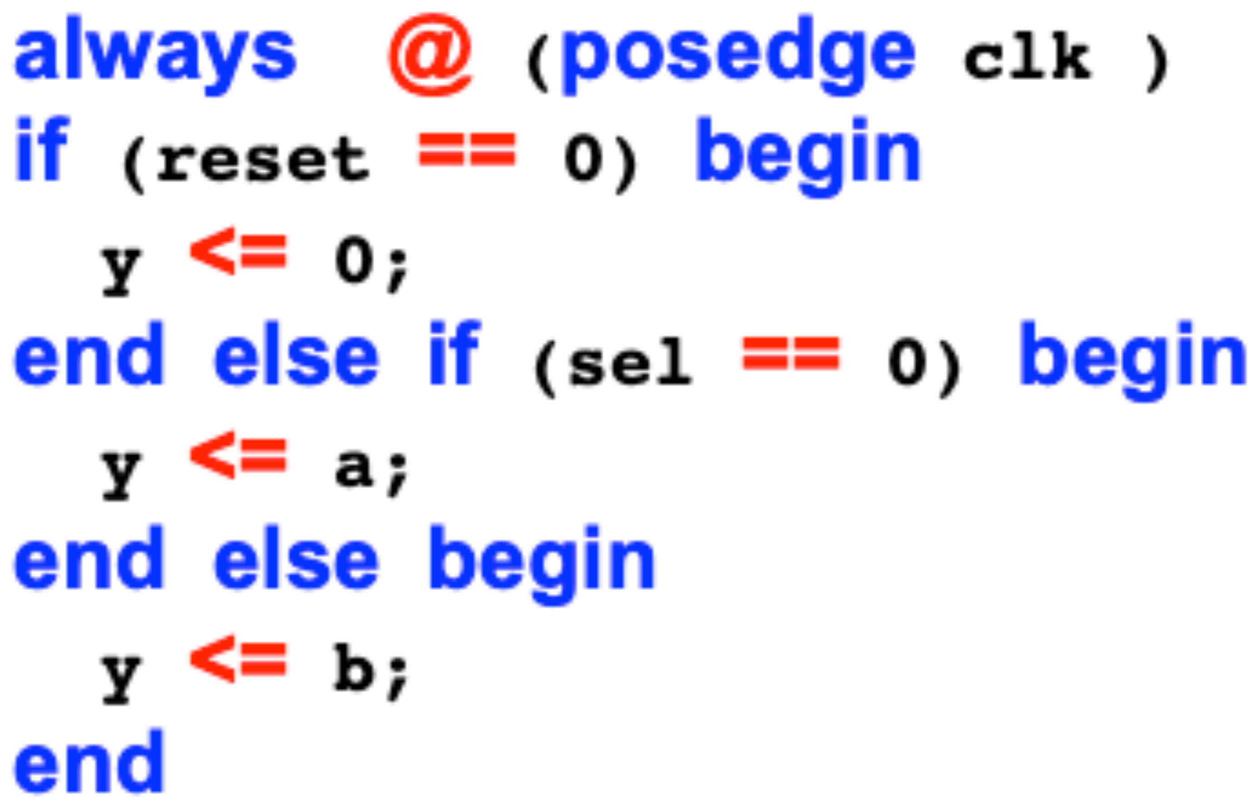
EECE 2322: Fundamentals of Digital Design and Computer Organization

Lecture 3_2: Verilog and FPGA

Xiaolin Xu
Department of ECE
Northeastern University

Blocking and Non-blocking



```
always  @ (posedge clk )
if (reset == 0) begin
  y <= 0;
end else if (sel == 0) begin
  y <= a;
end else begin
  y <= b;
end
```

- initial
- begin
- B <= A;
- C <= B;
- end
- All results evaluated first, then assigned
  - Result is that old contents of B are in C

How to Make a Device Reconfigurable to Any Logic?

In[1:0]	Out
00	?
01	?
10	?
11	?

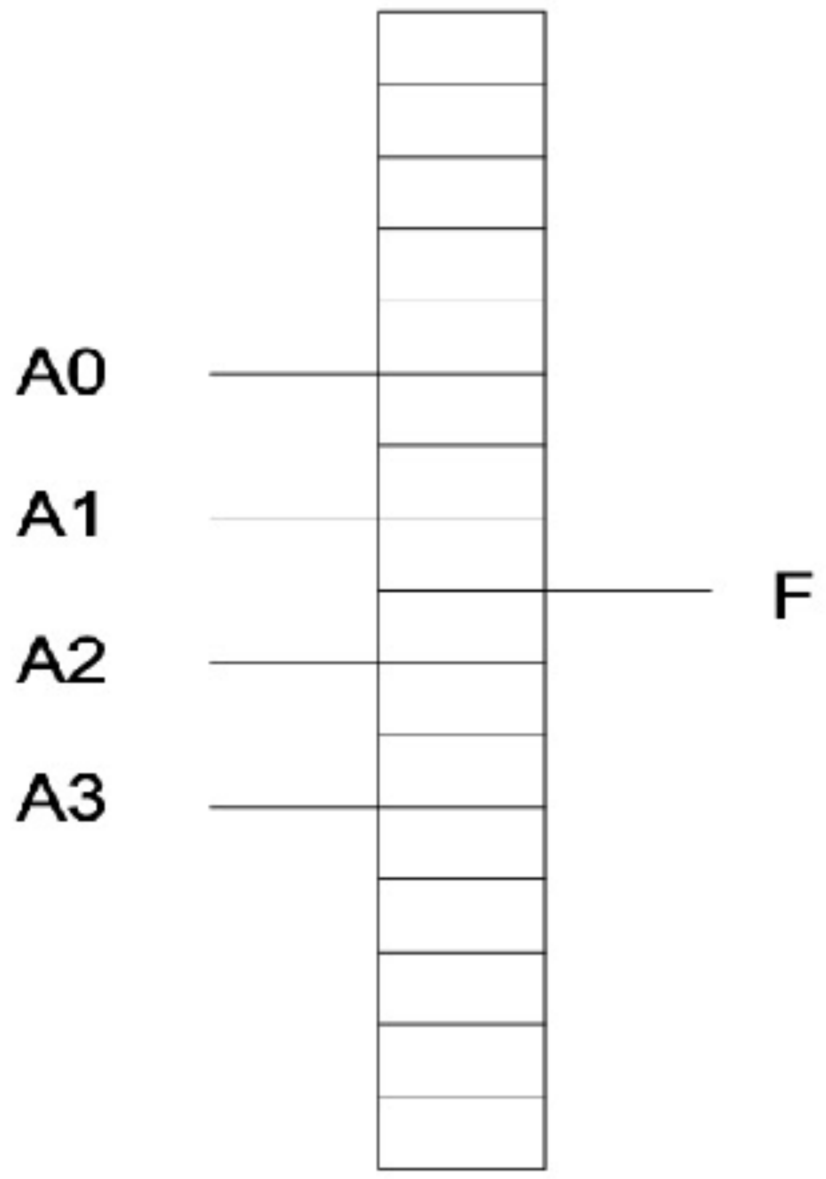
How to Make a Device Reconfigurable to Any Logic?

In[1:0]	Out
00	?
01	?
10	?
11	?

- **Reconfigure the memory!**
- **A memory cell can be written with a 1 or a 0**

Addressable Memory

- SRAM: Static Radom Accessible Memory
- A0-A3: address
- F: output
- How many memory cells?

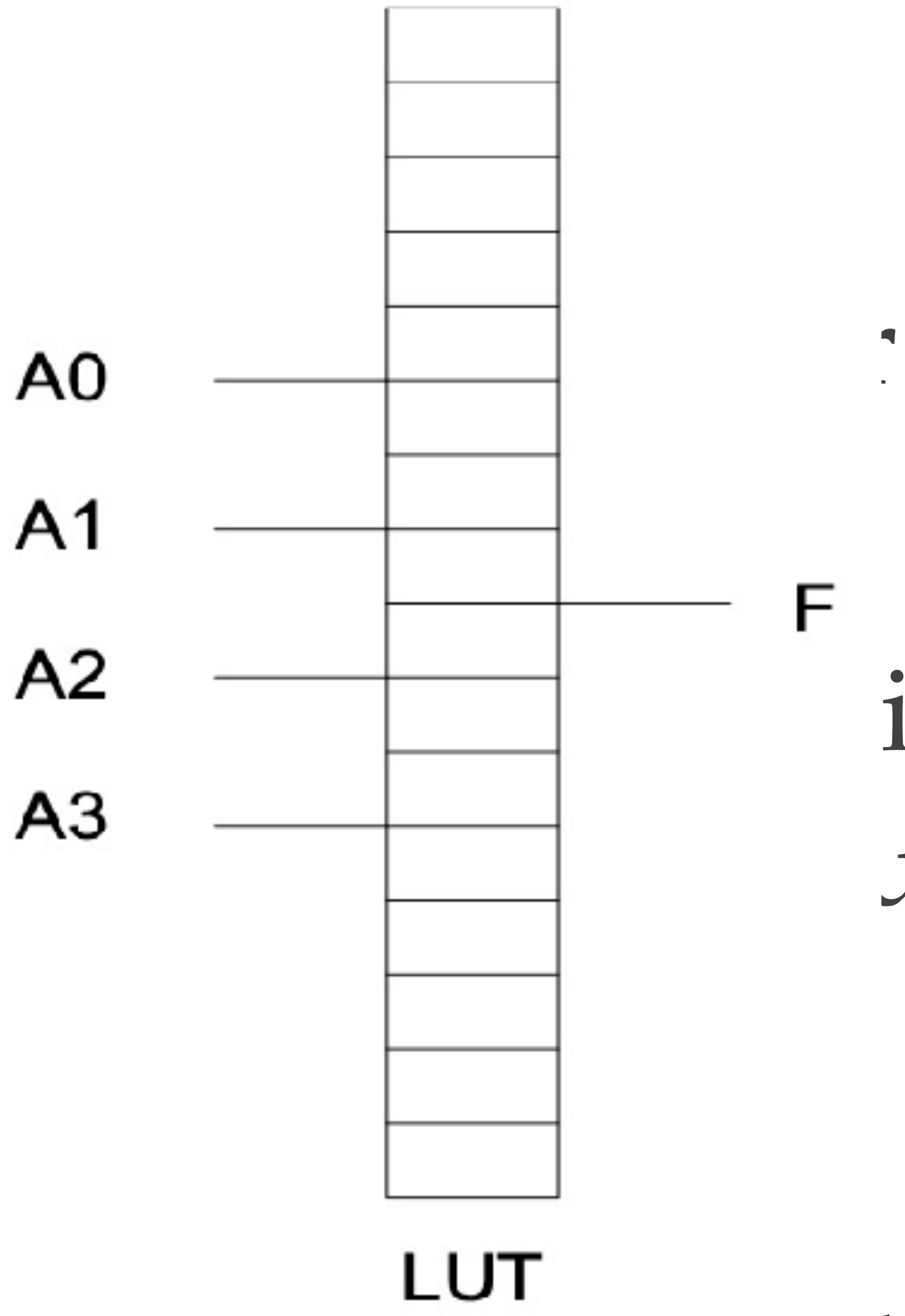


LUT: Look-Up Table

- Look-up tables are how your logic actually gets implemented. Users can program what the output should be for every single possible input
- A LUT consists of a block of RAM that is indexed by the LUT's inputs. The output of the LUT is whatever value is in the indexed location in its RAM cell

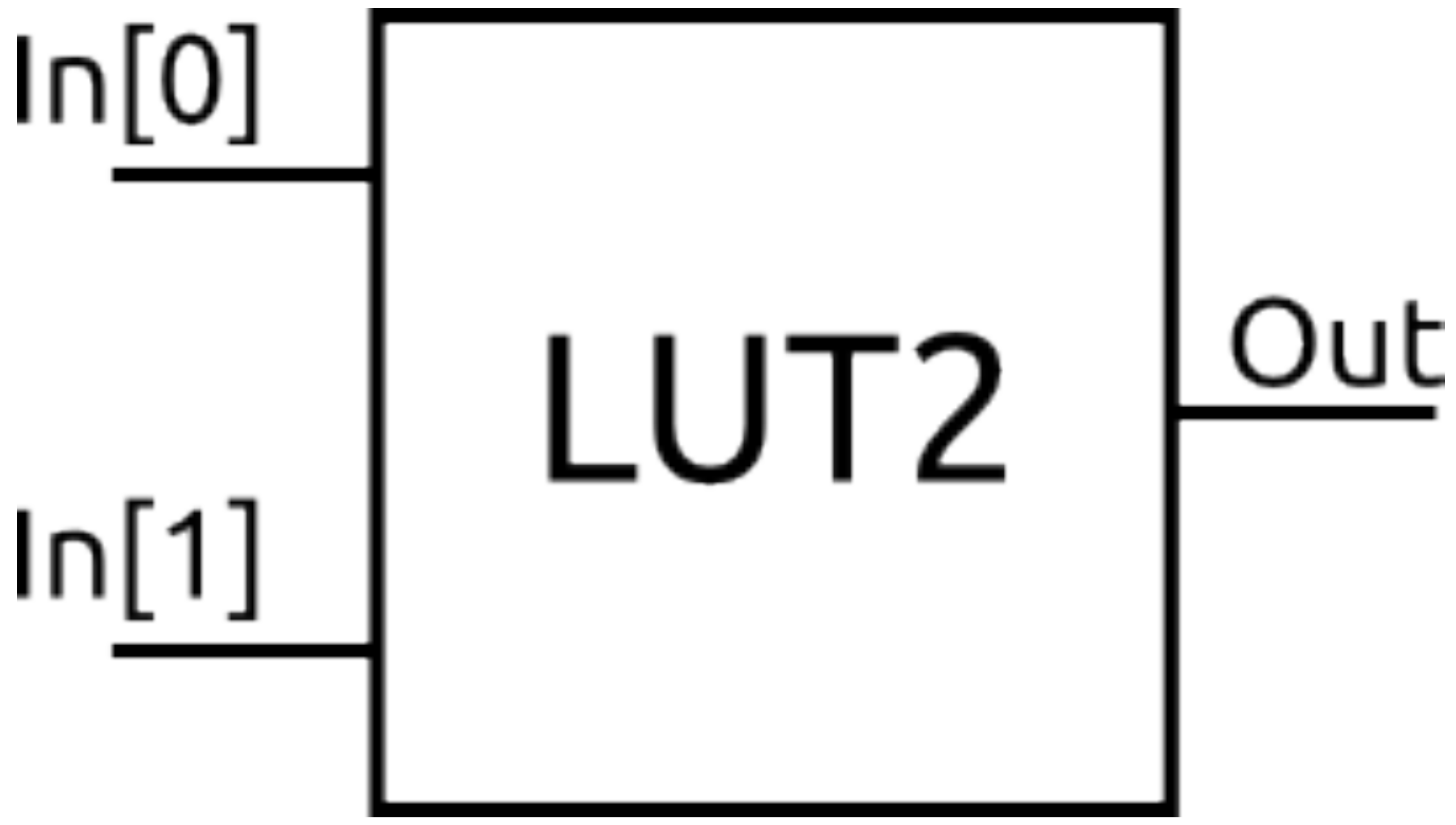
LUT: Look-Up Table

- Look-up tables are ho
actually gets impleme
program what the ou
every single possible
- A LUT consists of a bl is
indexed by the LUT's ıt
of the LUT is whateve
indexed location in its



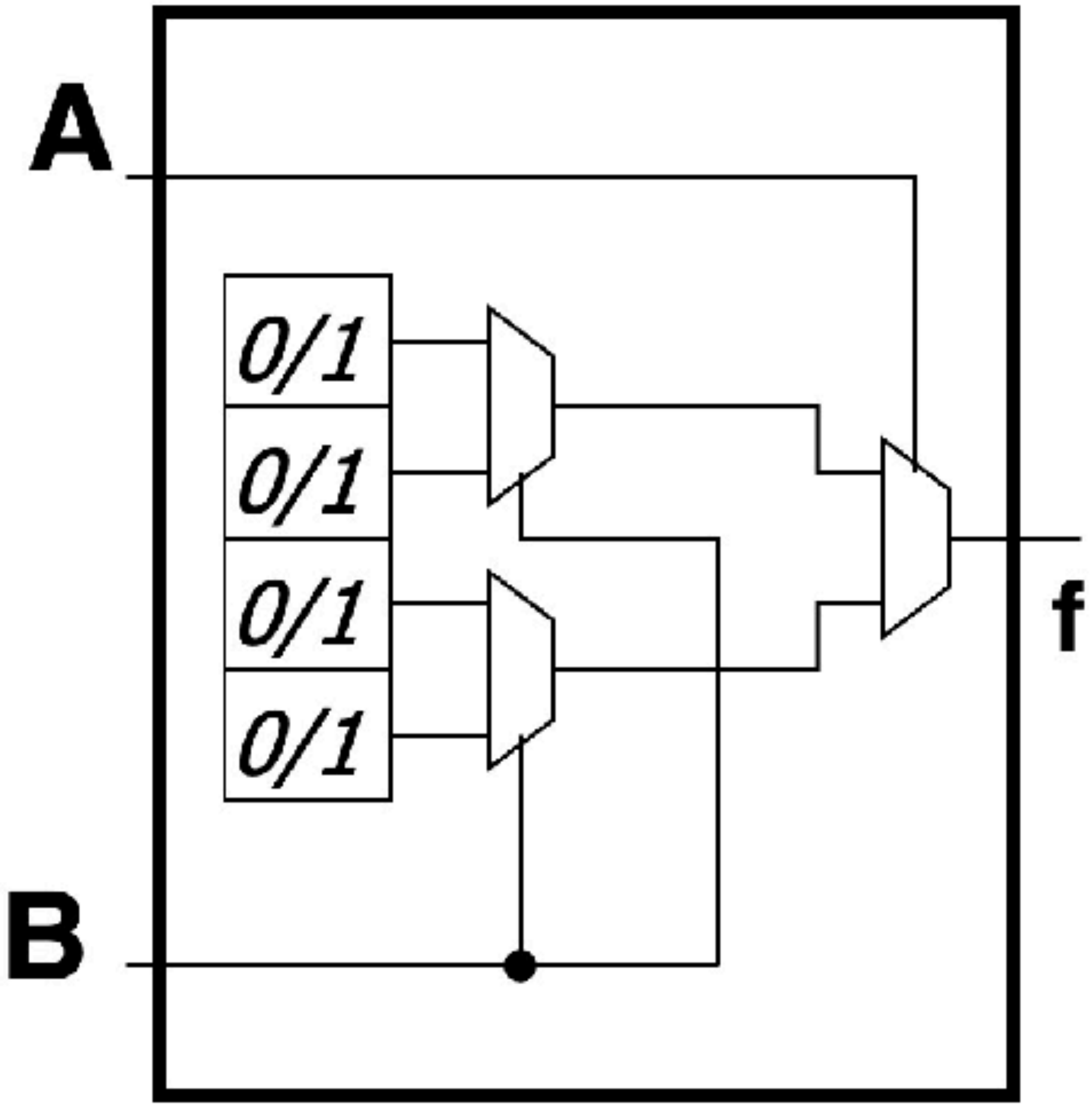
LUT: Look-Up Table

- A K input LUT requires 2k RAM cells to store function
- K Inputs are fed into LUT Mux to choose outputs
- An example 2-input LUT



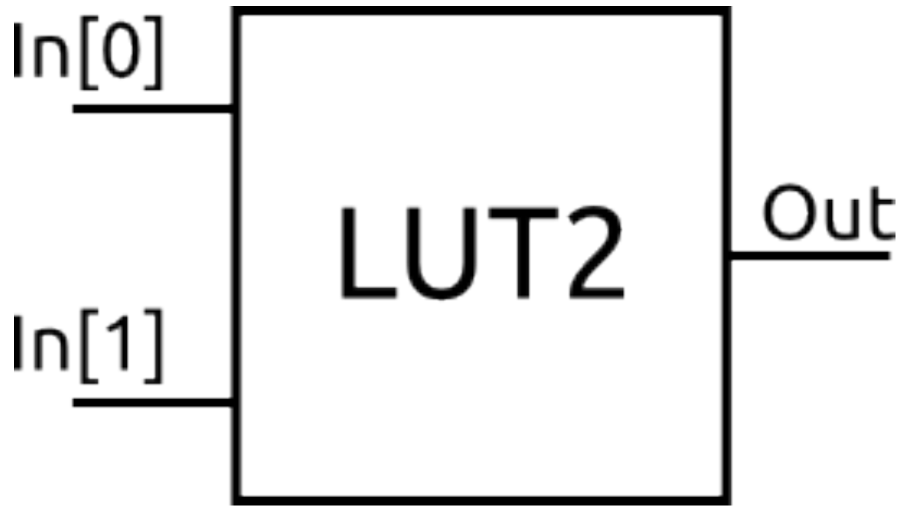
2-input LUT with MUXes

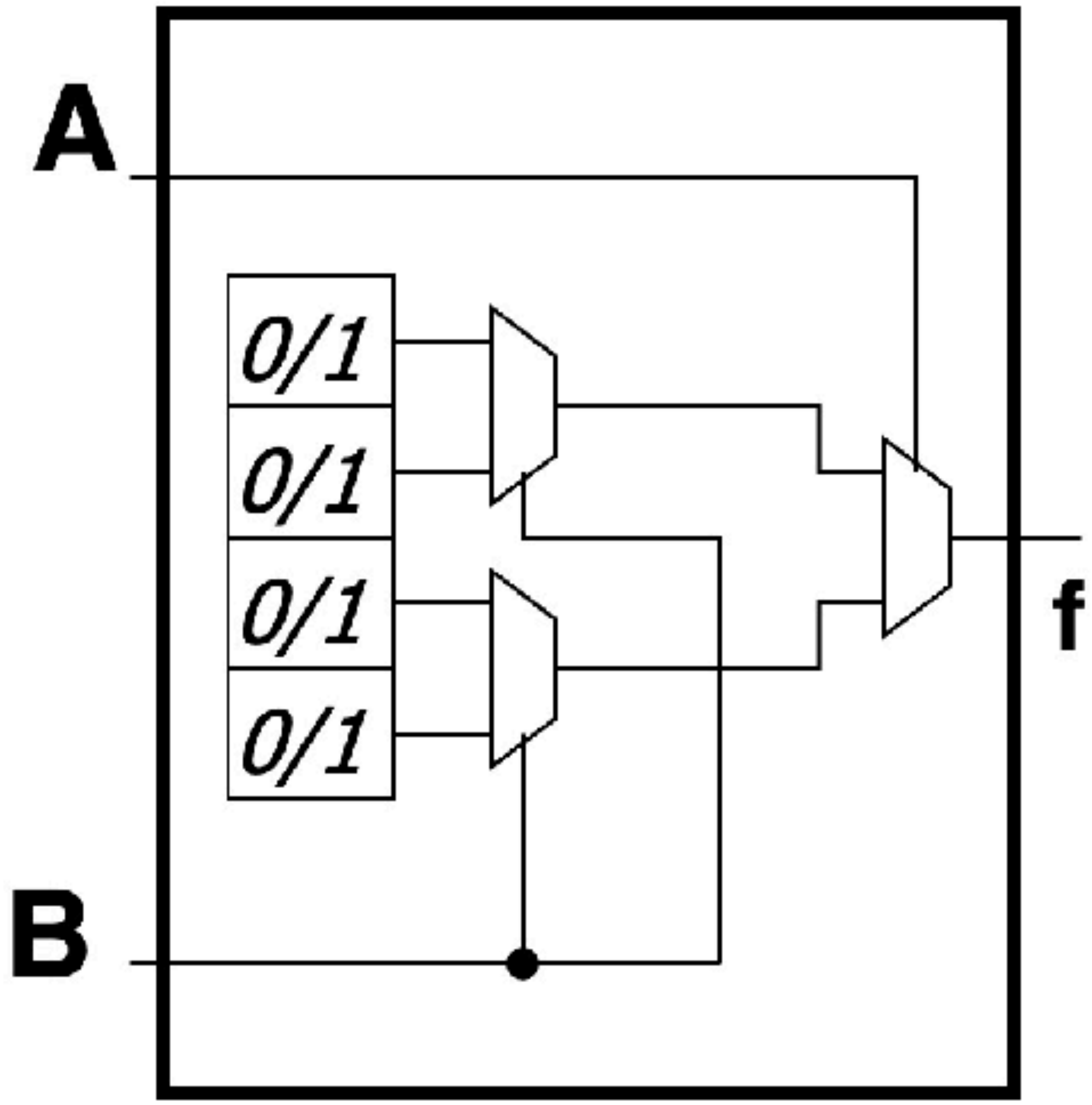
- Muxes are built as a tree of Muxes
-



LUT: Look-Up Table

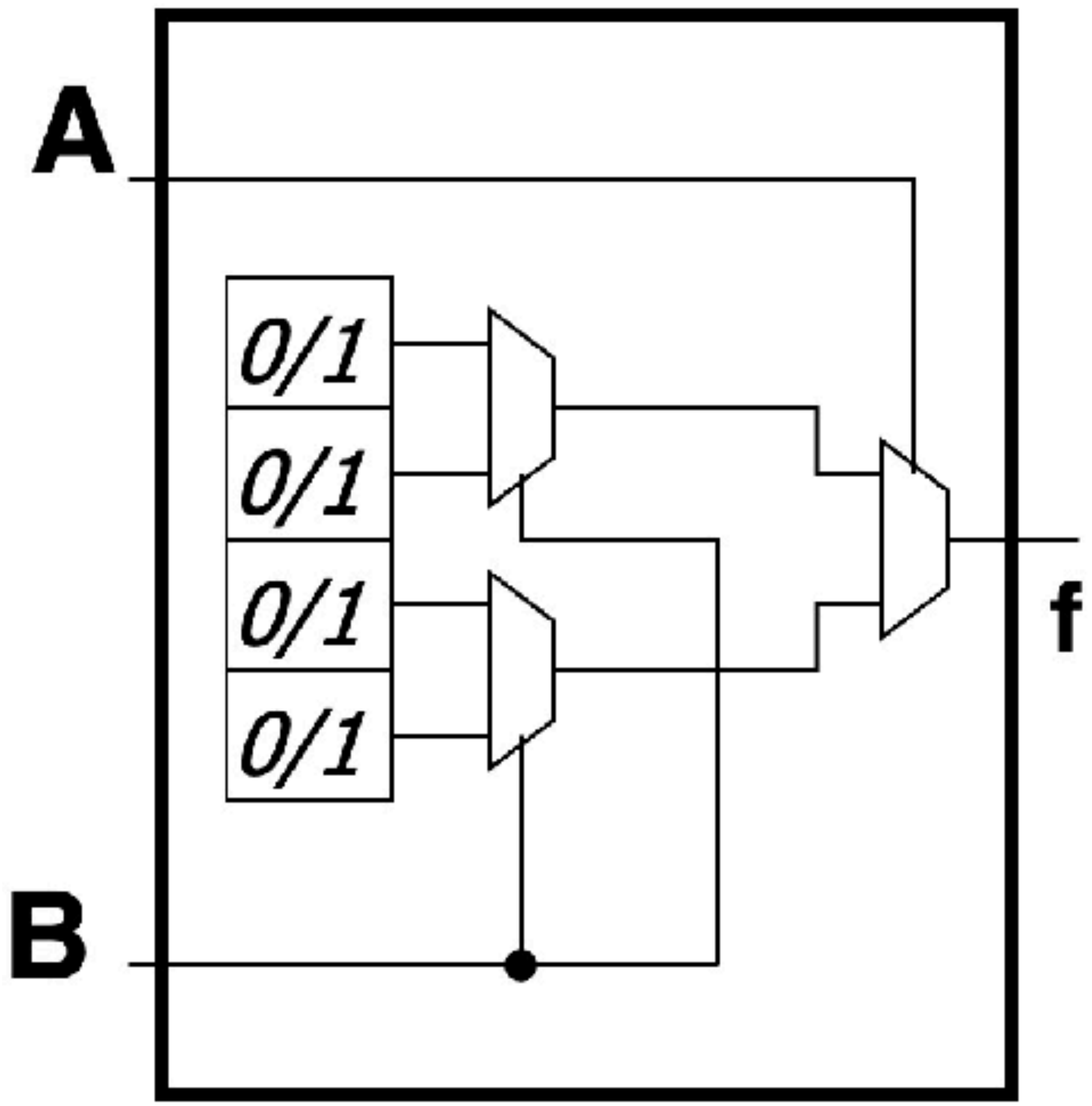
In[1:0]	Out
00	0
01	0
10	0
11	1





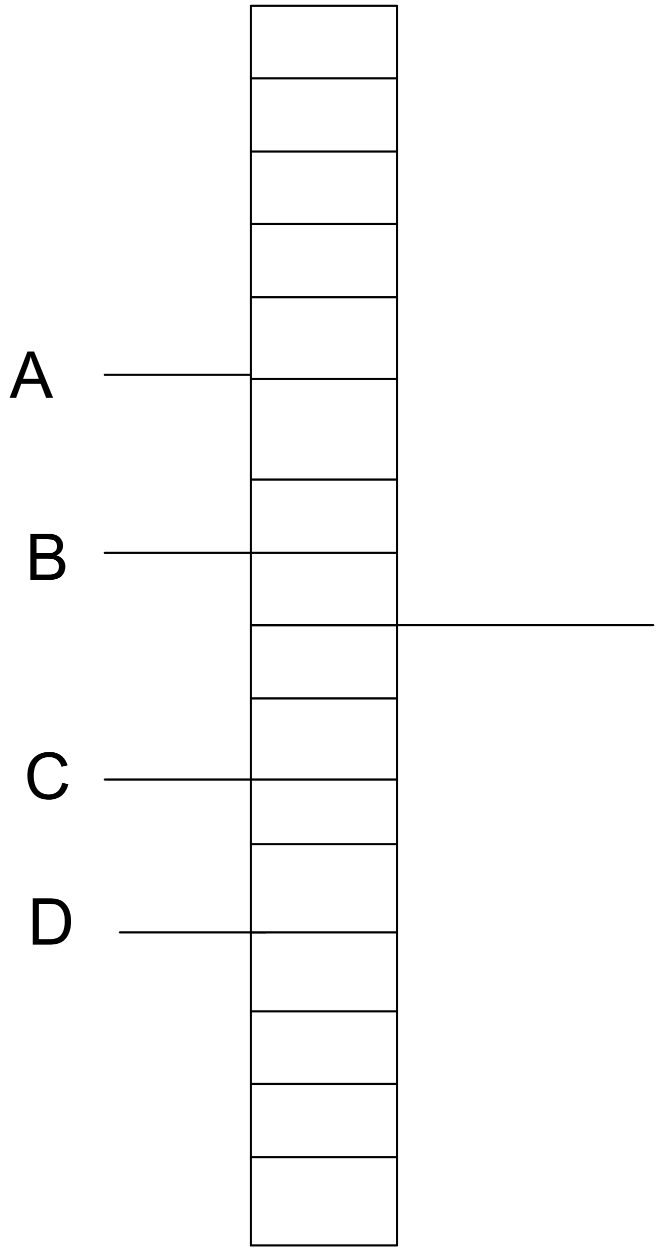
Implement an XOR gate with a 2-input LUT

- What are the values stored in the memory cells?



Practice: Mapping a Random Function to a 4-input LUT

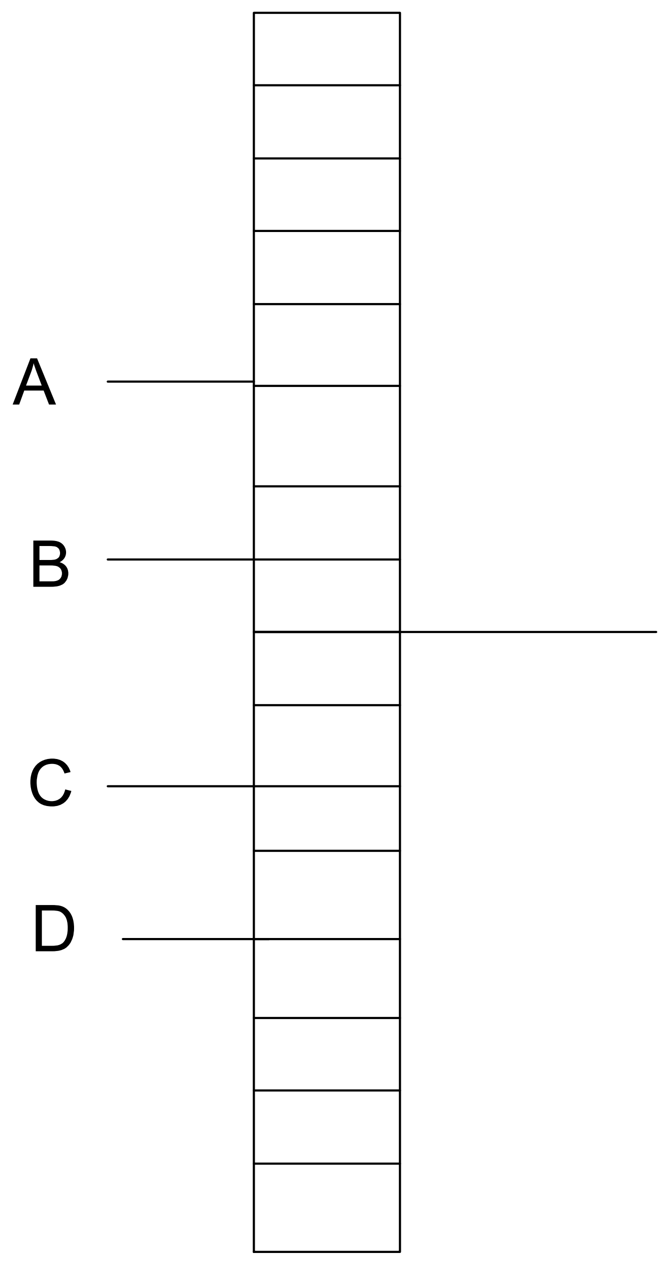
- Write down your answers



$F = A C' + A B D' + B C' D'$

A	B	C	D	F
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Practice: Mapping a Random Function to a 4-input LUT

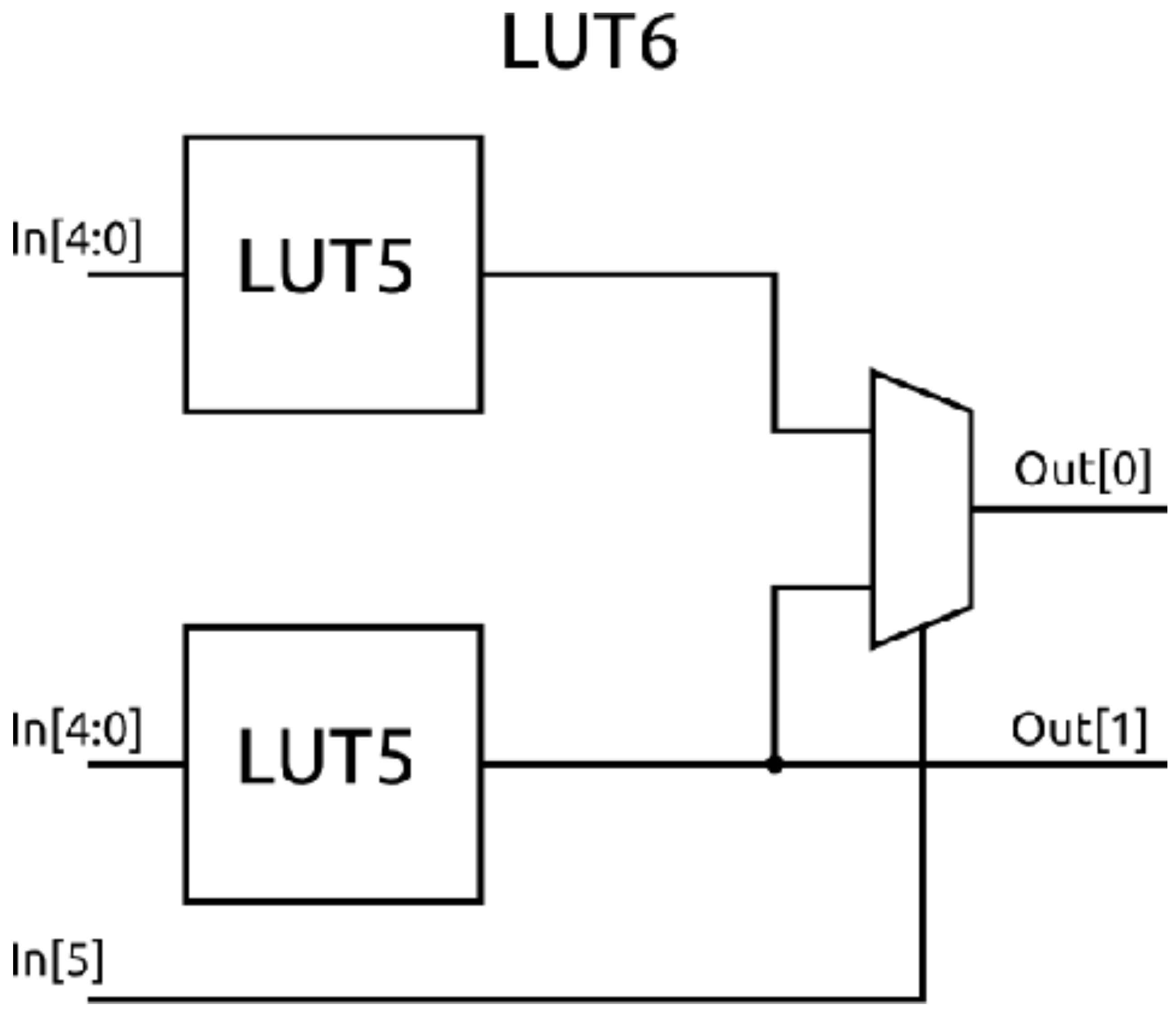


$$F = A\,C' + A\,B\,D' + B\,C'\,D'$$

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

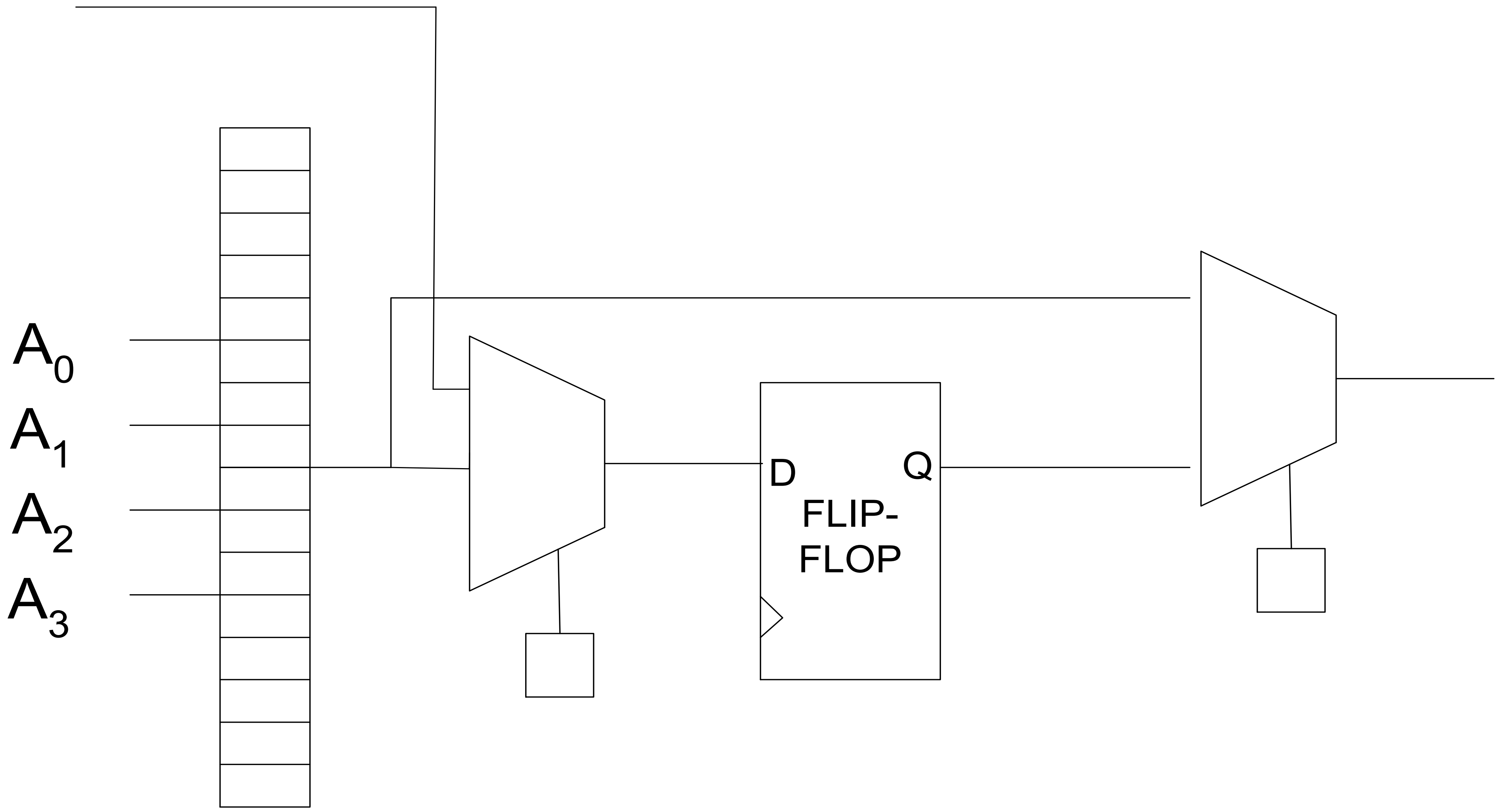
Building more Complicated Logic

- In the Xilinx Spartan 6 FPGA, each LUT is a 6-input LUT



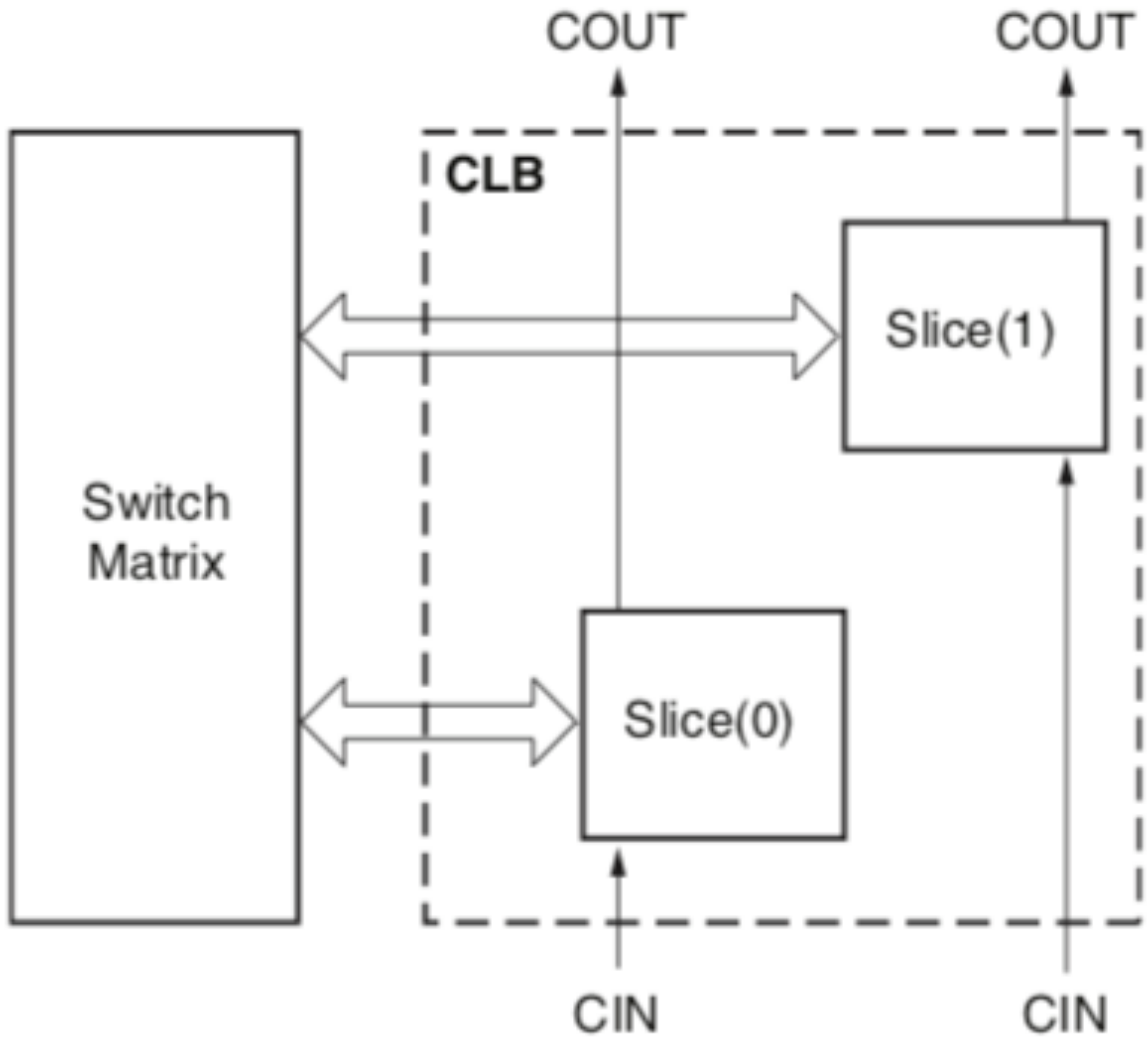
Logic Slice

- LUT + MUXes + Flip-flop

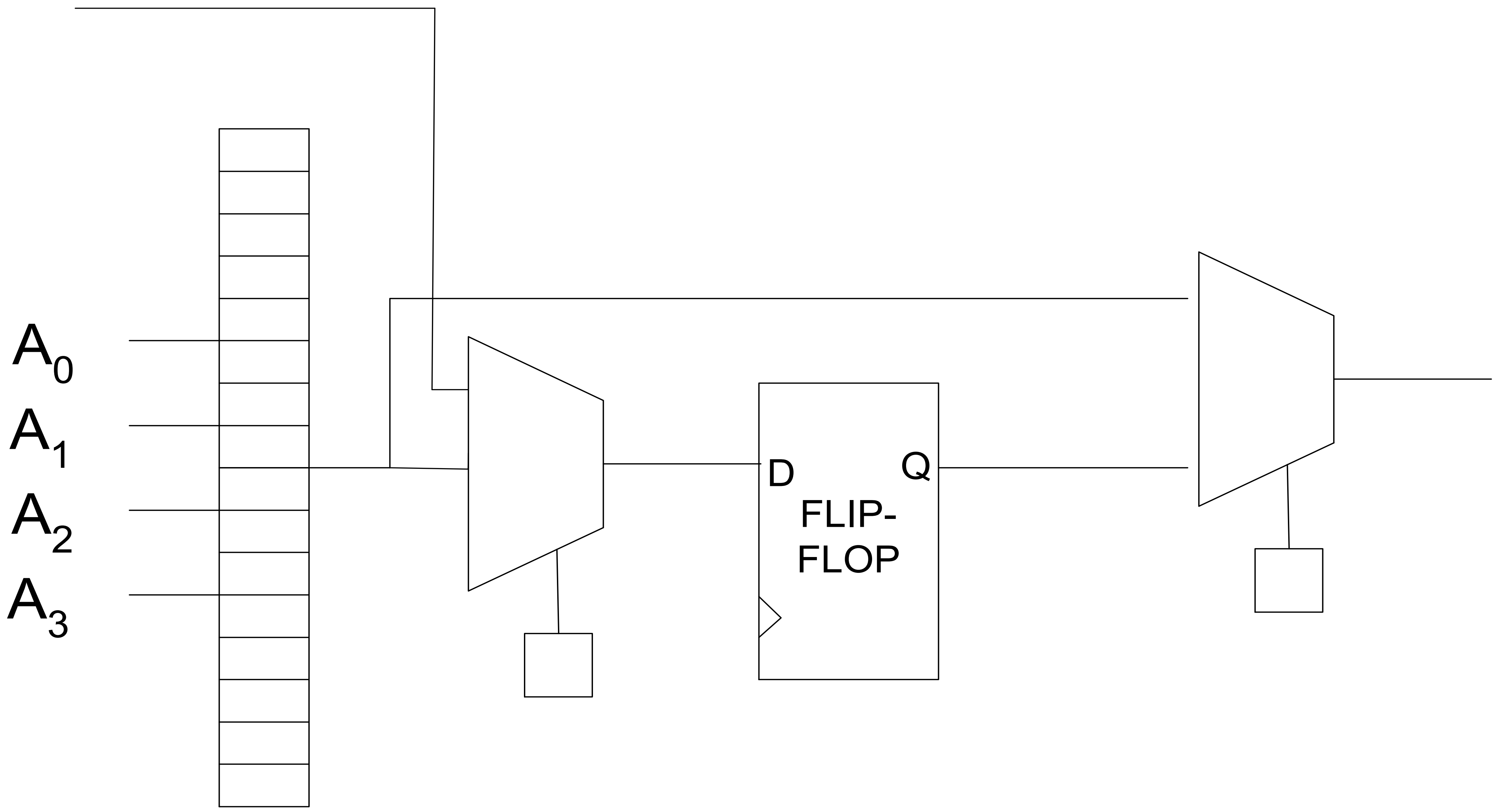


CLB: Configurable Logic Block

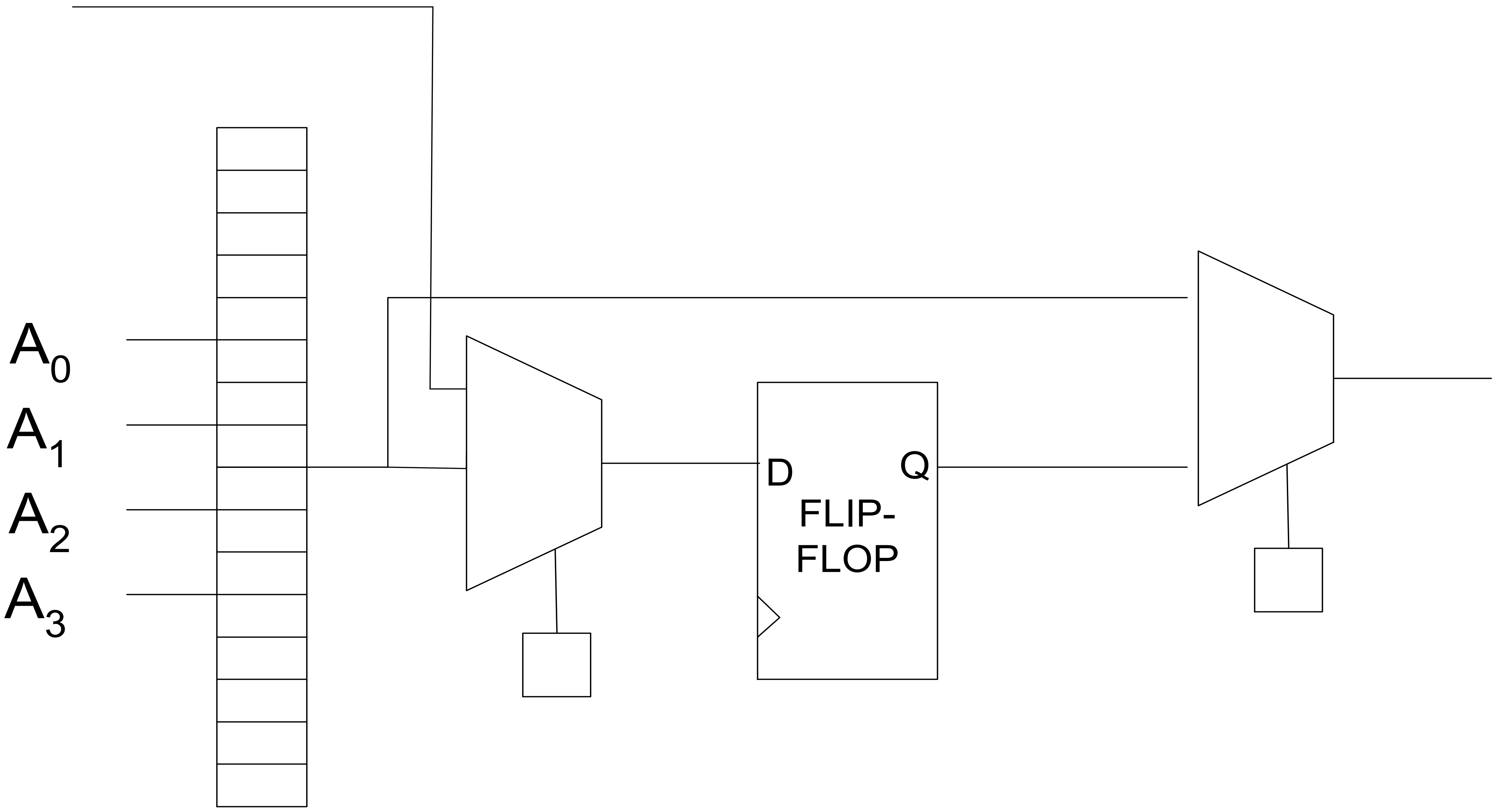
- Each CLB has two Slices



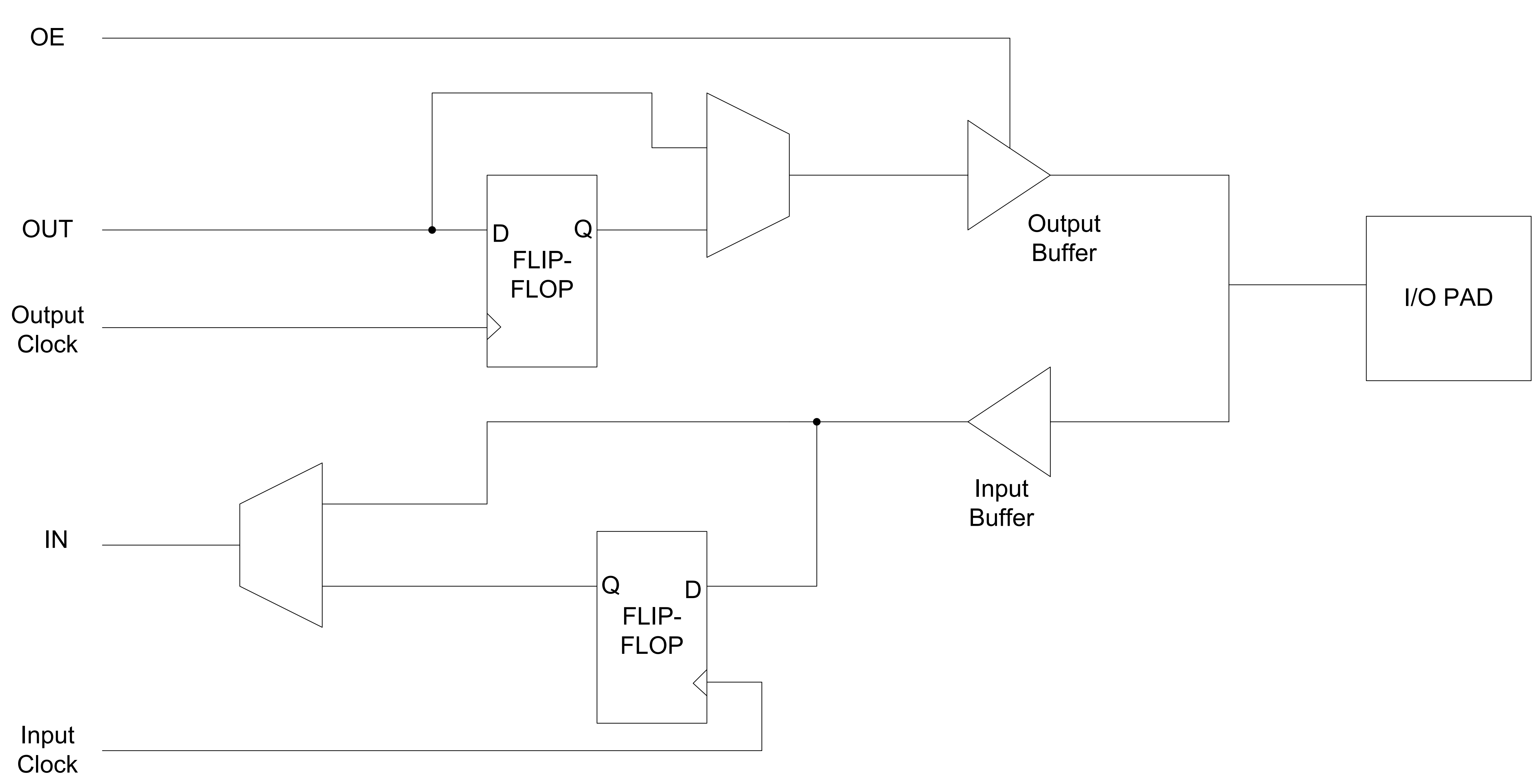
Combinational Logic Design



Sequential Logic Design

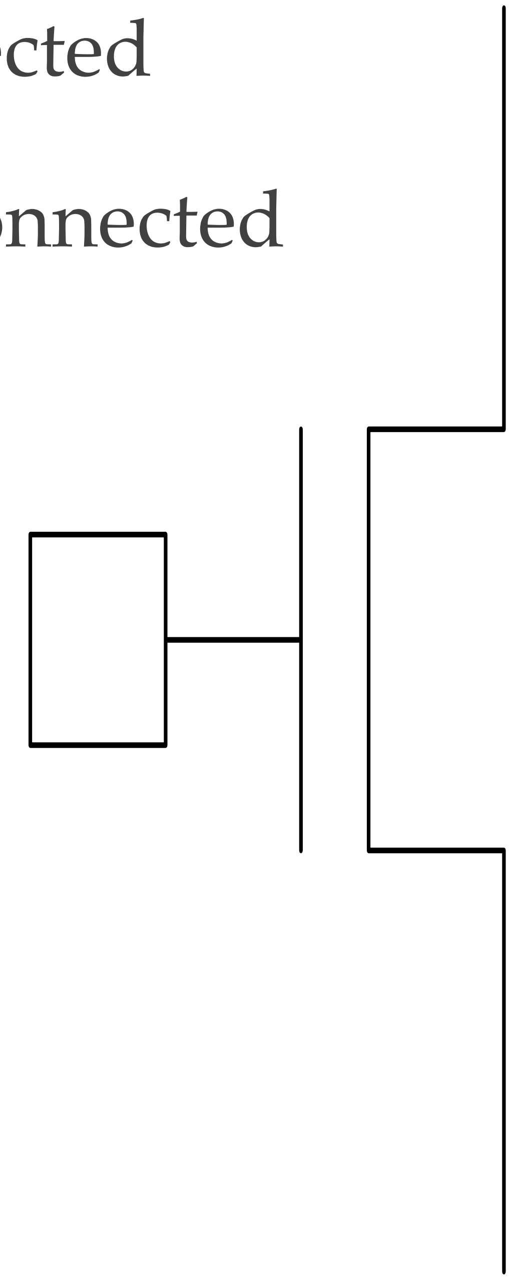


I/O Block Xilinx 4000 Family -- Simplified



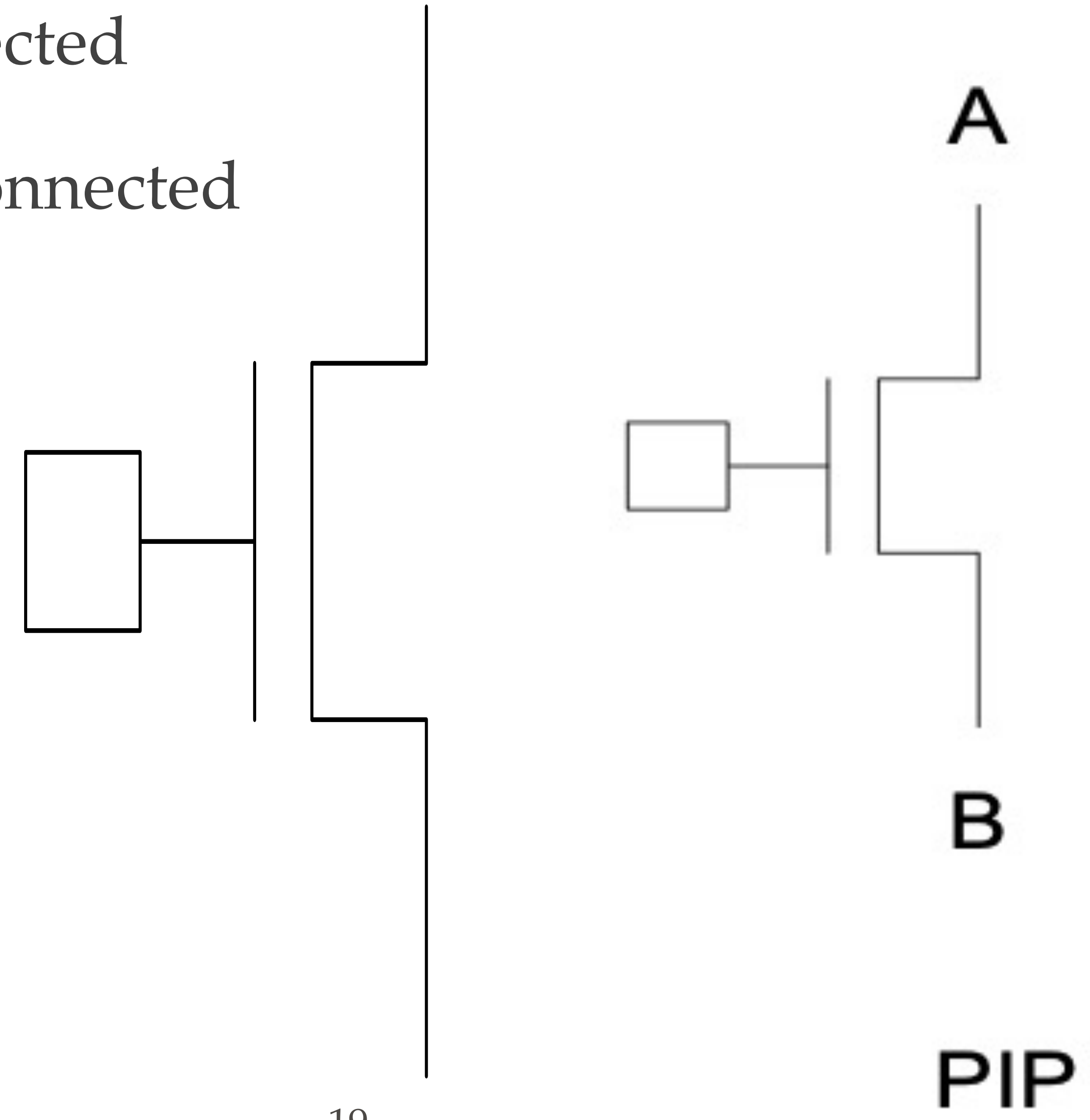
Programmable Interconnect Point (PIP)

- The basic unit of programmable interconnect
  - 1 stands for connected
  - 0 stands for dis-connected



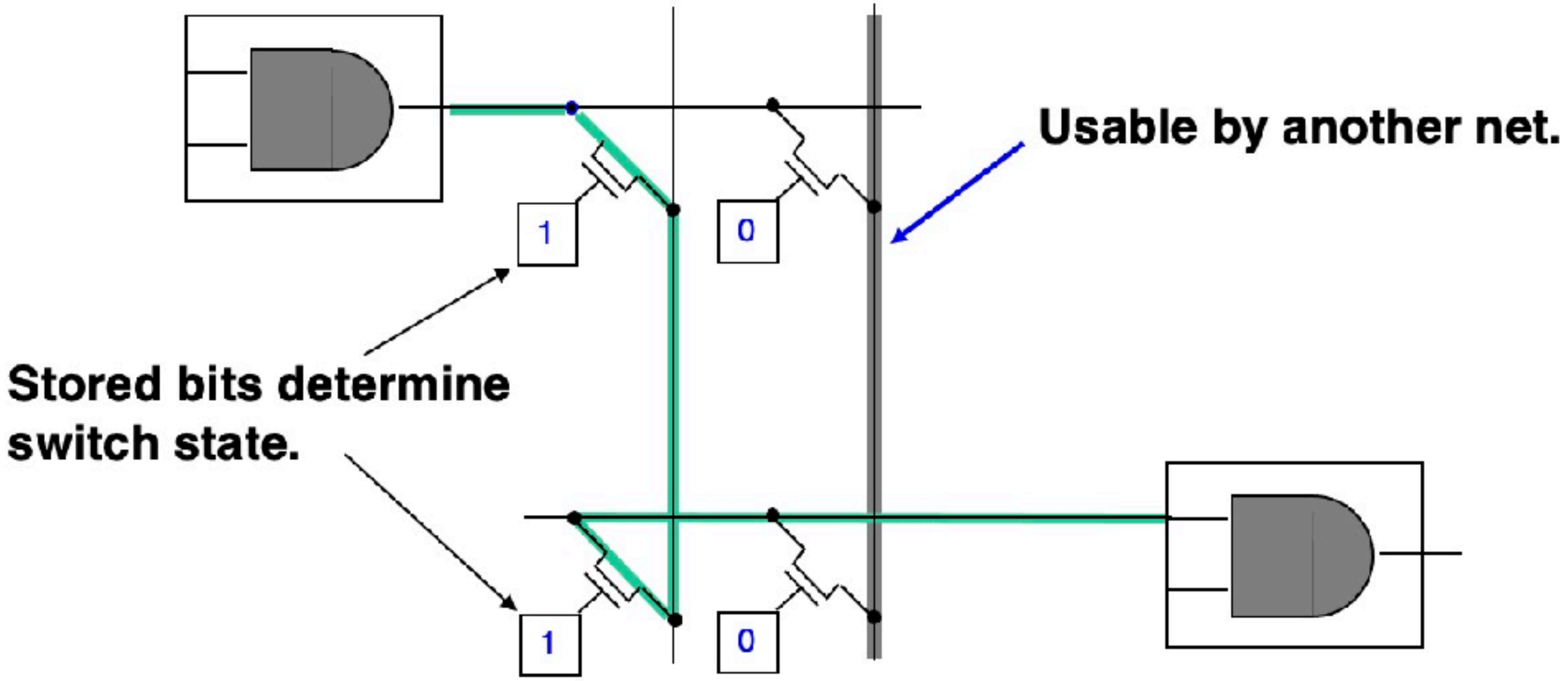
Programmable Interconnect Point (PIP)

- The basic unit of programmable interconnect
  - 1 stands for connected
  - 0 stands for dis-connected

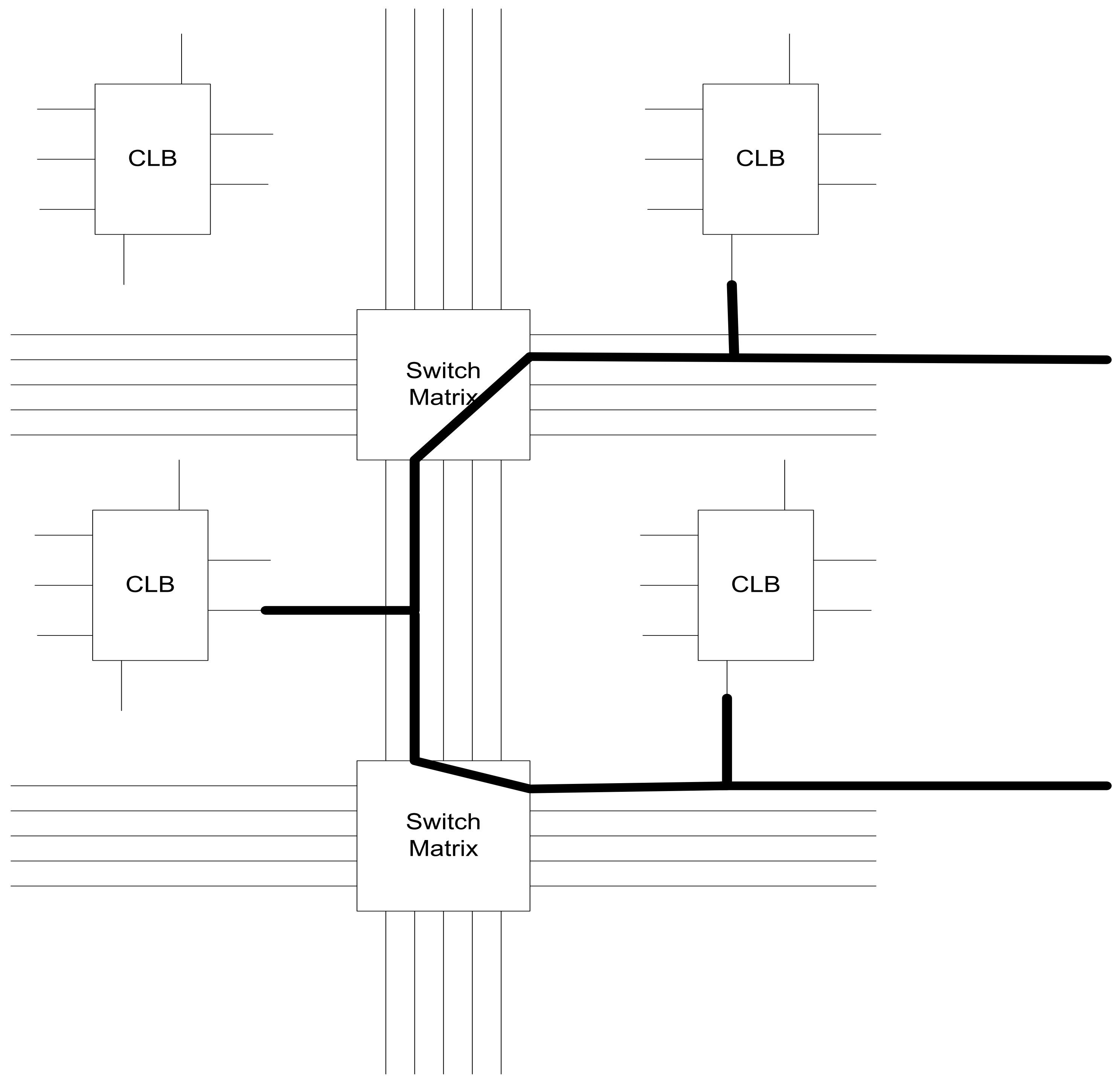


Xilinx FPGA Interconnect Example

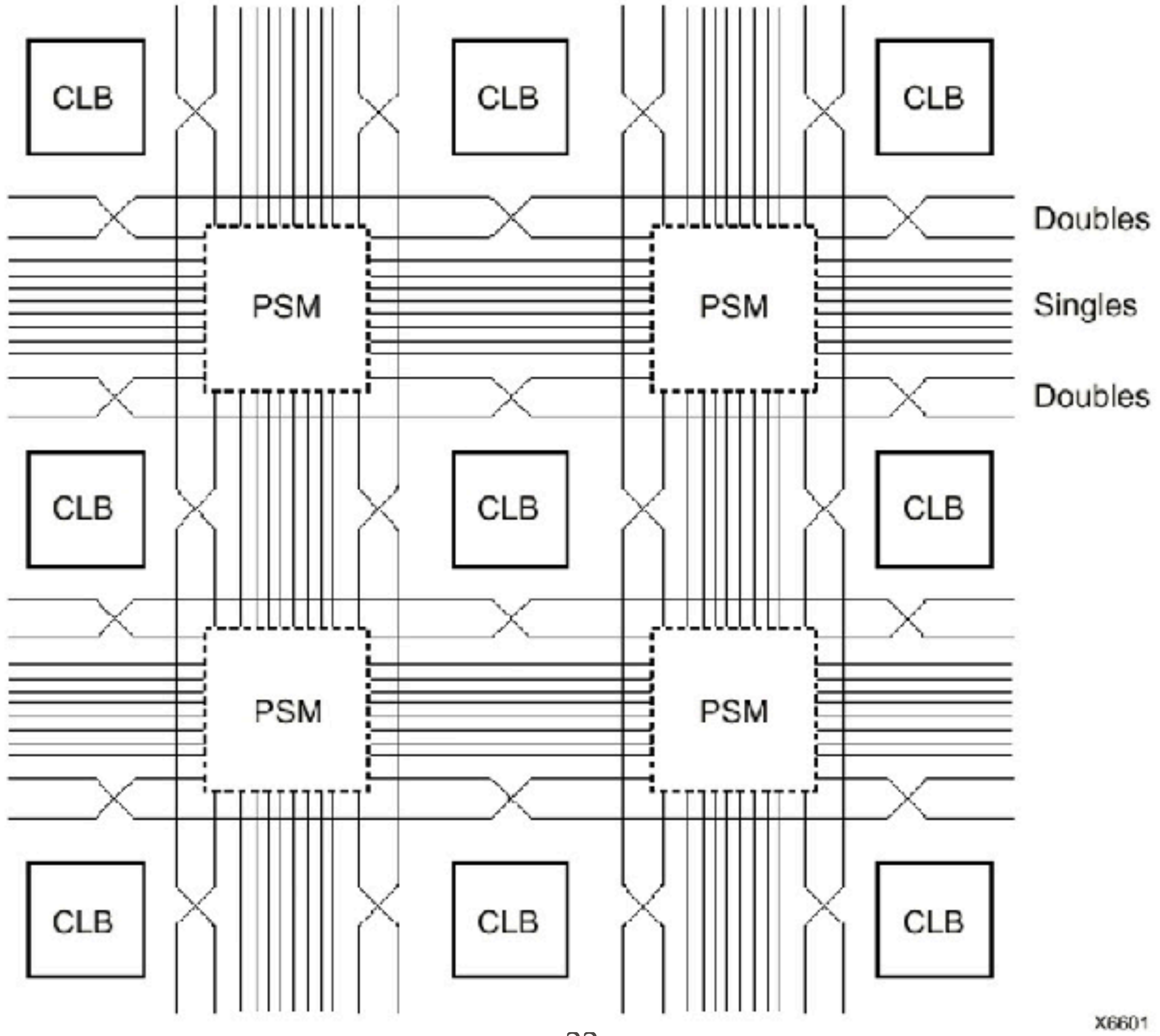
- Programmable Interconnect
  - Pass Transistors as switches



Interconnect:
Switch Matrixes and Programmable Interface Points

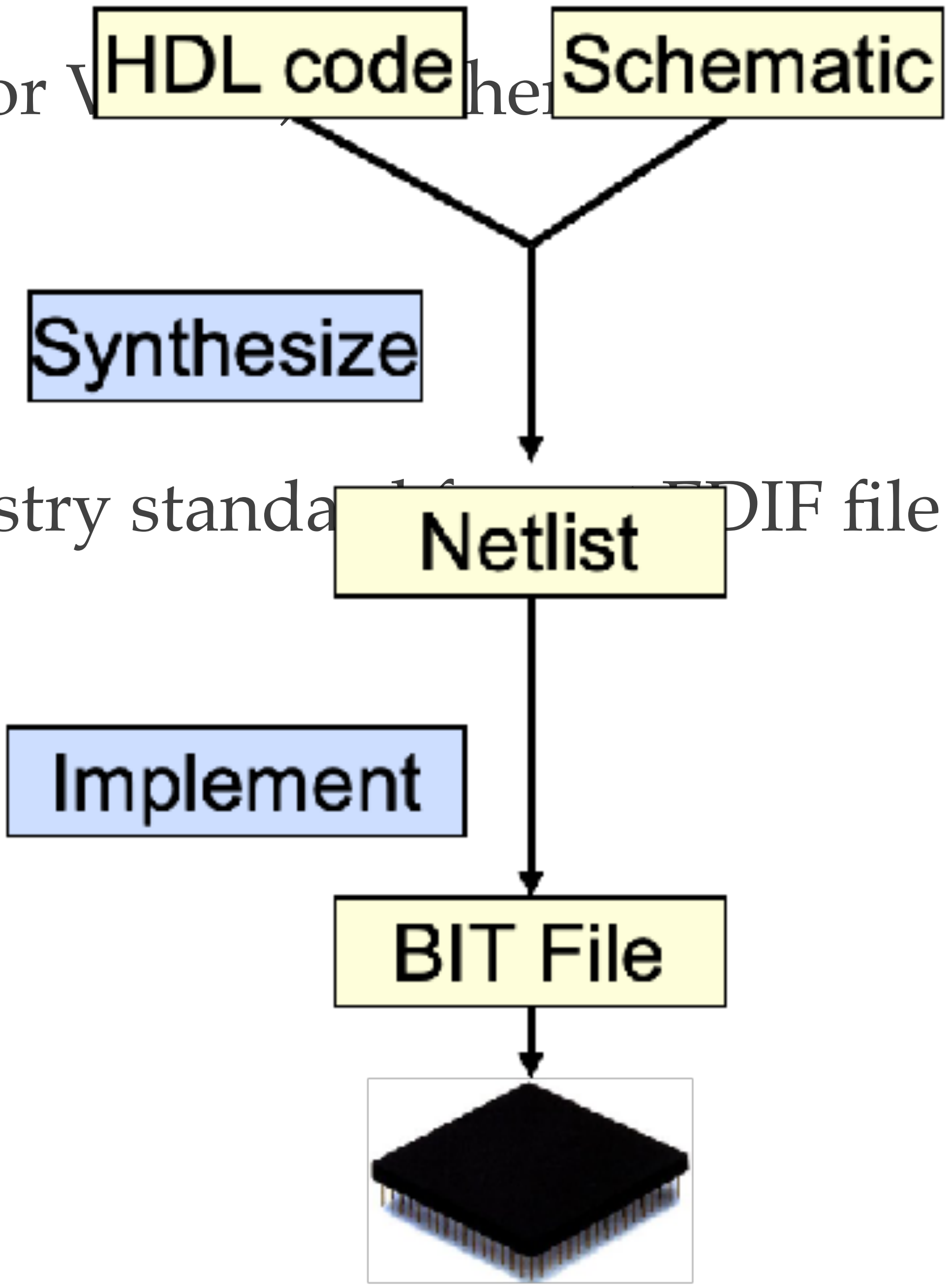


Xilinx 4000 Interconnect



Xilinx FPGA Design Flow

- Step1: Design
  - Two design entry methods: HDL(Verilog or V he drawings
- Step 2: Synthesize to create Netlist
  - Translates V, VHD, SCH files into an industry standa DIF file
- Step 3: Implement design (netlist)
  - Translate, Map, Place & Route
- Step 4: Configure FPGA
  - Download BIT file into FPGA

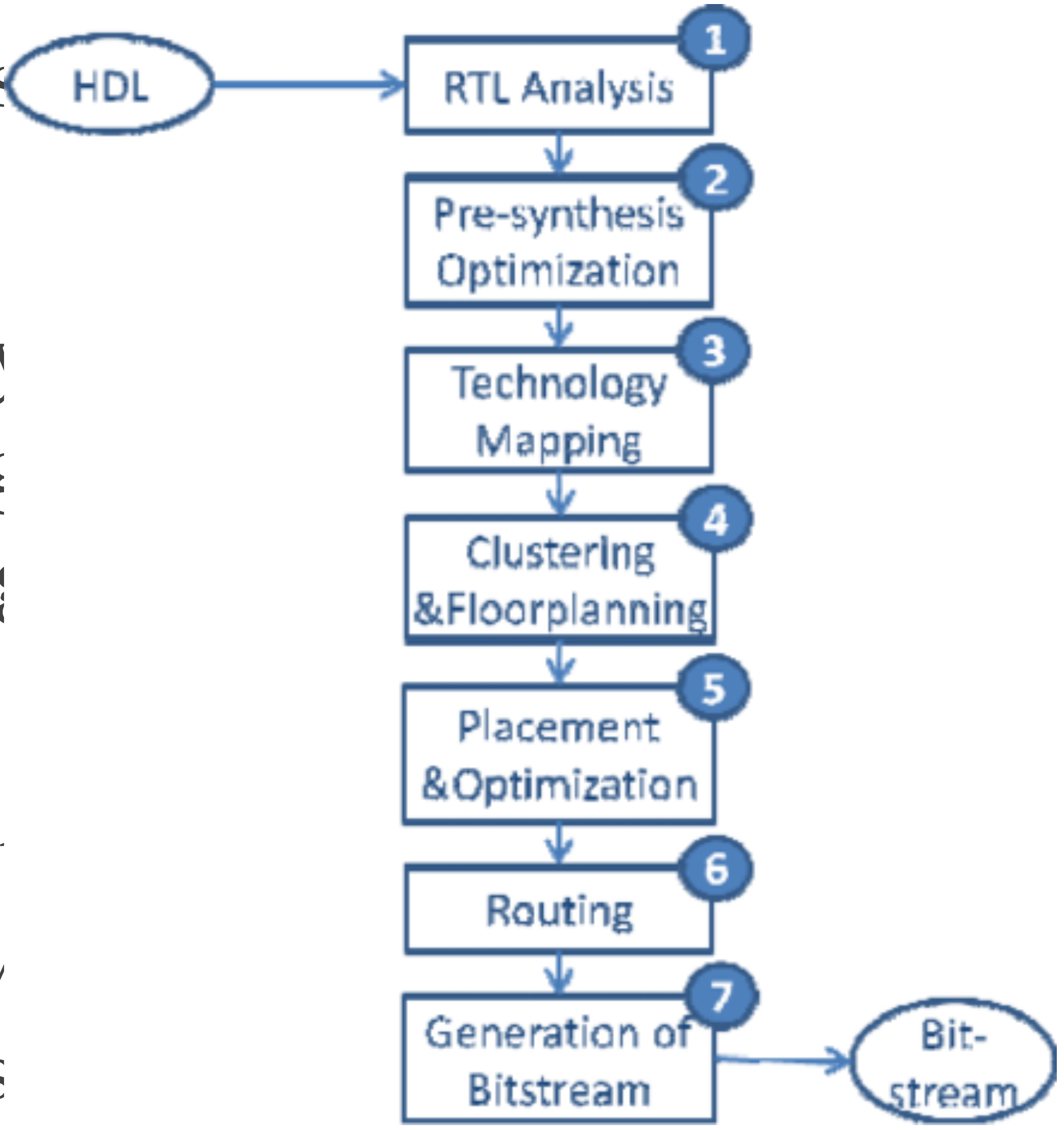


Important Step: Synthesis

- Enabled by sophisticated computer-aided design (CAD) tools
- A full set of programs and tools that allow automatic synthesis and compilation from a high level hardware description language, such as Verilog or VHDL to a string of bits, commonly termed a **bitstream**

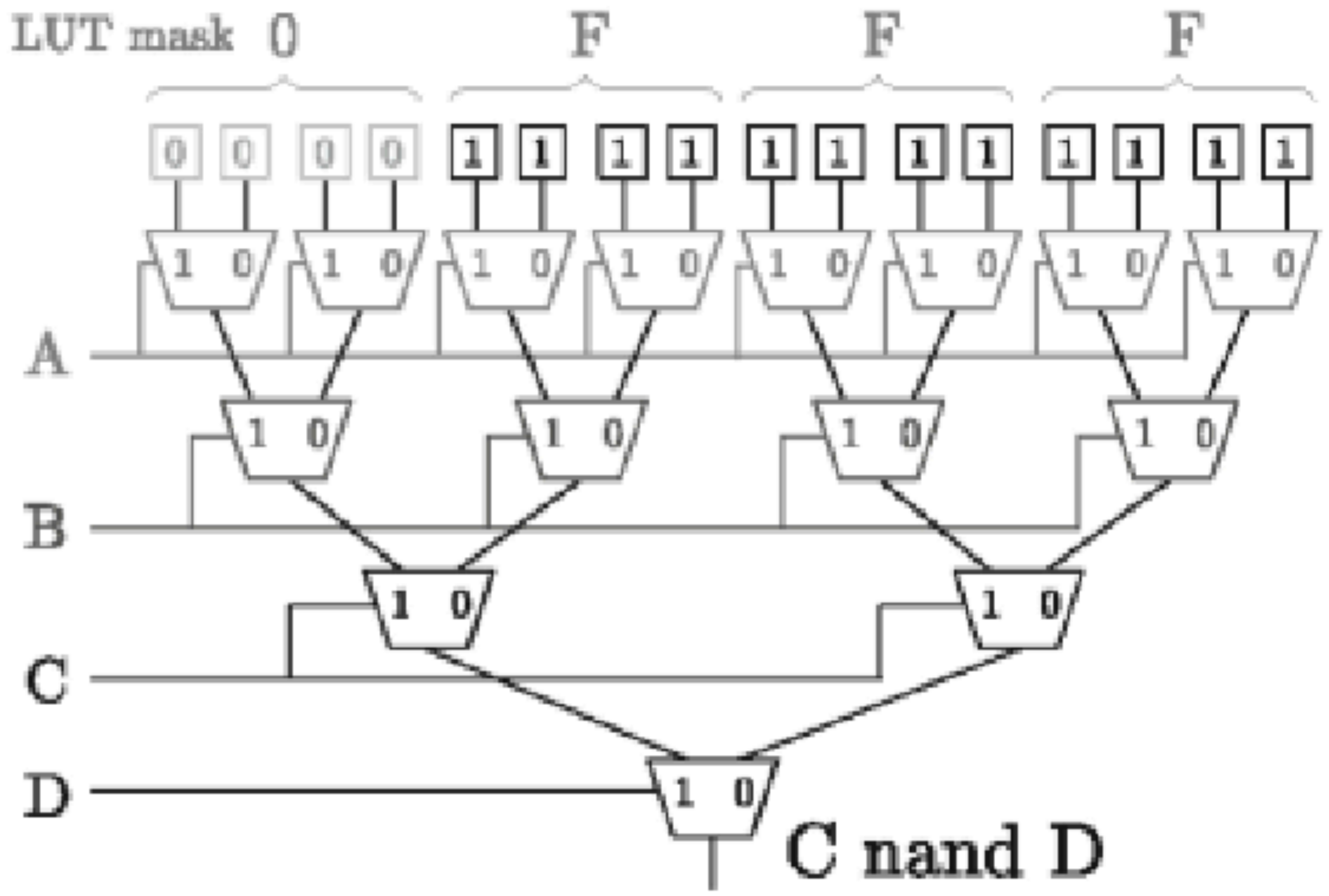
FPGA Synthesis Flow

- The input to the synthesis flow is the hardware specification, design constraints FPGA-specific commands
- The set of inputs is symbolically (hardware Description Languag contains the knowledge of desi specifications
- The design constraints include the input and output pads, betv registers, and between the regis



FPGA Design Example

- Lets reverse the function from LUT

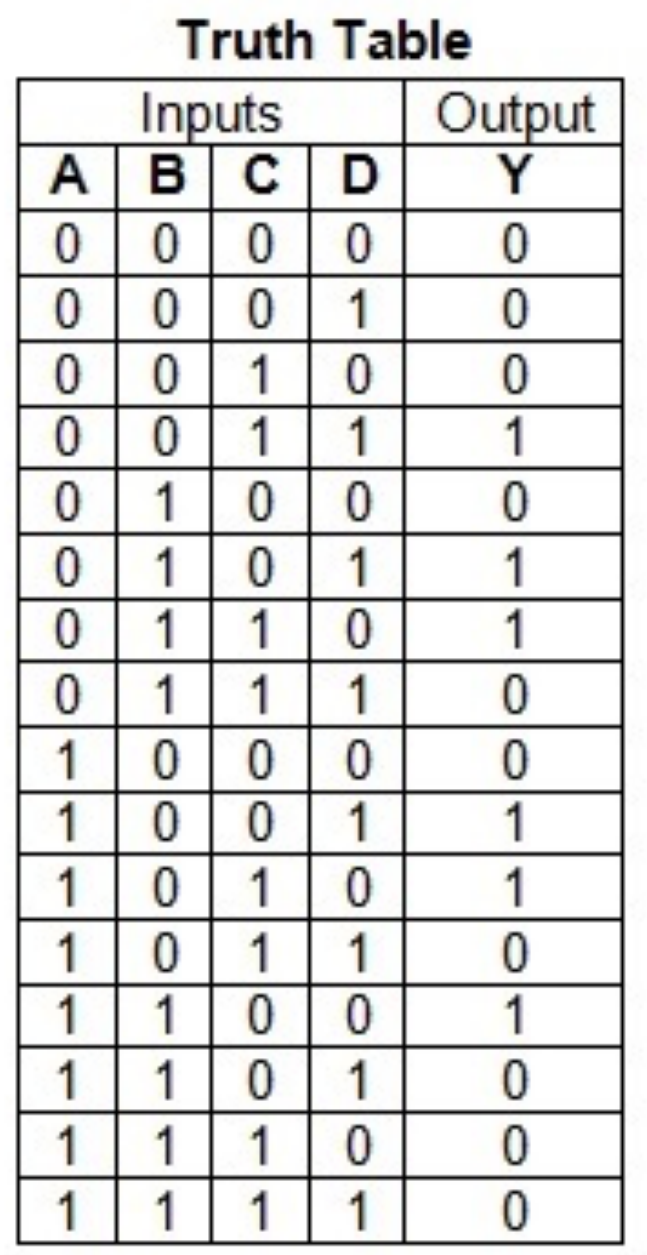


FPGA Design Example

- A Boolean Function of four input variables A, B, C and D using a 4-input LUT.
- Here, let the output become high only when any of the two input variables are one.
- What is the hardware realization?

FPGA Design Example

- A Boolean Function of four inp C and D using a 4-input LUT.
- Here, let the output become hi of the two input variables are one.
- What is the hardware realizatic



Truth Table

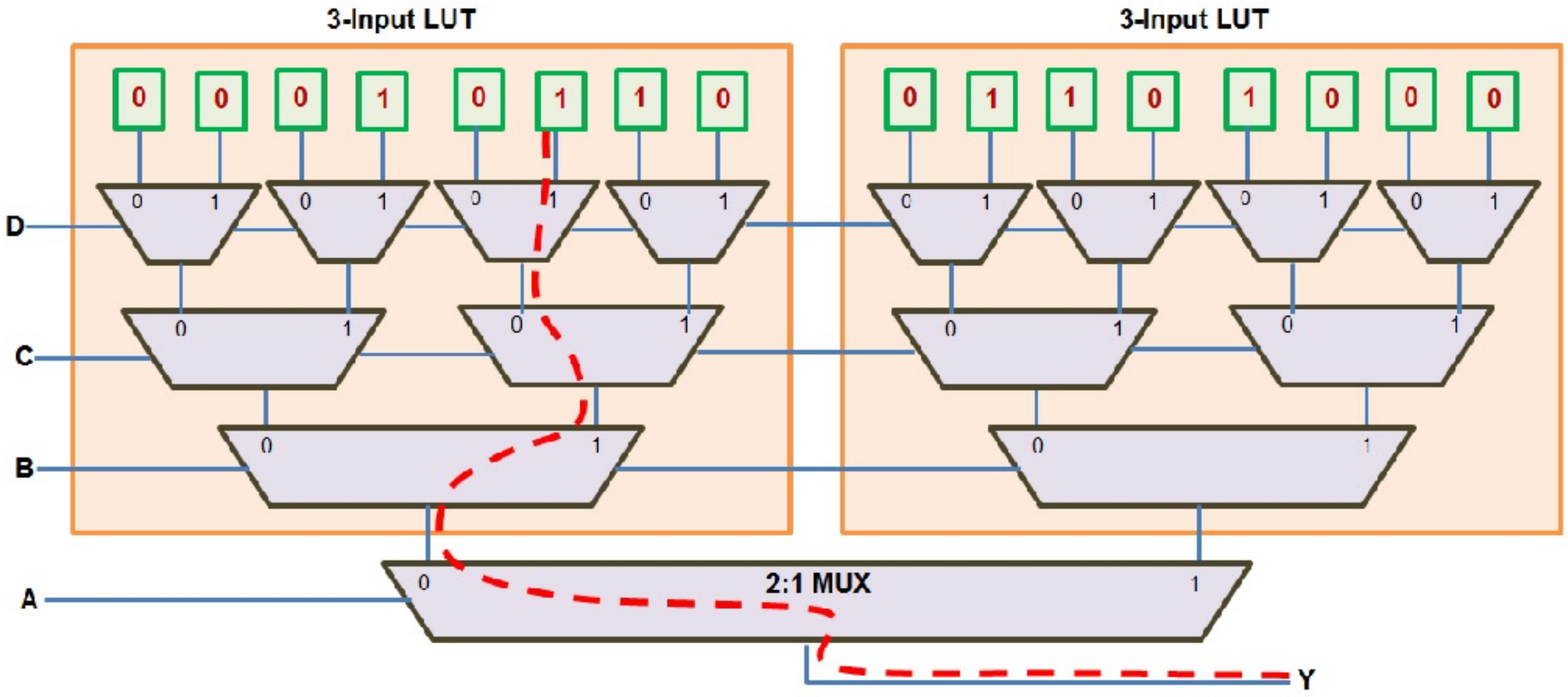
Inputs				Output
A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

FPGA Implementation on a Slice

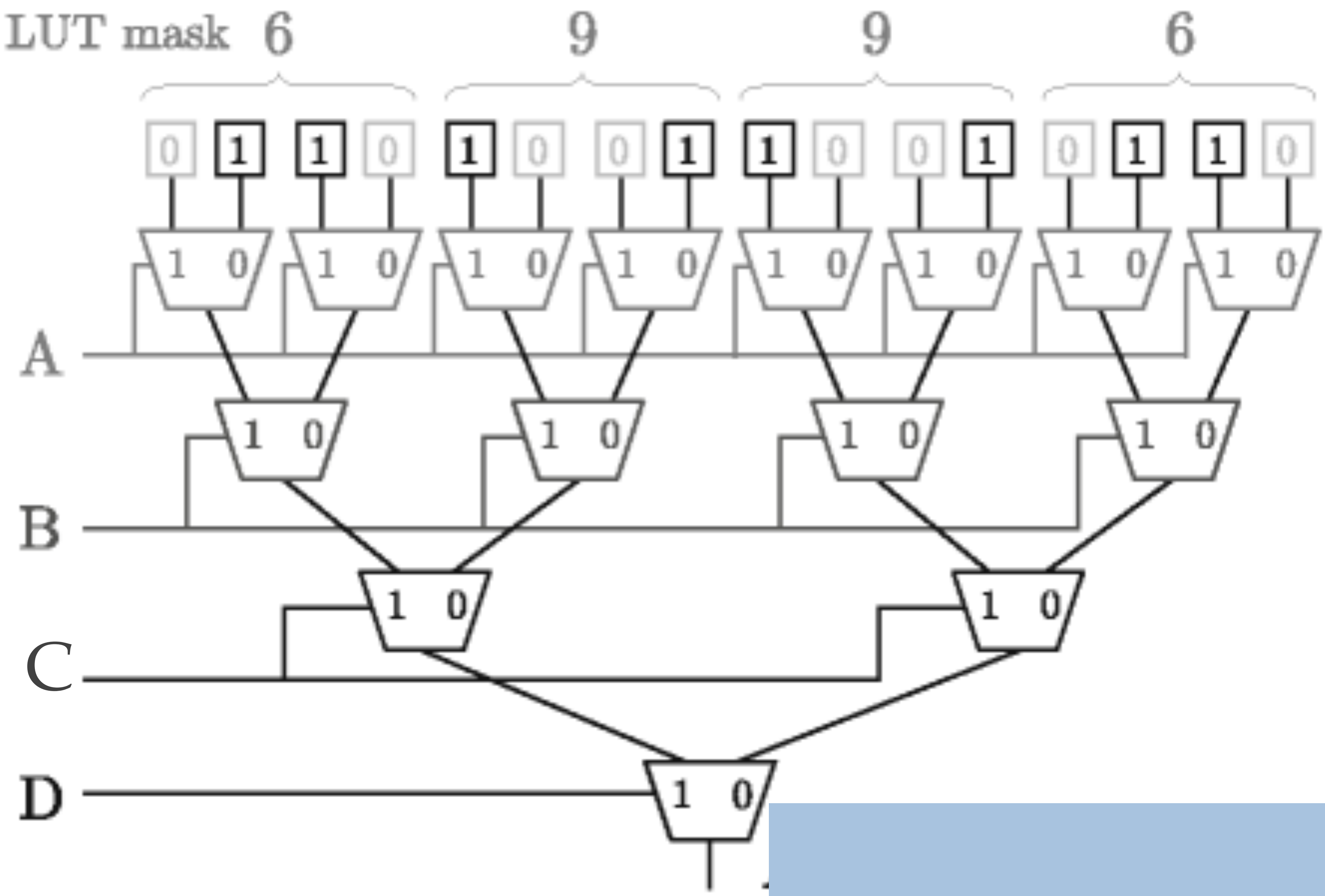
- Design the schematic yourself, using a 4-input LUT

FPGA Implementation on a Slice

- Design the schematic yourself, using a 4-input LUT

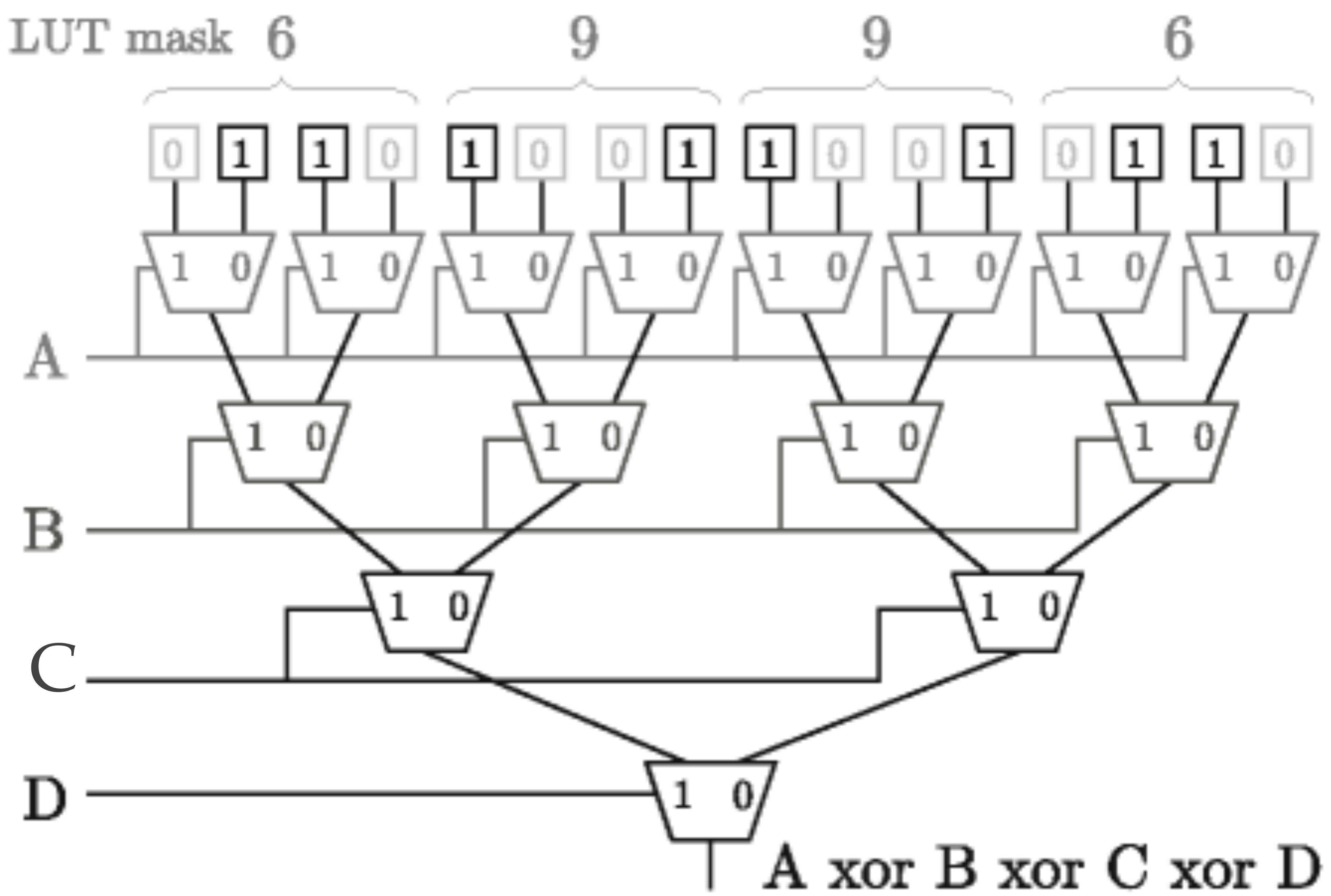


Practice

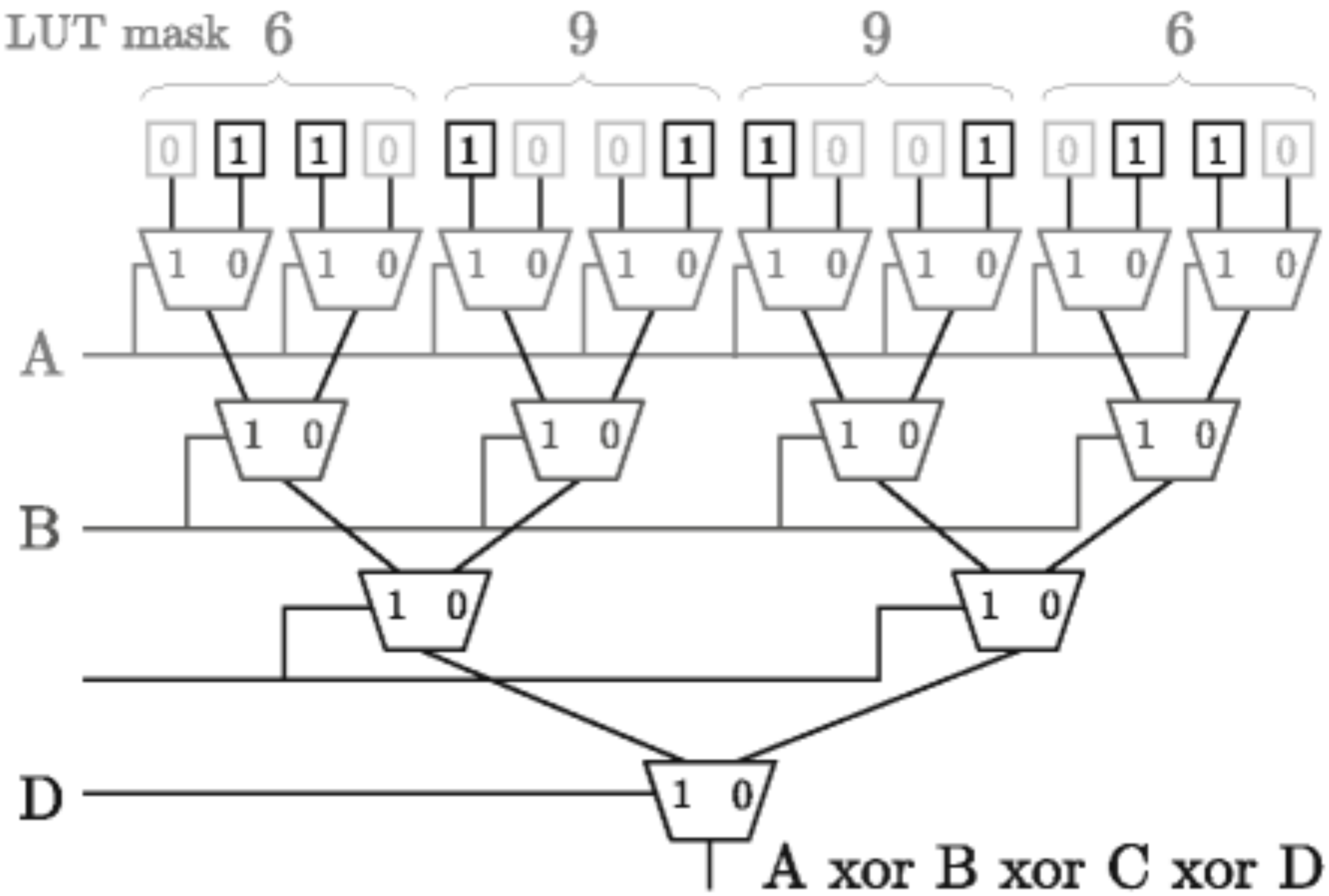


[redacted]

Practice



Practice

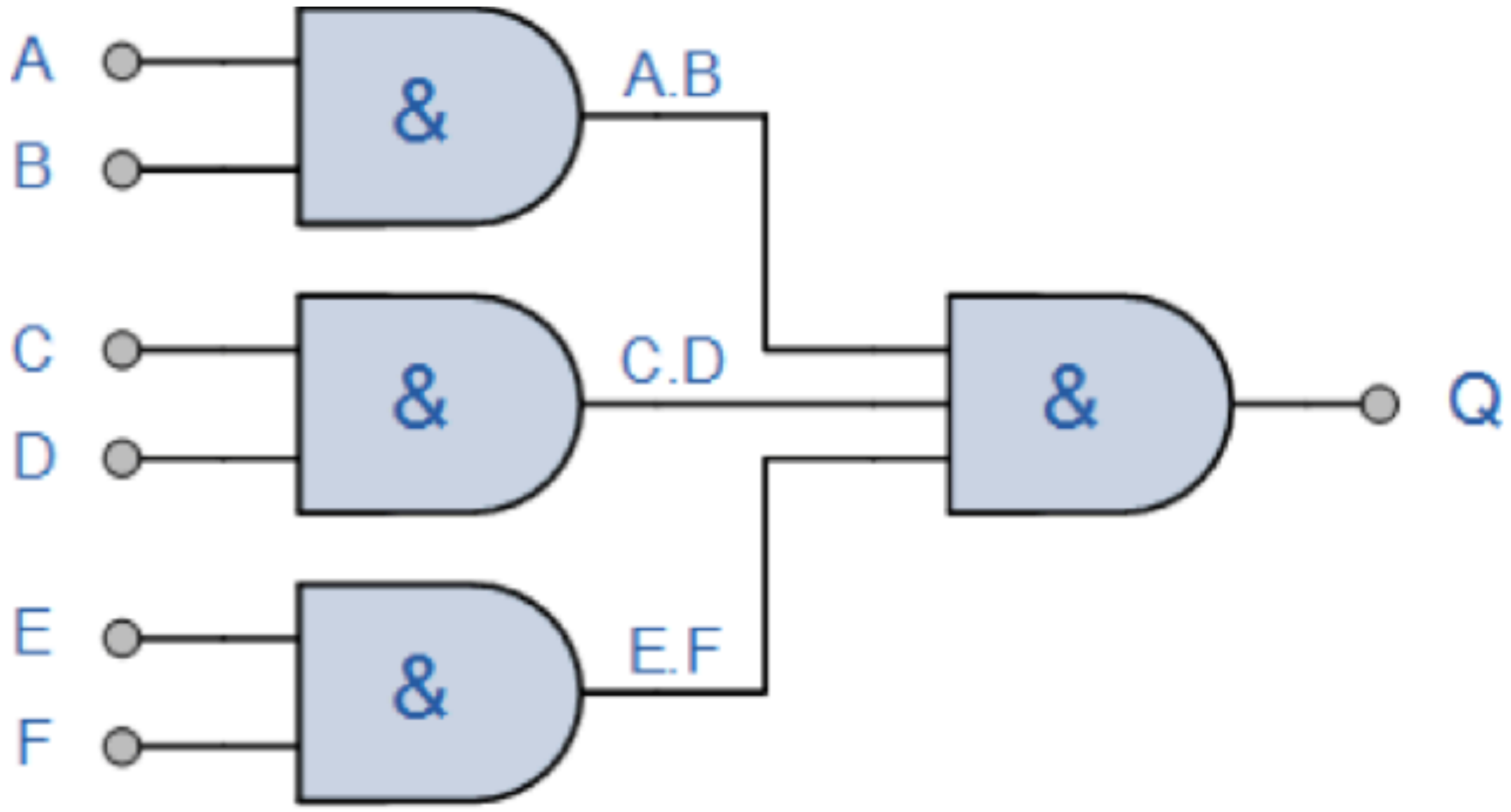


Direct Programming LUT

- LUT6 #(.INIT(64'h0000000000000000) / / Specify LUT Contents
- LUT6_inst (.O(O), / / LUT general output
- .I0(I0), / / LUT input
- .I1(I1), / / LUT input
- .I2(I2), / / LUT input
- .I3(I3), / / LUT input
- .I4(I4), / / LUT input
- .I5(I5) / / LUT input);/ / End of LUT6_inst instantiation

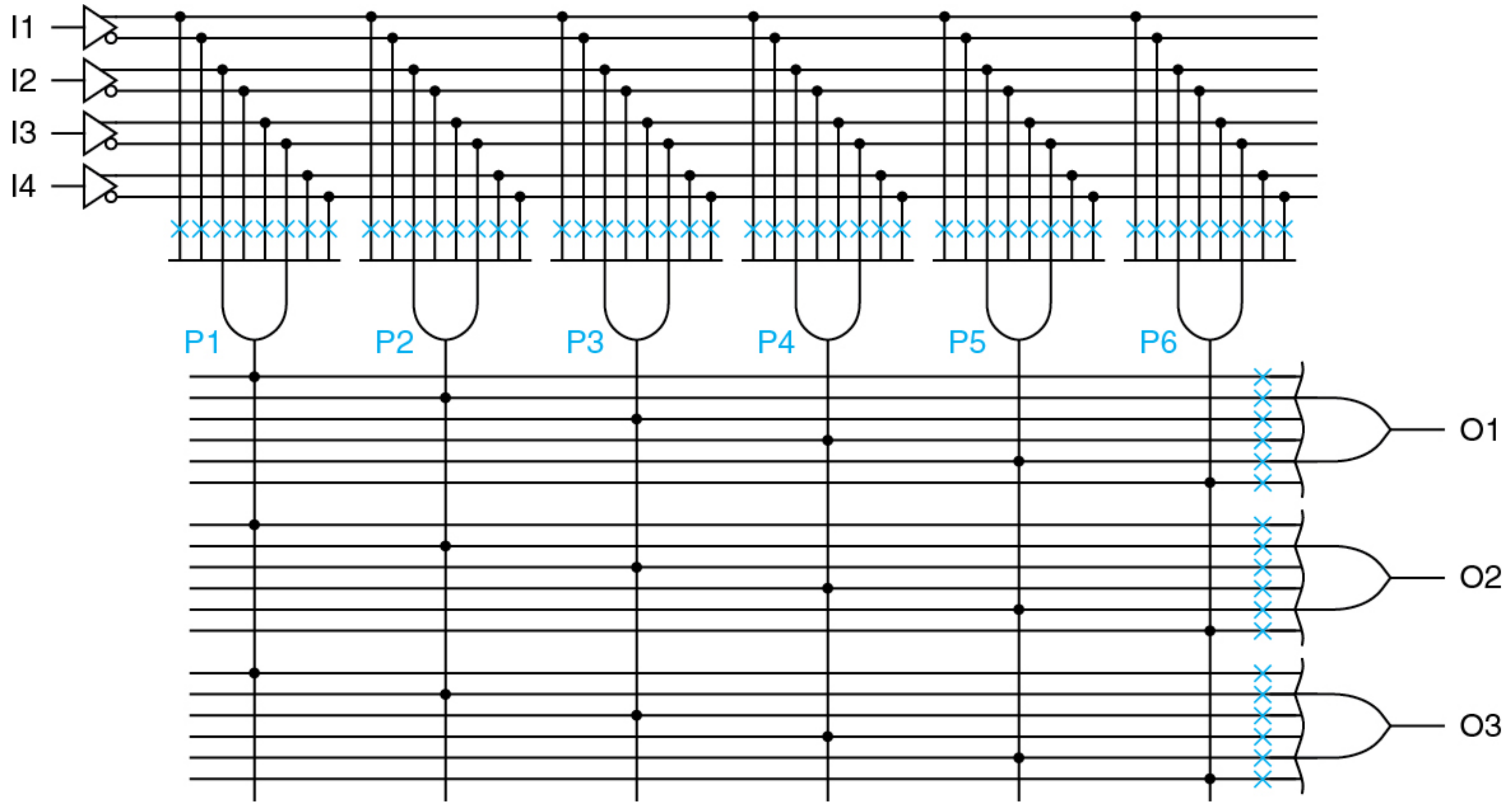
FPGA Design Example

- How should the LUT be programmed?
  - LUT6 #(.INIT(64’h0000_0000_0000_0001))
  - AND_6IN(.I0(A),.I1(B),.I2(C),.I3(D),.I4(E),.I5(F),.O(Q));



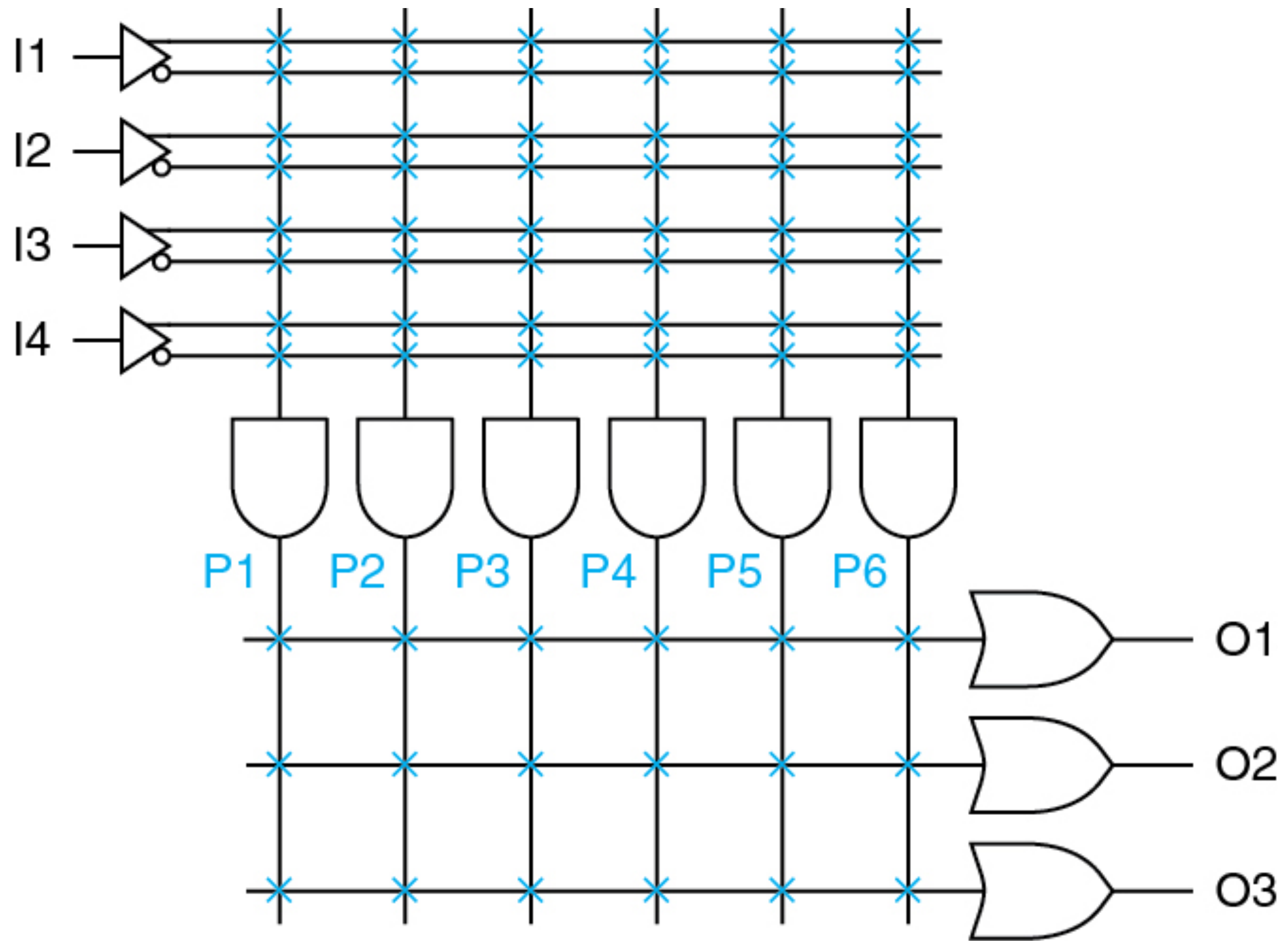
Other Programmable Devices

- The first programmable logic device (PLD) was programmable logic arrays (PLAs)
  - Connection with fuse



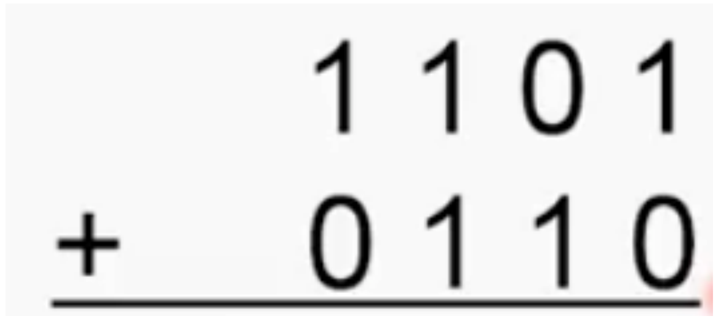
Other Programmable Devices

- Compact representation of a 4 × 3 PLA with six product terms



HW2: Adder

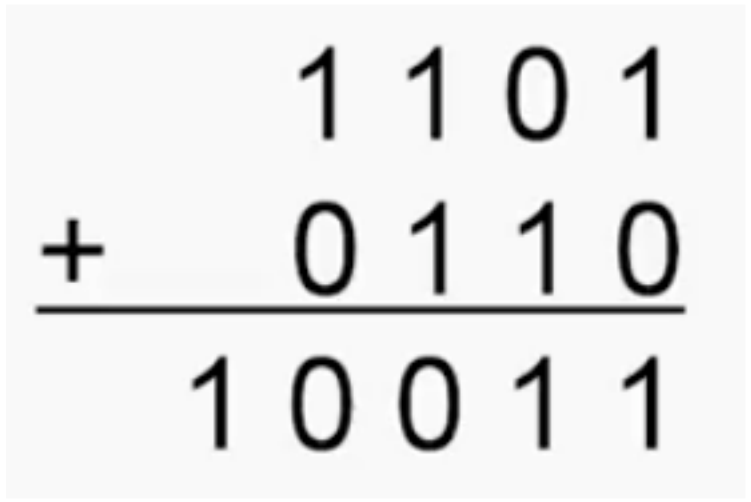
- 1101 + 0110 = ?



$$\begin{array}{r} 1\,1\,0\,1 \\ +\quad 0\,1\,1\,0 \\ \hline \end{array}$$

Binary Adder

- 1101 + 0110 = ?
- Calculation by hand…



```
     1 1 0 1
+    0 1 1 0
------------
   1 0 0 1 1
```

Half Adder

- What is a half adder?

Half Adder

- What is a half adder?
- Two 1-bit numbers are being added

Half Adder

- What is a half adder?
- Two 1-bit numbers are being added
  - Two inputs A and B

Half Adder

- What is a half adder?
- Two 1-bit numbers are being added
  - Two inputs A and B
  - Two outputs S(um) and C(arry out)

Half Adder

- What is a half adder?
- Two 1-bit numbers are being added
  - Two inputs A and B
  - Two outputs S(um) and C(arry out)
- No carry in

Half Adder

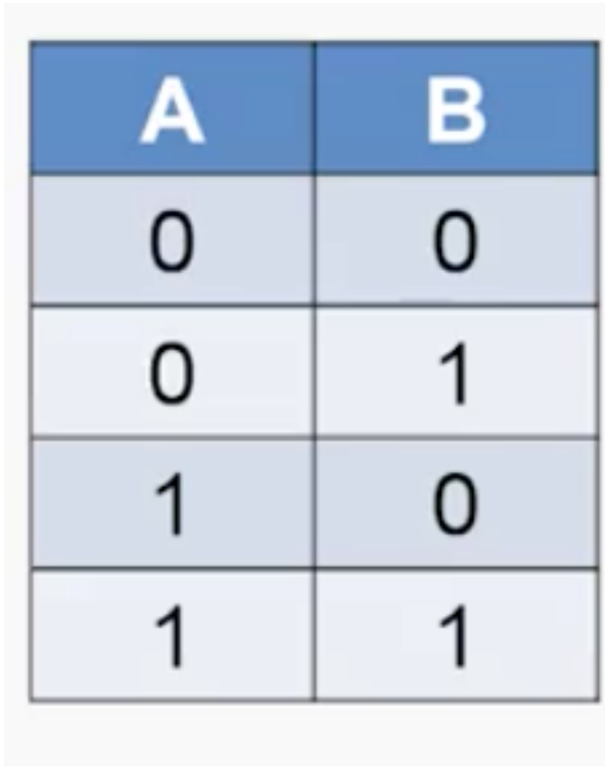
- What is a half adder?
- Two 1-bit numbers are being added
  - Two inputs A and B
  - Two outputs S(um) and C(arry out)
- No carry in
- S = ?

Half Adder

- What is a half adder?
- Two 1-bit numbers are being added
  - Two inputs A and B
  - Two outputs S(um) and C(arry out)
- No carry in
- S = ?
- C= ?

Half Adder

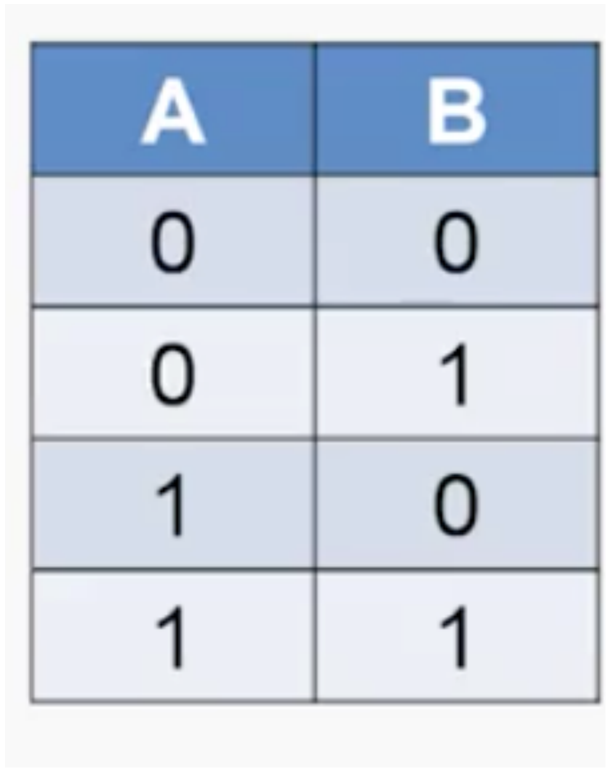
- What is a half adder?
- Two 1-bit numbers are being added
  - Two inputs A and B
  - Two outputs S(um) and C(arry out)
- No carry in
- S = ?
- C= ?



A	B
0	0
0	1
1	0
1	1

Half Adder

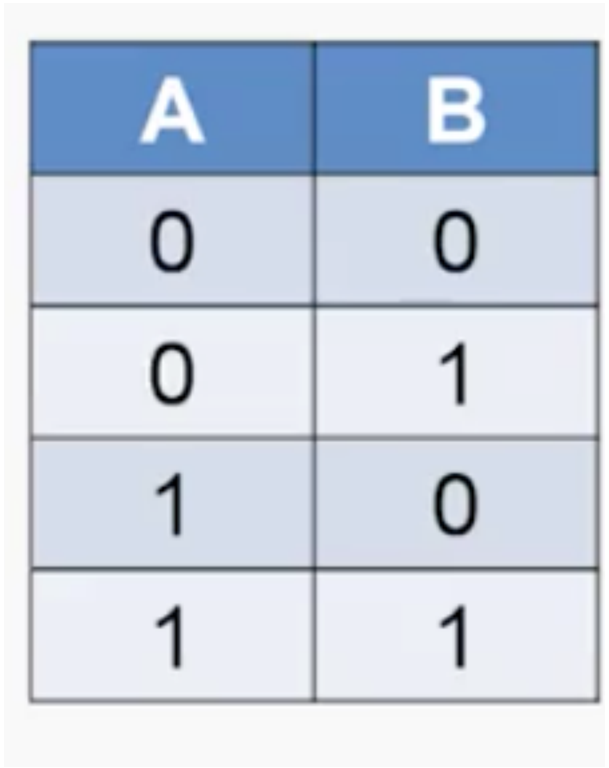
- What is a half adder?
- Two 1-bit numbers are being added
  - Two inputs A and B
  - Two outputs S(um) and C(arry out)
- No carry in
- S = ?
- C= ?
- Please write down your answer



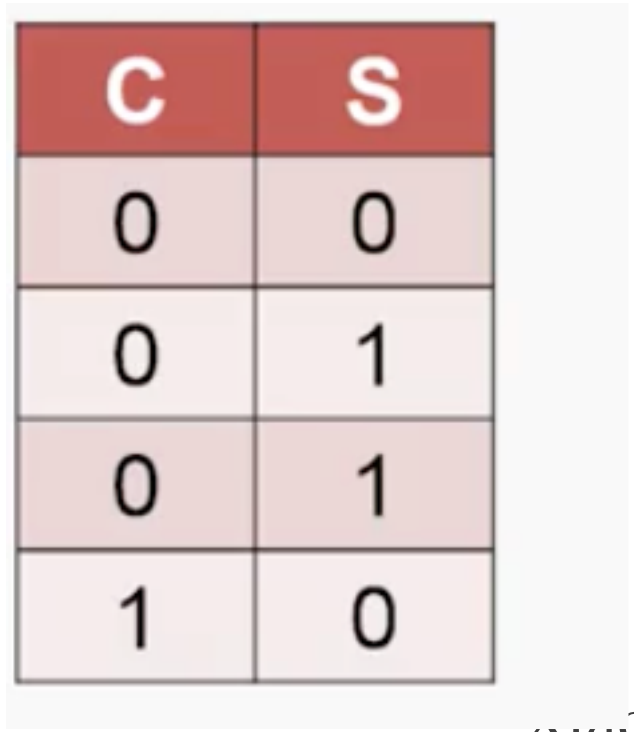
A	B
0	0
0	1
1	0
1	1

Half Adder

- What is a half adder?
- Two 1-bit numbers are being added
  - Two inputs A and B
  - Two outputs S(um) and C(arry out)
- No carry in
- S = ?
- C= ?
- Please write down your answer



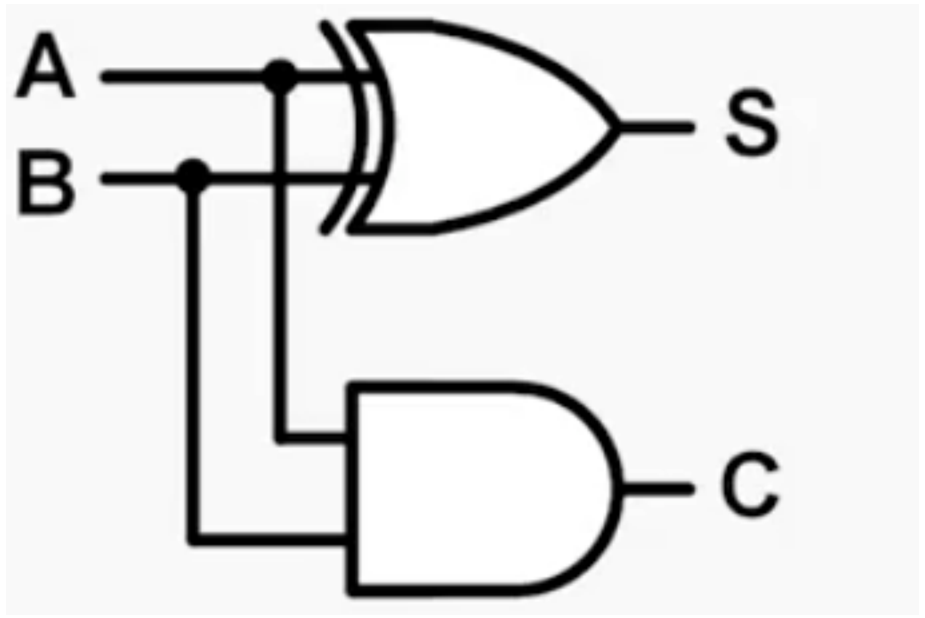
A	B
0	0
0	1
1	0
1	1

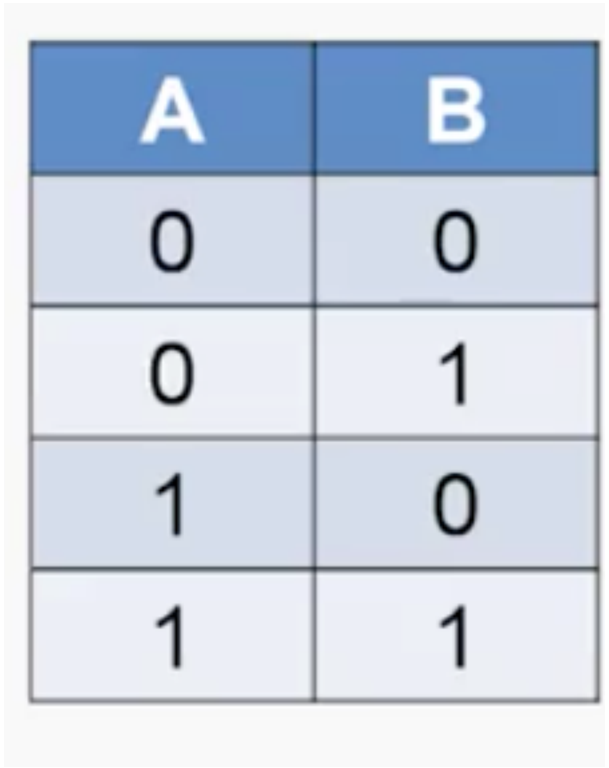


C	S
0	0
0	1
0	1
1	0

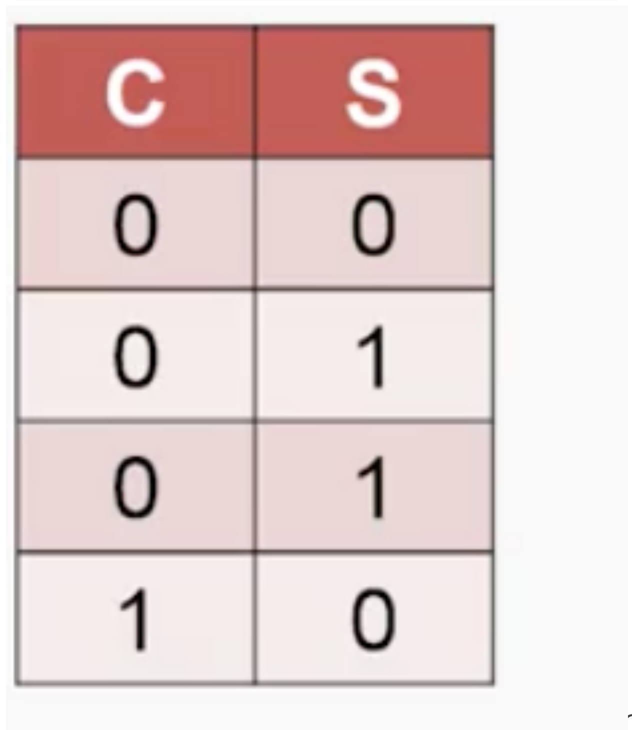
Half Adder

- What is a half adder?
- Two 1-bit numbers are being added
  - Two inputs A and B
  - Two outputs S(um) and C(arry out)
- No carry in
- S = ?
- C= ?
- Please write down your answer





A	B
0	0
0	1
1	0
1	1



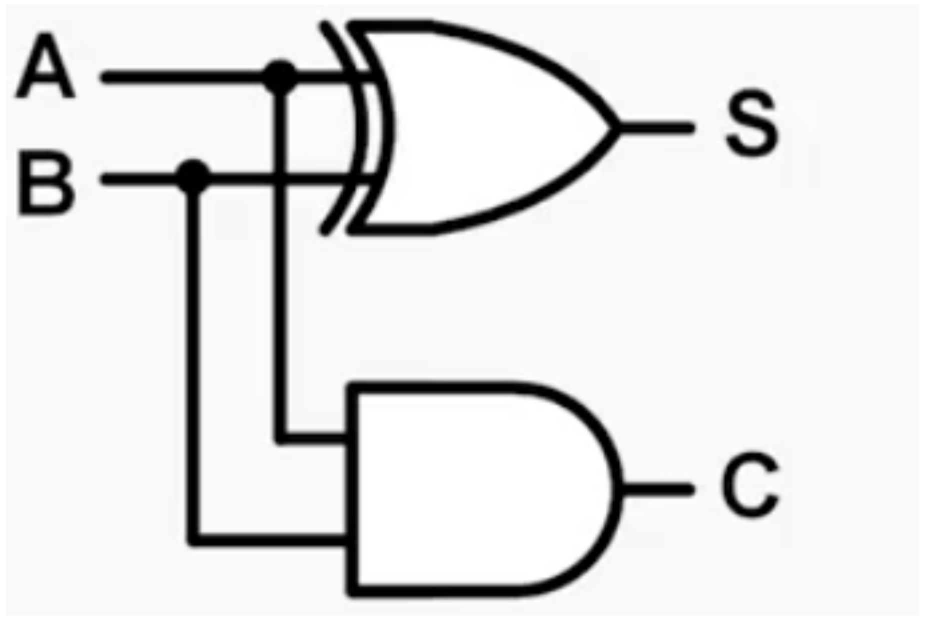
C	S
0	0
0	1
0	1
1	0

Half Adder

- What is a half adder?
- Two 1-bit numbers are being added
    - Two inputs A and B
    - Two outputs S(um) and C(arry out)
- No carry in
- S = A XOR B
- C=A AND B

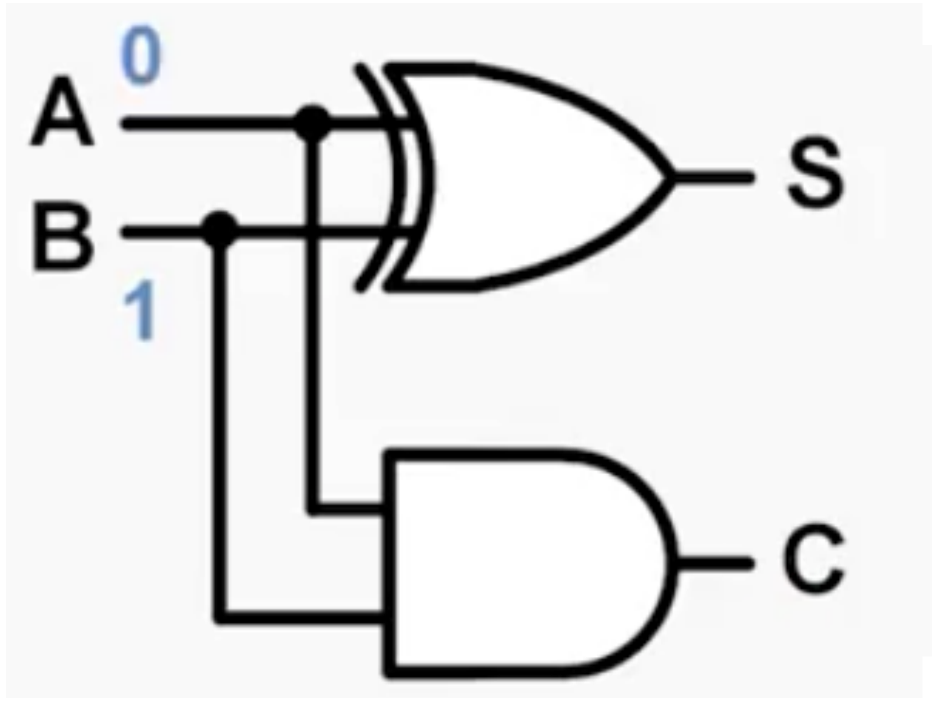
Half Adder

- What is a half adder?
- Two 1-bit numbers are being added
  - Two inputs A and B
  - Two outputs S(um) and C(arry out)
- No carry in
- S = A XOR B
- C=A AND B



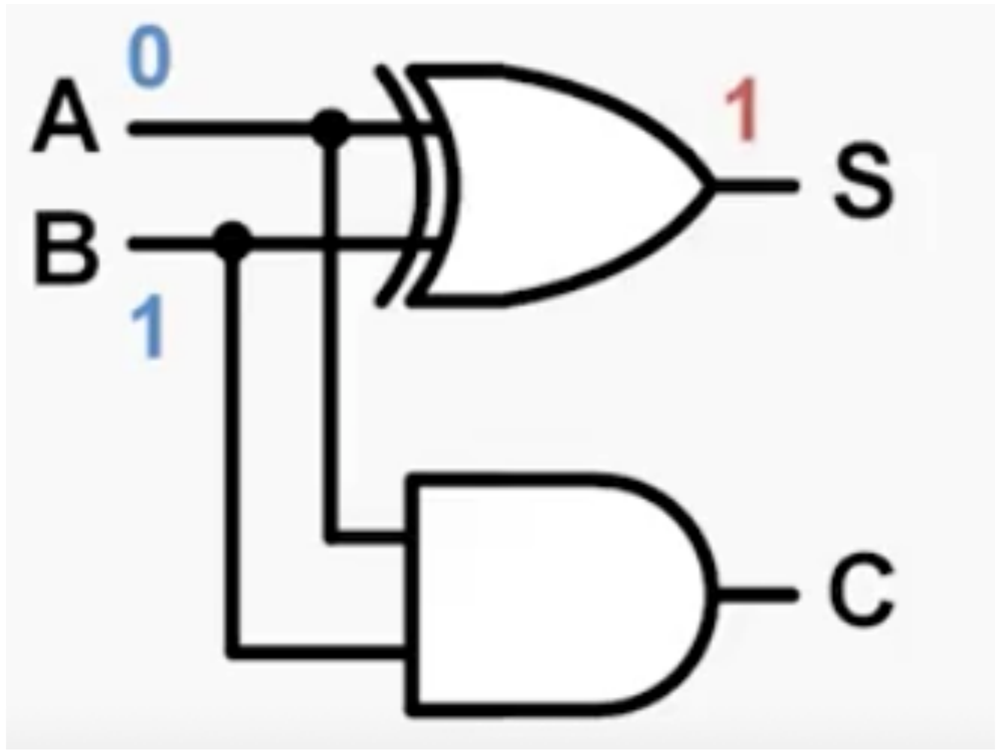
Half Adder

- What is a half adder?
- Two 1-bit numbers are being added
  - Two inputs A and B
  - Two outputs S(um) and C(arry out)
- No carry in
- S = A XOR B
- C=A AND B



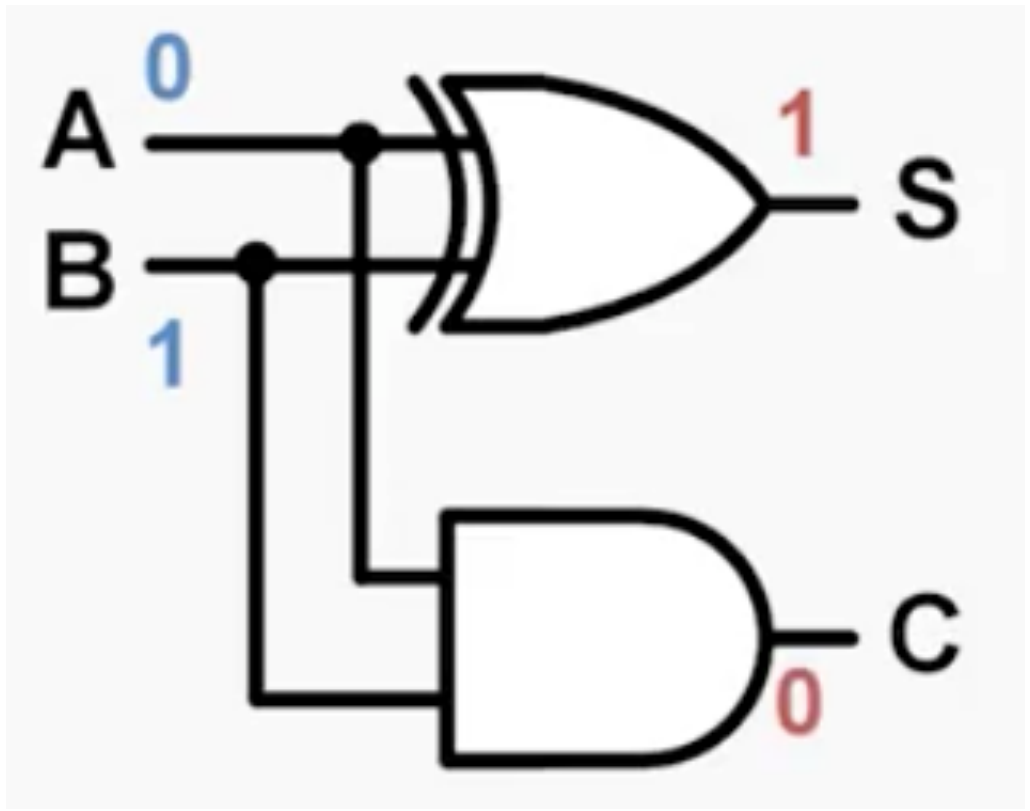
Half Adder

- What is a half adder?
- Two 1-bit numbers are being added
  - Two inputs A and B
  - Two outputs S(um) and C(arry out)
- No carry in
- S = A XOR B
- C=A AND B



Half Adder

- What is a half adder?
- Two 1-bit numbers are being added
  - Two inputs A and B
  - Two outputs S(um) and C(arry out)
- No carry in
- S = A XOR B
- C=A AND B



Full Adder

- What is a full adder?

Full Adder

- What is a full adder?
  - Has 3-bit input

Full Adder

- What is a full adder?
  - Has 3-bit input
  - Has a Cin (carry in) bit

Full Adder

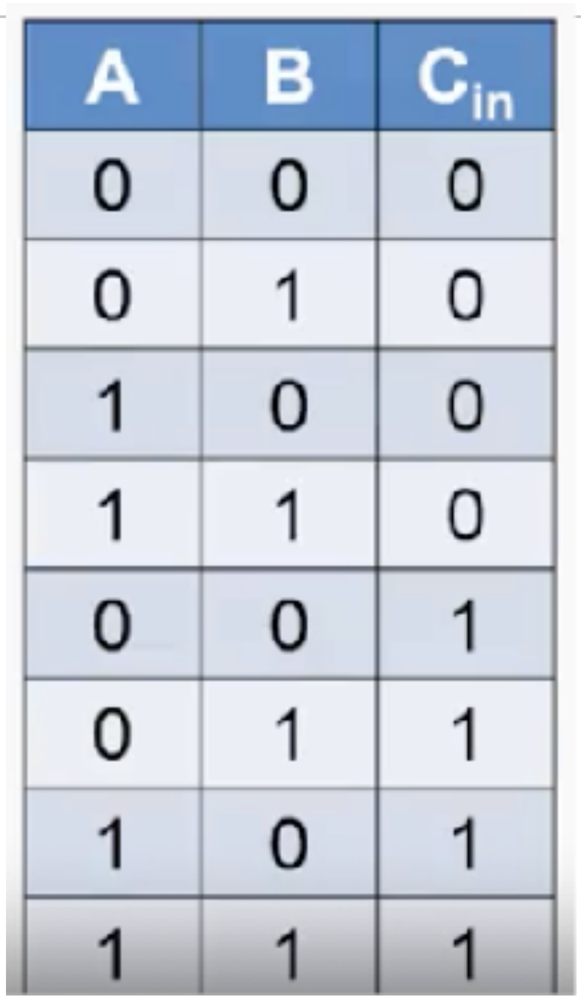
- What is a full adder?
  - Has 3-bit input
  - Has a Cin (carry in) bit
  - Has two outputs: S(um) and Cout

Full Adder

- What is a full adder?
  - Has 3-bit input
  - Has a Cin (carry in) bit
  - Has two outputs: S(um) and Cout

Full Adder

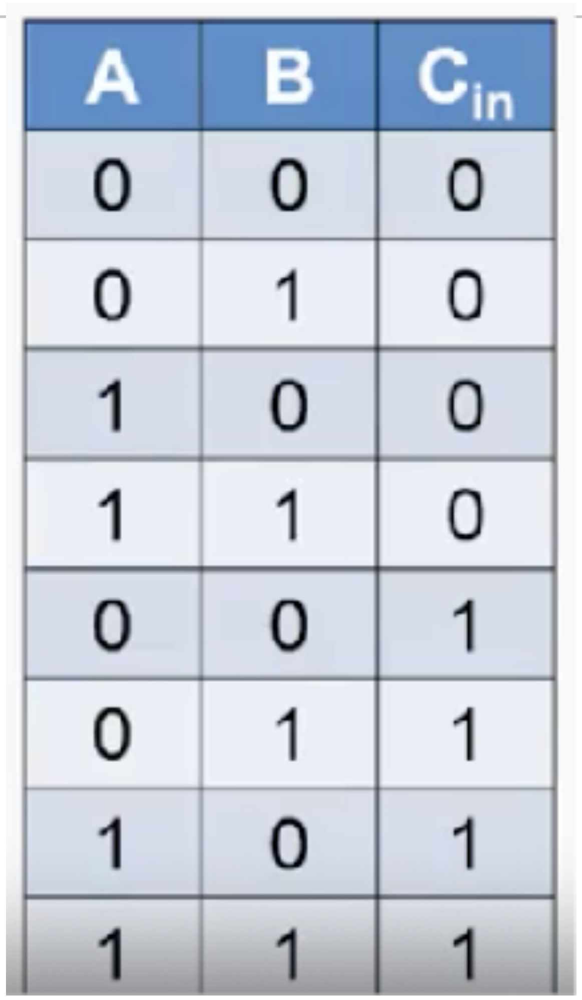
- What is a full adder?
  - Has 3-bit input
  - Has a Cin (carry in) bit
  - Has two outputs: S(um) and Cout



A	B	C_{in}
0	0	0
0	1	0
1	0	0
1	1	0
0	0	1
0	1	1
1	0	1
1	1	1

Full Adder

- What is a full adder?
  - Has 3-bit input
  - Has a Cin (carry in) bit
  - Has two outputs: S(um) and Cout

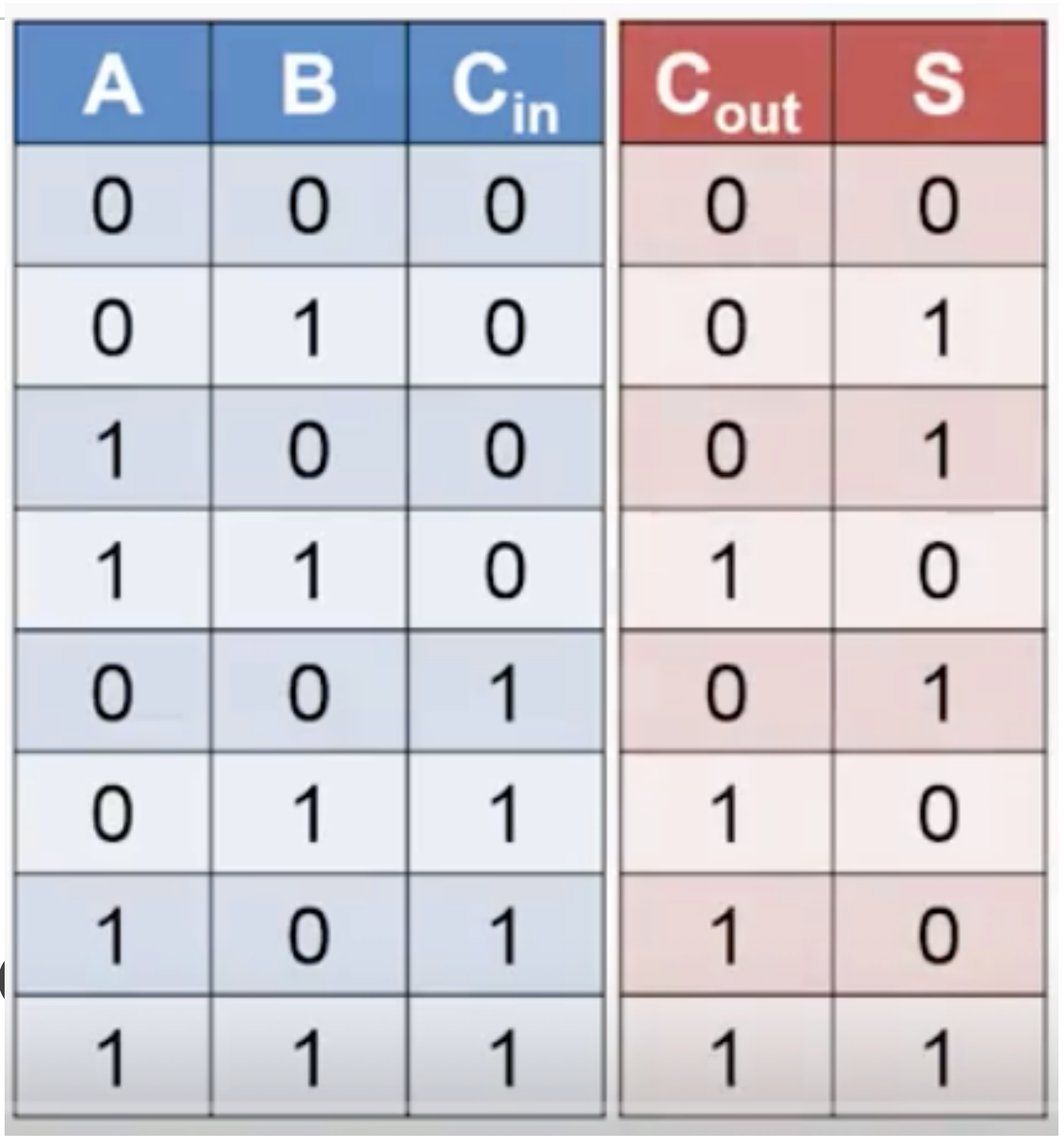


A	B	C_{in}
0	0	0
0	1	0
1	0	0
1	1	0
0	0	1
0	1	1
1	0	1
1	1	1

- Write down the truth table for Cout and S

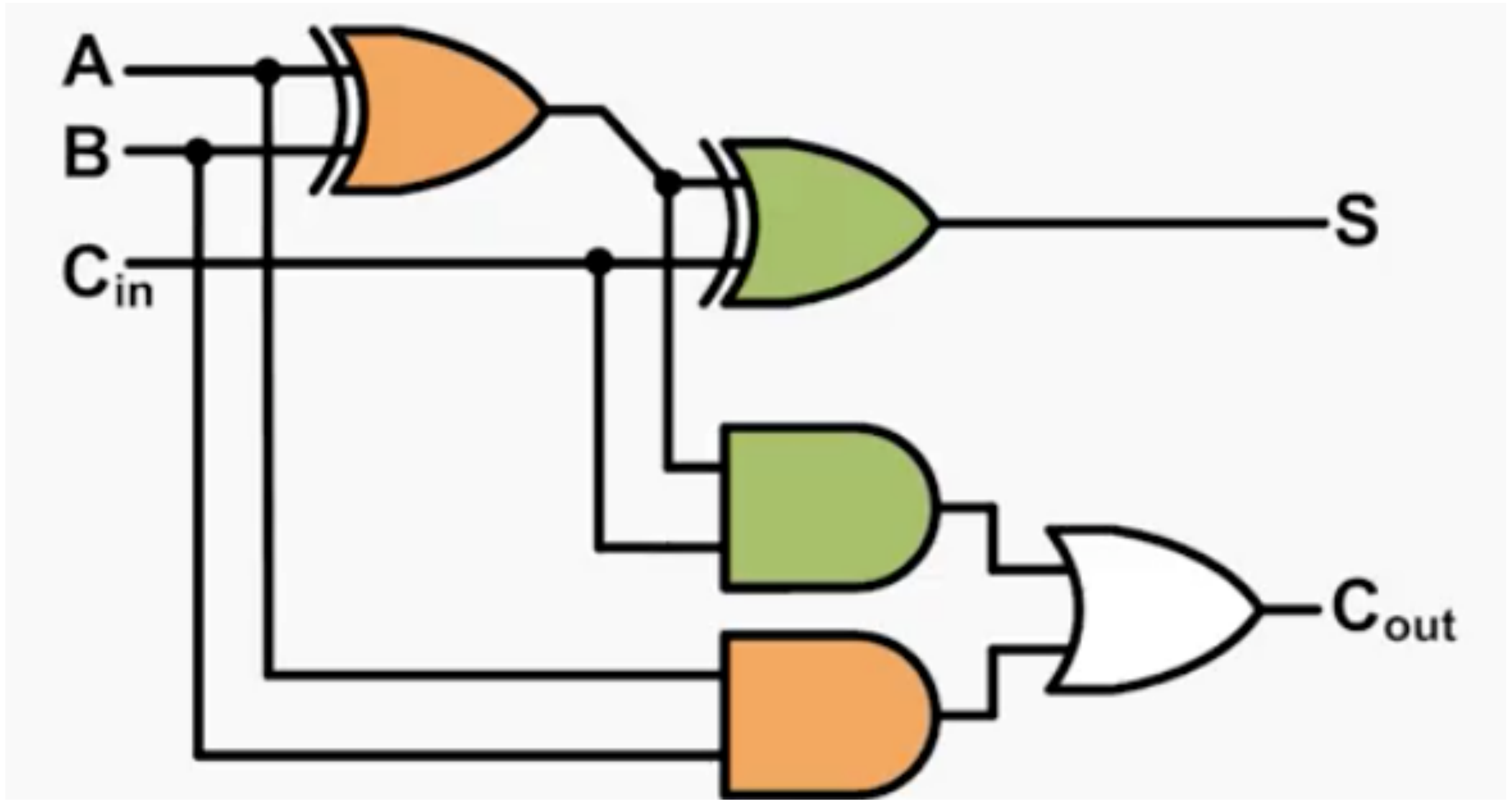
Full Adder

- What is a full adder?
  - Has 3-bit input
  - Has a Cin (carry in) bit
  - Has two outputs: S(um) and Cout
- Write down the truth table for



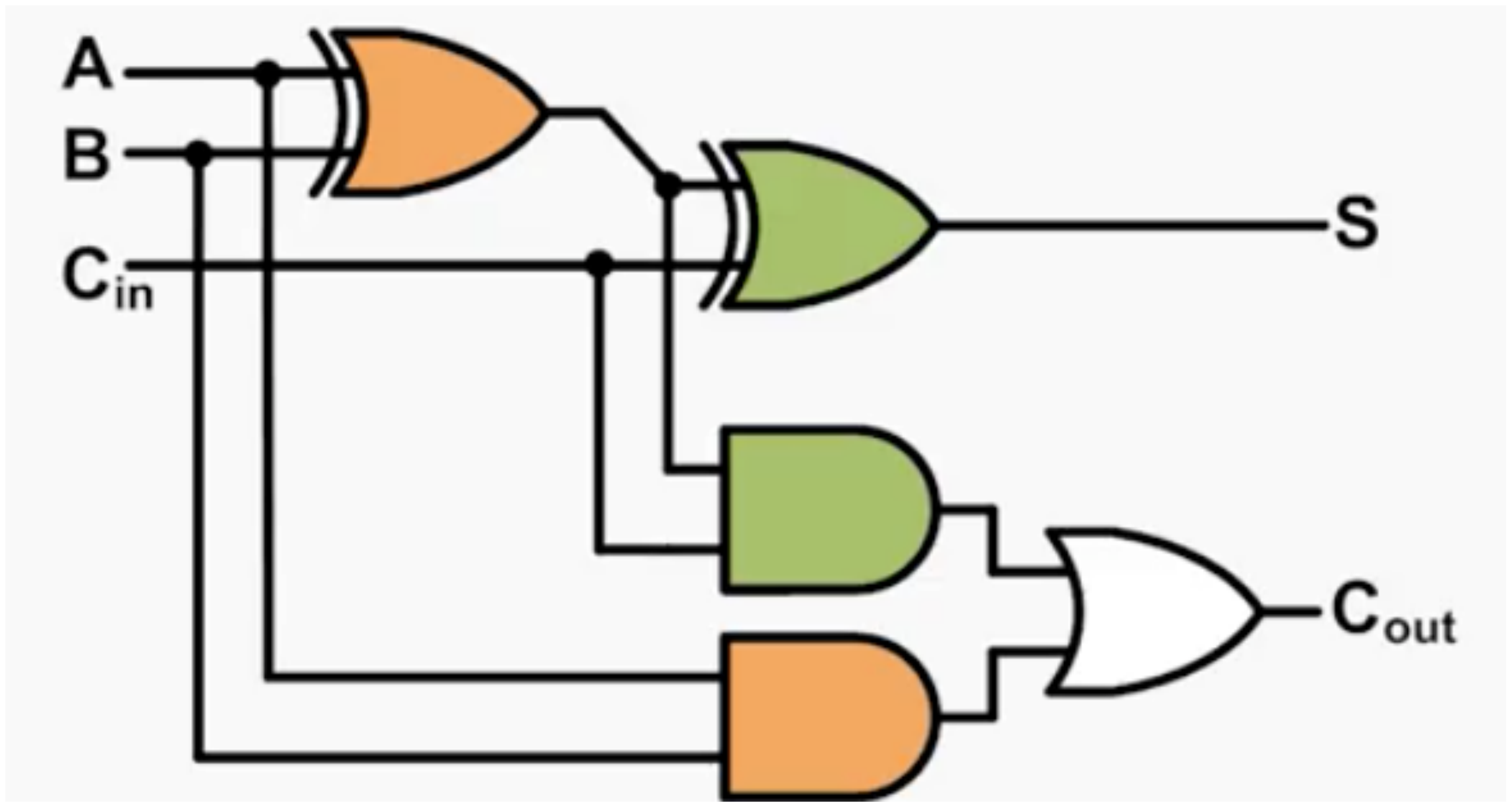
A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	1	0	0	1
1	0	0	0	1
1	1	0	1	0
0	0	1	0	1
0	1	1	1	0
1	0	1	1	0
1	1	1	1	1

Full Adder



Full Adder

- Composition
  - Two half-adders: colored differently!

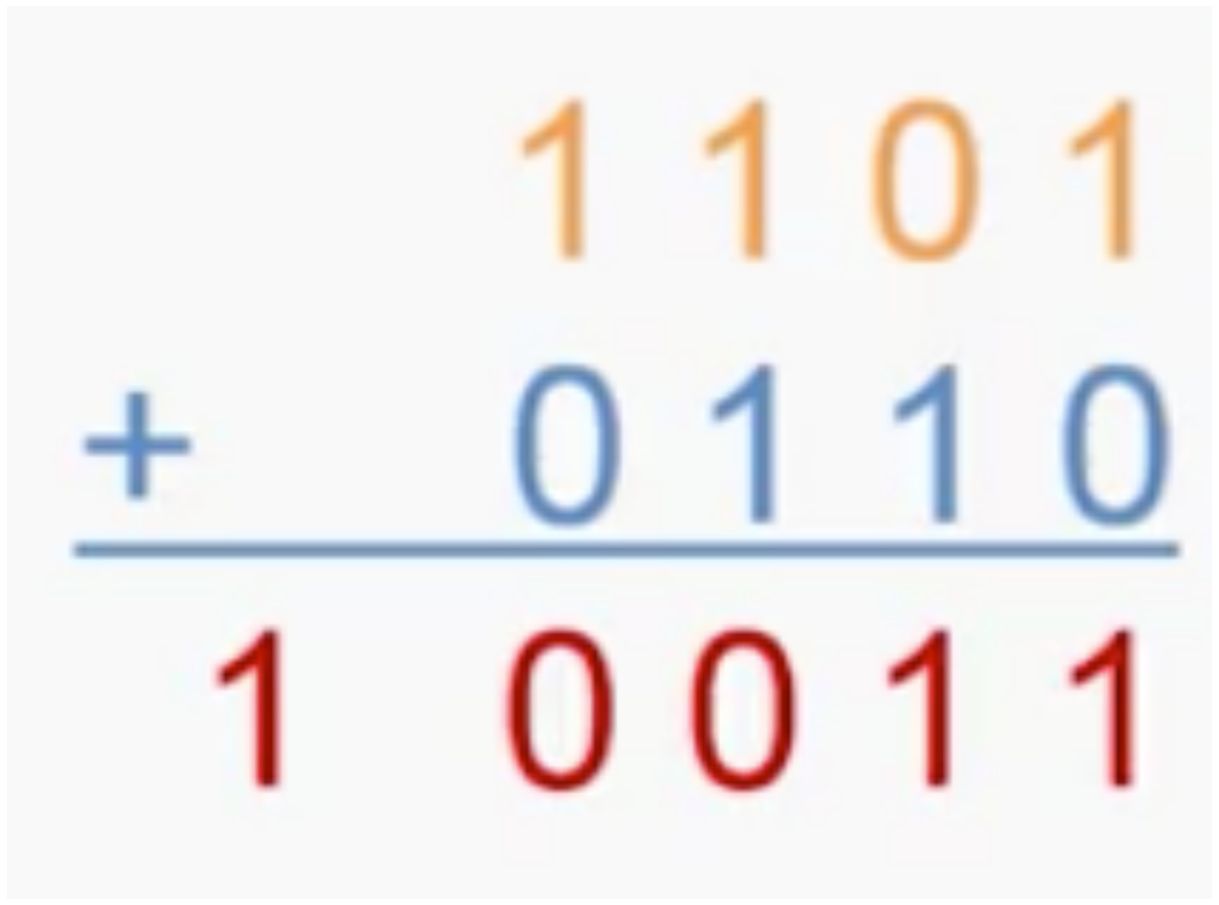


4-bit Full Adder

- A=1101
- B=0110

4-bit Full Adder

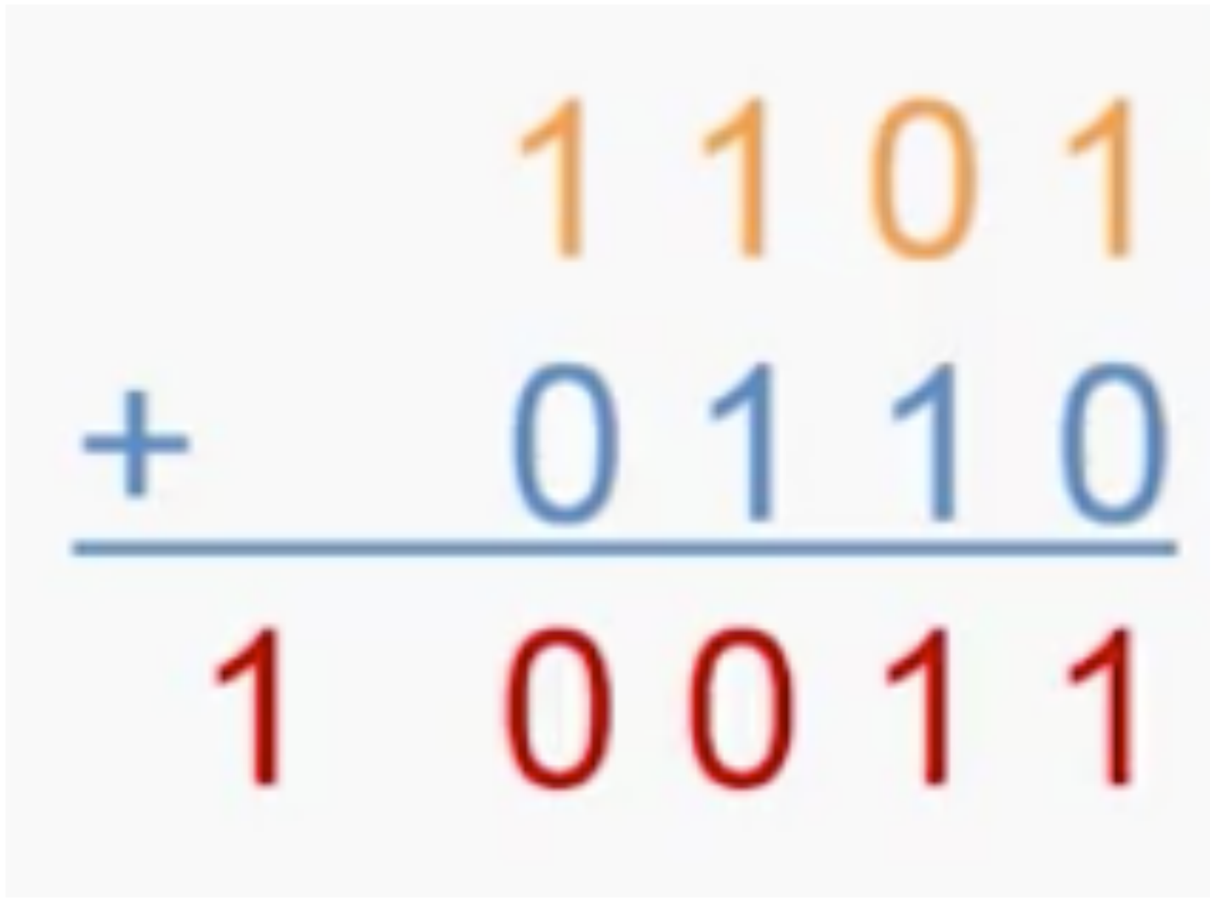
- A=1101
- B=0110



$$\begin{array}{r} 1101 \\ +\ 0110 \\ \hline 10011 \end{array}$$

4-bit Full Adder

- A=1101
- B=0110



$$\begin{array}{r} 1101 \\ +\ 0110 \\ \hline 10011 \end{array}$$

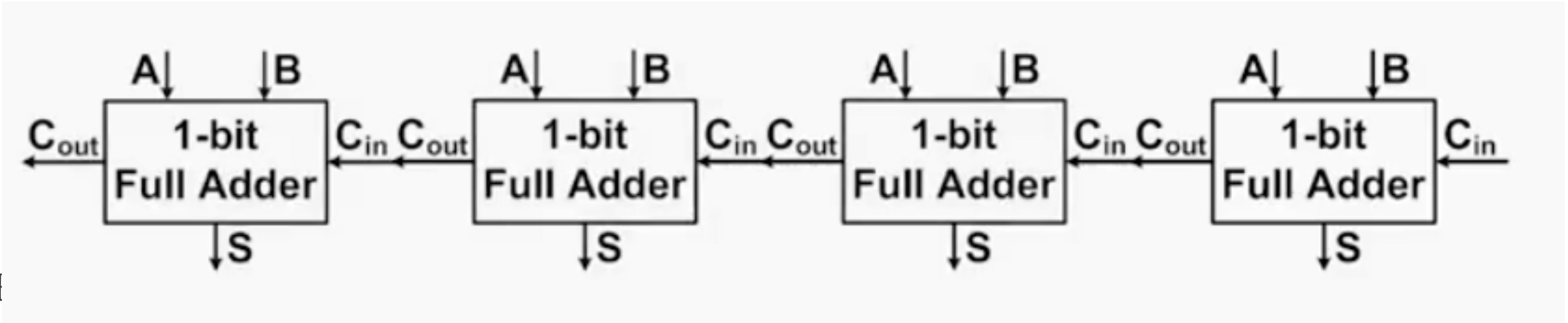
- How to build a 4-bit full adder?

4-bit Full Adder

- A=1101
- B=0110

$$\begin{array}{r} 1\,1\,0\,1 \\ +\quad 0\,1\,1\,0 \\ \hline 1\,0\,0\,1\,1 \end{array}$$

- How to build a 4-bit full adder?



4-bit Full Adder

- How it works?

4-bit Full Adder

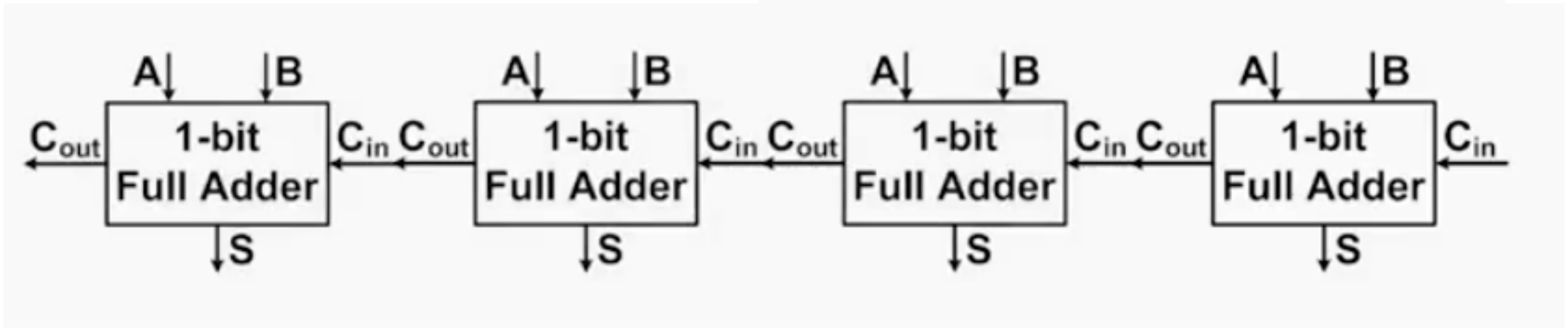
- How it works?

$$\begin{array}{r} 1101 \\ +\quad 0110 \\ \hline 10011 \end{array}$$

4-bit Full Adder

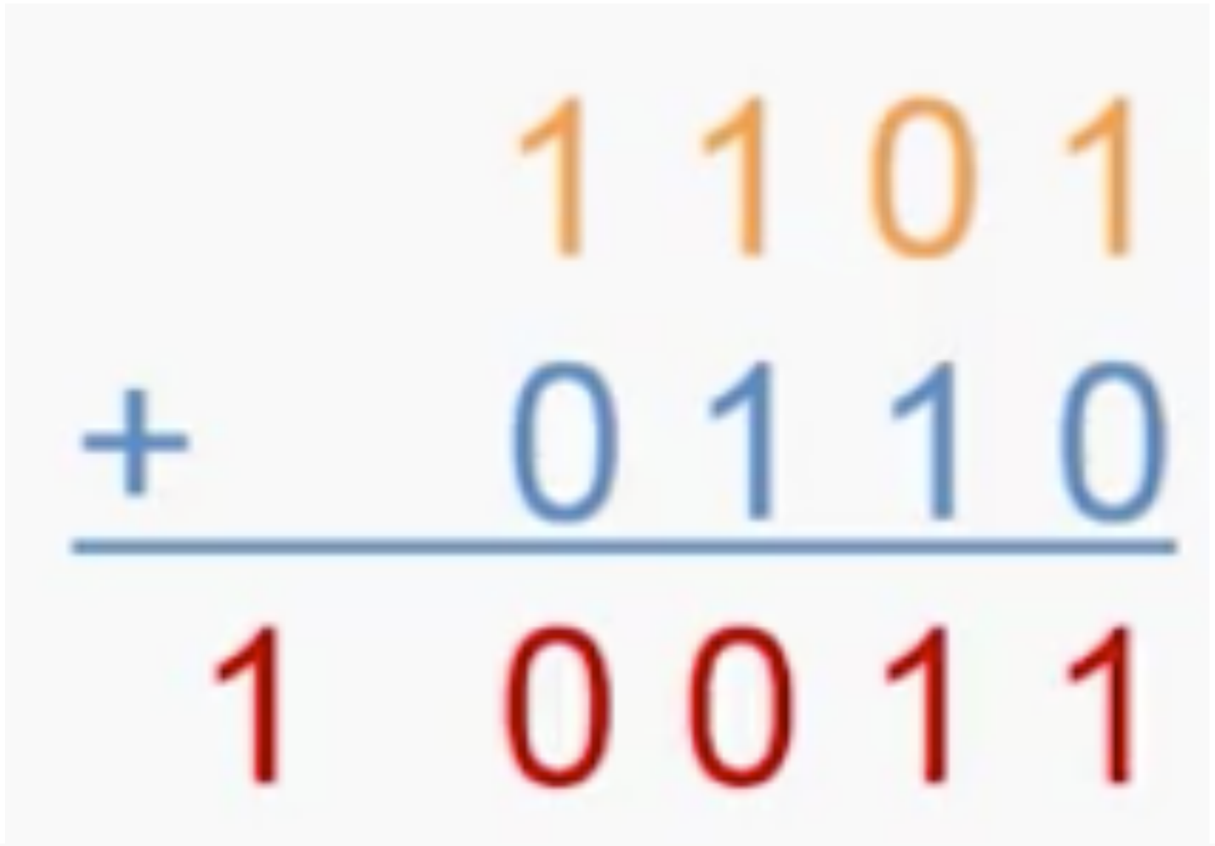
- How it works?

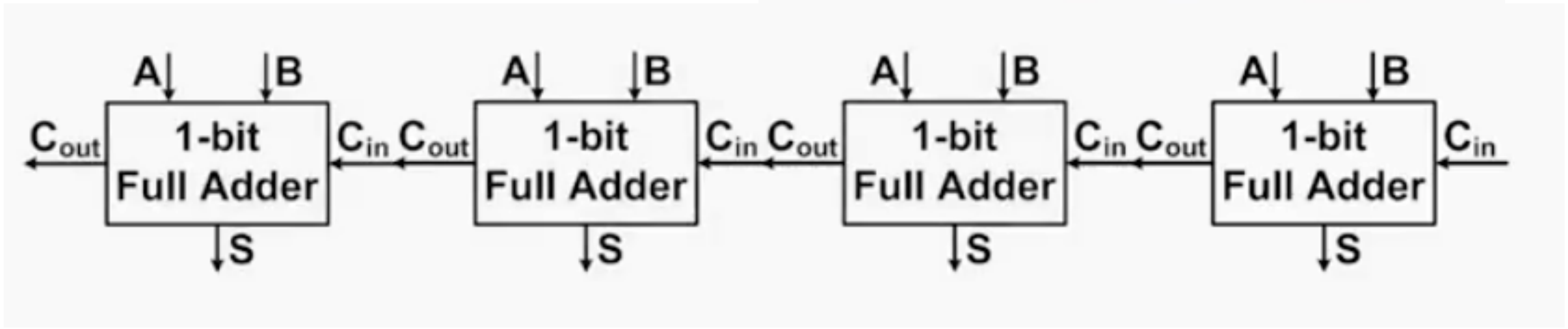
$$\begin{array}{r} 1101 \\ +\ 0110 \\ \hline 10011 \end{array}$$



4-bit Full Adder

- How it works?
- Each bit assigned!

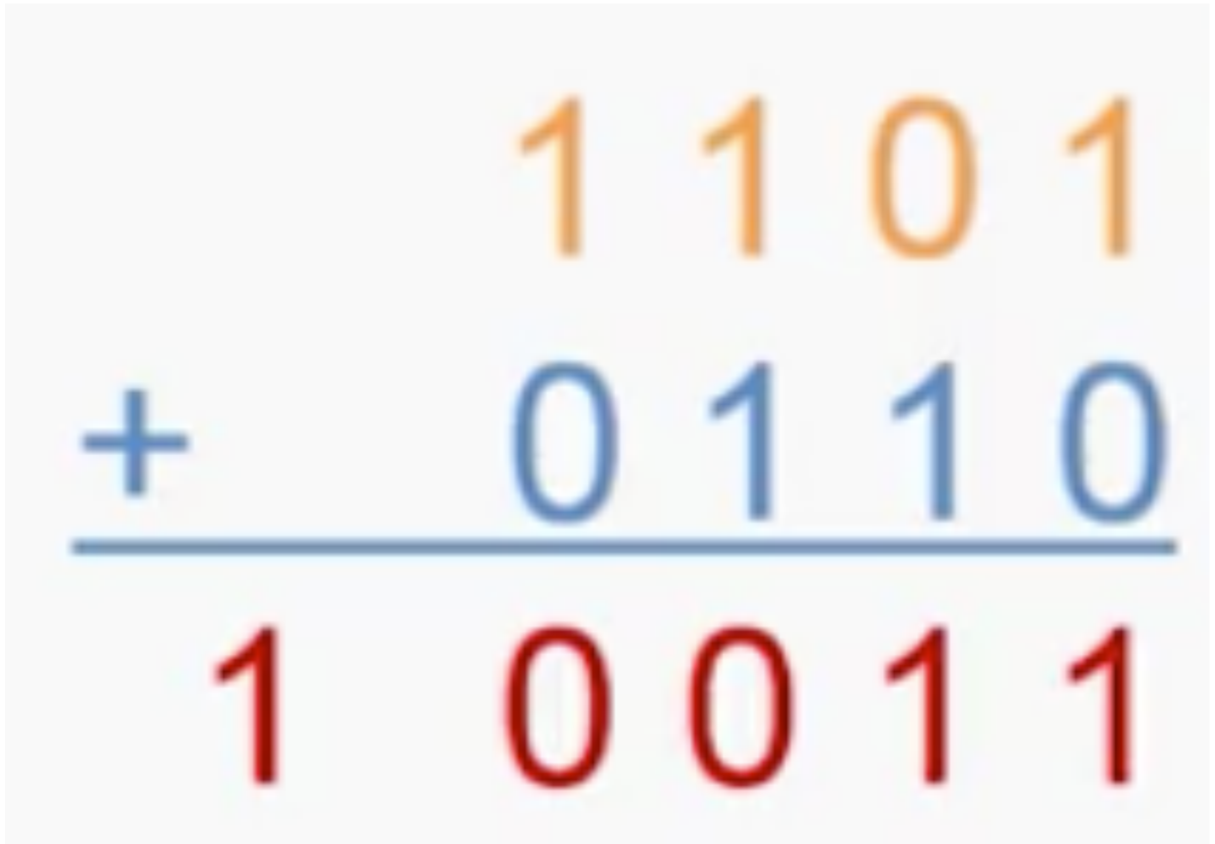




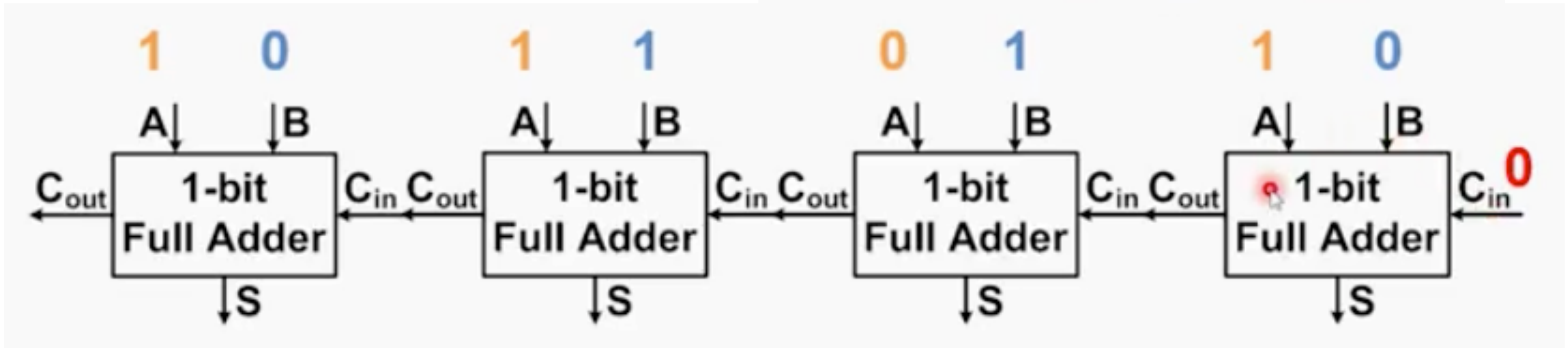
4-bit Full Adder

- How it works?

- Each bit assigned!

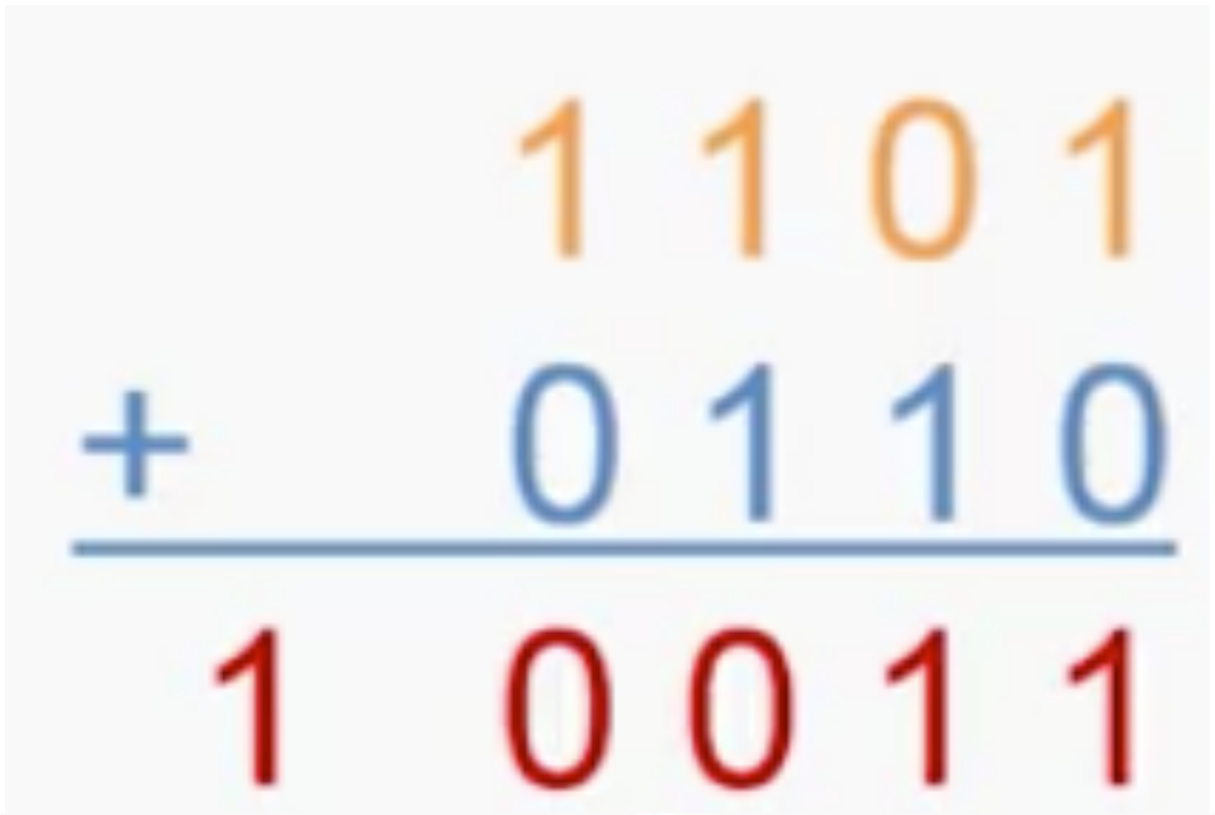


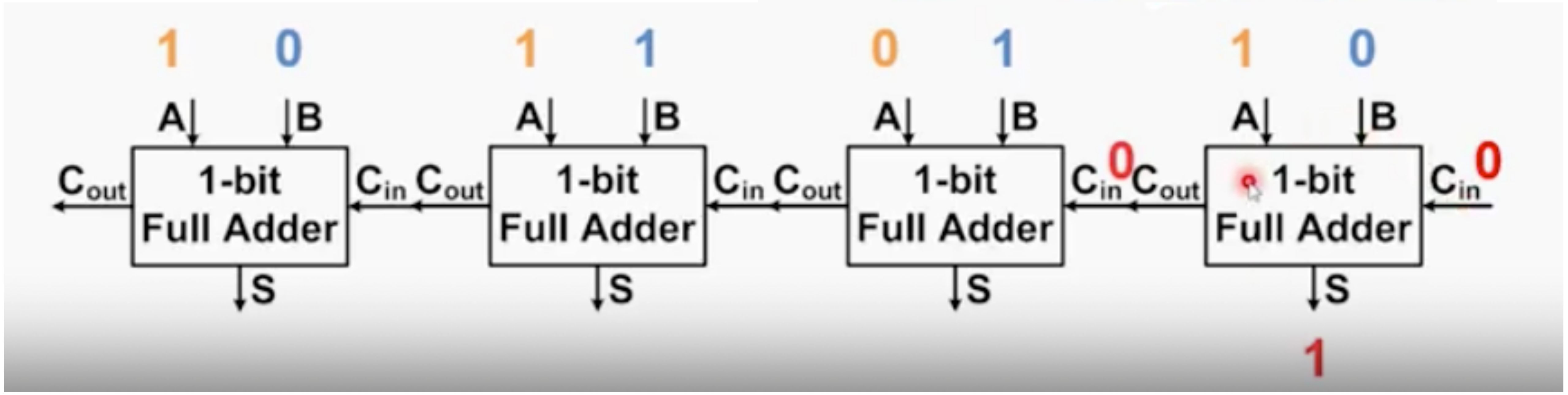
$$\begin{array}{r} 1\,1\,0\,1 \\ +\quad 0\,1\,1\,0 \\ \hline 1\;0\,0\,1\,1 \end{array}$$



4-bit Full Adder

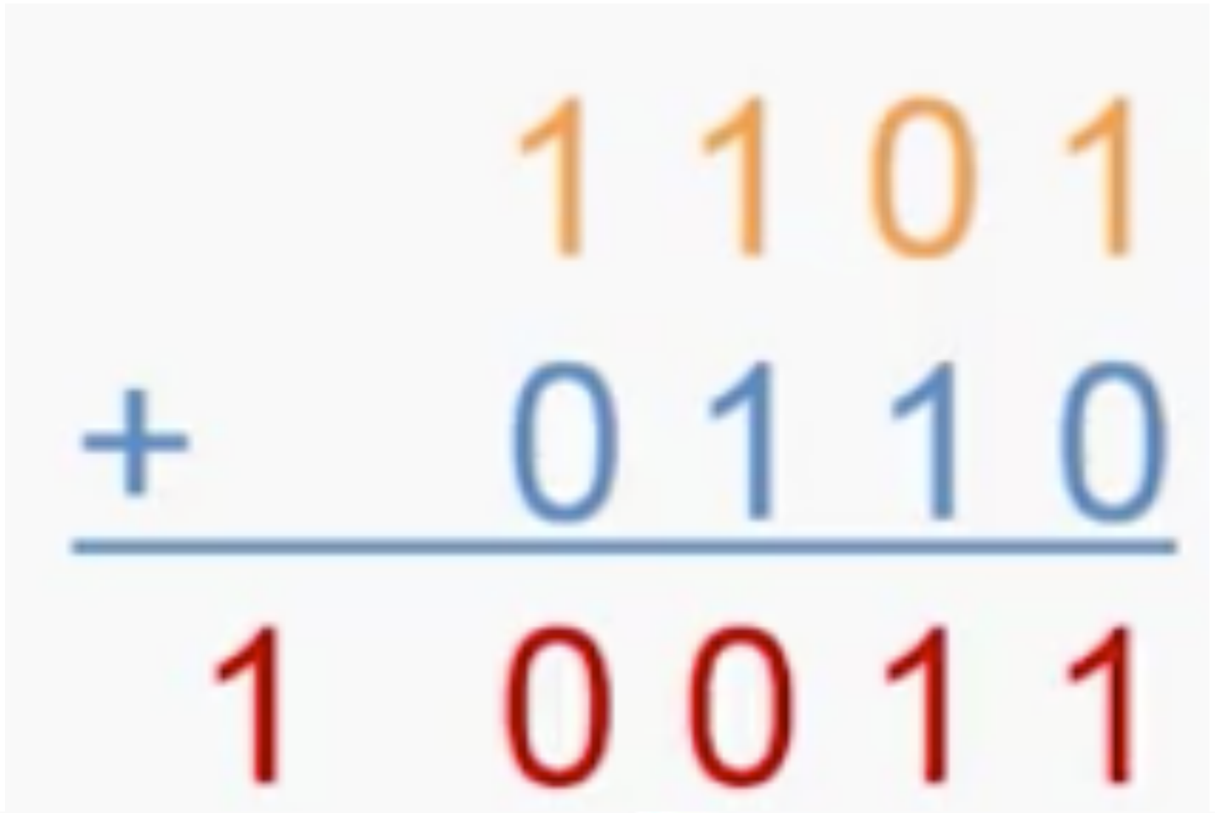
- How it works?
- Each bit assigned!

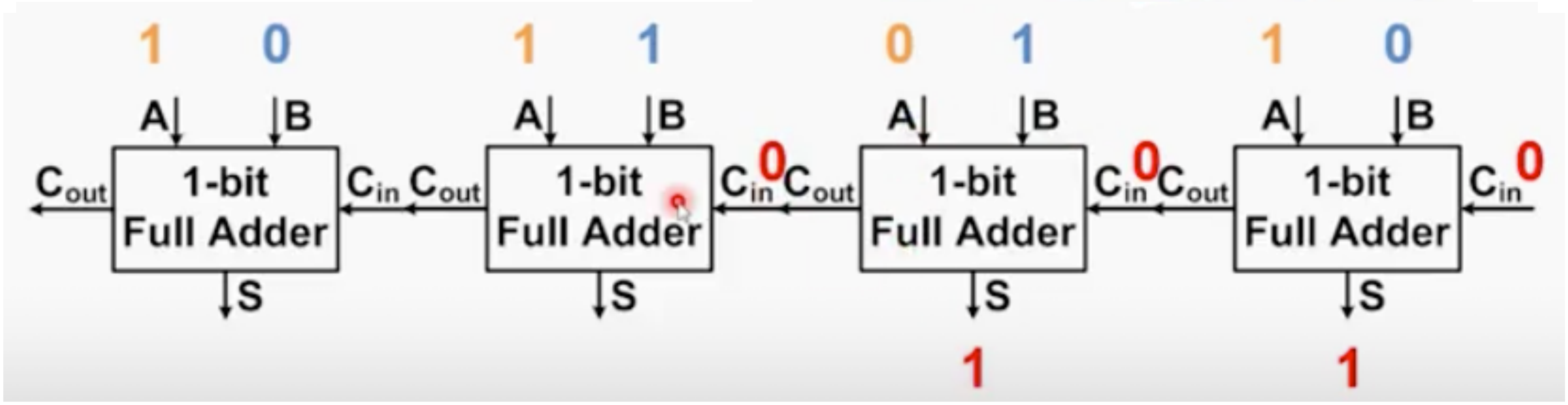




4-bit Full Adder

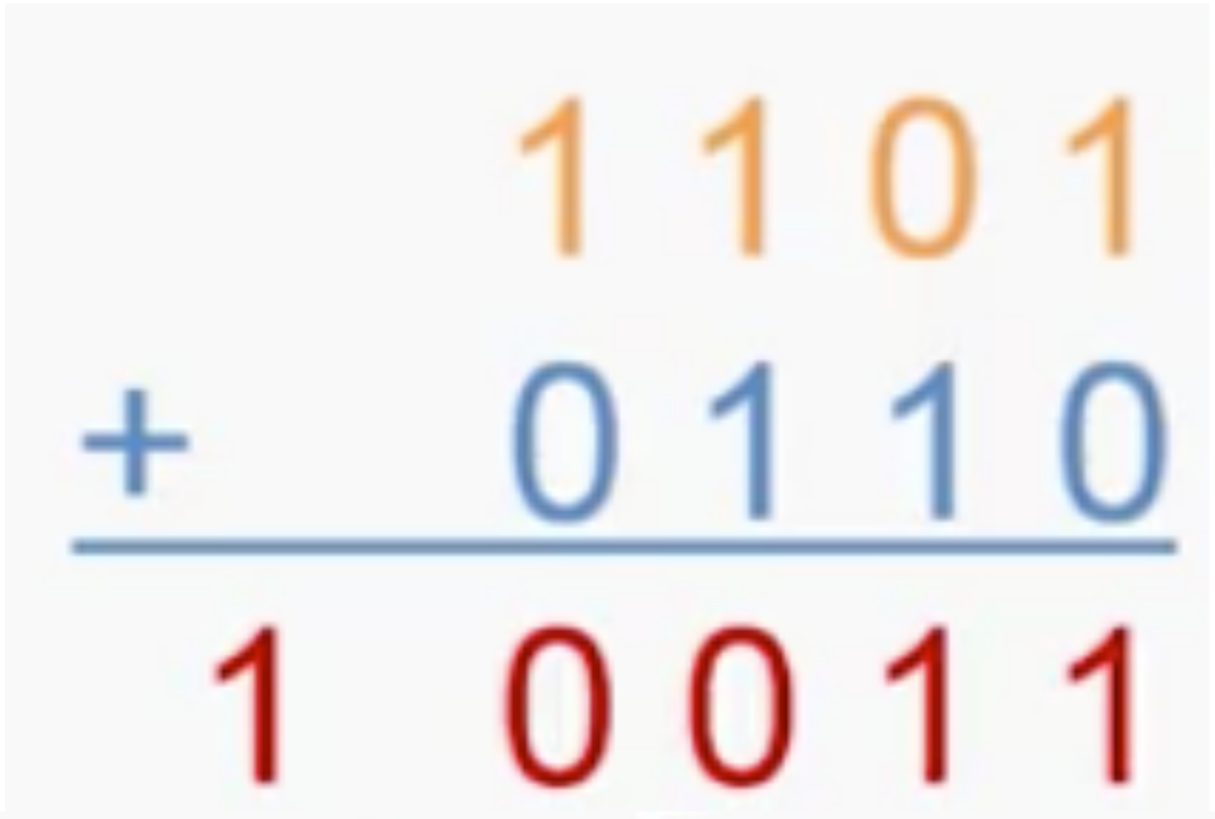
- How it works?
- Each bit assigned!

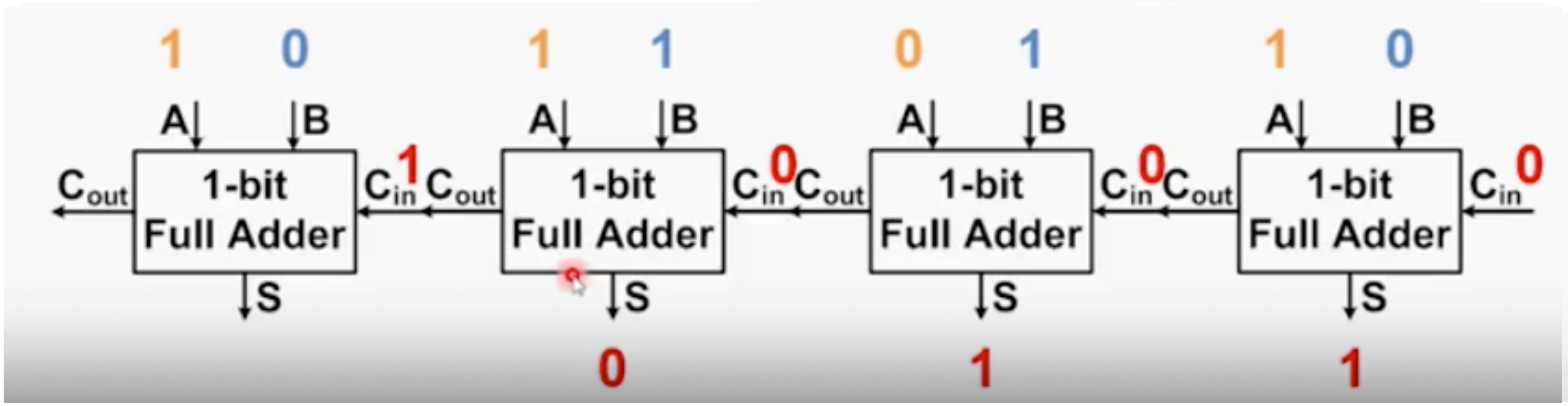




4-bit Full Adder

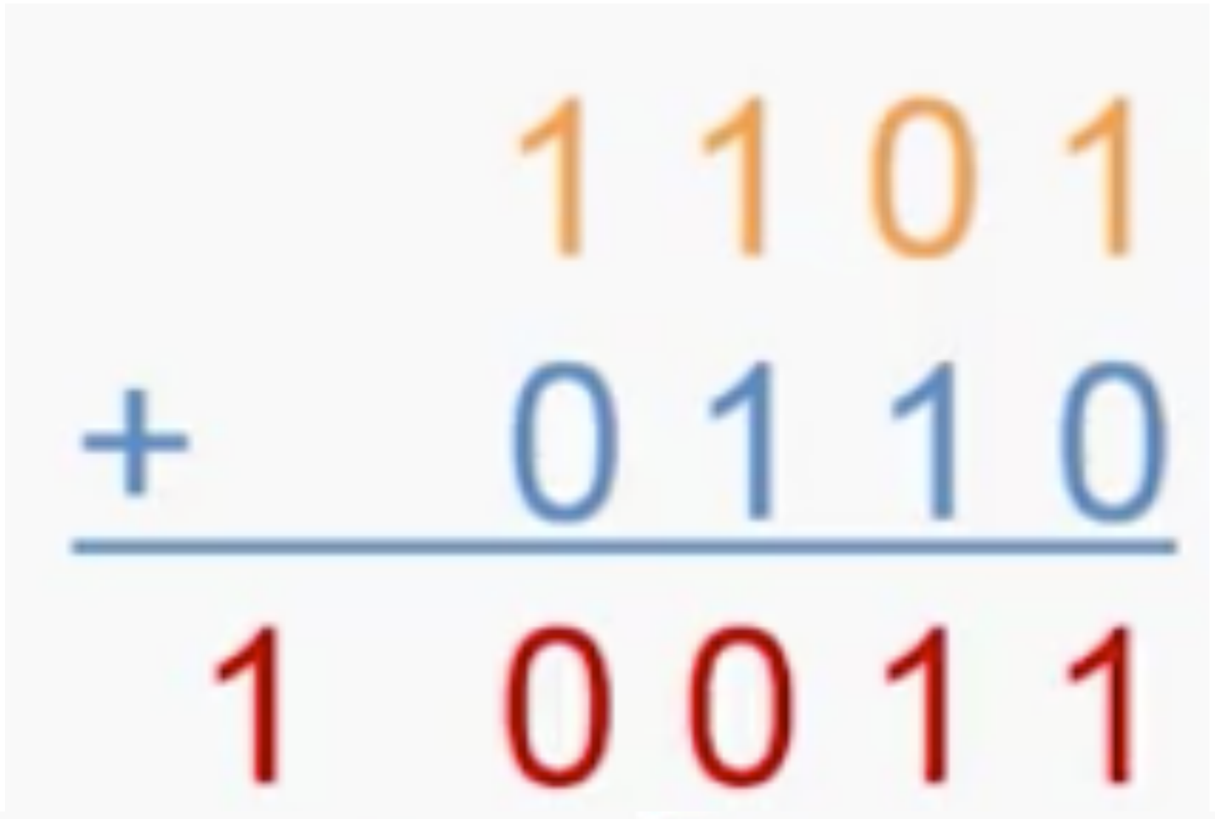
- How it works?
- Each bit assigned!

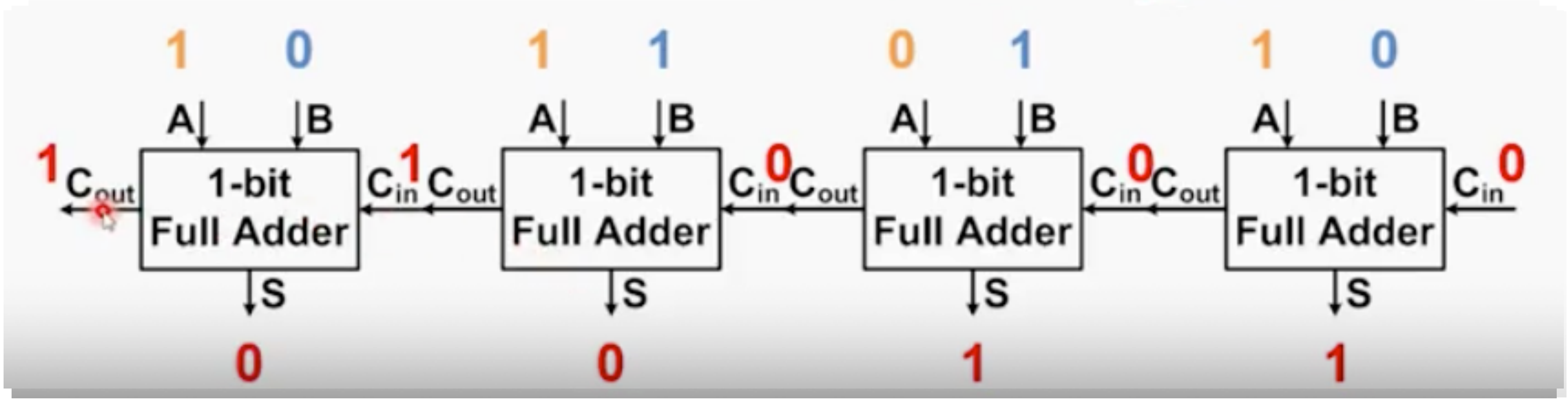




4-bit Full Adder

- How it works?
- Each bit assigned!





More bits Added

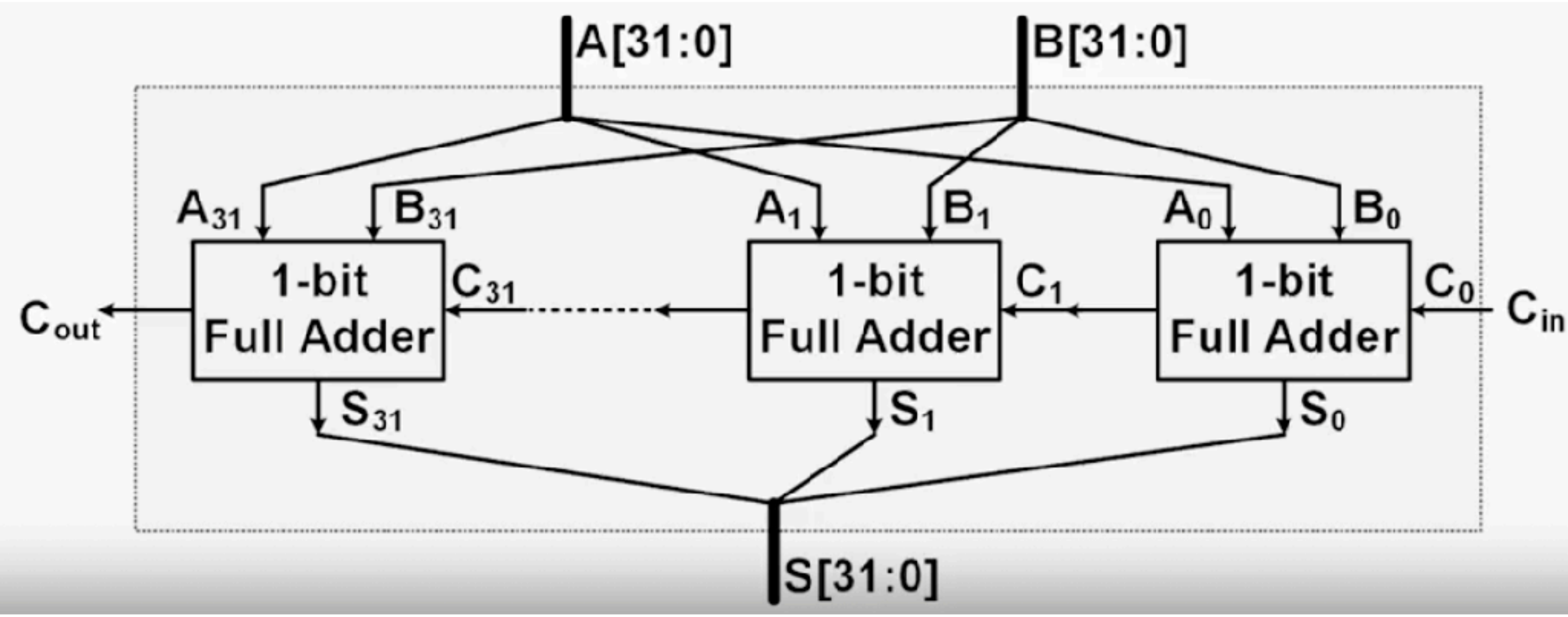
- 32-bit full adder

More bits Added

- 32-bit full adder
  - How to design?

More bits Added

- 32-bit full adder
  - How to design?



A[31:0]
B[31:0]
A_{31}
B_{31}
A_1
B_1
A_0
B_0
1-bit Full Adder
1-bit Full Adder
1-bit Full Adder
C_{out}
C_{31}
C_1
C_0
C_{in}
S_{31}
S_1
S_0
S[31:0]