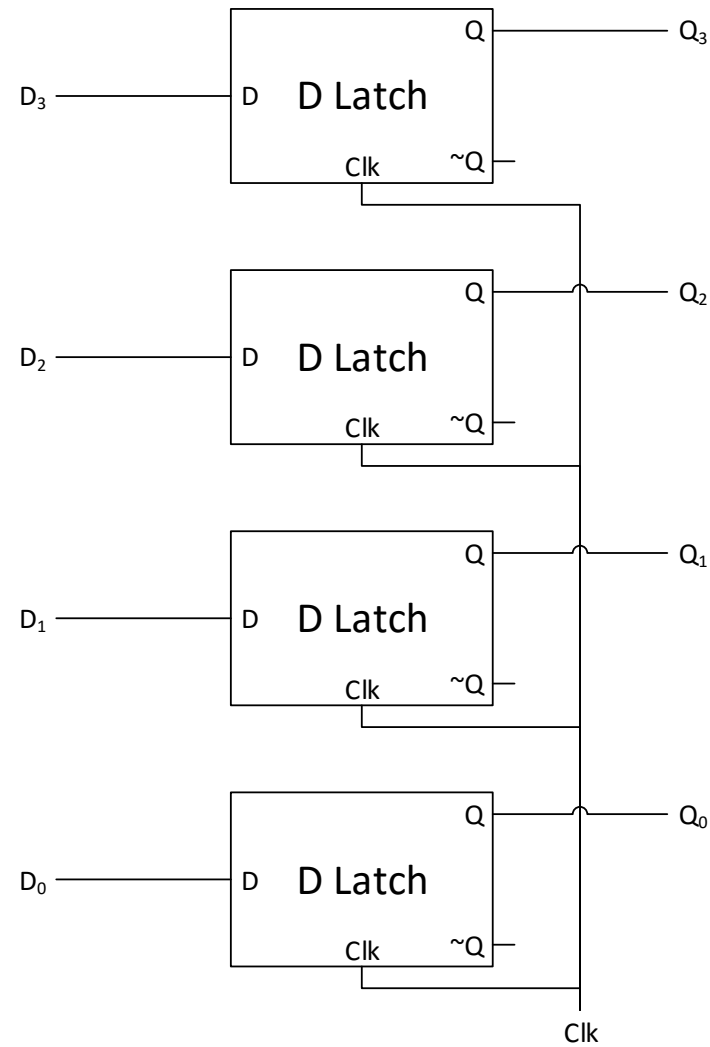
D Latch Constraints

- Clk should be low when D changes
- Before Clk goes high, D should be stable for at least one τ
 - We refer to this time as the **setup time** or t_{su}
- D should not change when Clk is high
- Clk should be high until Flip Flop is stable
 - One τ for Clk to propagate to the Flip Flop
 - A second τ for Flip Flop input to propagate to R and S
 - Therefore, Clk should be high for at least 2τ

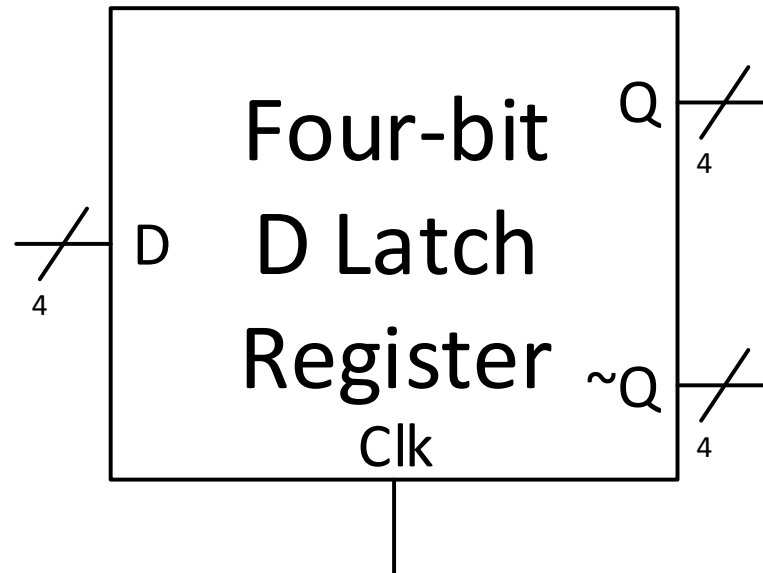
Encapsulation of D Latch



Four-bit Register Built from D Latches



Encapsulation of D Latch Four-bit Register

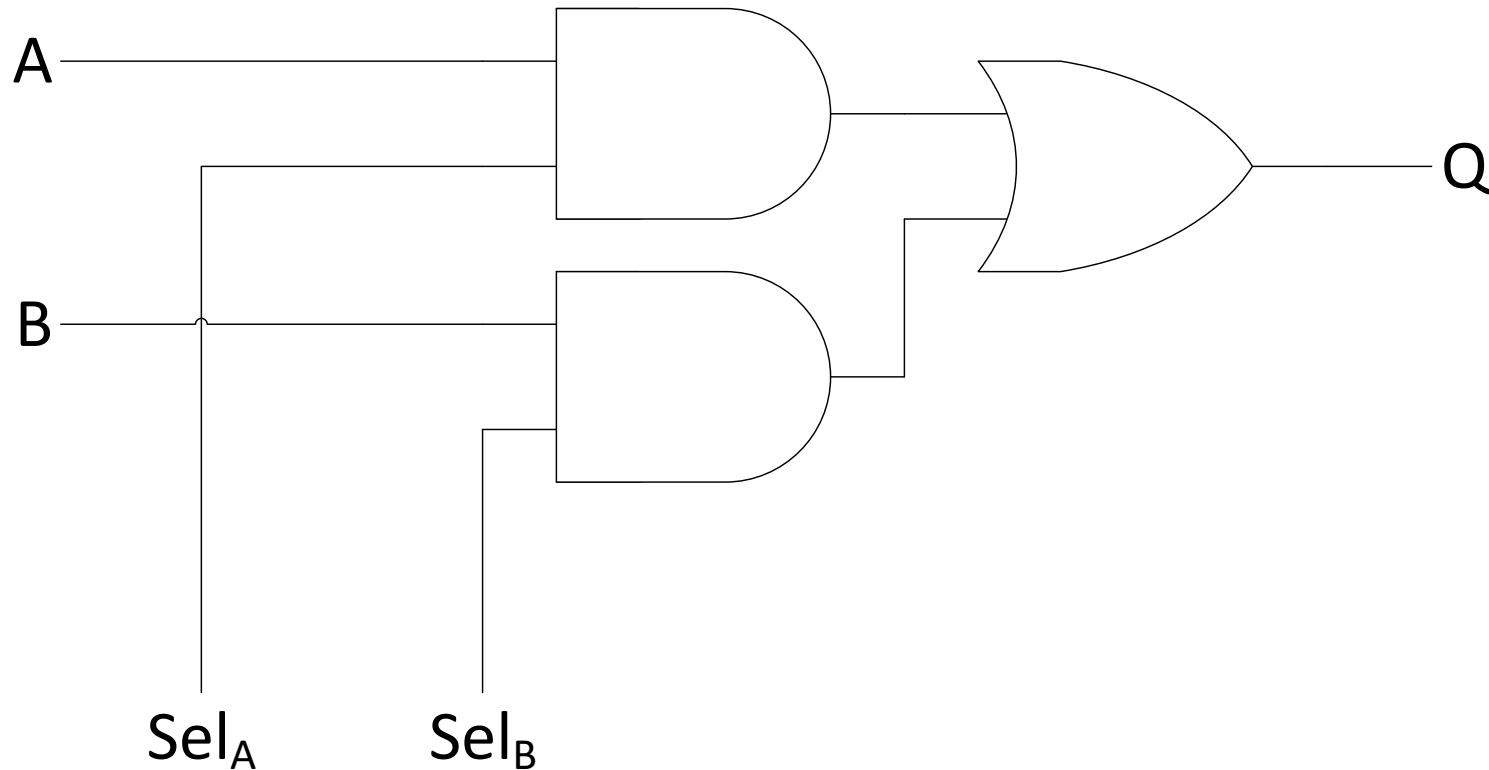


D Latch Register Encapsulation Observations

- The slashes on wires to the D Latch Register Encapsulation indicate the **bus width**
- This simplifies the diagram by collapsing replicated inputs or outputs
- In Visio, the **bus width** symbol is available under Engineering → Electrical Engineering → Transmission Paths → Bus width

Multiplexer, Mux, or Data Selector – Initial

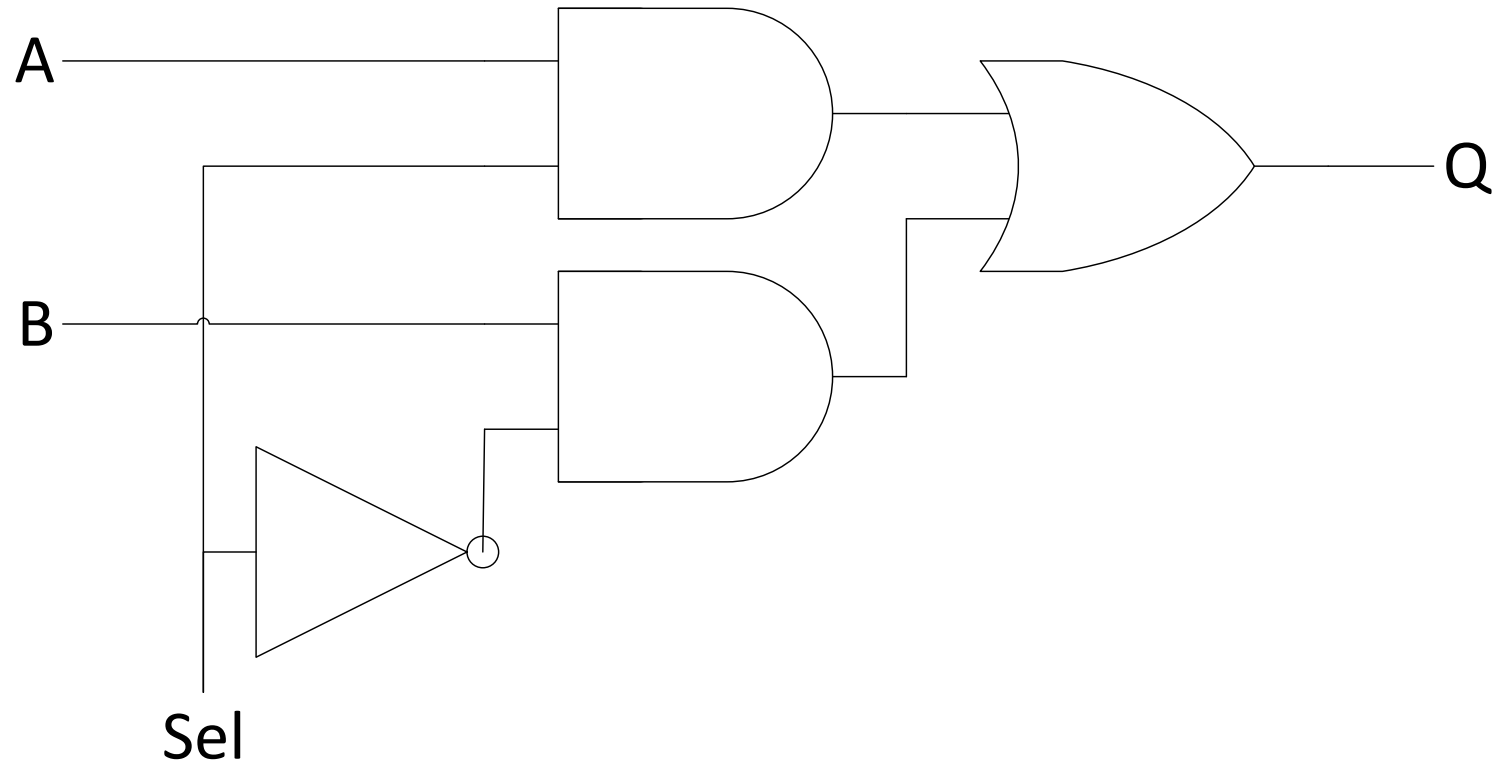
- A multiplexer selects one of multiple inputs to be the output



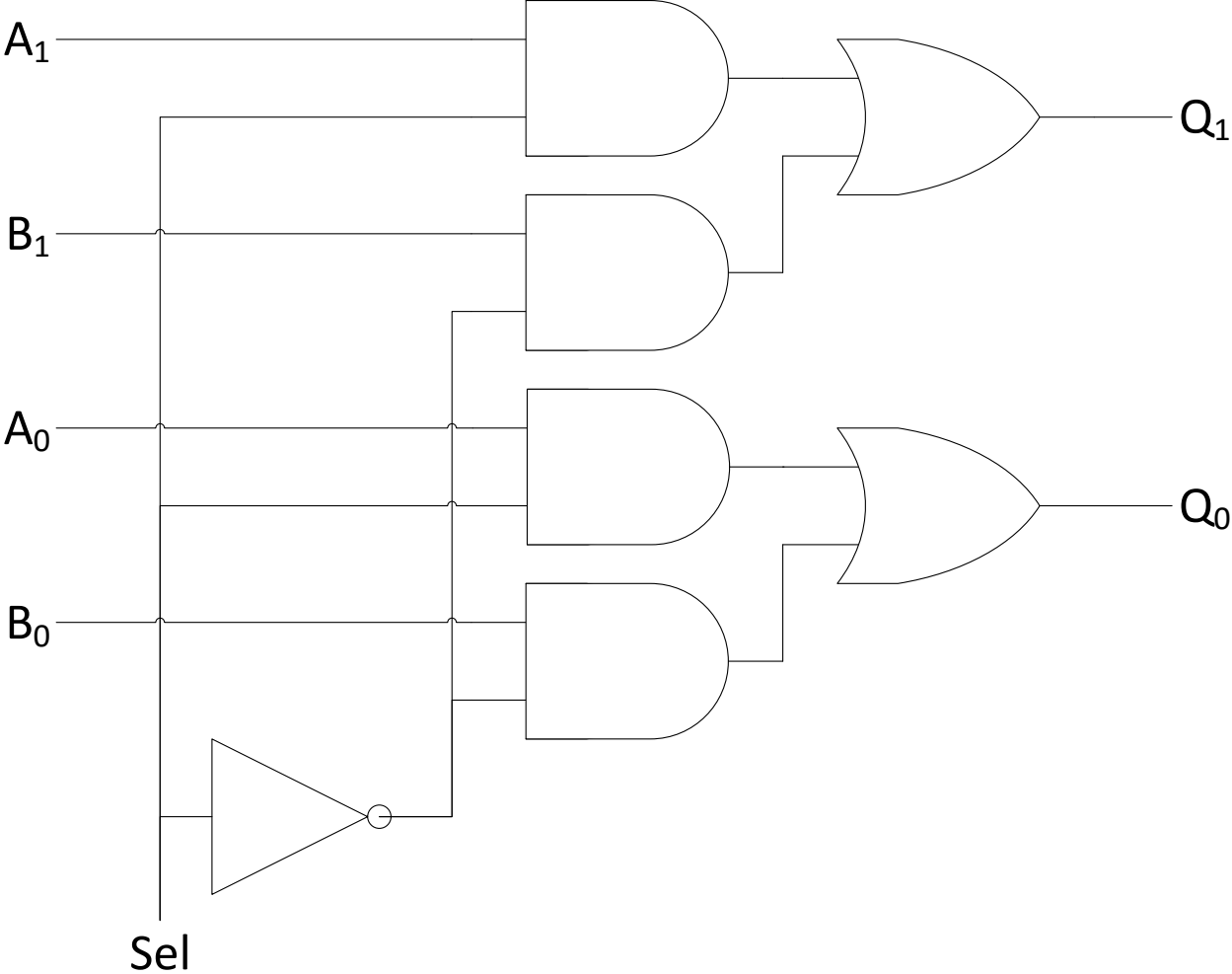
Multiplexer, Mux, or Data Selector

- Design on previous slide also allows ORing the inputs together
- Often, there is no need to be able to OR inputs
- And, this design requires multiple *Sel* inputs
- So, we can use a single *Sel* input to select one of two inputs as follows

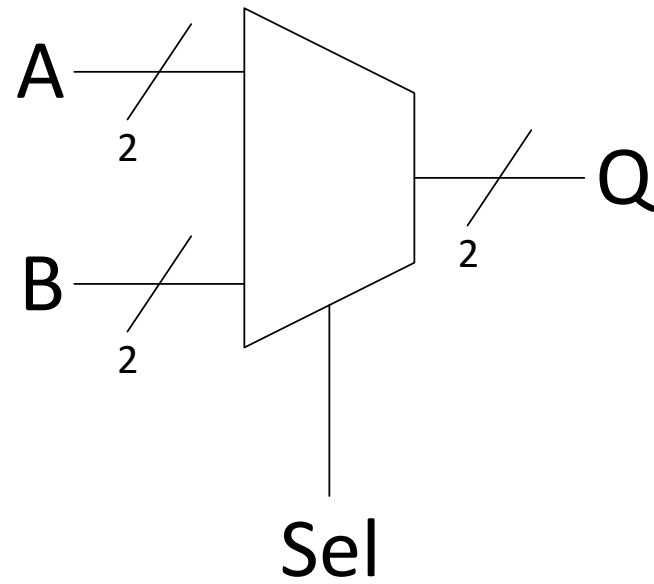
Multiplexer, Mux, or Data Selector – Final



Two-bit Mux



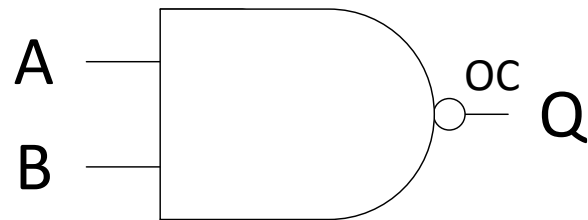
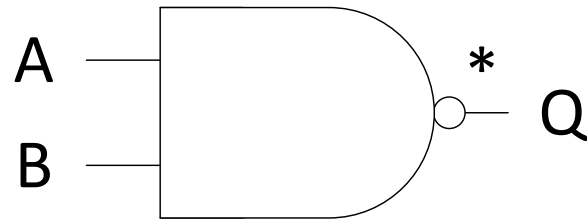
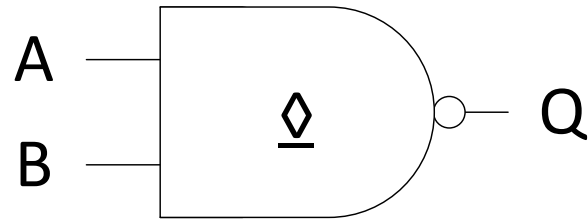
Mux Encapsulation



Open-Collector Output

- A device with an open-collector output will pull its output low when the output would be 0, but otherwise appears to be disconnected from the output
 - This disconnected state is really a high impedance state signified by Z
- Devices with open-collector outputs *can* have their outputs connected together
- Thus, when several devices with open-collector outputs have their outputs connected together, any one (or more than one) of those device(s) can pull the output low
- An external device must be used to default the connected outputs to be high when no open-collector output is pulling the output low
 - Typically, this is a resistor

Open-Collector Schematic Symbols



Truth Table for Open-Collector NAND Gate

A	B	Q
0	0	Z
0	1	Z
1	0	Z
1	1	0

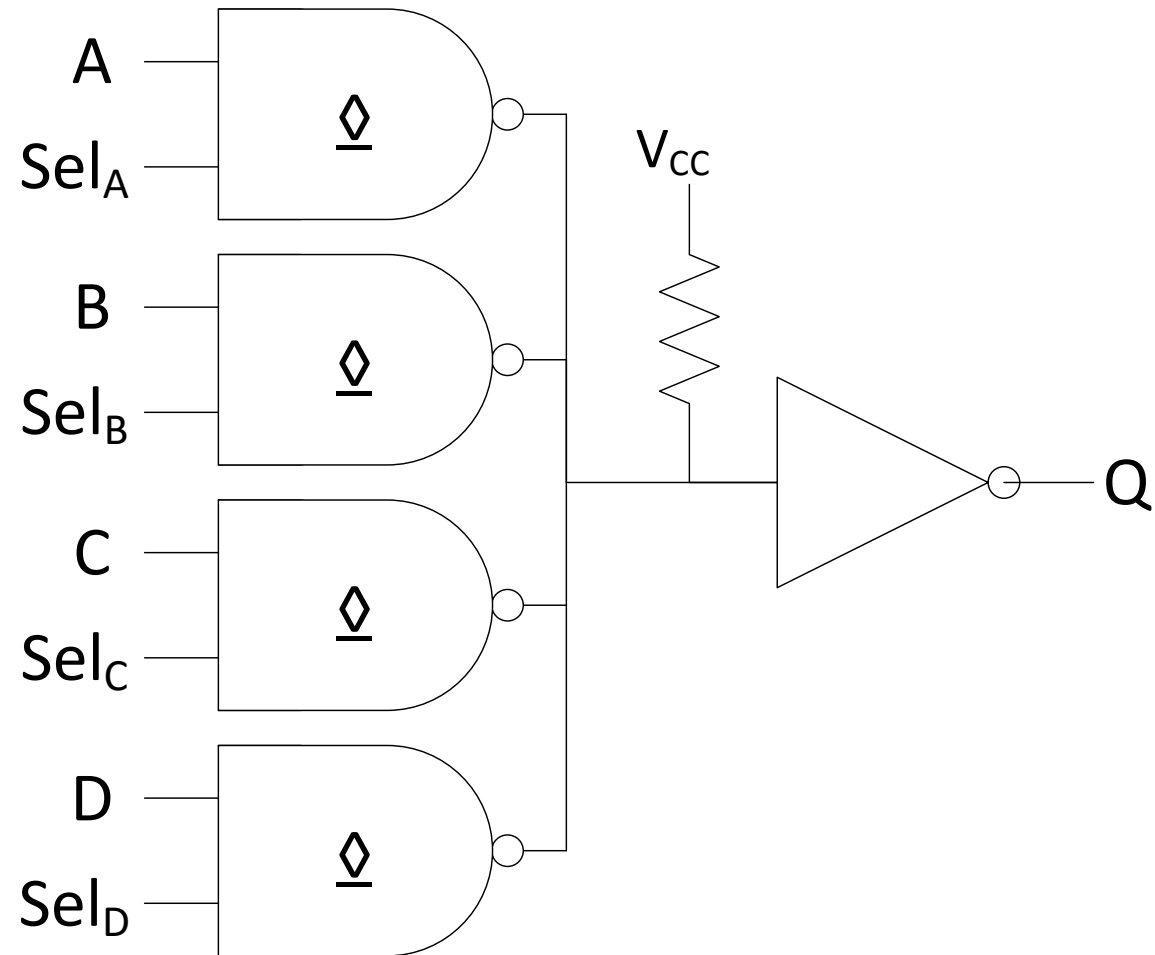
Open-Collector Output Observations

- Of course, any gate can have an open-collector output
- There is a limit to the number of devices with open-collector outputs whose outputs can be connected together
- Open-collector circuits are often called “active-low wired-OR” or “active-high wired-AND”
 - Why?

Implementing a Bus Using Open-Collector Output Devices

- A bus is a wire that allows any one of several signals to be driven onto it
 - Thus, it functions as a mux, but is implemented as a wire
 - It may be driven by open-collector outputs
- Implements a mux, but doesn't require as many gates

Open-Collector Bus



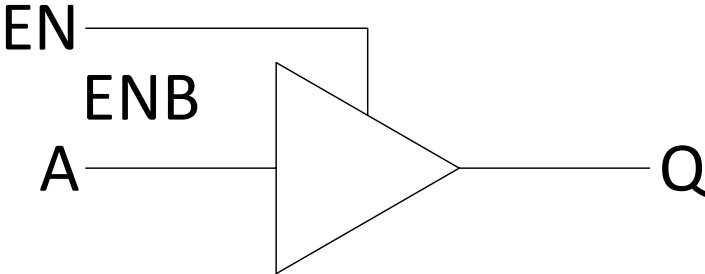
Open-Collector Devices on our Altera FPGA

- The bad news is that, alas, our Altera FPGA does not have any open-collector devices
 - Therefore, we can't use them in our designs

Tri-State Output Devices

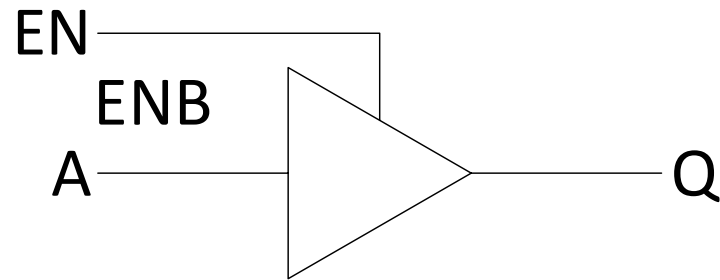
- Another device that can be used to directly implement a bus is a device with a tri-state output
- These devices are able to drive their outputs either high or low *or* place their output into a high impedance state
- The output is put into a high impedance state using an additional **enable** input
- Devices with tri-state outputs can have their outputs connected together, but must be carefully managed

Tri-State Buffer or Driver



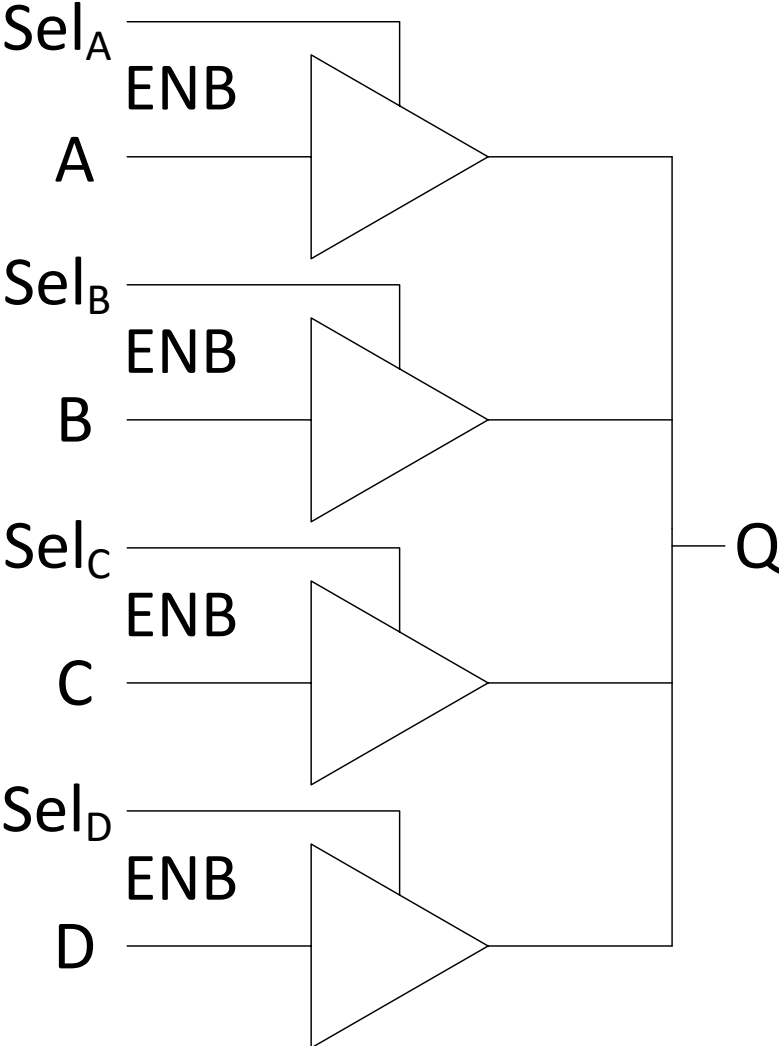
A	EN	Q
X	0	Z
0	1	0
1	1	1

Simplified Truth Table for Tri-State Buffer or Driver



EN	Q
0	Z
1	A

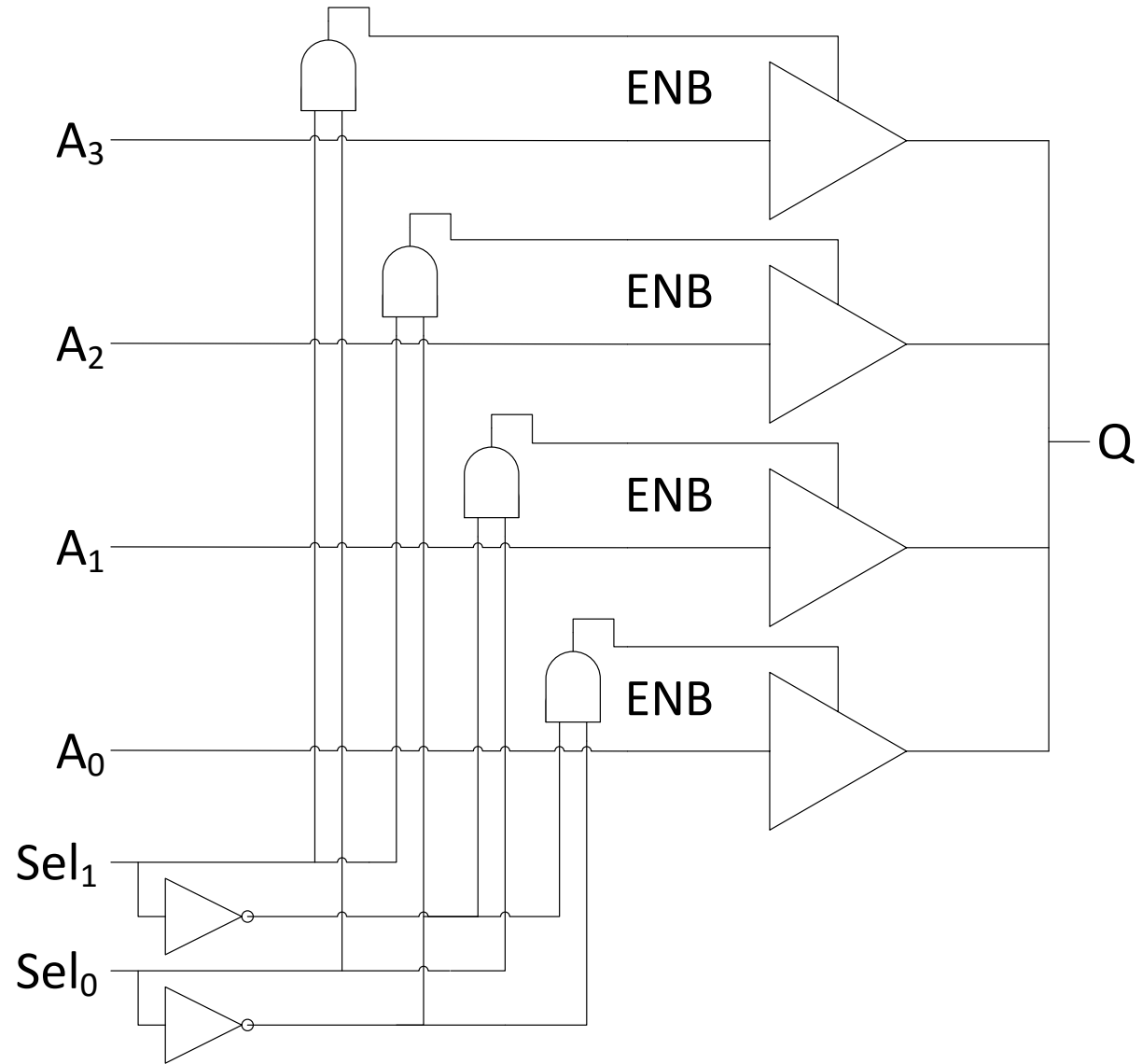
Tri-State Bus



Tri-State Output Device Observations

- Because a tri-state device will drive its output either high or low if it is enabled,
 - ***Never, never, never*** enable more than one tri-state device driving the same wire
- The previous circuit was potentially dangerous
 - What if more than one *Sel* line was asserted?
- We make sure that this can never happen by building devices that use tri-state drivers with circuitry that selects only one enable line at a time for tri-states that drive the same wire

Safe Tri-State Bus



Truth Table for Safe Tri-State Bus

A ₃	A ₂	A ₁	A ₀	Sel ₁	Sel ₀	Q
X	X	X	0	0	0	0
X	X	X	1	0	0	1
X	X	0	X	0	1	0
X	X	1	X	0	1	1
X	0	X	X	1	0	0
X	1	X	X	1	0	1
0	X	X	X	1	1	0
1	X	X	X	1	1	1

Simplified Truth Table for Safe Tri-State Bus

Sel_1	Sel_0	Q
0	0	A_0
0	1	A_1
1	0	A_2
1	1	A_3

Further Simplified Truth Table for Safe Tri-State Bus

Sel _{1..0}	Q
0	A ₀
1	A ₁
2	A ₂
3	A ₃

Tri-State Devices on our Altera FPGA

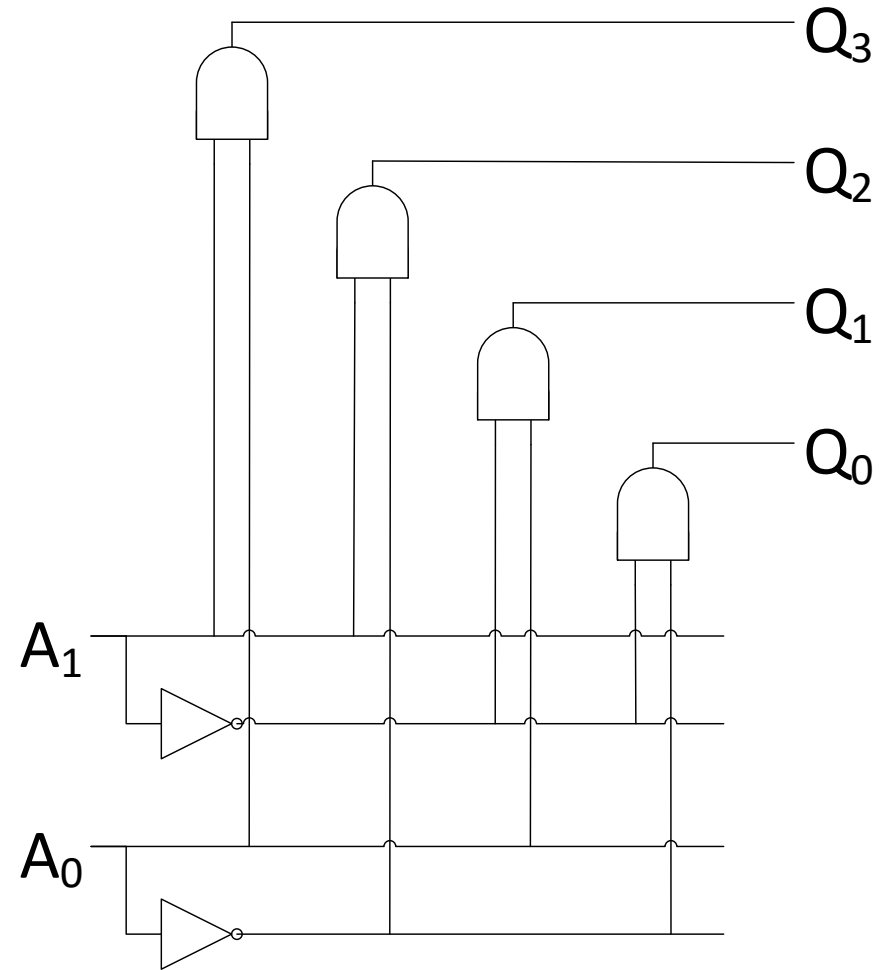
- The bad news is that, alas, our Altera FPGA does not have any tri-state devices
 - Therefore, we can't use them in our designs

Decoder

- A decoder asserts one of multiple outputs based on an input binary number
- We used one in our Safe Tri-State Bus
- Here is a truth table for a 2-to-4 line decoder

A_1	A_0	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Decoder Schematic



Decoder Encapsulation

