

CPS 104
Computer Organization and Programming
Lecture-10: Digital Logic Design.

Sep. 27, 1999

Dietolf (Dee) Ramm

<http://www.cs.duke.edu/~dr/cps104.html>

Overview of Today's Lecture:

- **Review: Truth tables, Boolean functions, Gates and Circuits**
- **Digital circuit examples: 2-1 MUX, Full Adder**
- **The ALU**
- **Shift Unit**
- **Memory elements: Latch, Data-FlipFlop**

Read Appendix B

Boolean Functions and Gates (Cont.)

- **Examples:** Boolean functions: NOT, AND, OR, XOR, . . .

a	NOT(a)
0	1
1	0

a	b	AND(a,b)
0	0	0
0	1	0
1	0	0
1	1	1

a	b	OR(a,b)
0	0	0
0	1	1
1	0	1
1	1	1

a	b	XOR(a,b)
0	0	0
0	1	1
1	0	1
1	1	0

a	b	XNOR(a,b)
0	0	1
0	1	0
1	0	0
1	1	1

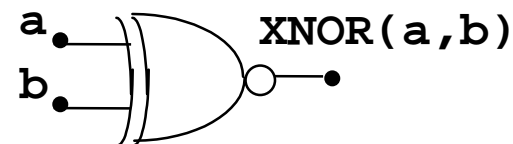
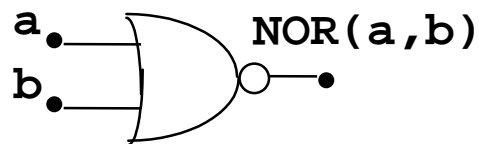
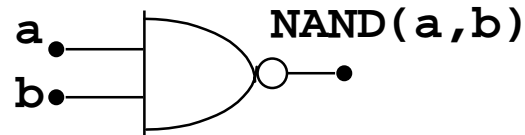
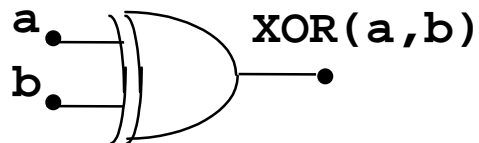
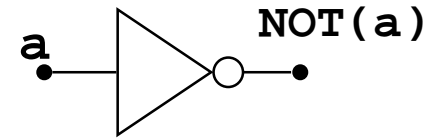
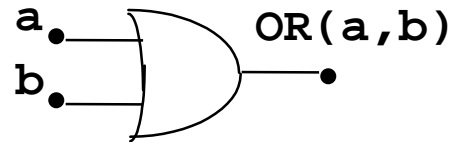
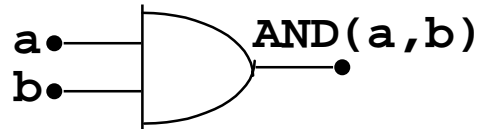
a	b	NOR(a,b)
0	0	1
0	1	0
1	0	0
1	1	0

Boolean Functions and Gates

(Cont.)

- **Gates are electronics devices that implements simple Boolean functions:**

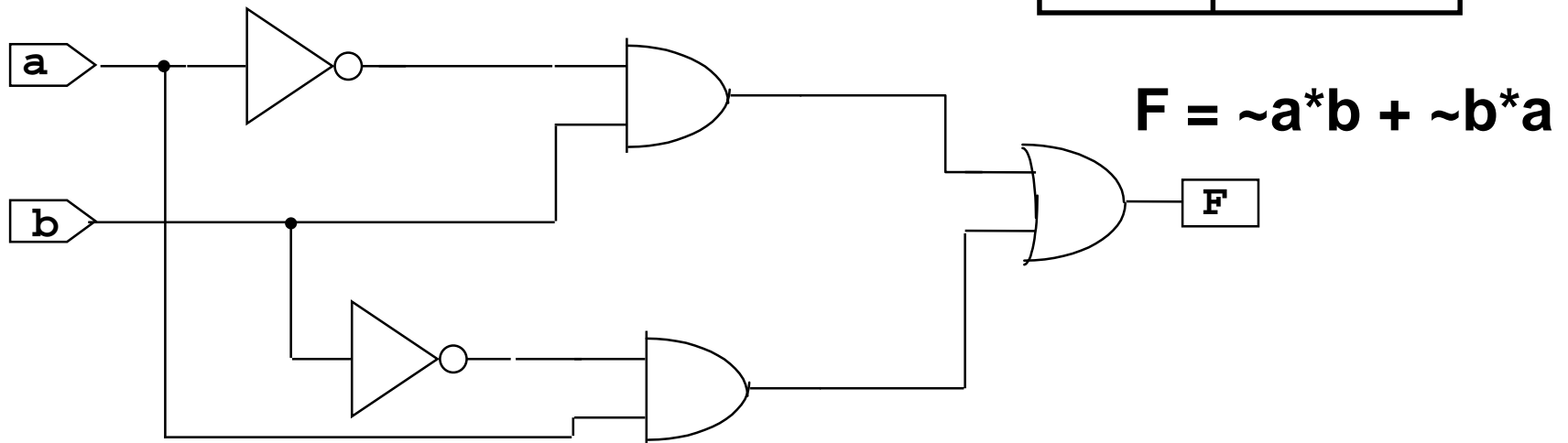
- **Examples:**



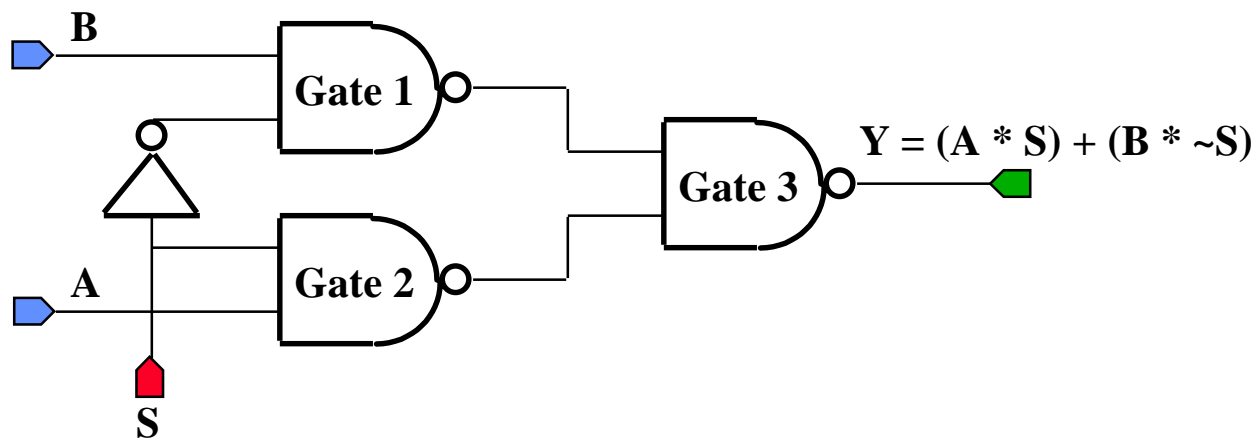
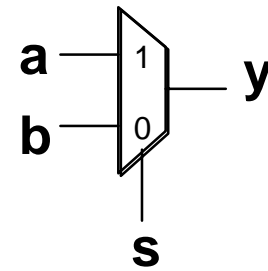
Boolean Functions, Gates and Circuits

- **Circuits** are made from a network of gates. (function compositions).
- **Example:**

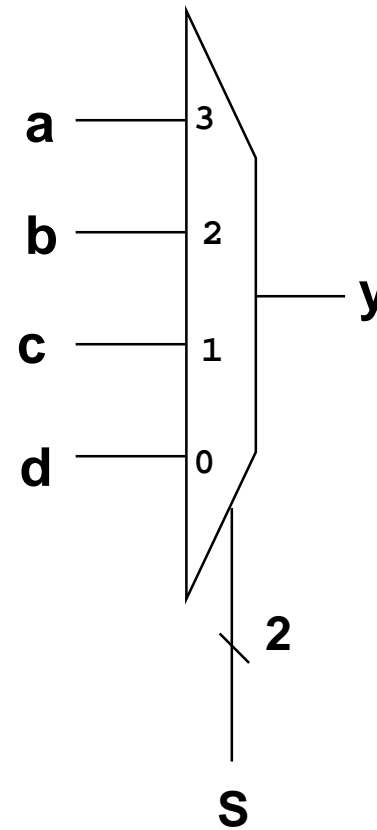
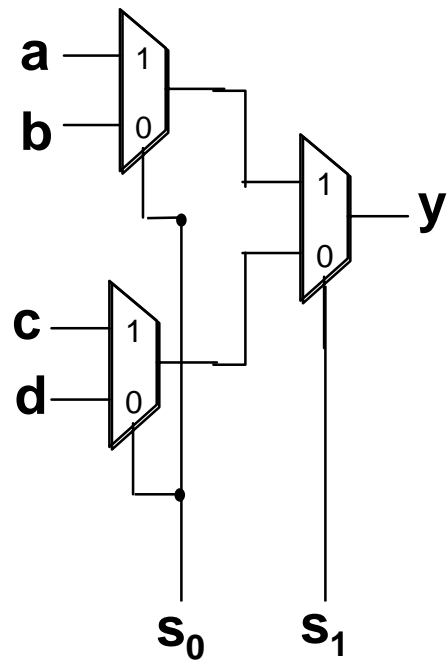
a	b	XOR(a,b)
0	0	0
0	1	1
1	0	1
1	1	0



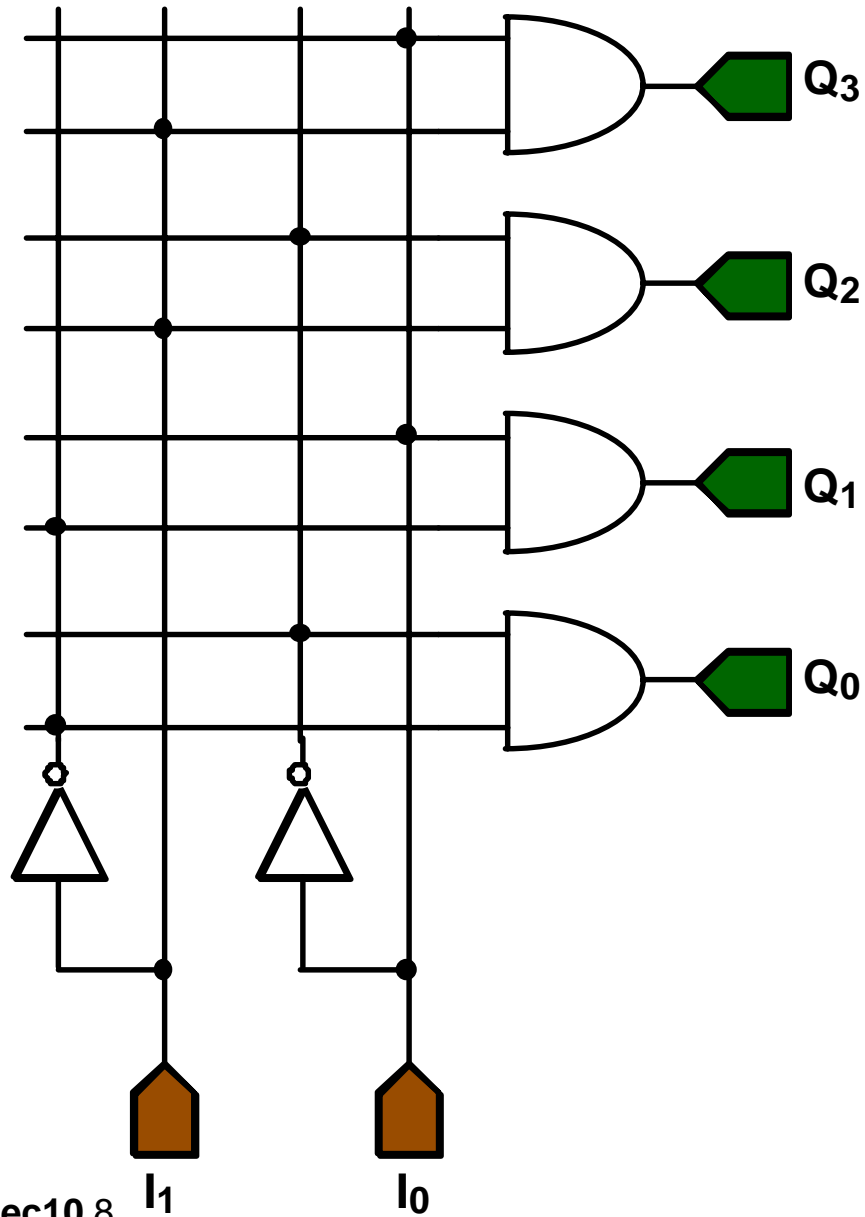
Circuit Example: 2x1 MUX



Example 4x1 MUX

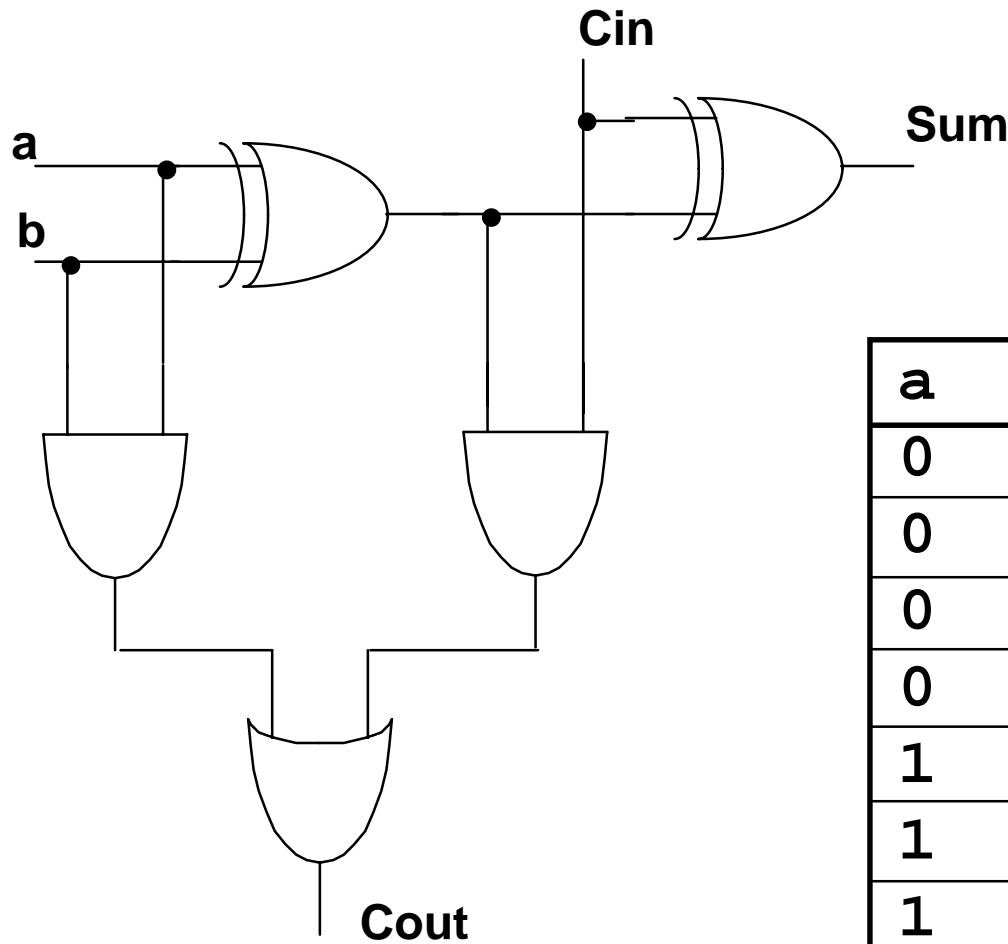


Circuit Example: Selector



I_1	I_0	Q_0	Q_1	Q_2	Q_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

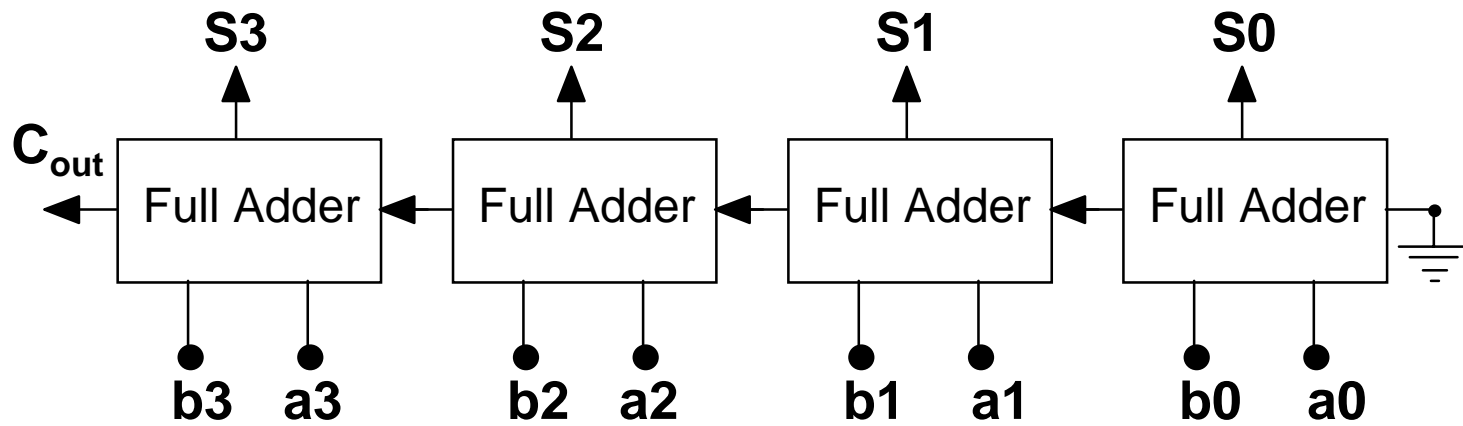
Full Adder



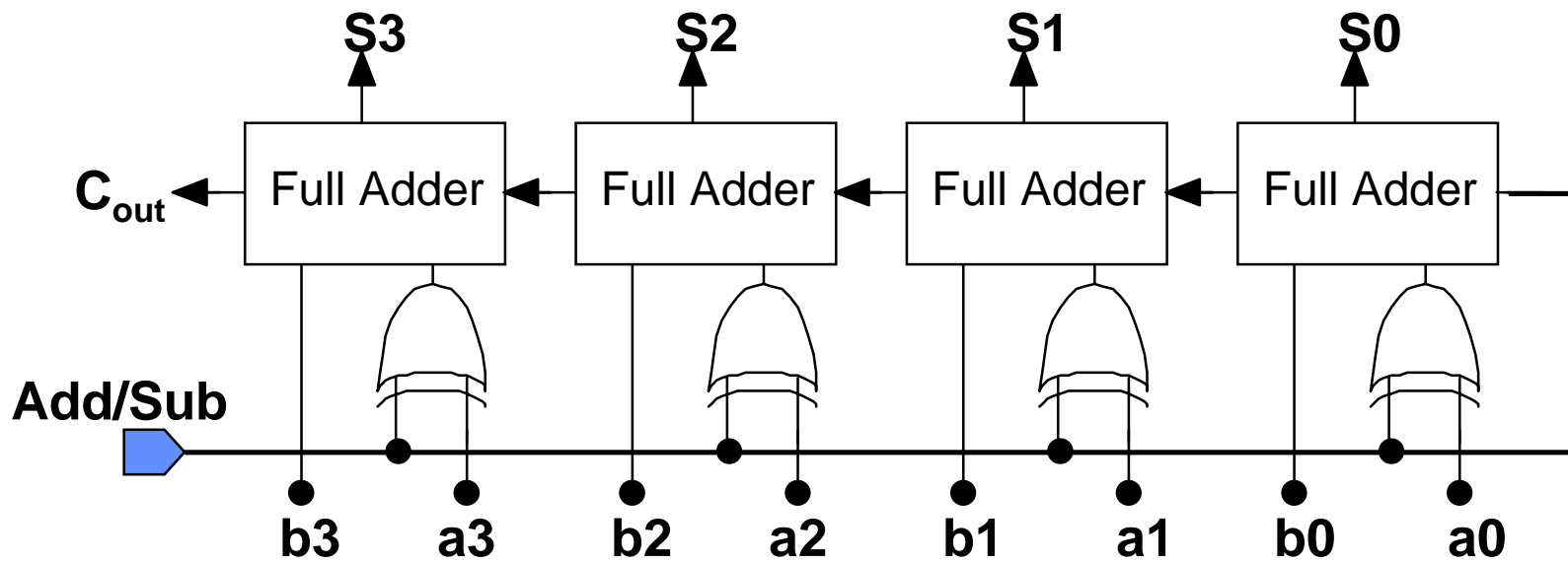
01101100
01101101
+00101100
10011001

a	b	C _{in}	Sum	C _{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Example: 4-bit adder



Example: Adder/Subtractor



Overflow

Example1:

$$\begin{array}{r} 01000000 \\ \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \\ 0110101_2 \quad (= 53_{10}) \\ + 0101010_2 \quad (= 42_{10}) \\ \hline 1011111_2 \quad (= -33_{10}) \end{array}$$

Example2:

$$\begin{array}{r} 10000000 \\ \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \\ 1010101_2 \quad (= -43_{10}) \\ + 1001010_2 \quad (= -54_{10}) \\ \hline 0011111_2 \quad (= 31_{10}) \end{array}$$

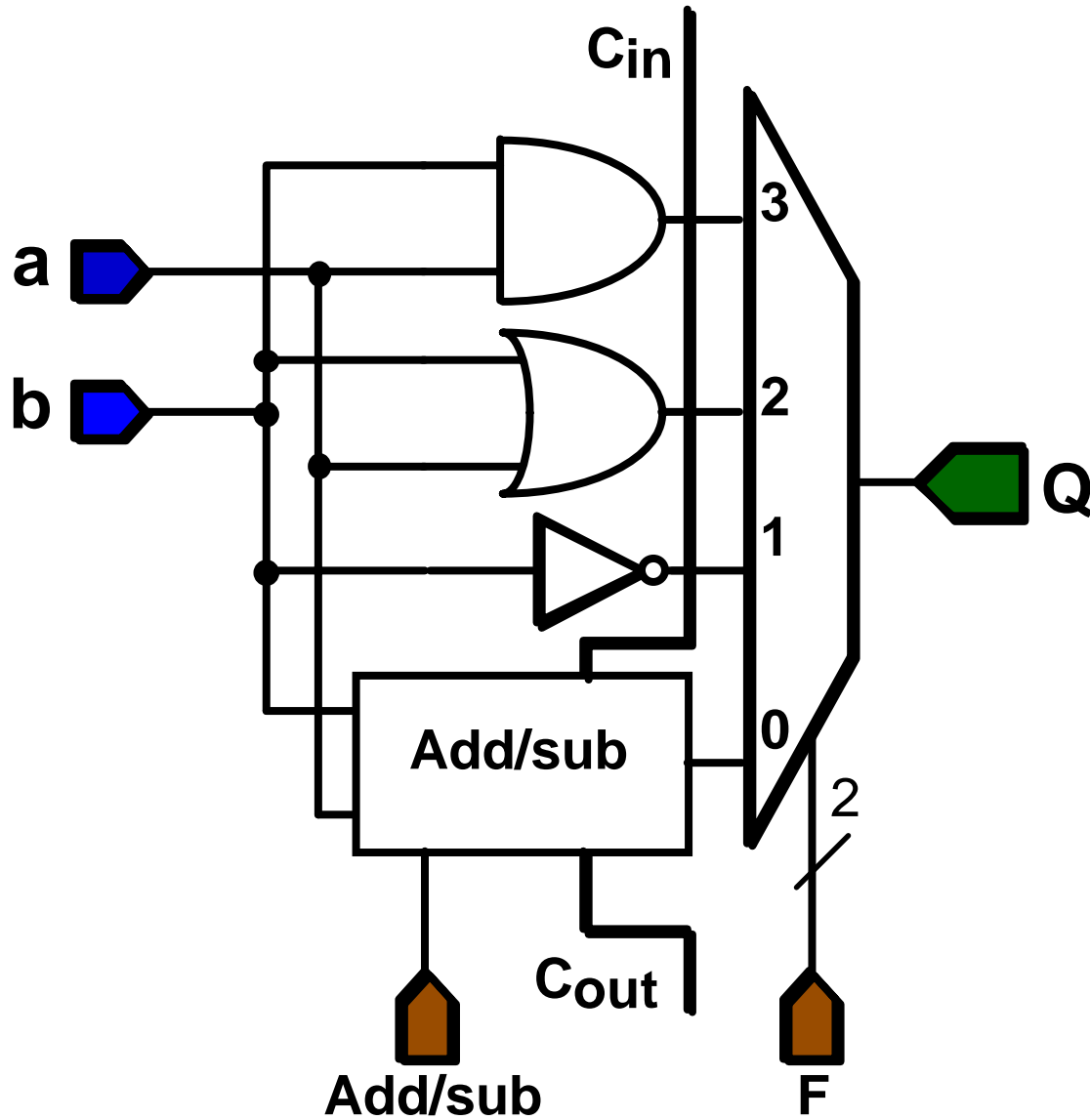
Example3:

$$\begin{array}{r} 11000000 \\ \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \\ 0110101_2 \quad (= 53_{10}) \\ + 1101010_2 \quad (= -22_{10}) \\ \hline 0011111_2 \quad (= 31_{10}) \end{array}$$

Example4:

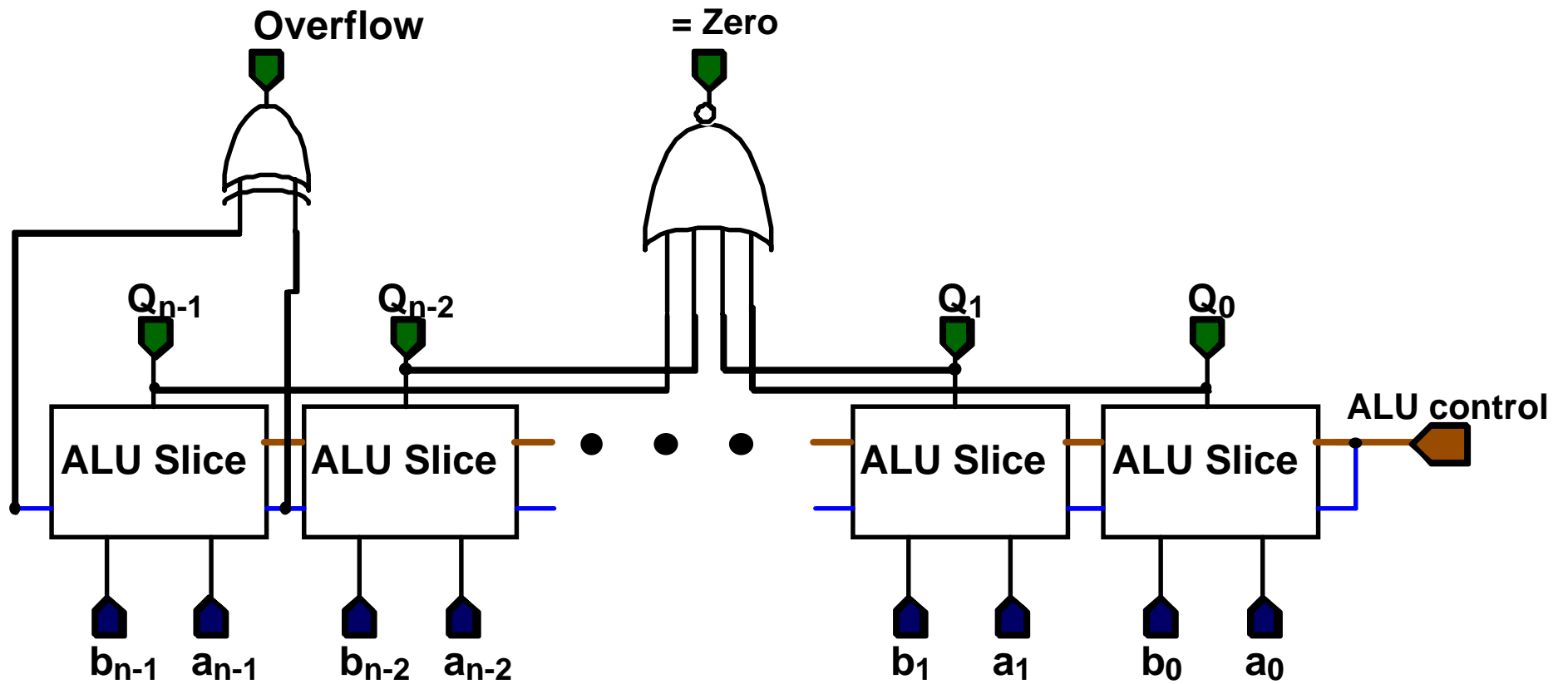
$$\begin{array}{r} 00000000 \\ \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \\ 0010101_2 \quad (= 21_{10}) \\ + 0101010_2 \quad (= 42_{10}) \\ \hline 0111111_2 \quad (= 63_{10}) \end{array}$$

ALU Slice

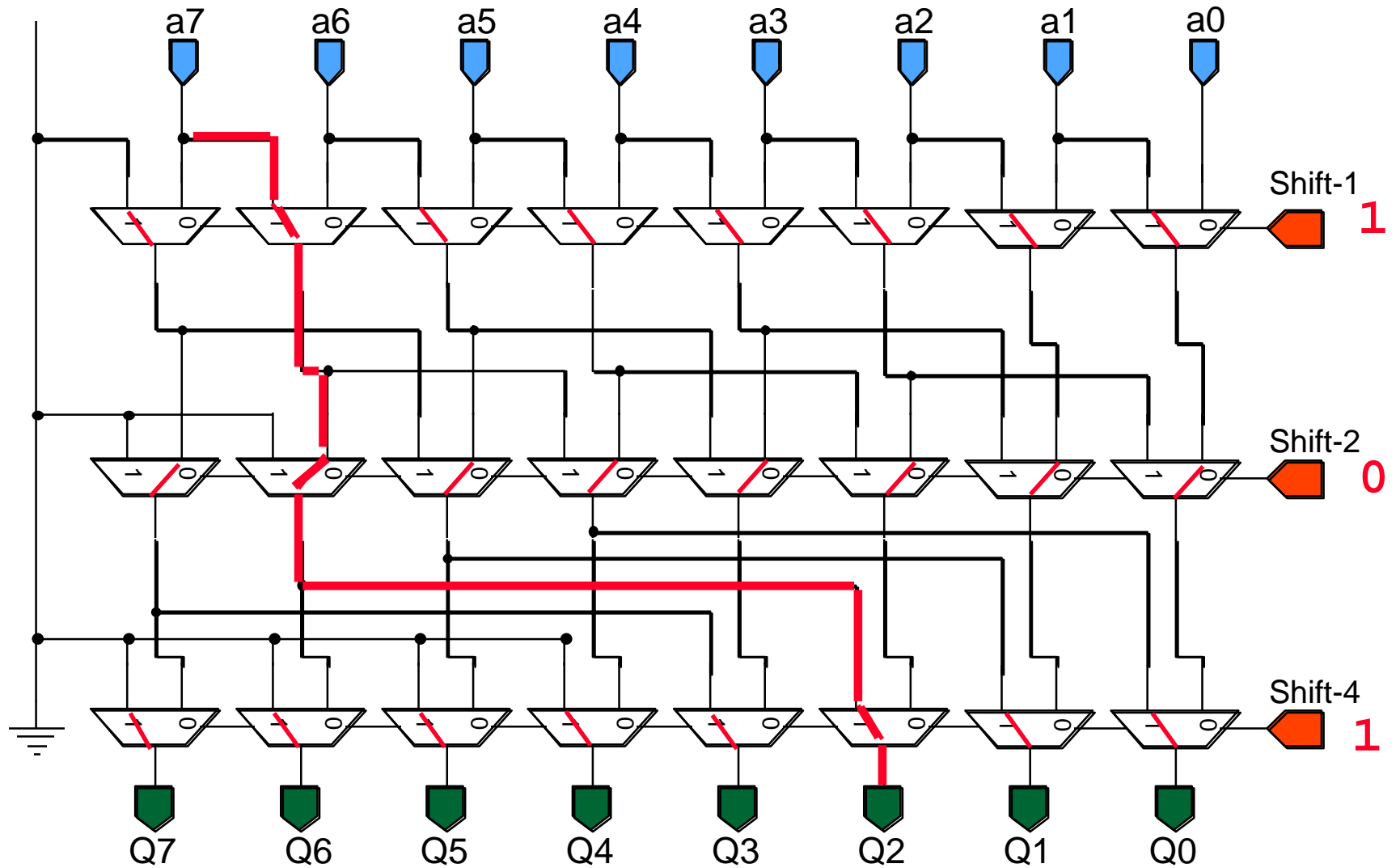


A	F	Q
0	0	$a + b$
1	0	$a - b$
-	1	NOT b
-	2	a OR b
-	3	a AND b

The ALU



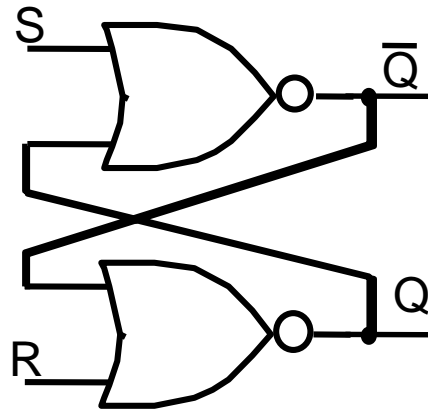
Shifter



Memory Elements

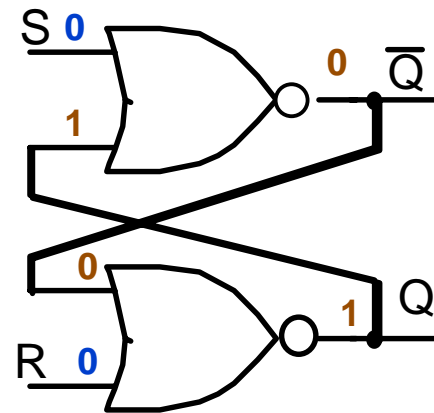
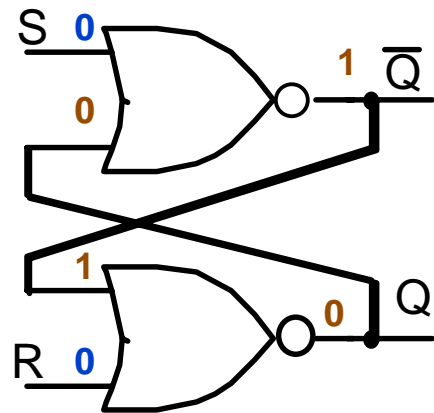
- All the circuit we looked at so far are **combinational circuits**: the output is a Boolean function of the inputs.
- We need circuits that can remember values. (registers)
- The output of the circuit is a function of the input AND a function of a stored values (state) .
- Circuits with memory are called **sequential circuits**.

Rest-Set Latch

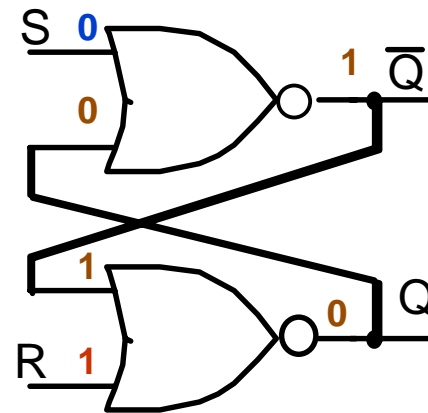
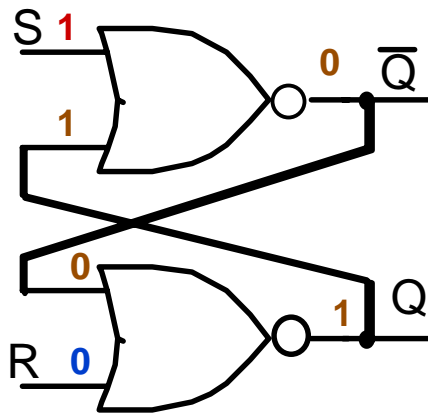


R	S	Q
0	0	Q
0	1	1
1	0	0
1	1	-

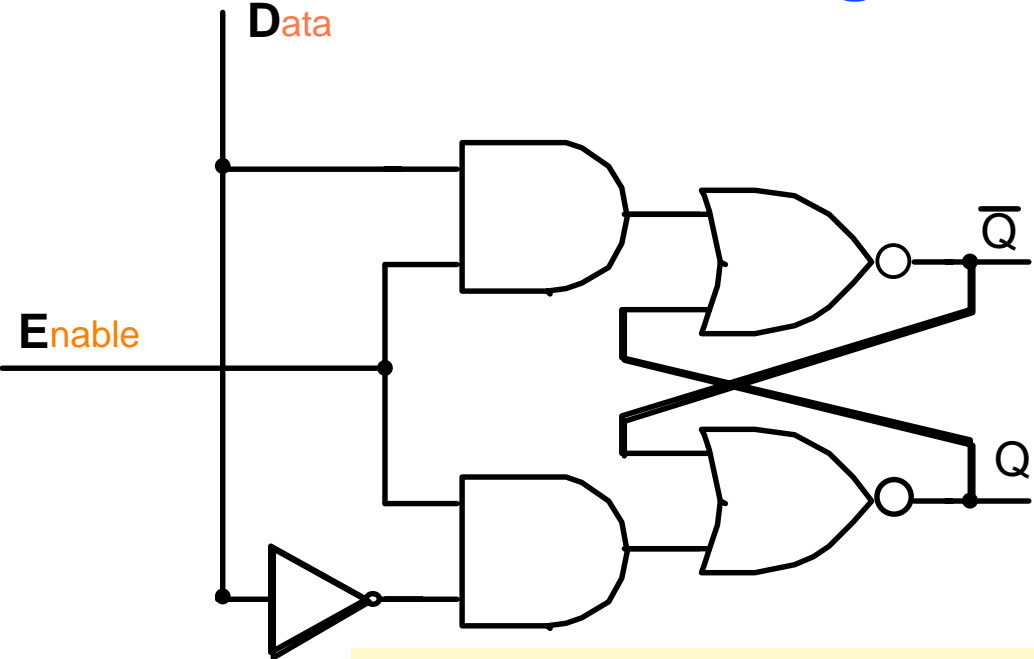
Rest-Set Latch (cont.)



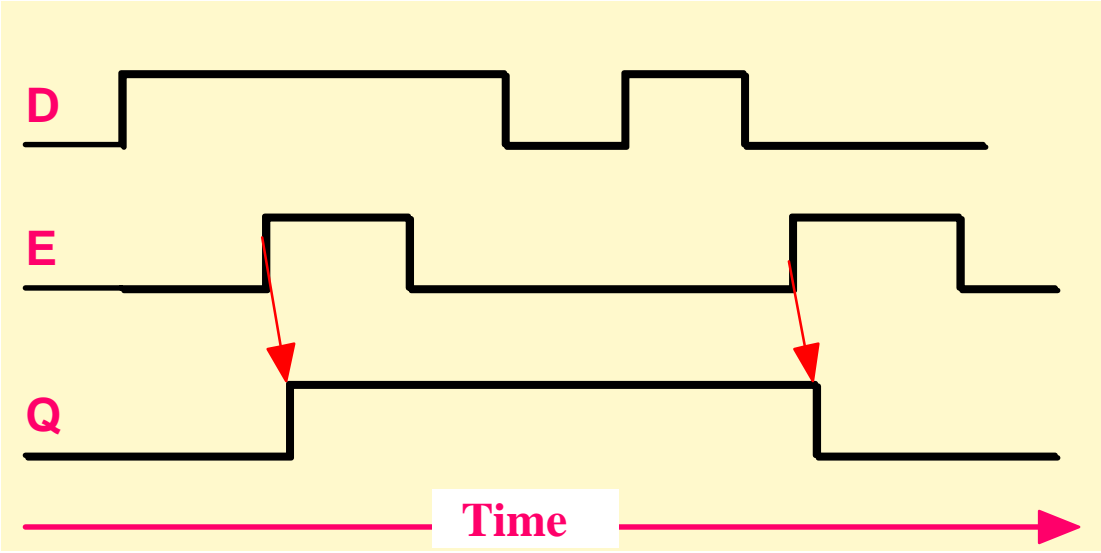
Rest-Set Latch (cont.)



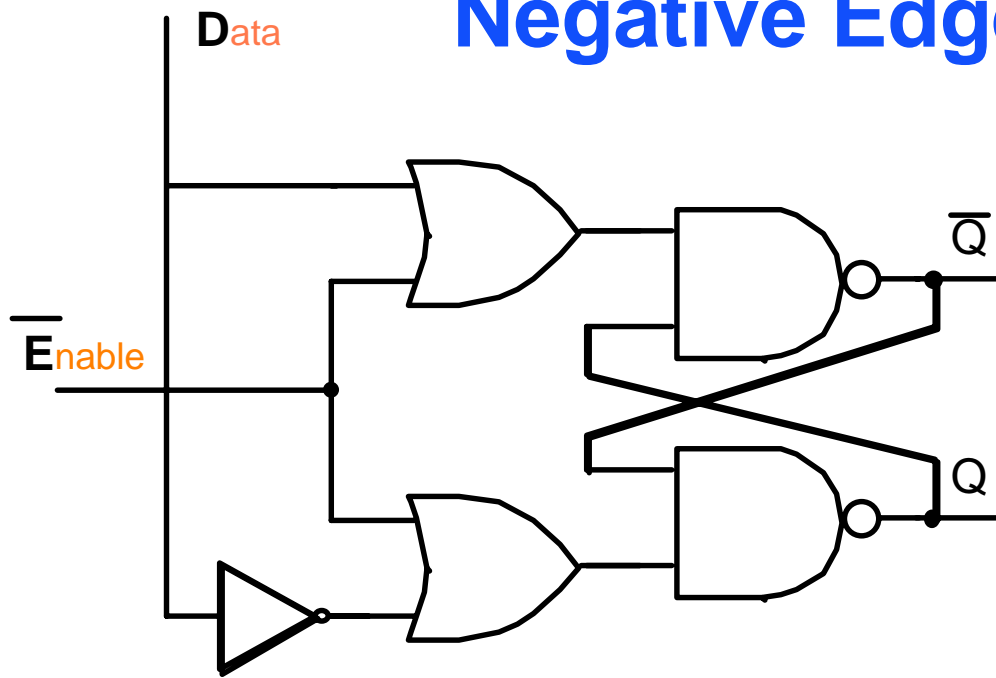
Positive Edge Data-Latch



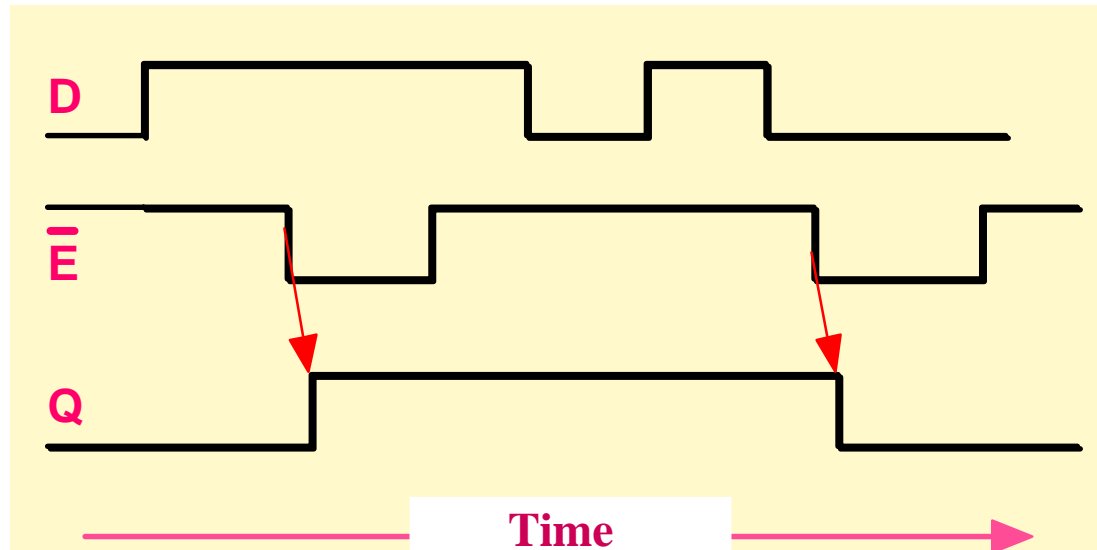
D	E	Q
0	1	0
1	1	1
-	0	Q



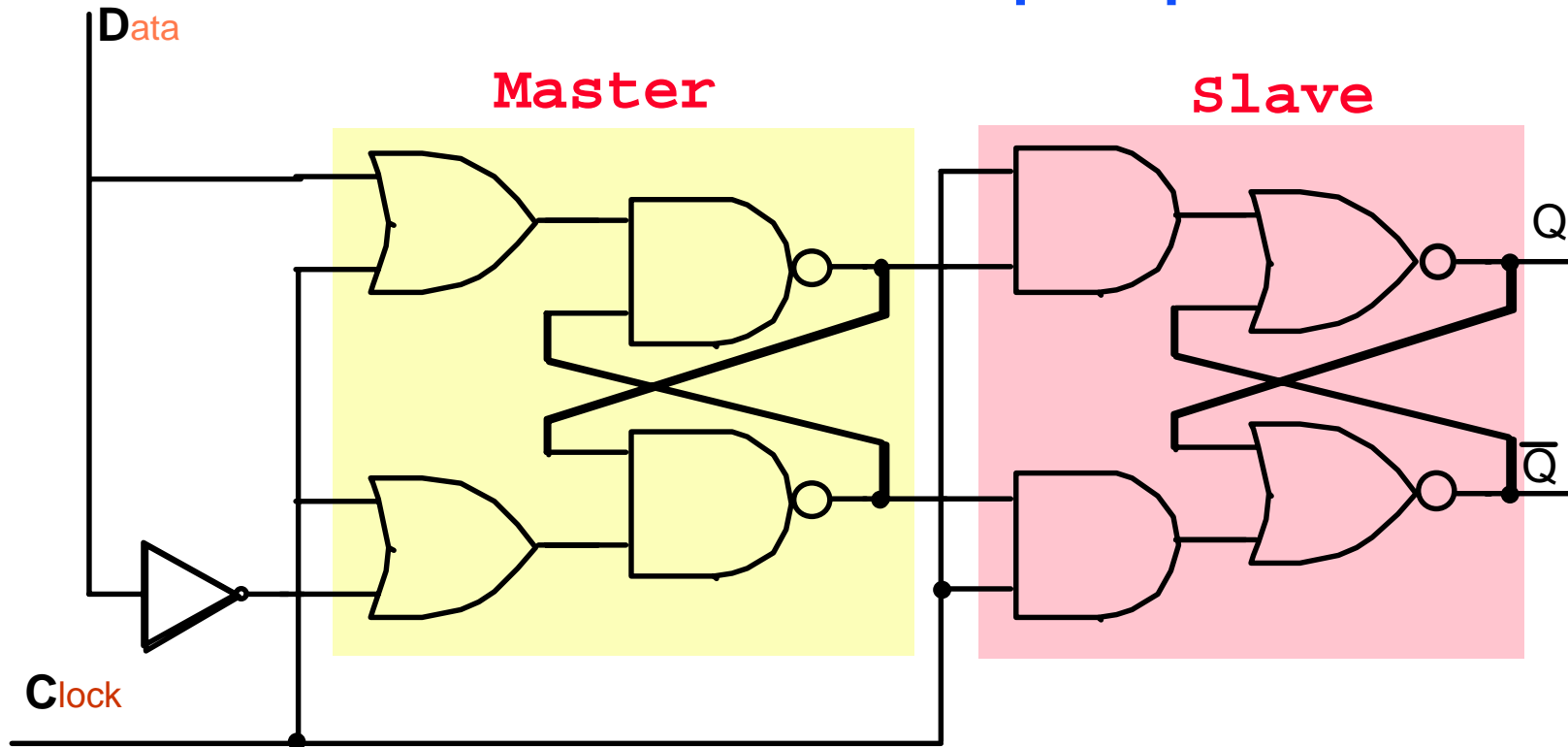
Negative Edge D-Latch



D	E	Q
0	1	0
1	1	1
-	0	Q

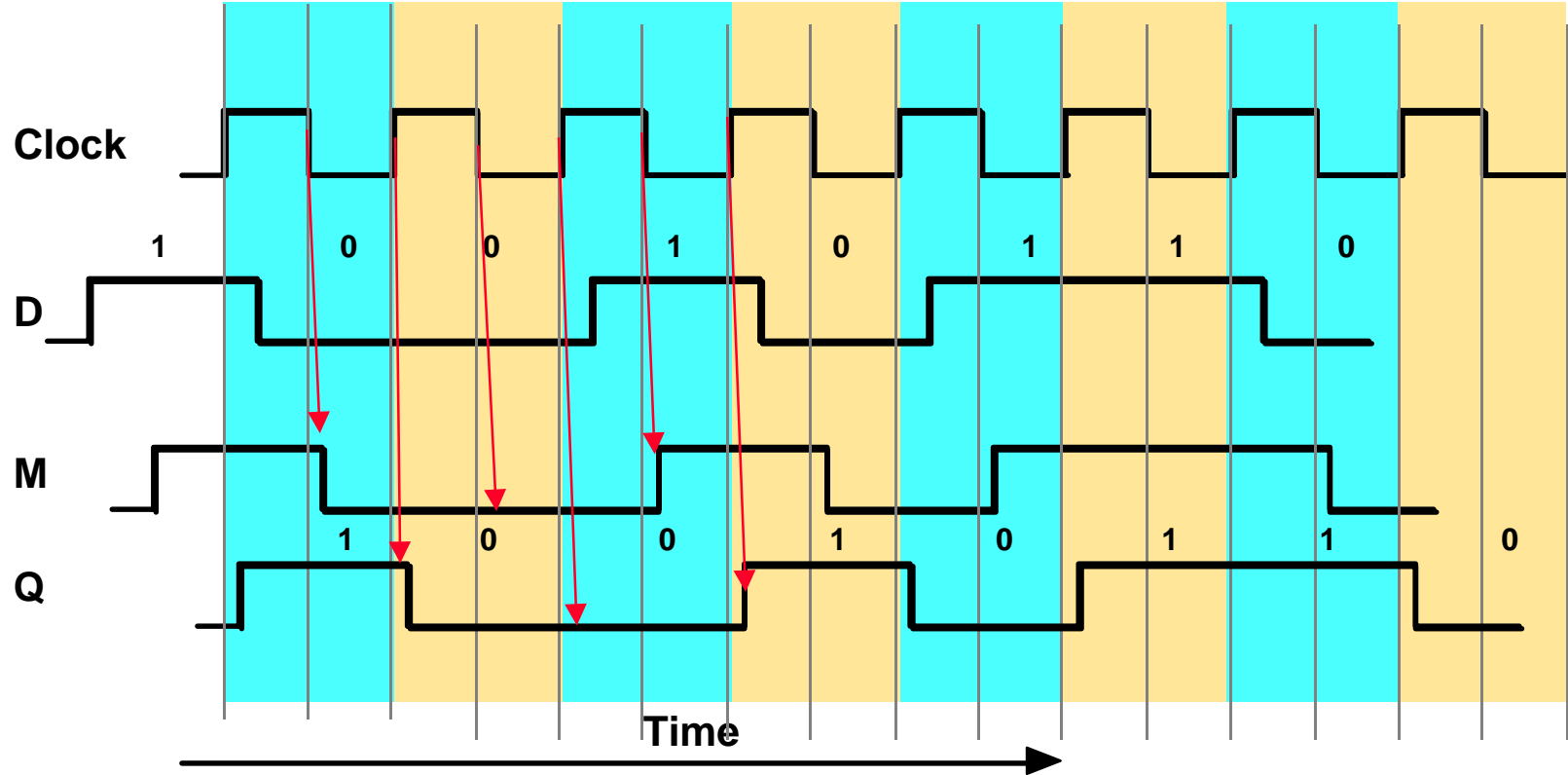
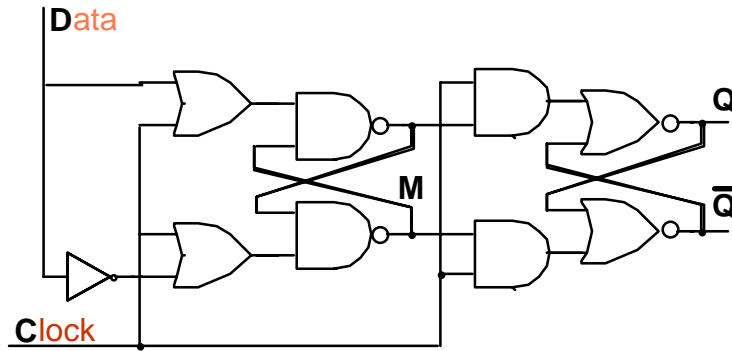


Master-Slave Data-Flip-Flop



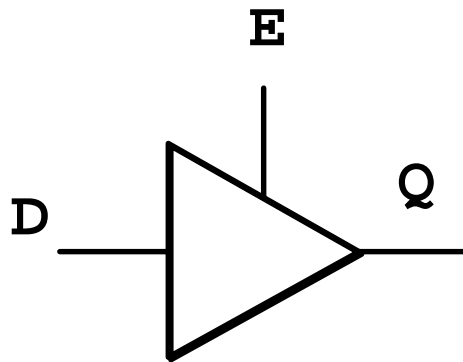
- On C ↓ D is transferred to the master stage and the slave is stable.
- On C ↑ the Master stage is transferred into the slave stage (output), and the master stage is stable.

DFF Timing



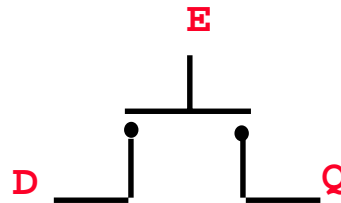
Tri-State Driver

- The Tri-State driver is like a (one directional) switch:
 - * When the Enable is on ($E=1$) it transfers the input to the output.
 - * When the Enable is off ($E=0$) it disconnects the output.



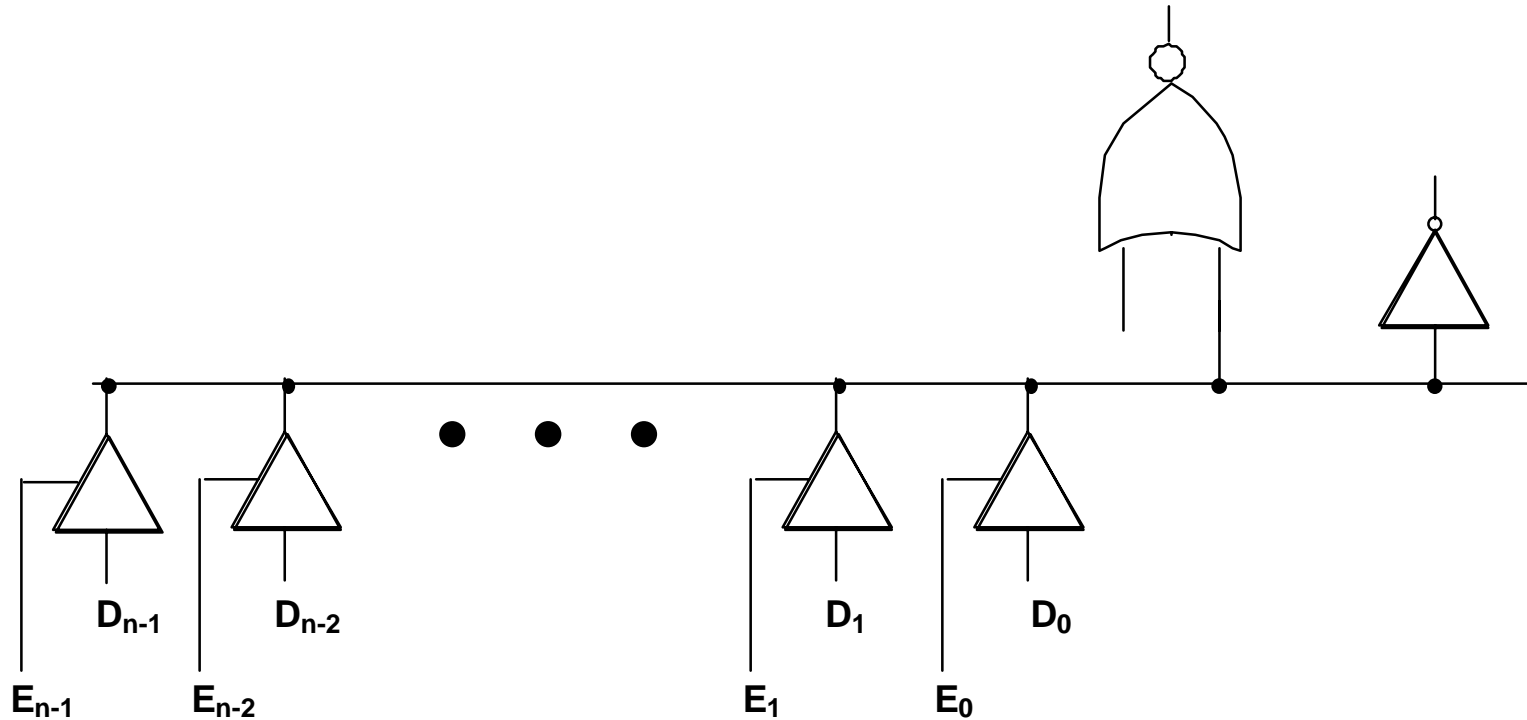
D	E	Q
0	1	0
1	1	1
-	0	Z

Z :- High Impedance

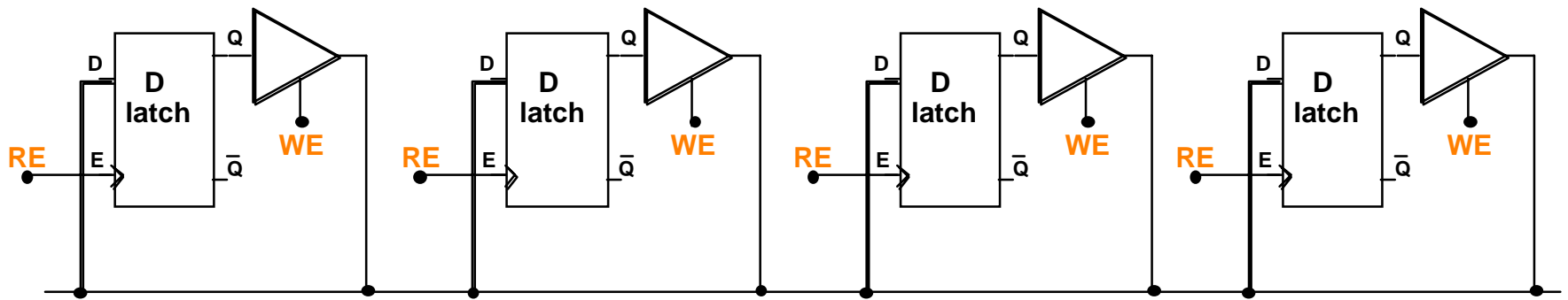


Bus Connections

- The Bus: Many to many connections.
- Mutual exclusion: **At most one Enable is on!**

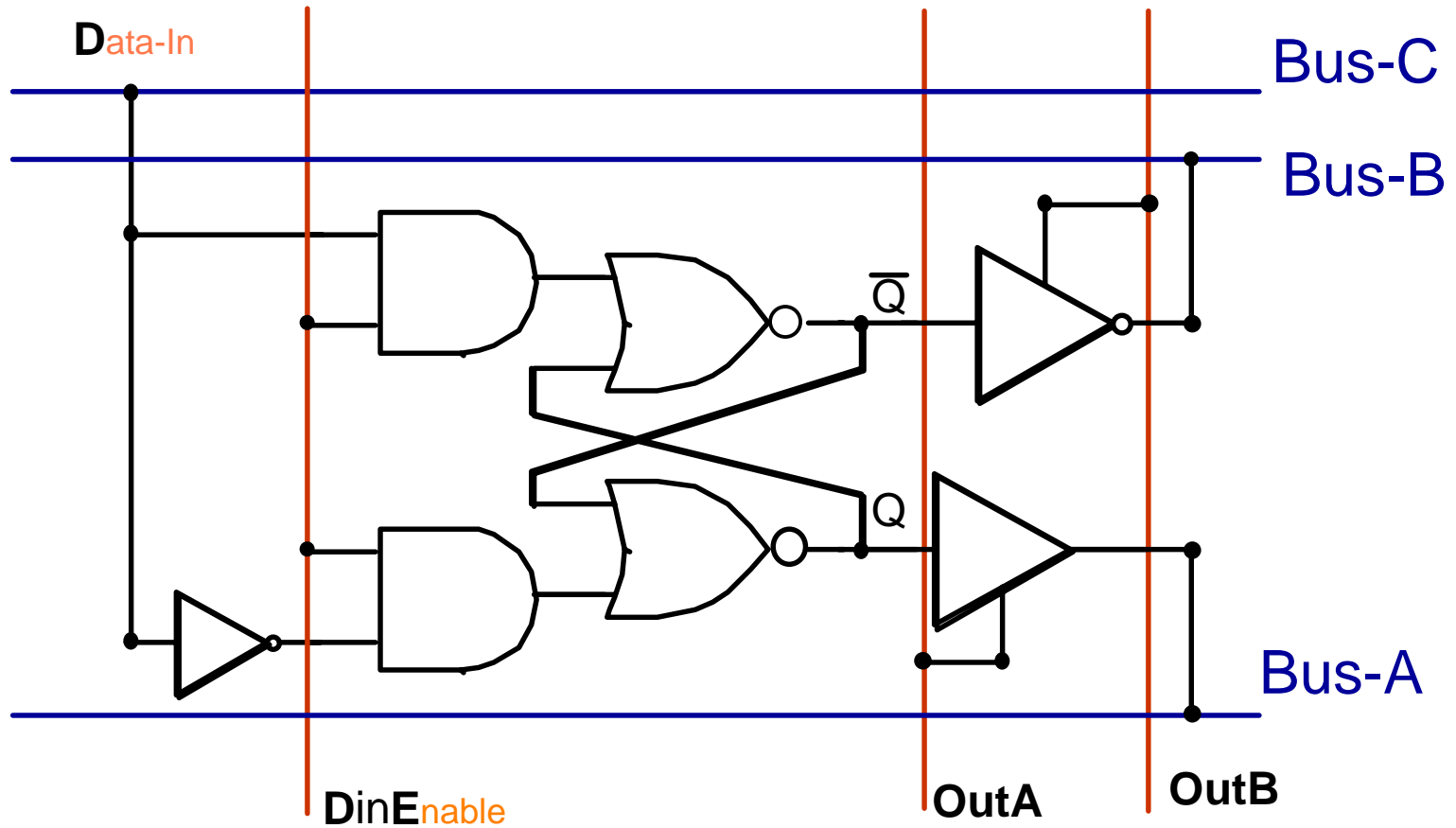


Register Cells on a bus

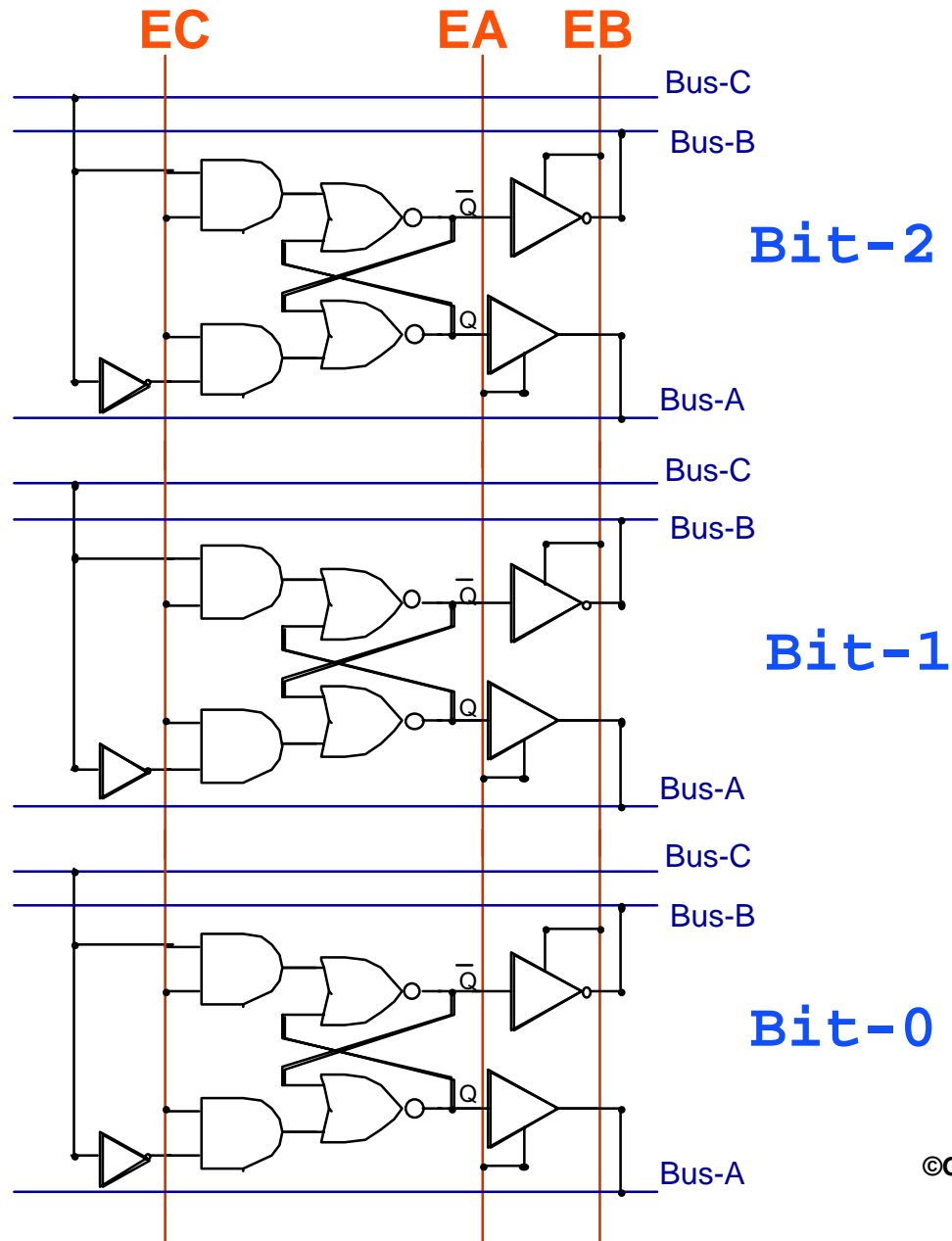


One can “source” and “sink” from any cell on the bus by activating the right controls (**WE** and **RE**).

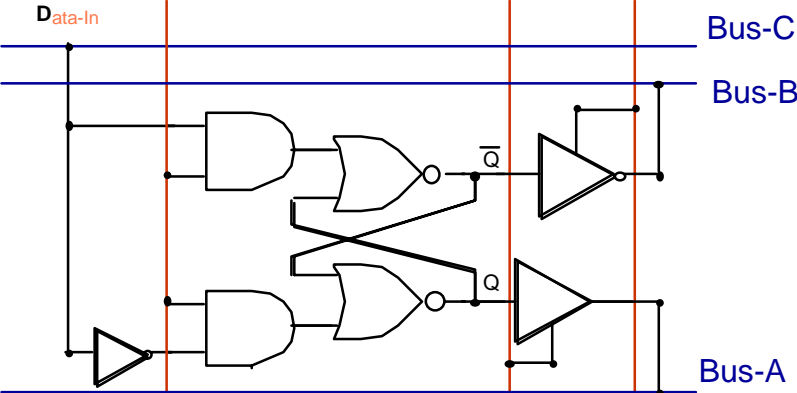
3-Port Register Cell



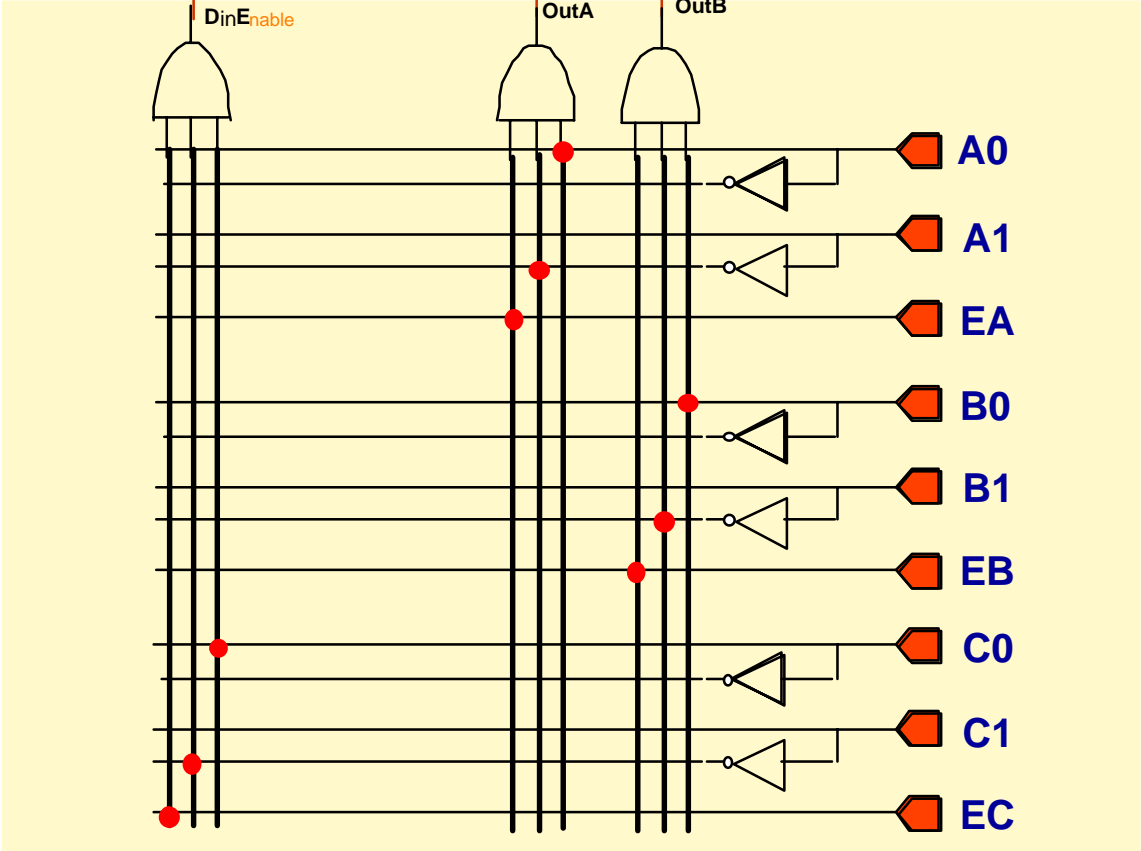
3-Port Register



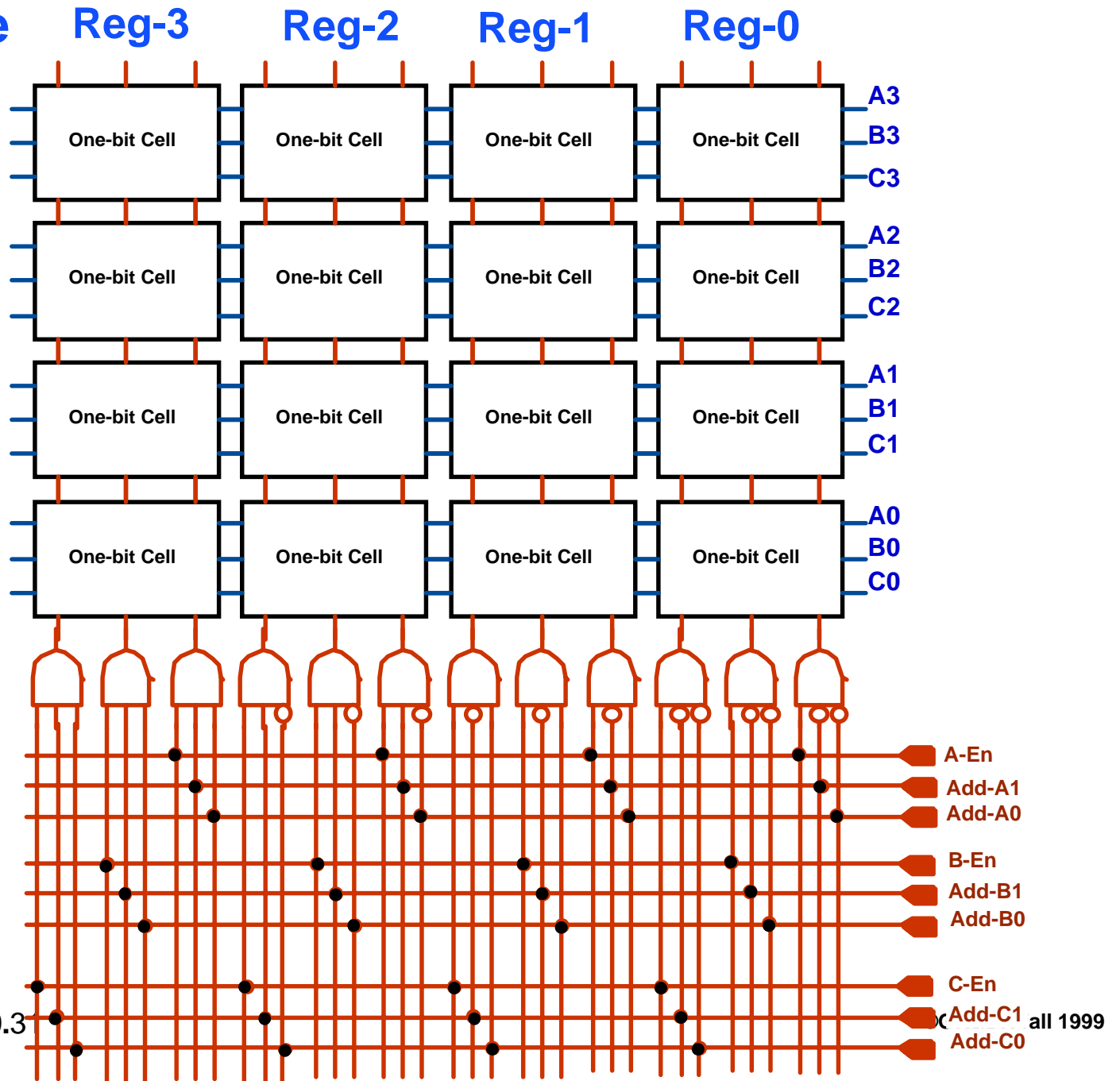
Address Decode circuit



Register address: 01



Register File



Summary

- So far we saw how to take a Boolean function and generate a circuit that “realize” the function.
- We learned to construct circuits that can **add** and **subtract**.
- We learned about the **ALU**: a circuit that can **add, subtract, detect overflow, compare**, and do **bit-wise operations (AND, OR, NOT)**
- Saw how to construct **a shifter** circuit.
- Learned about the memory elements: **RS-Latch, D-Latches** and **D-Flip-flops**.
- Learned about **Tri-State** drivers and **BUS** Communication. (many-to many)
- Learned about how to construct a **register file**.
- Saw how **control signals** can modify what the circuit will do with **inputs**.
 - * **Examples:** ALU, Shift, Register read-write, ...