Tri-State Logic
FPGAs

Andrew Lukefahr

Always specify defaults for **`always_comb`**!

BLOCKING (=) FOR
`always_comb`

NON-BLOCKING (<=) for
`always_ff`

UART RX/TX LEDs on Basys3

- A word of caution:
- The Basys3’s RX + TX LEDs are backwards from what you expect.
- They are the USB adaptor chip’s RX+TX, not the FPGAs.

Stack with RAMs

```
push( 4'b 0001)
```

head = 00, input = 0001, $Rd\overline{Wr}$ = 0, memEn = 1

head <= head + 1;

```
push( 4'b 0010)
```

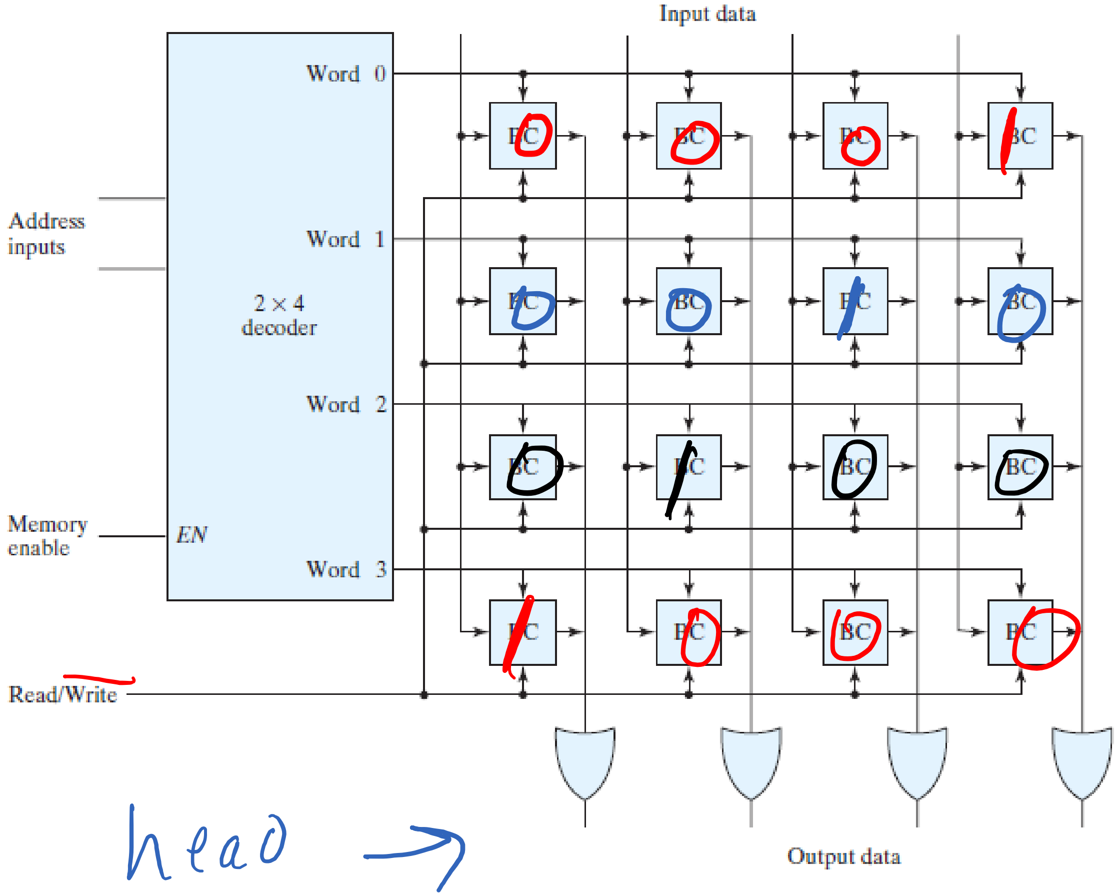
head = 01, input = 0010, $Rd\overline{WR}$ = 0, memEn = 1

head <= head + 1

```
push( 4'b 0100)
```

```
push( 4'b 1000)
```

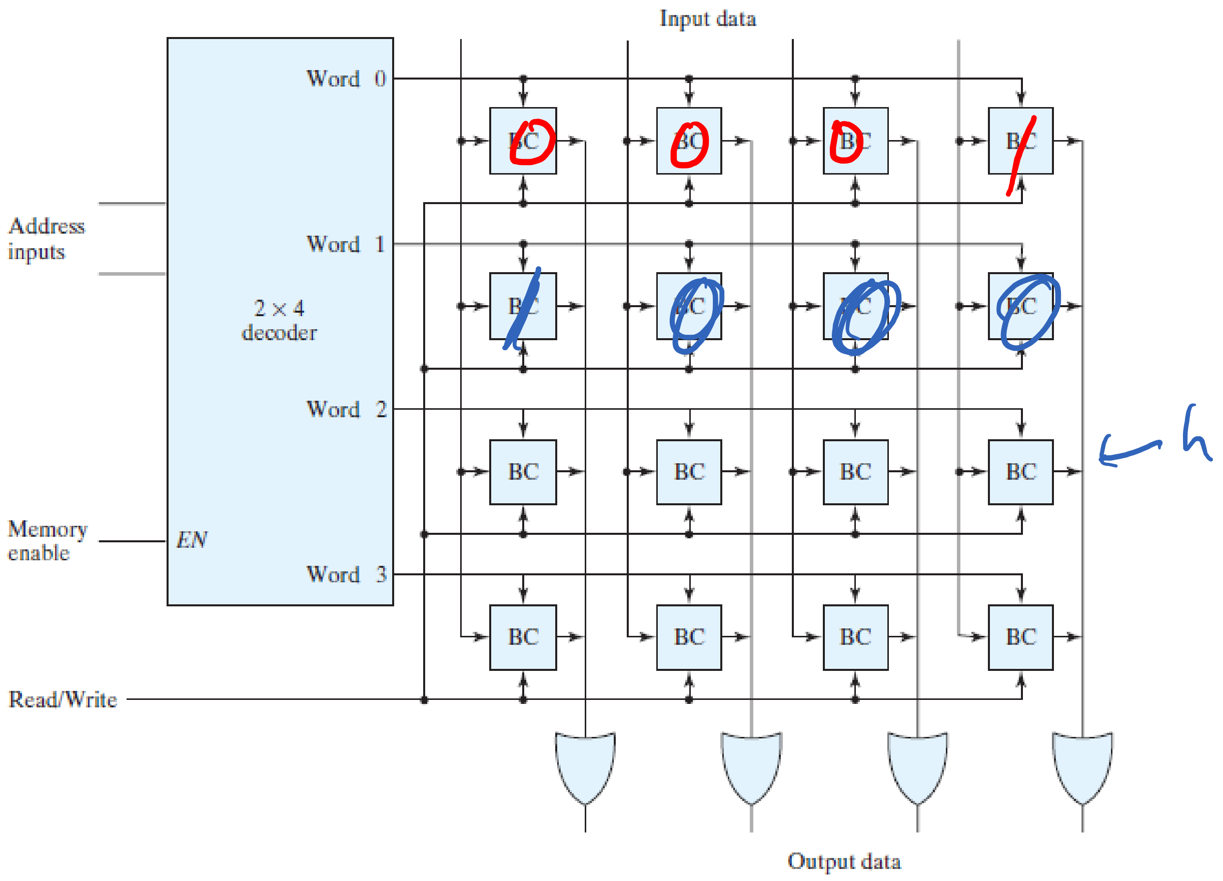
head =



head →
100

push/pop with RAMs

push(4’b 0001) ✓
push(4’b 0010) ✓
push(4’b 0100) ✓
pop() ⇒ 0100
pop() ⇒ 0010
push(4’b 1000) ✓
push(4’b 0011)
pop()
pop()
push(4’b 0110)
pop()



Busses

- Boolean Logic is bi-state:
  - 1: logical true
  - 0: logical false

  - X: The simulation tools don’t know if it’s 1 or 0

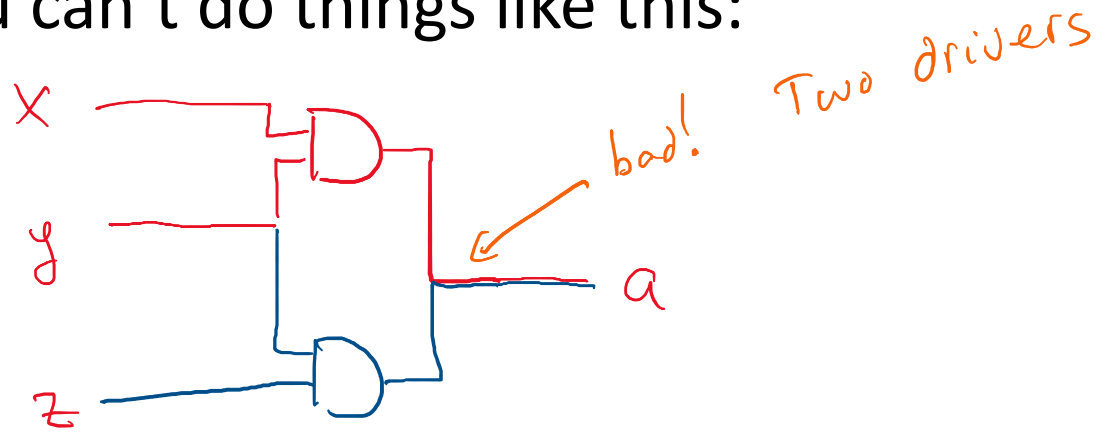
- So you can’t do things like this:

Busses

- Boolean Logic is bi-state:
  - 1: logical true
  - 0: logical false

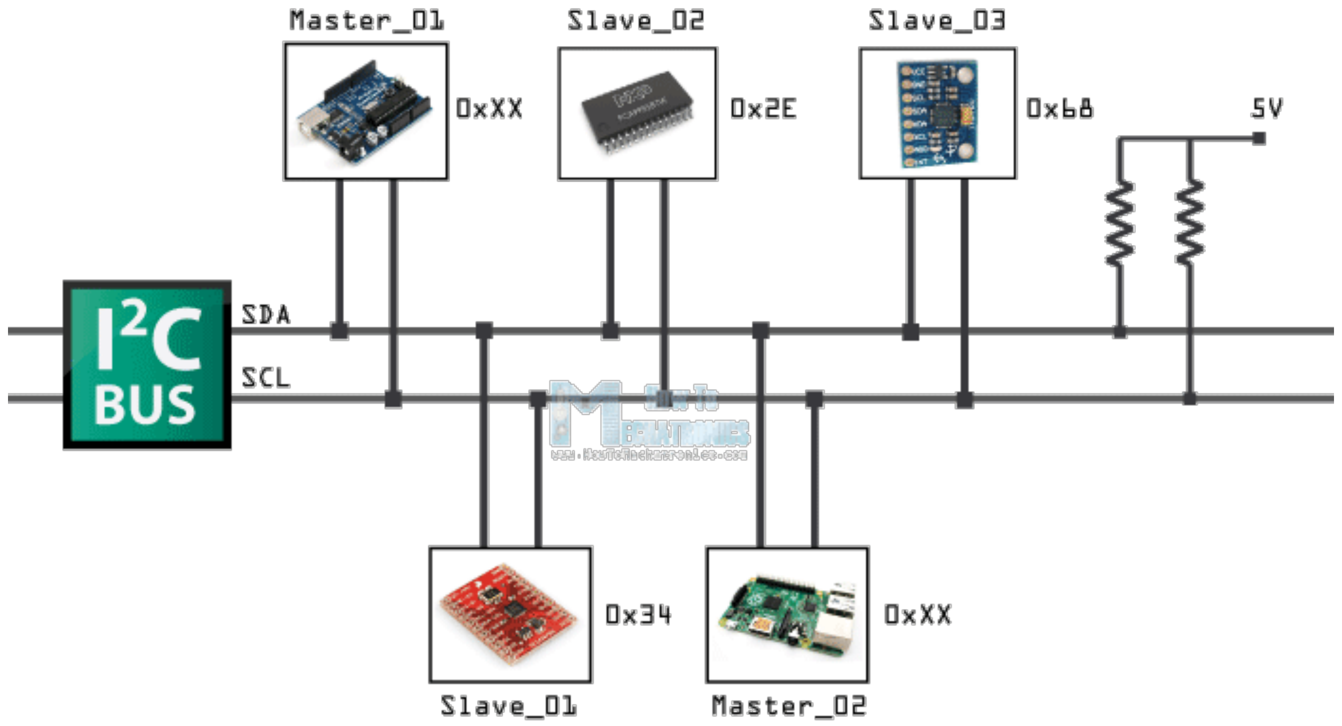
  - X: The simulation tools don't know if it's 1 or 0

- So you can't do things like this:



Question: Then how does I2C work?

- I2C: **Inter** - **Integrated** Circuit **Communication**



- **Clearly multiple "drivers" for 1 wire?**

Answer: A “Tri-State” Bus

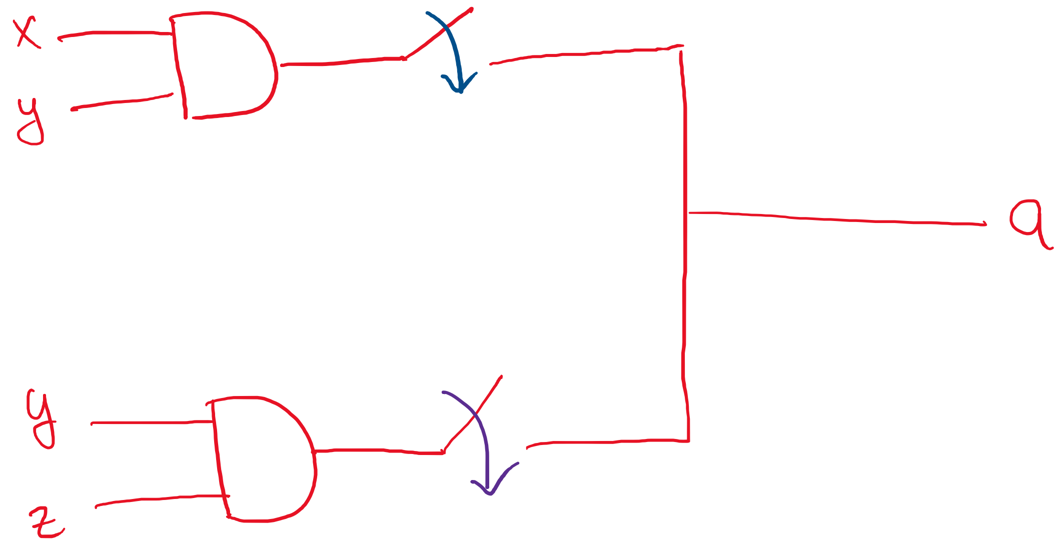
- “Tri-State” signals:
  - 1: this is logical true
  - 0: this is logical false
  - X: The simulation tools don’t know if it’s 1 or 0
  - Z: this is “high impedance”

- Z: High Impedance
  - Stop driving a logical value
  - Pretend I’m not connected

Tri-State logic

```
wire enable;
wire enabled_output = ‘h0;
wire tri;
assign tri = (enable ? enabled_output : ‘hZ);
```

Tri-State logic

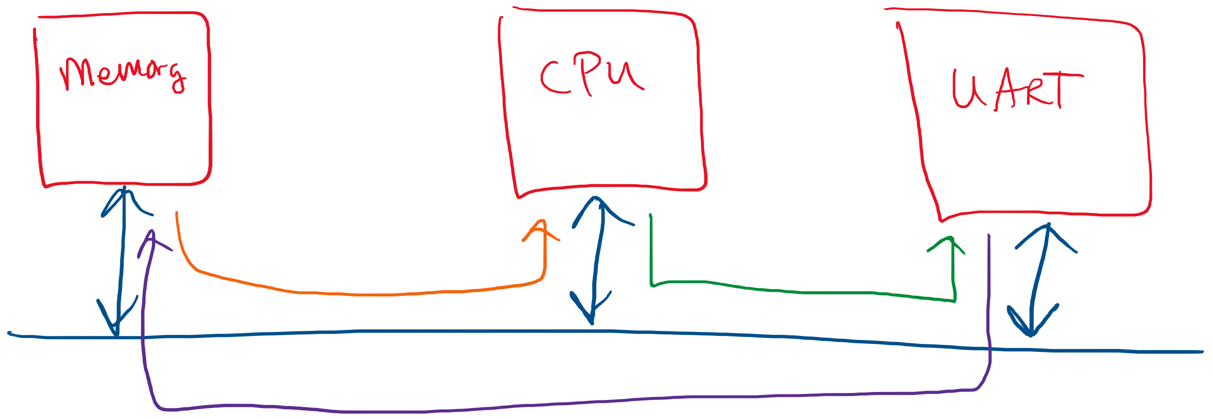


$= x \& y$

$= y \& z$

Tri-State Bus

Tri-State Bus

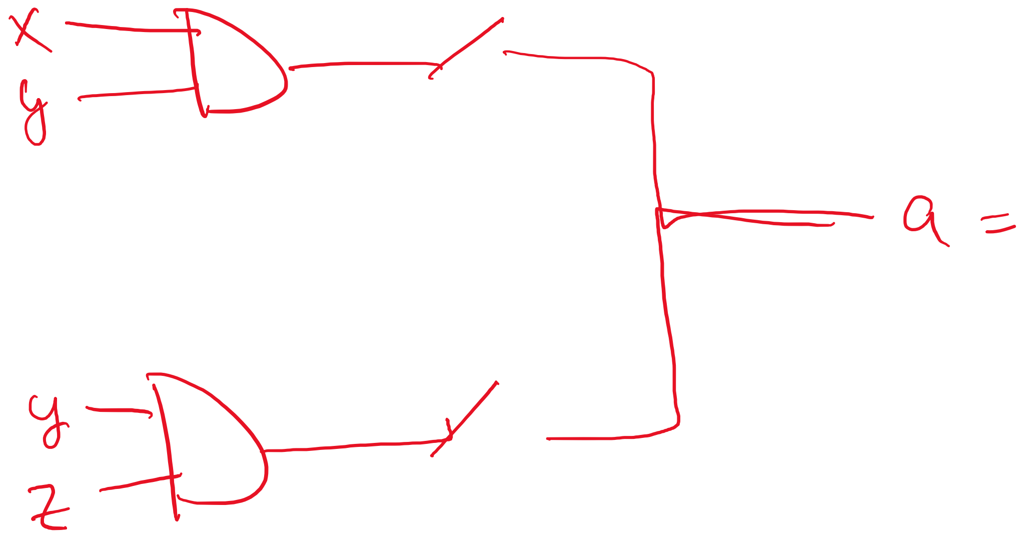


all on same bus

but not at same time

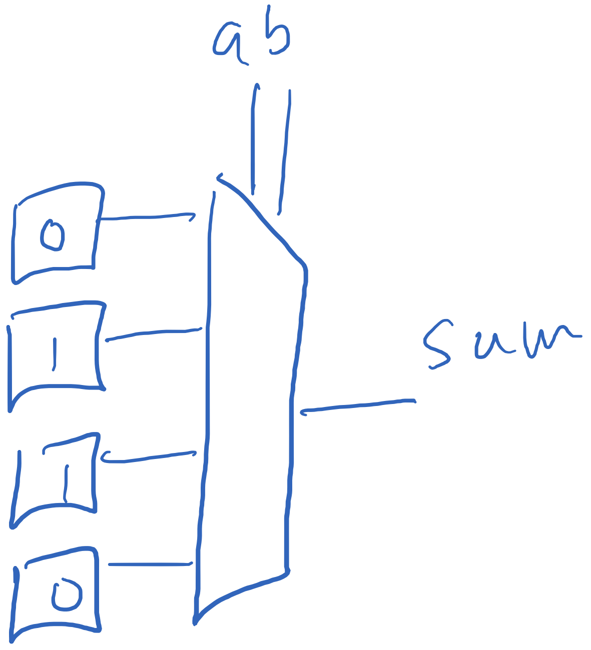
Problems with Tri-State Logic

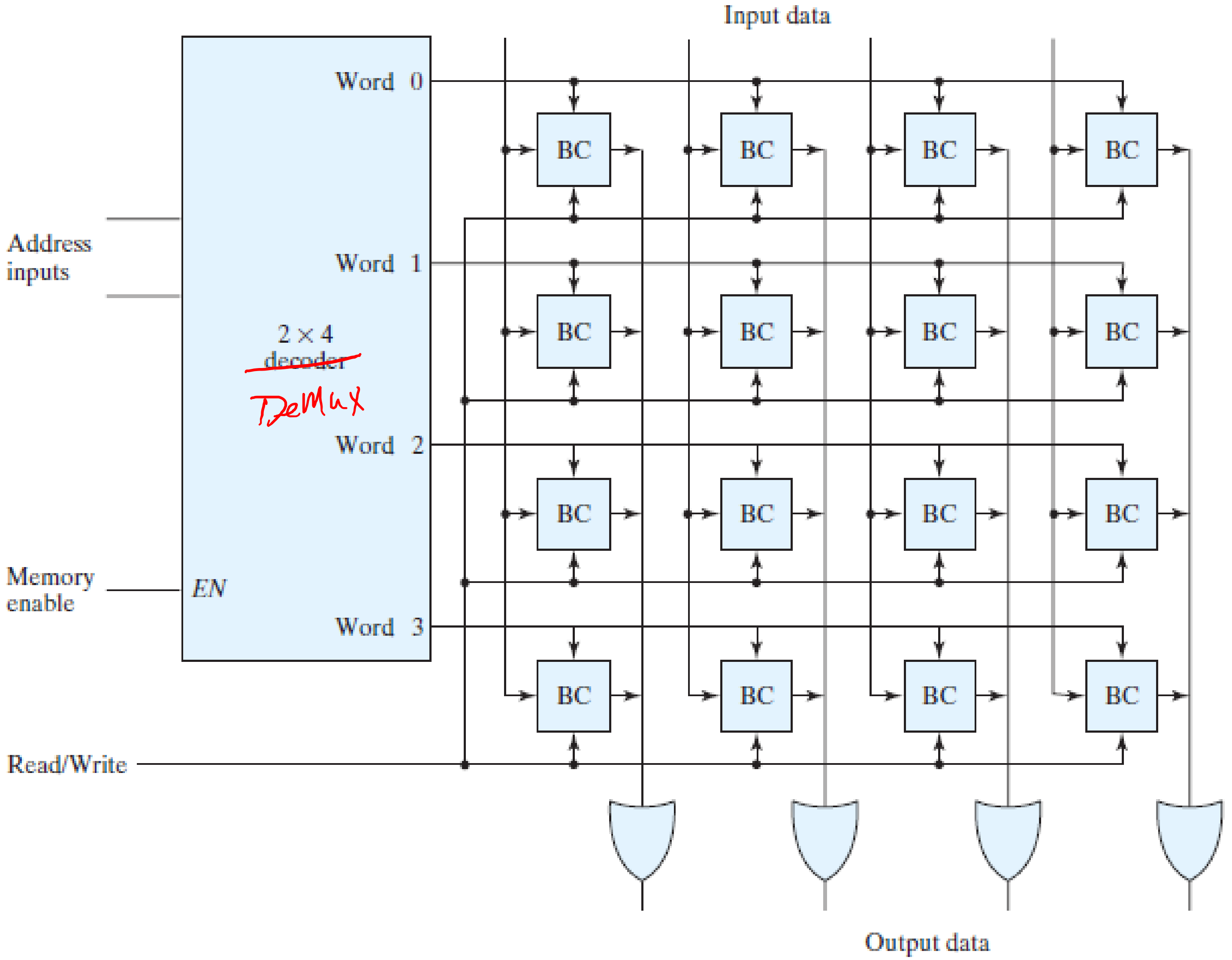
- What if two signals “drive” at once?



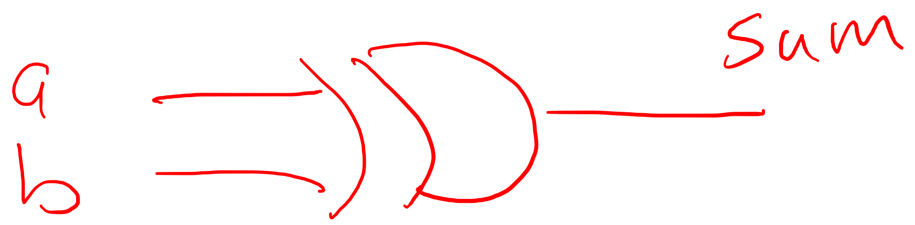
Solution: Don’t Do That!

Review: RAM


a b
0
1
1
0
sum


Input data
Word 0
Word 1
Word 2
Word 3
Address inputs
2 × 4
~~decoder~~
DeMux
EN
Memory enable
Read/Write
BC
Output data

Look-Up Table (LUT)

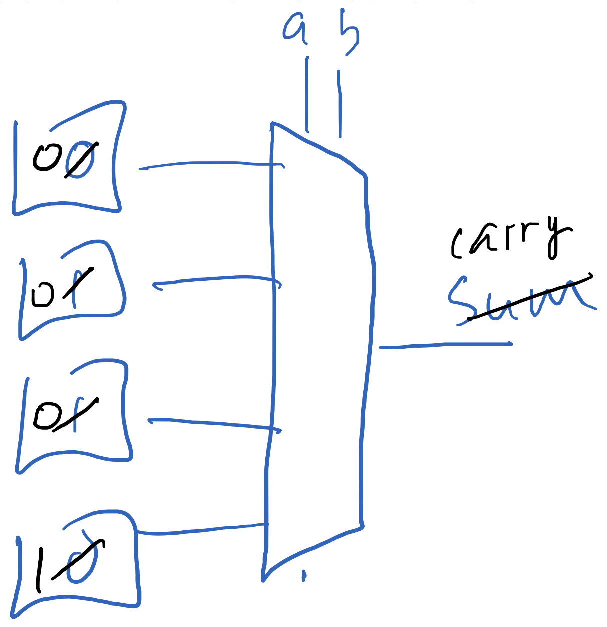


- DON'T compute a Boolean equation
- DO pre-compute all solutions in a table
- DO look up the Boolean result in the table

a	b	Sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- Examples:

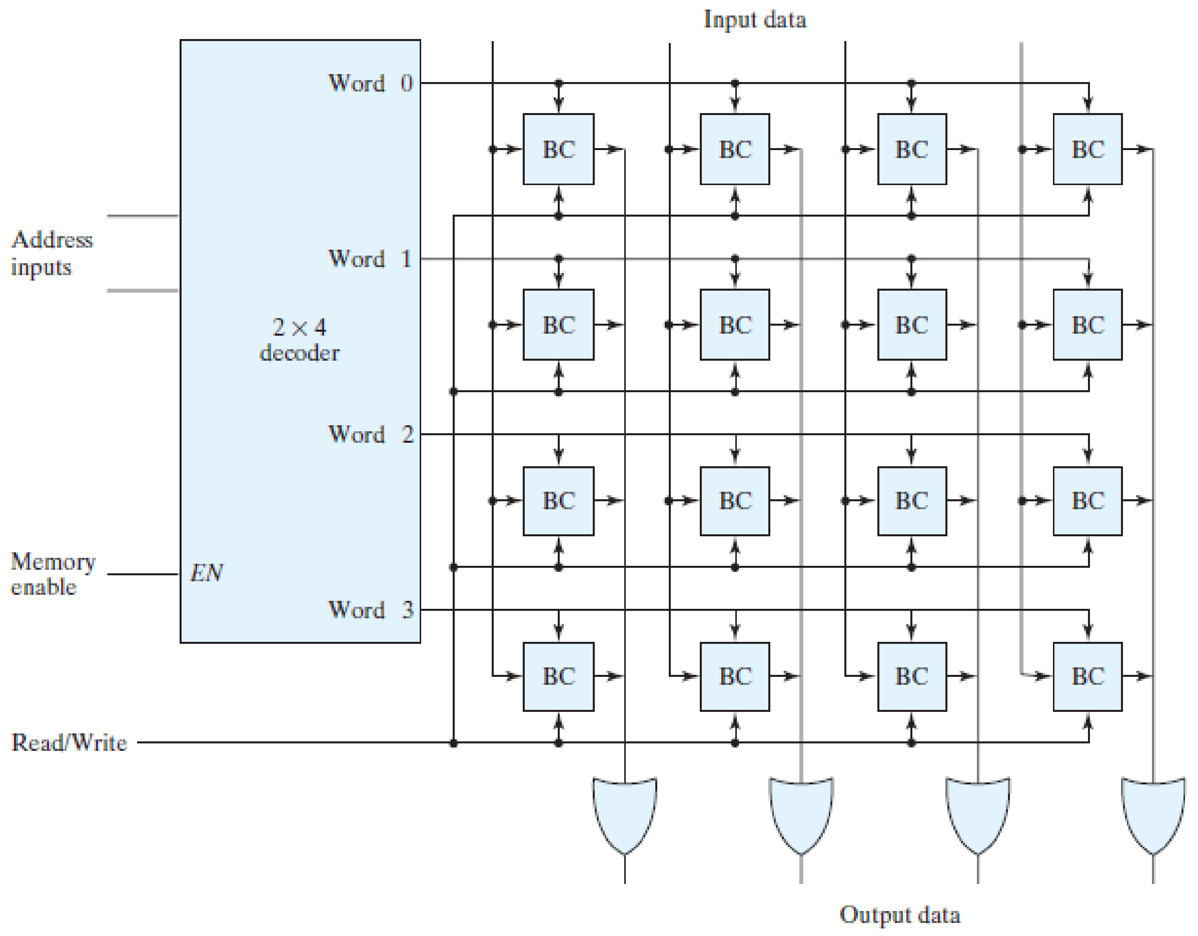
```
s = a ^ b;
c = a & b;
```



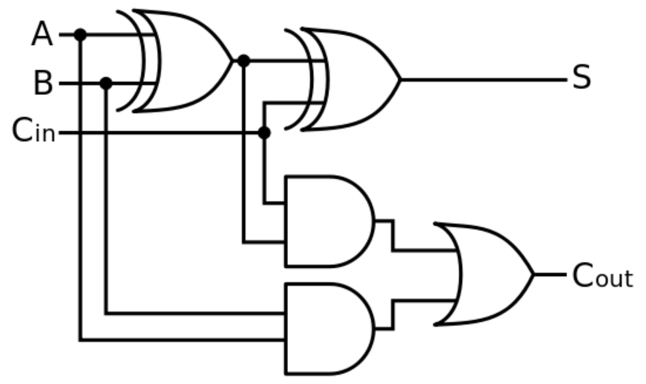
RAM to LUT

- Can I use a RAM to build a Half-Adder LUT?

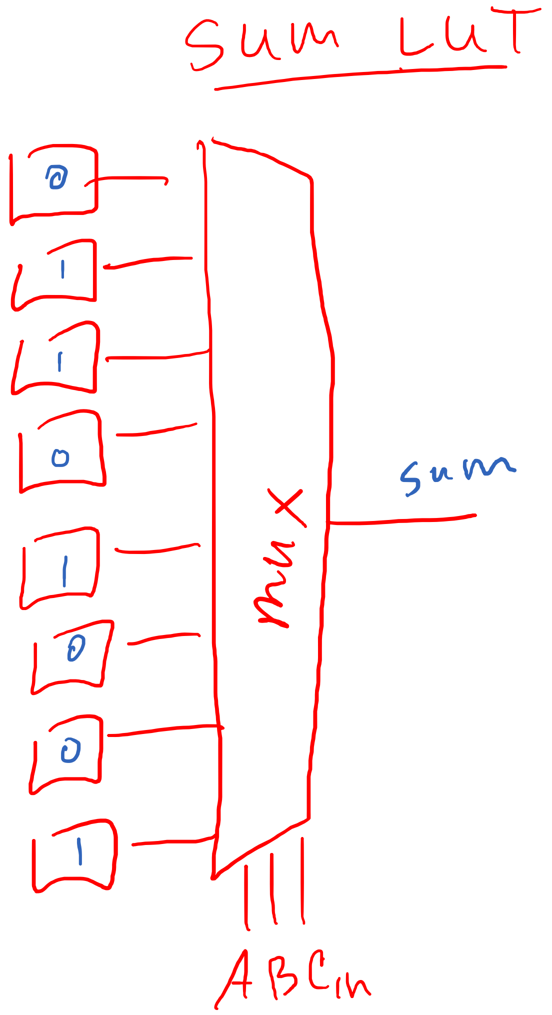
```
s = a ^ b;
c = a & b;
```

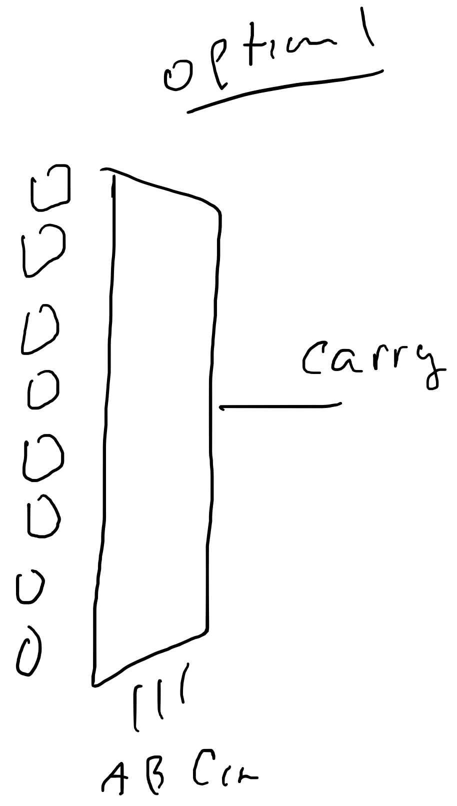


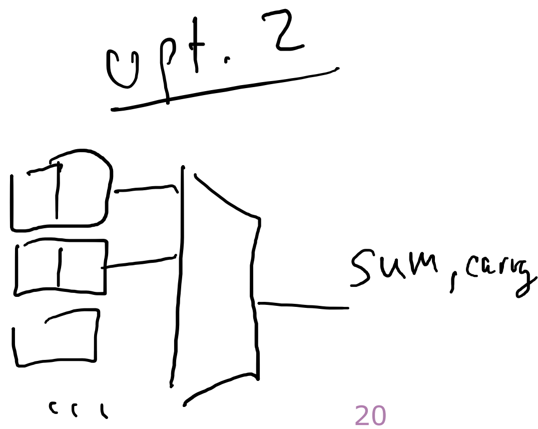
Full-Adder LUT



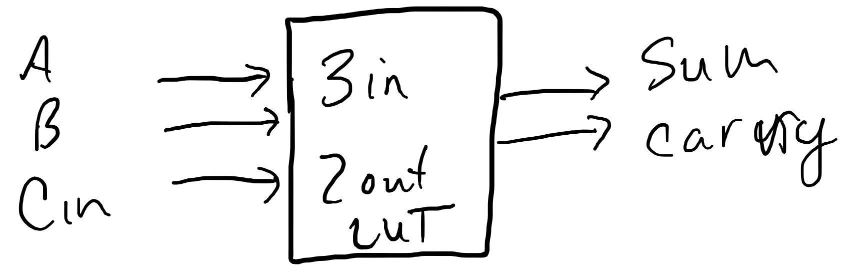
Input			Output	
A	B	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

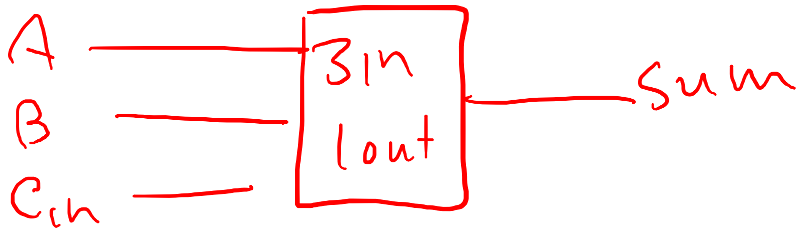


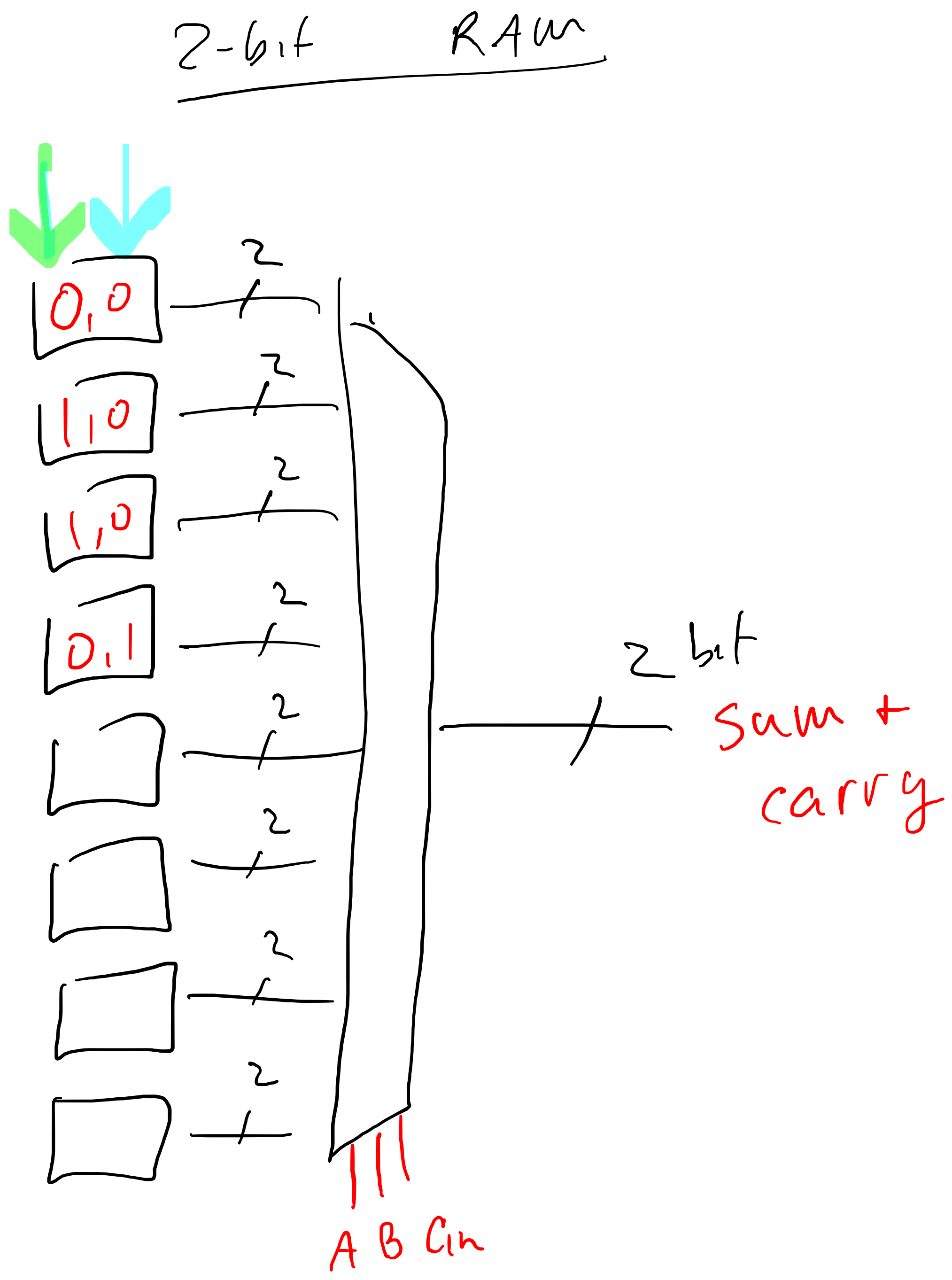




N-input, M-output LUT







1 output

2 output

LUT size

3 ~~inp~~

- Why not a 1000-input,100-output LUT?

- 3 inputs => 2^3 rows = 8 rows ← 3in, 1 out
- 4 inputs => 2^4 rows = 16 rows ← 4in, 1 out
- 5 inputs => 2^5 rows = 32 rows
- …
- 64 inputs => 2^{64} rows = 1.85 x10^{19} rows

- LUT input size does **not** scale well.

$8 \cdot 2 = 16$

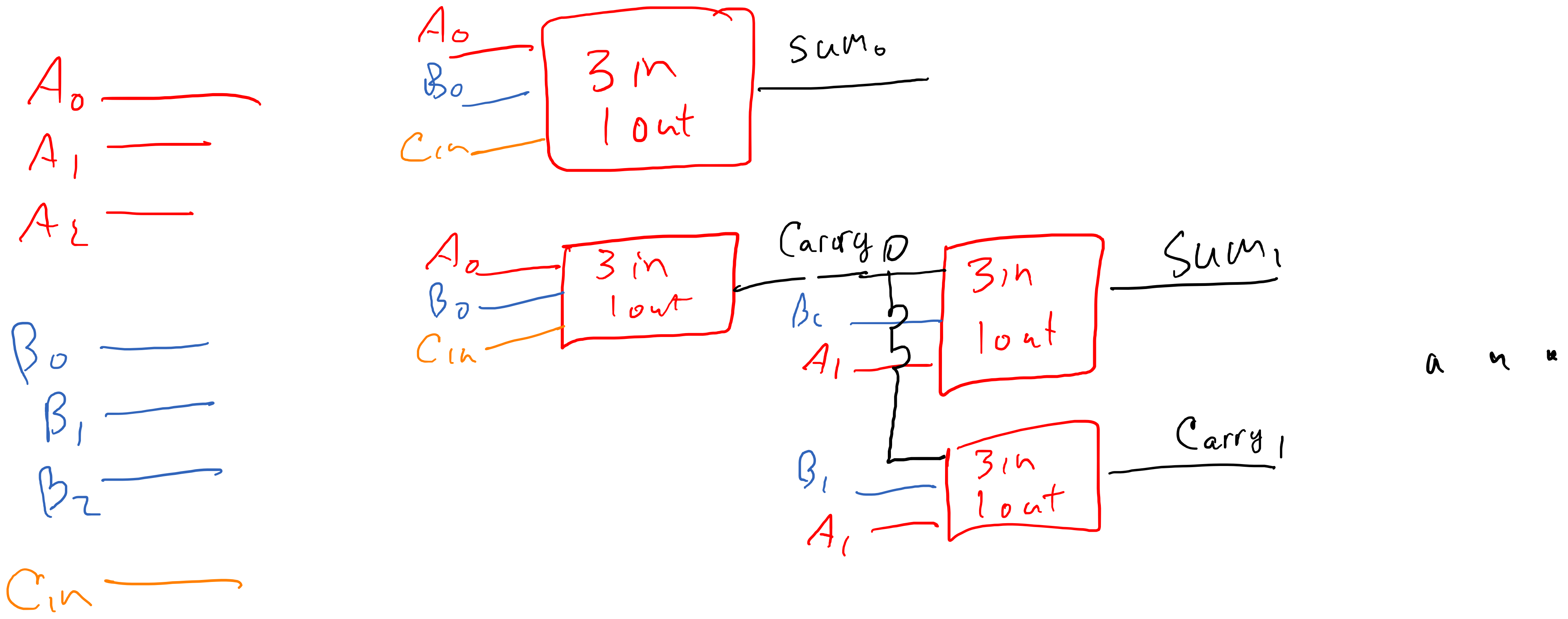
$16 \cdot 2 = 32$

$32 \cdot 2 = 64$

...

Divide and Conquer with LUTs

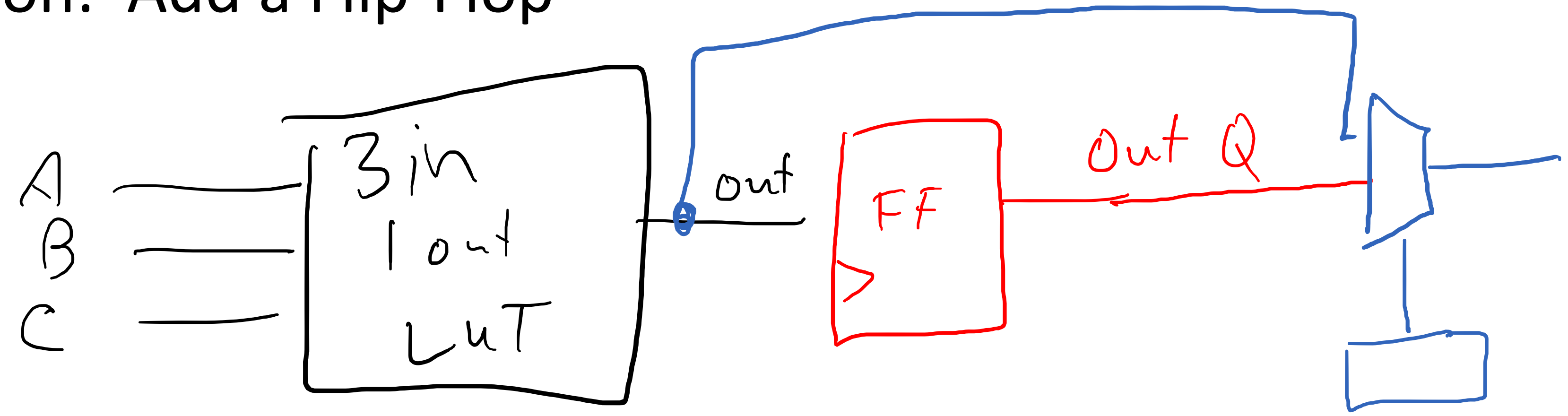
- 3-Bit Full Adder



Sequential Logic

- Problem: How do we handle sequential logic?
  - LUTs cannot contain state

- Solution: Add a Flip-Flop



Basic Logic Element (BLE)

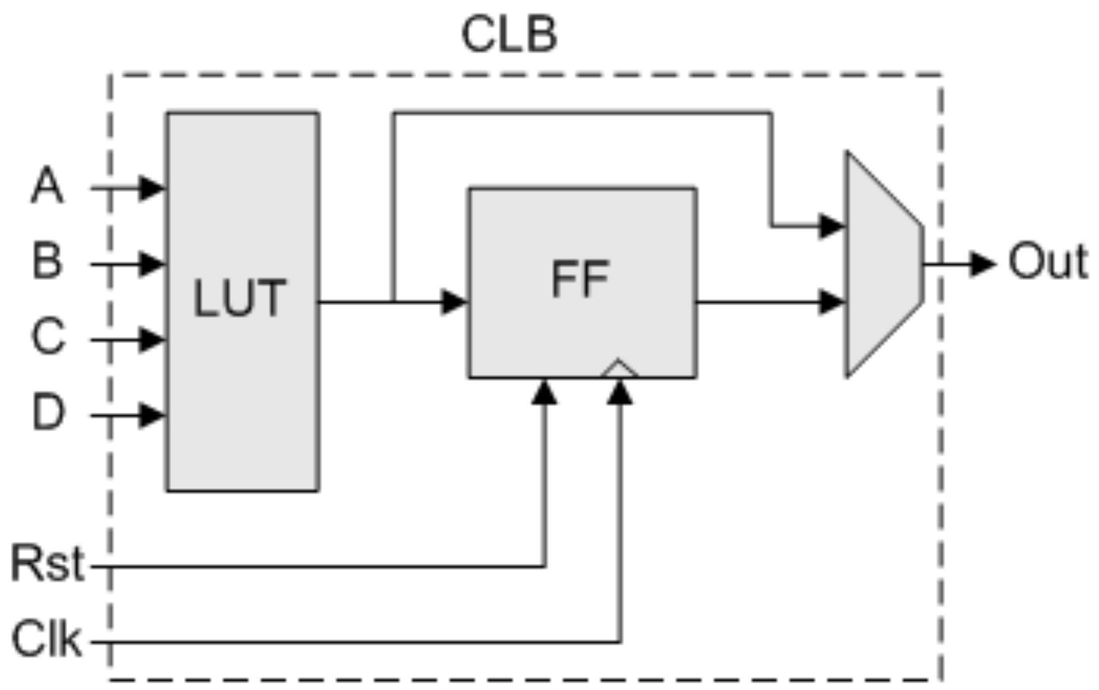


Fig.4 Example of Configurable Logic Cell.

Basic Logic Element (BLE)

- What if I only want to store a value?

A	B	C	Z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

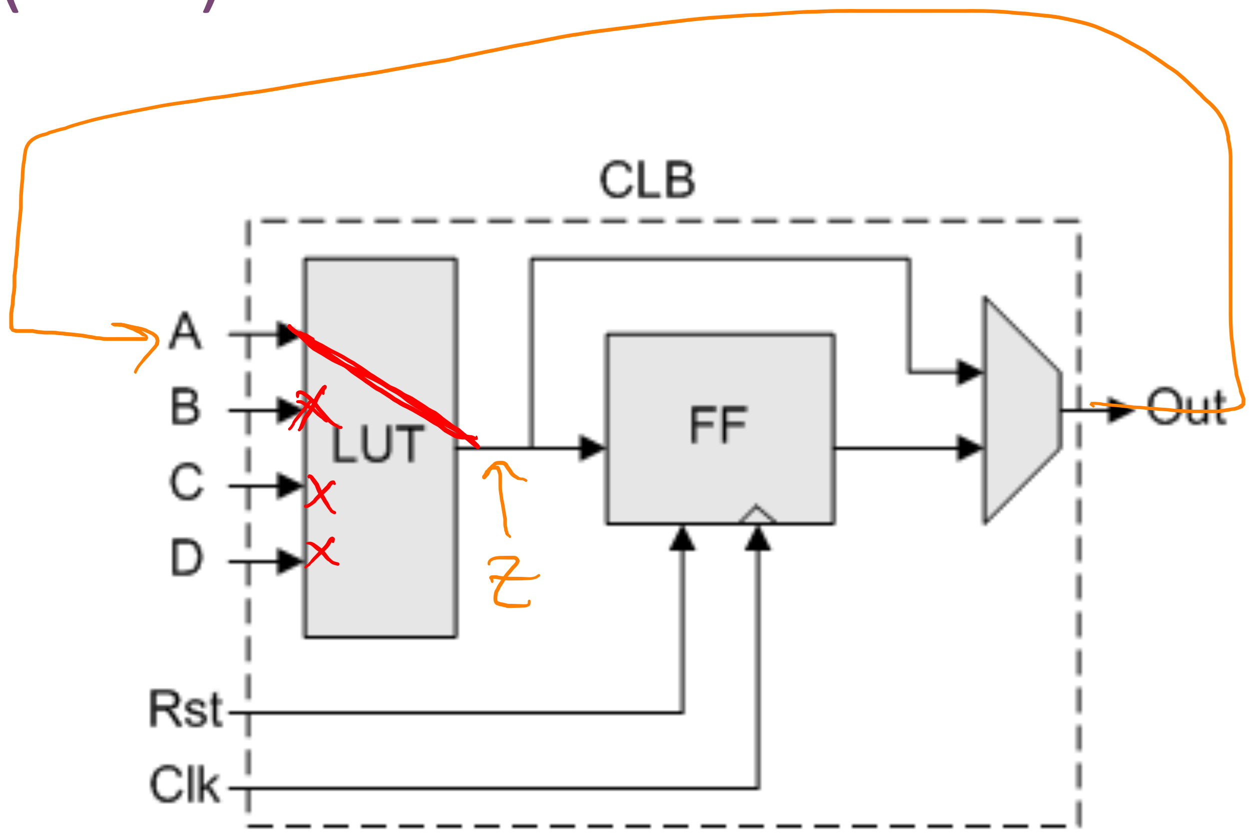
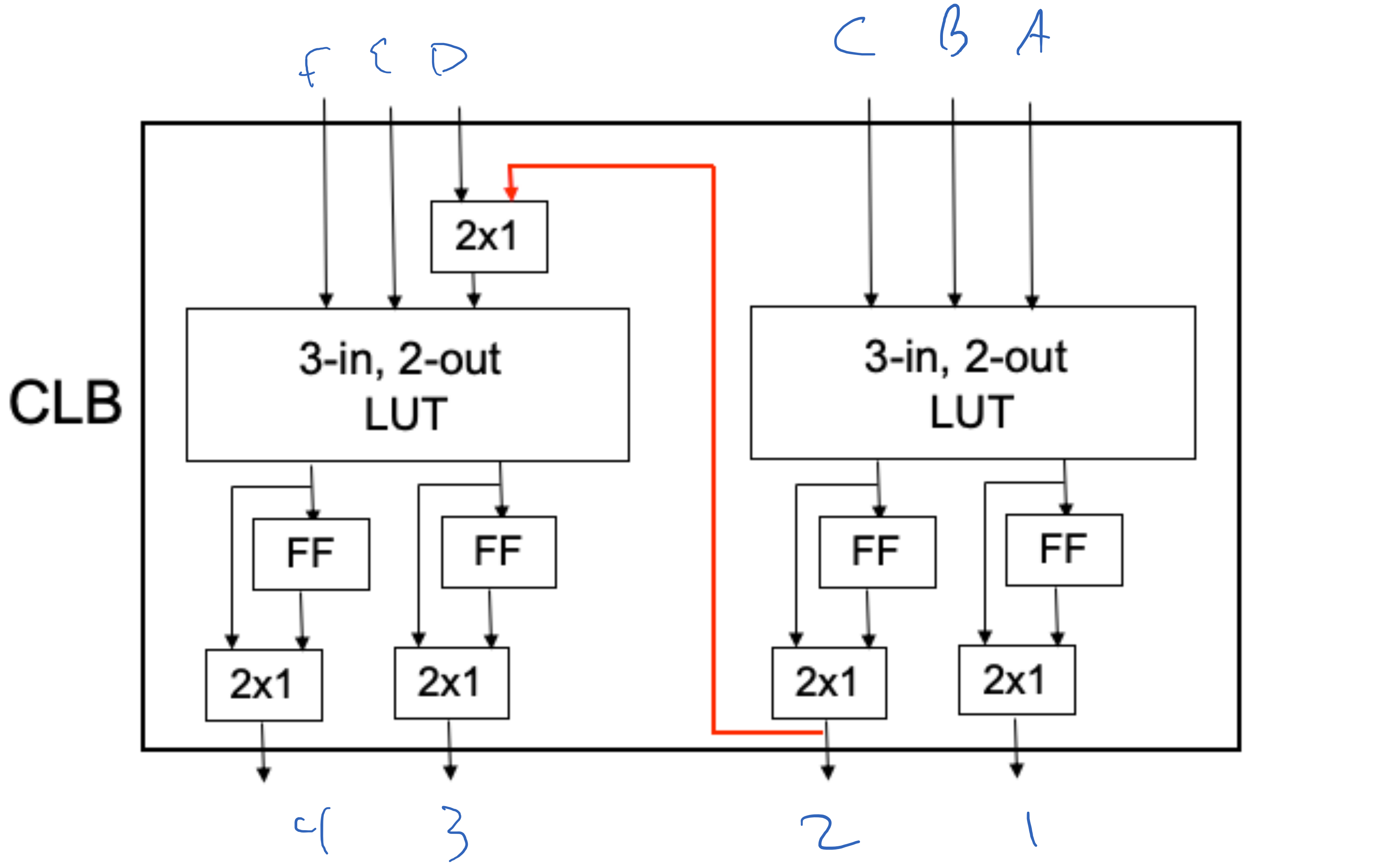


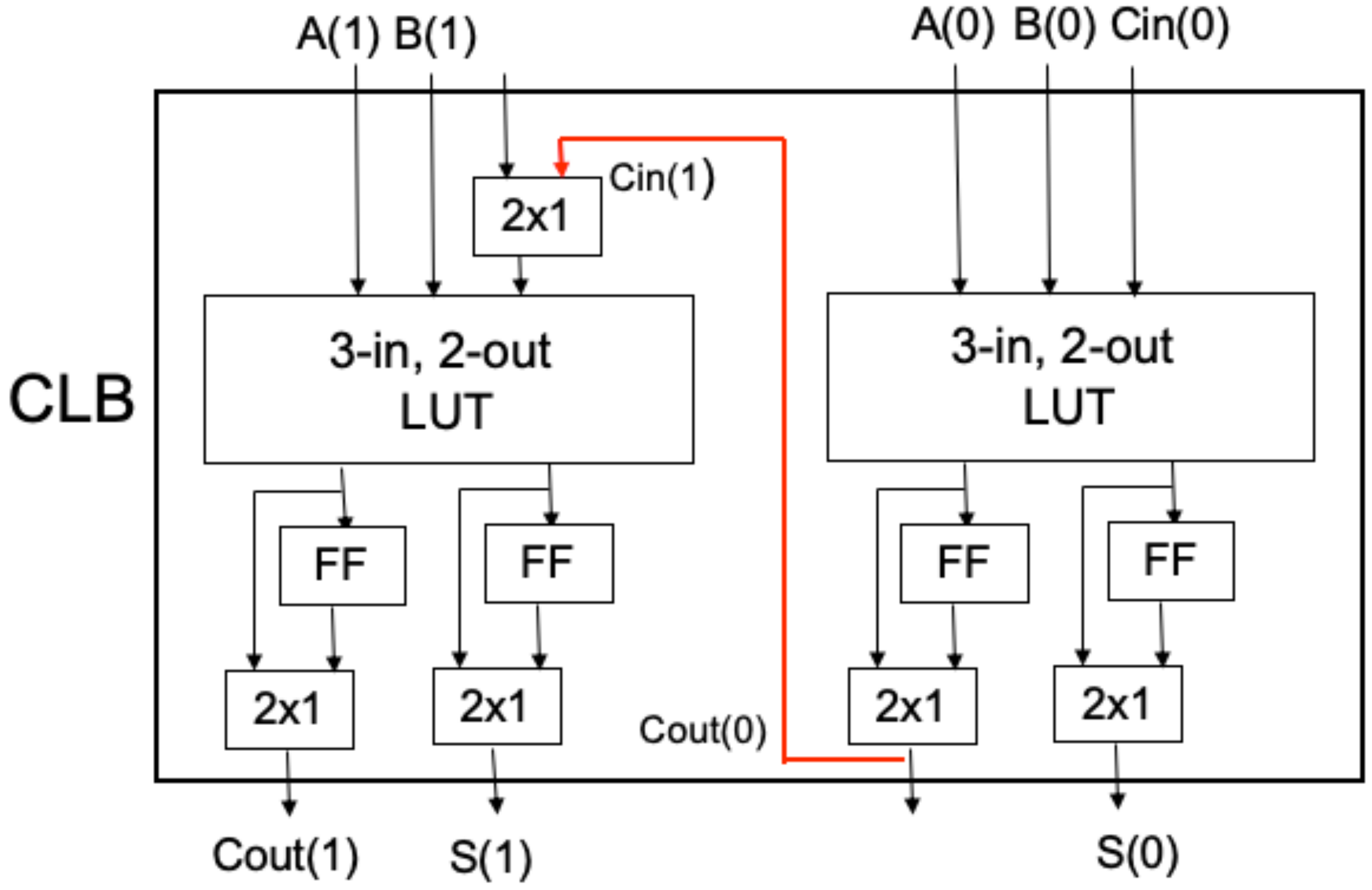
Fig.4 Example of Configurable Logic Cell.

6 input, 4 output CLB

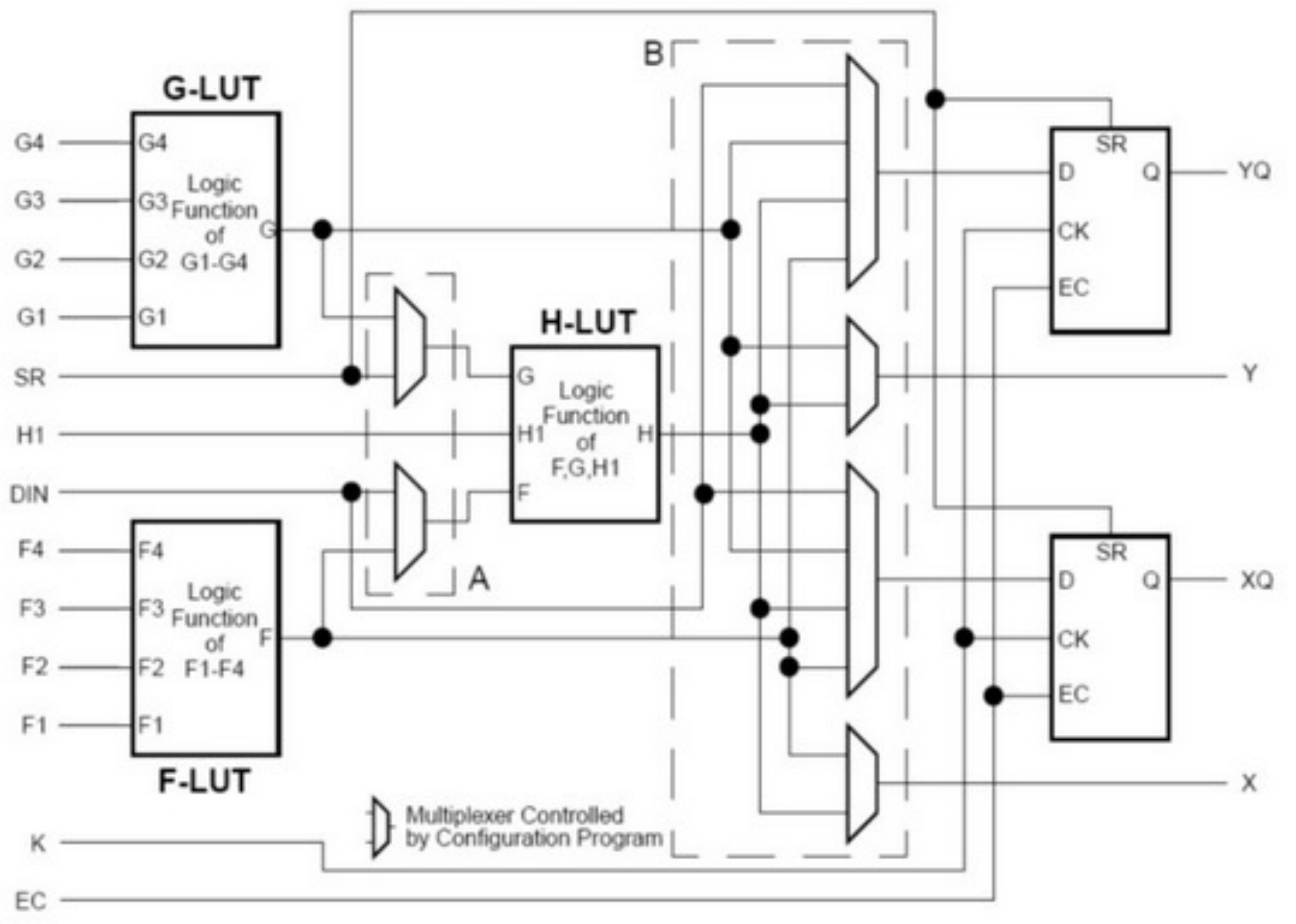
Improved BLE



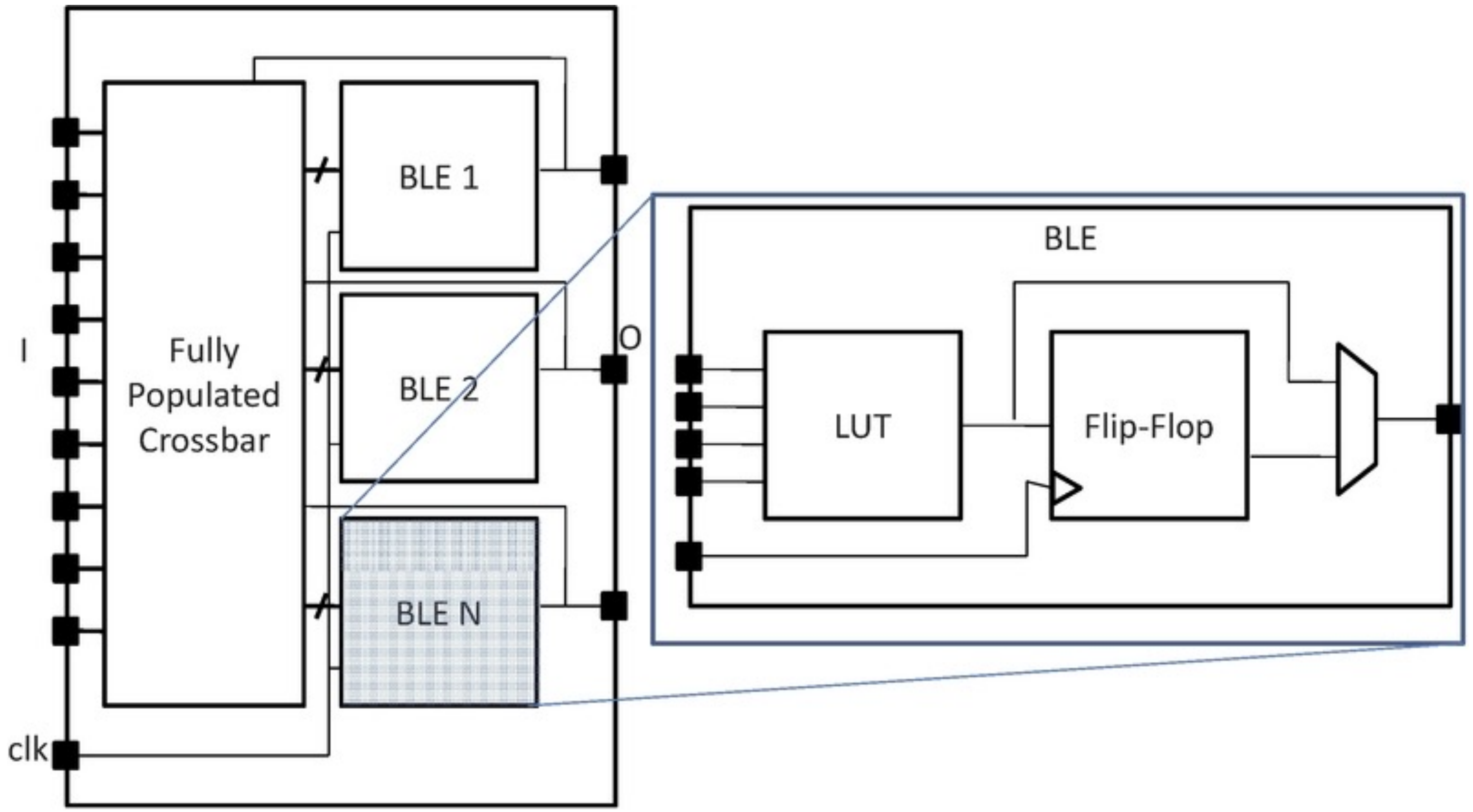
2-Bit Ripple-Carry w/ BLE



Realistic BLE: Xilinx

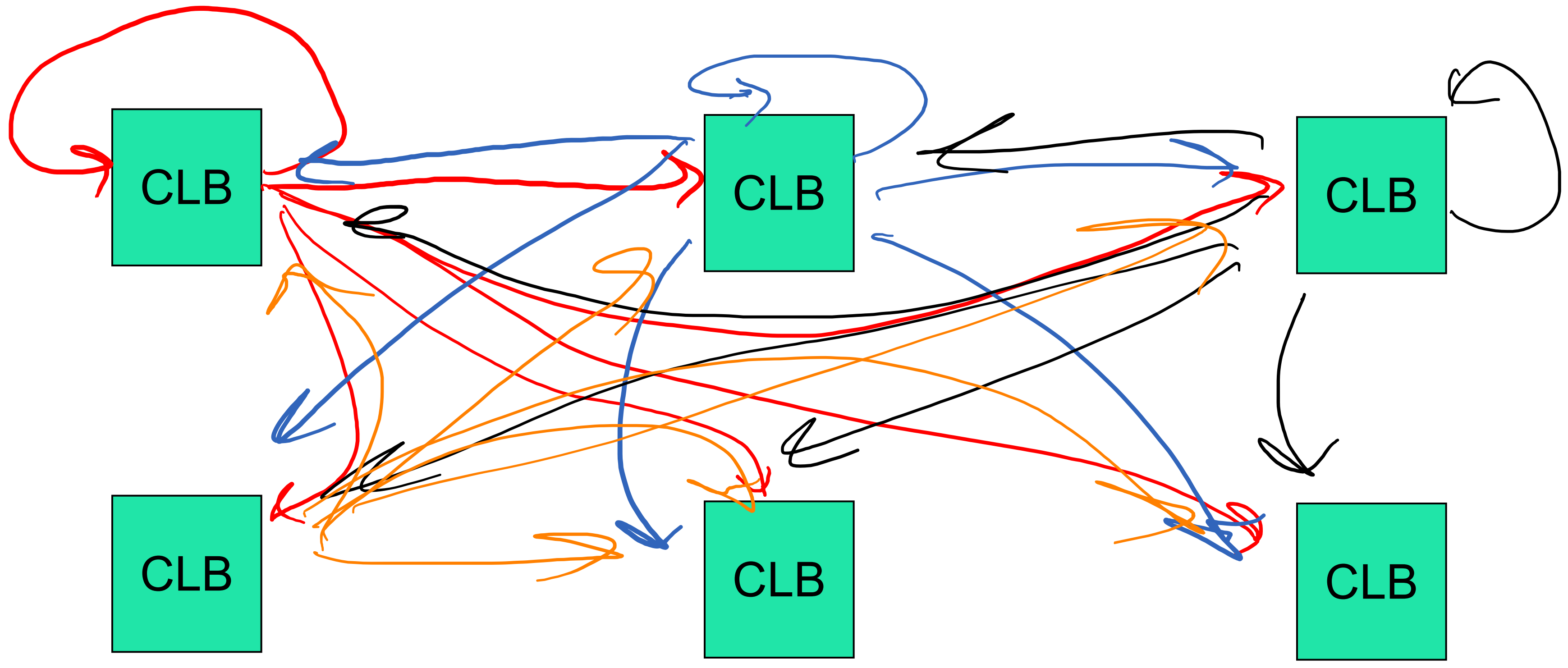


CLBs: Configurable Logic Block



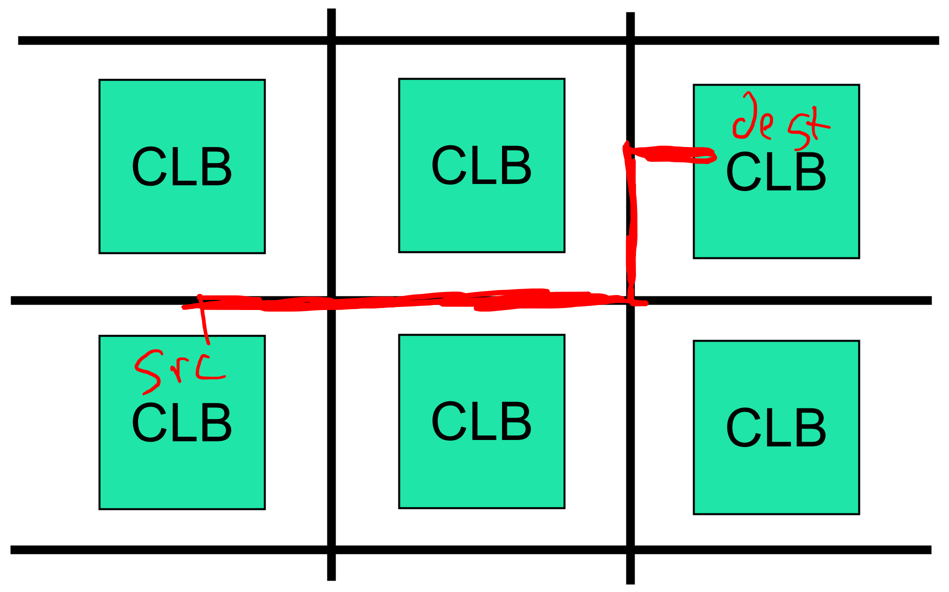
Connecting CLBs

- Q: How do CLBs talk to each other?
- A: Put wires everywhere!



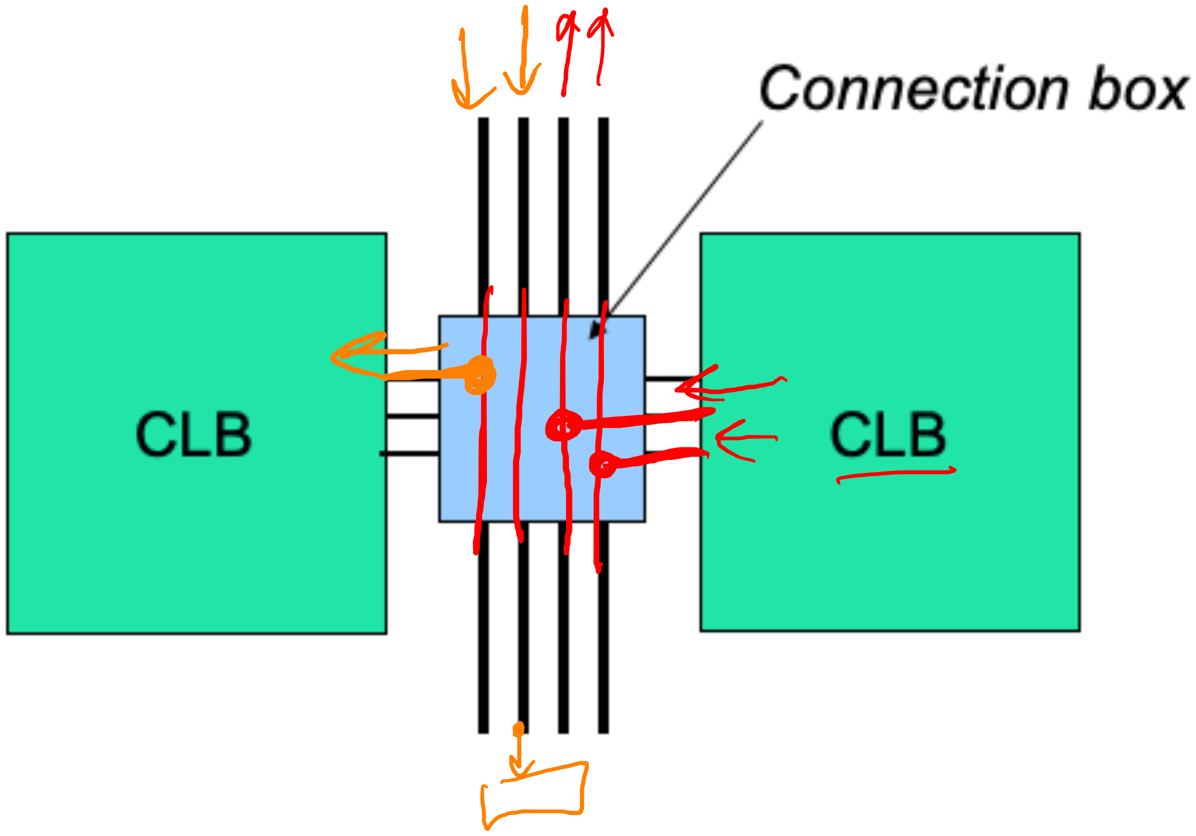
Connecting CLBs

- Q: How do CLBs talk to each other?
- A: Put wires everywhere (ok, almost everywhere)!

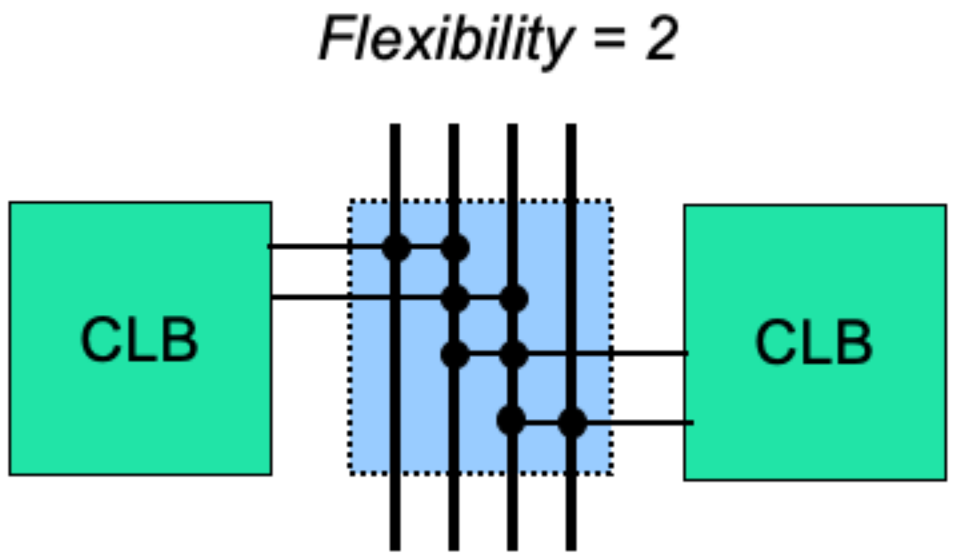


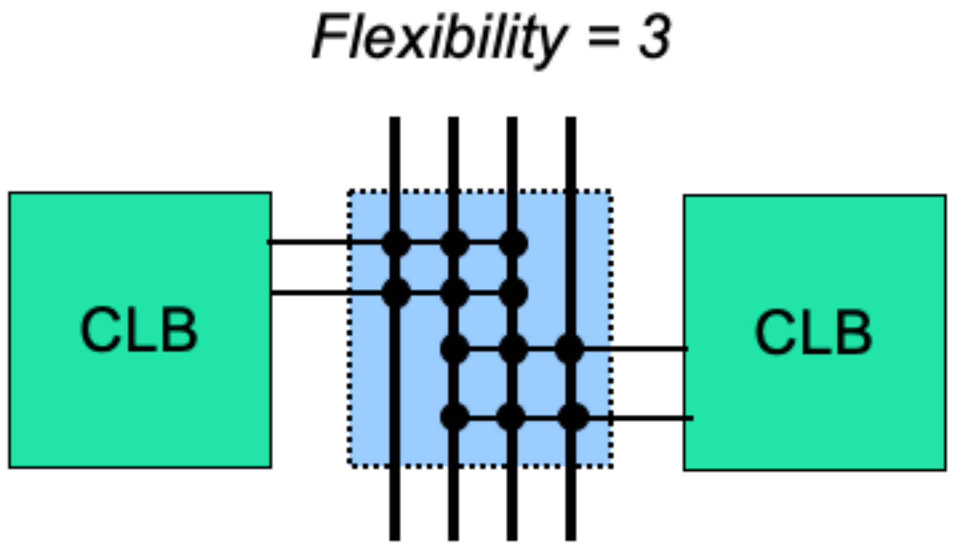
How to connect CLBs to wires?

- "Connection box"
  - Device that allows inputs and outputs of CLB to connect to different wires



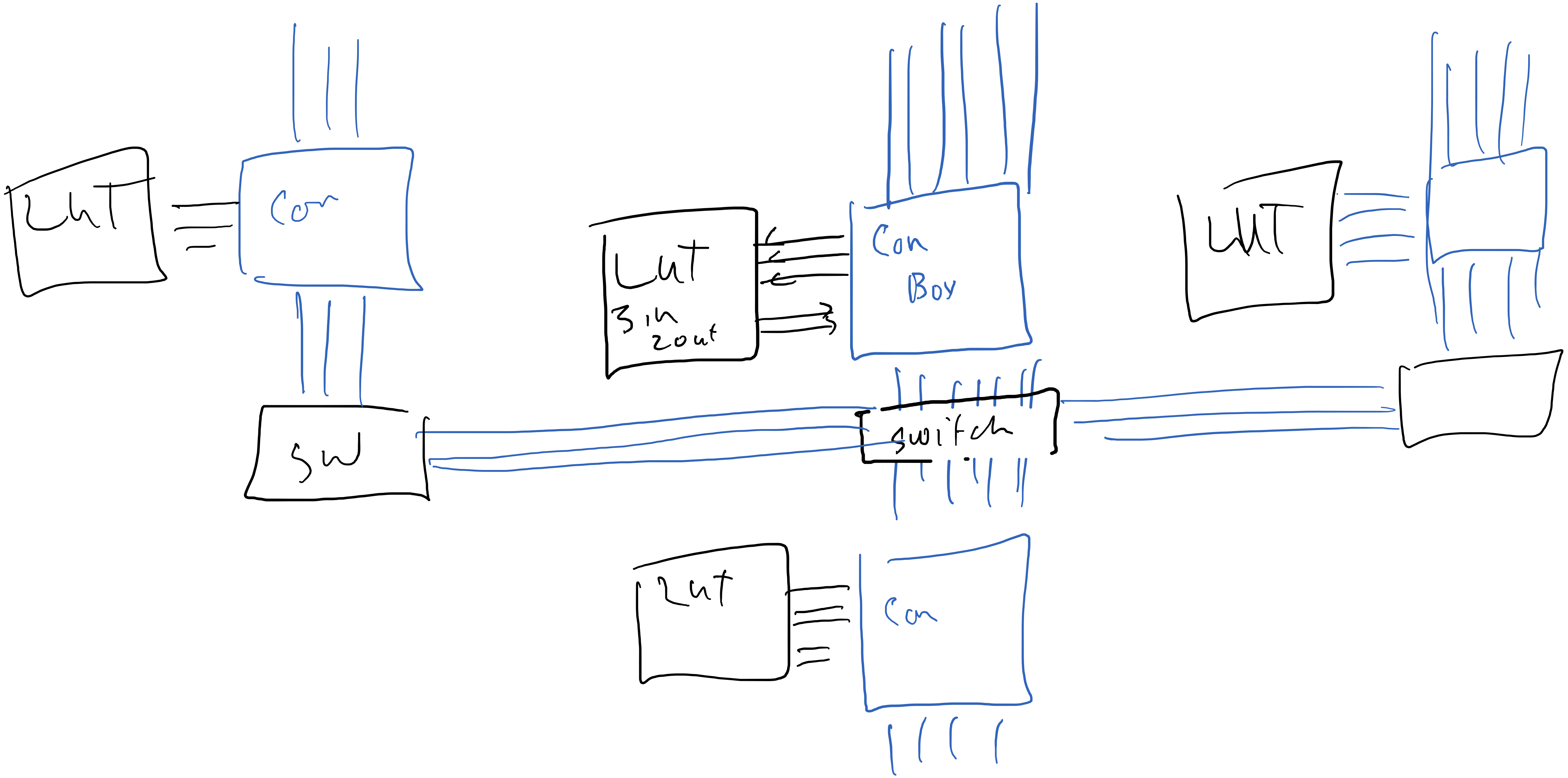
Connection Box Flexibility





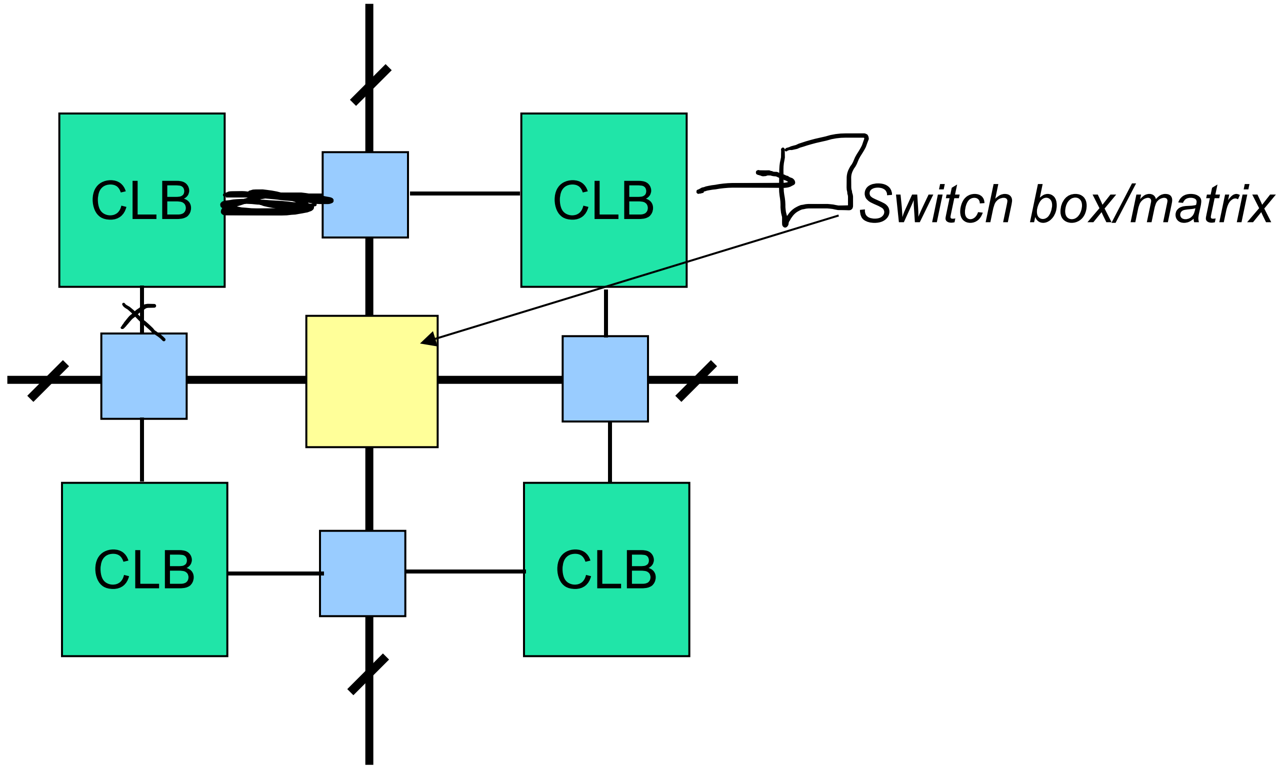
Dots represent* ***possible* *connections*

How to connect wires to each other?

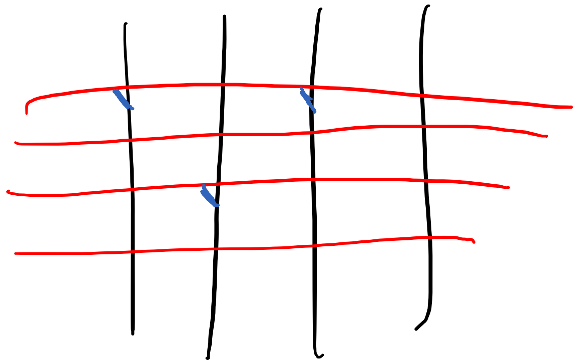


Switch Box

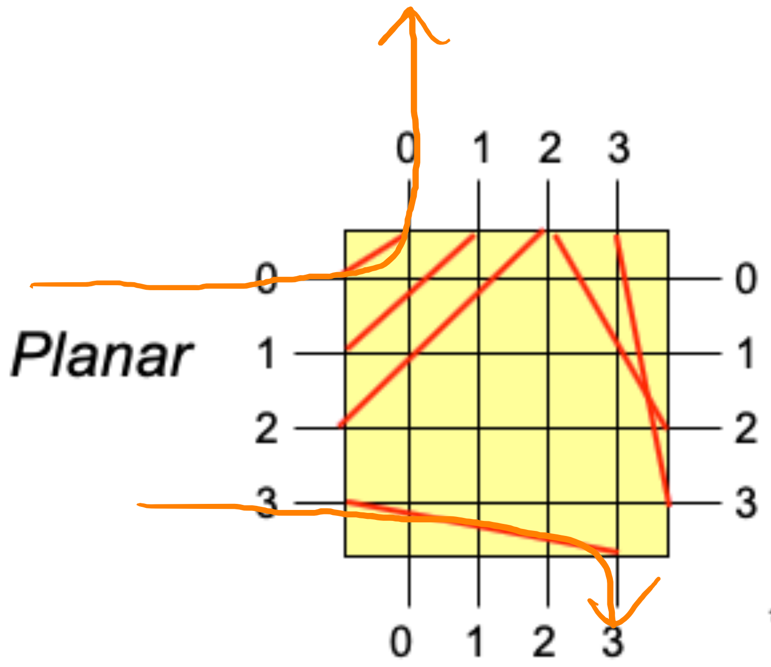
- Connects horizontal and vertical routing channels



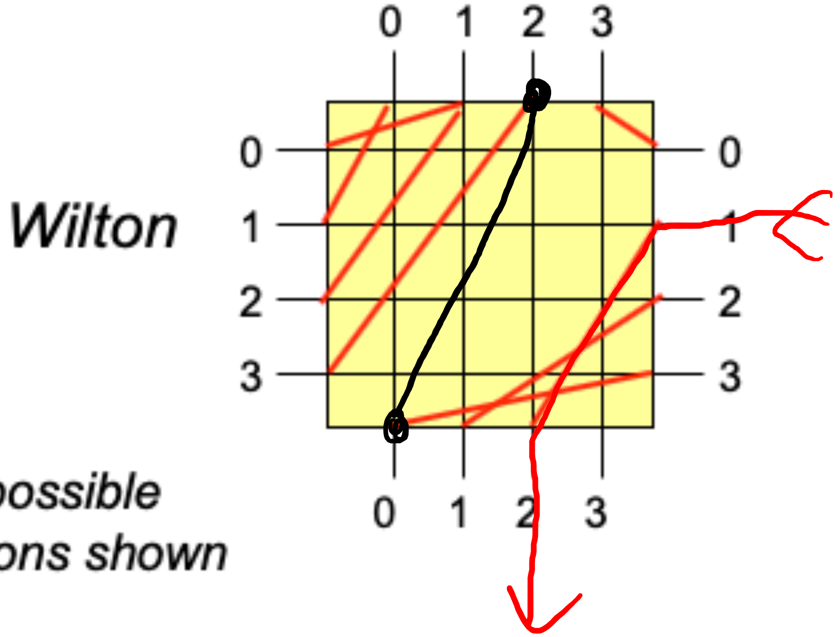
Switch Box Connections



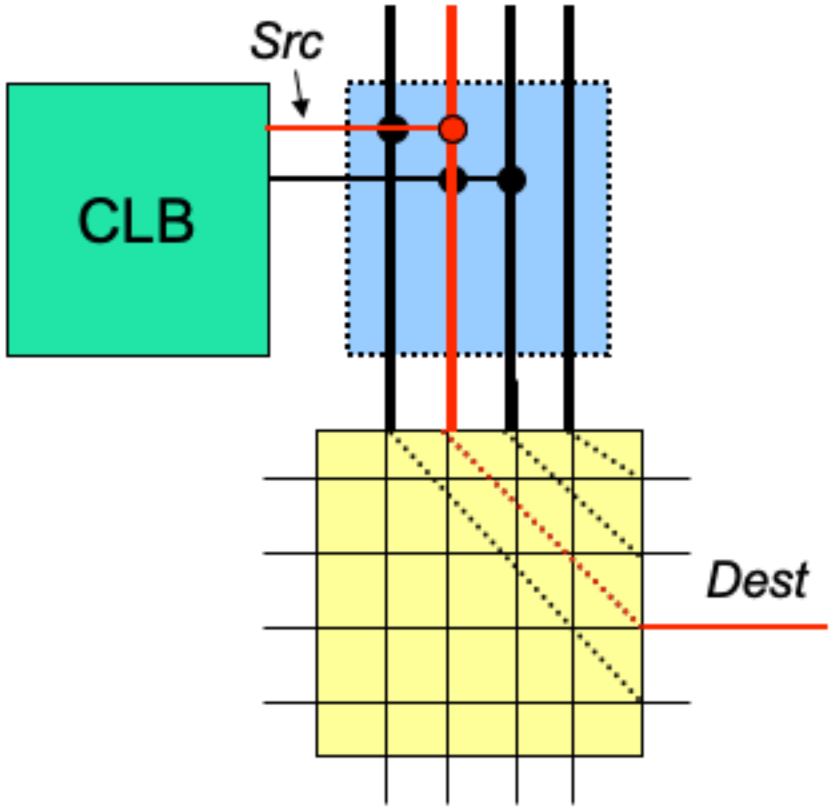
- Programmable connections between inputs and outputs

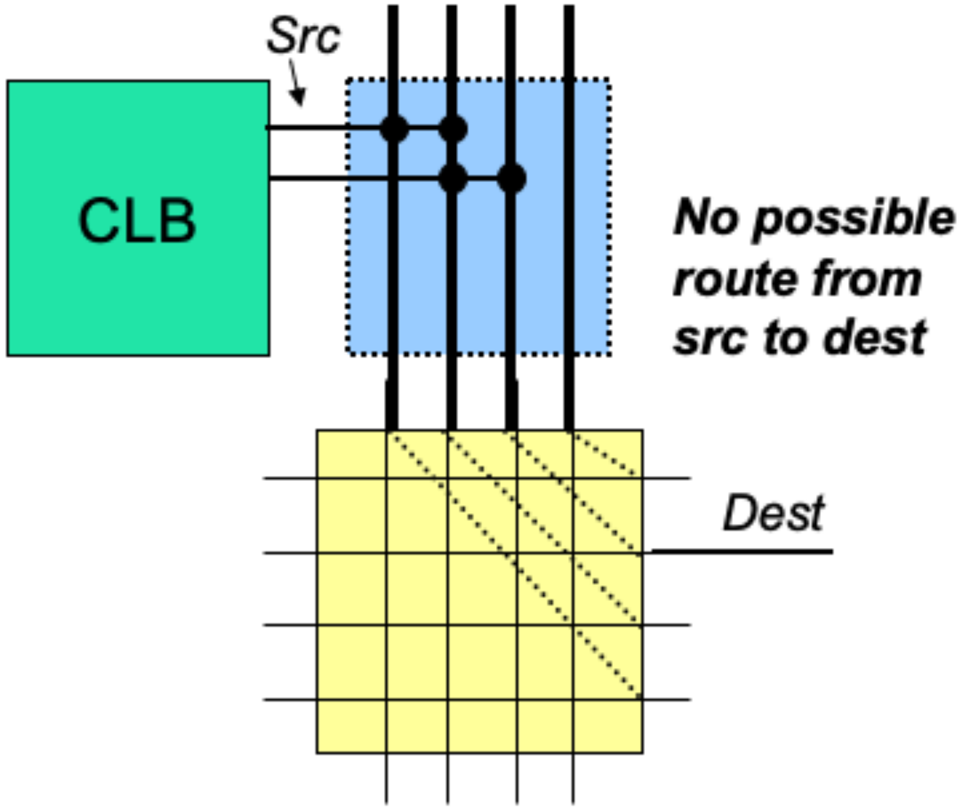


**Not all possible connections shown*



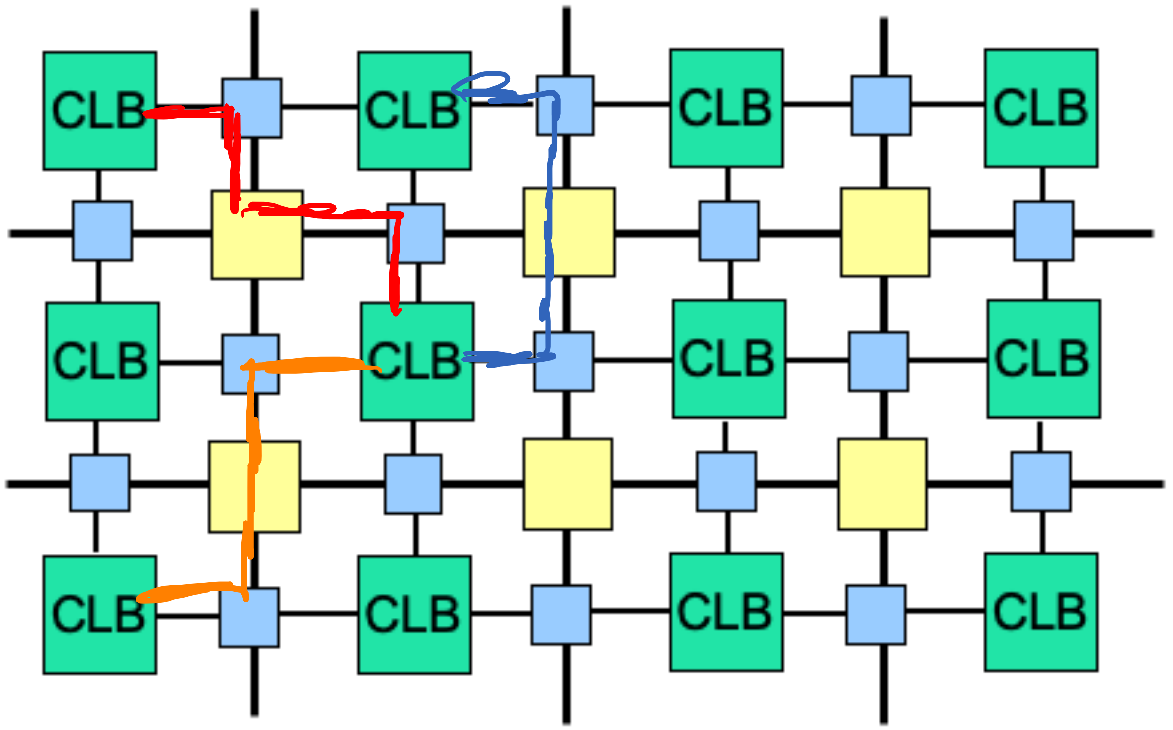
Switch Box Connections





FPGA "Fabric"

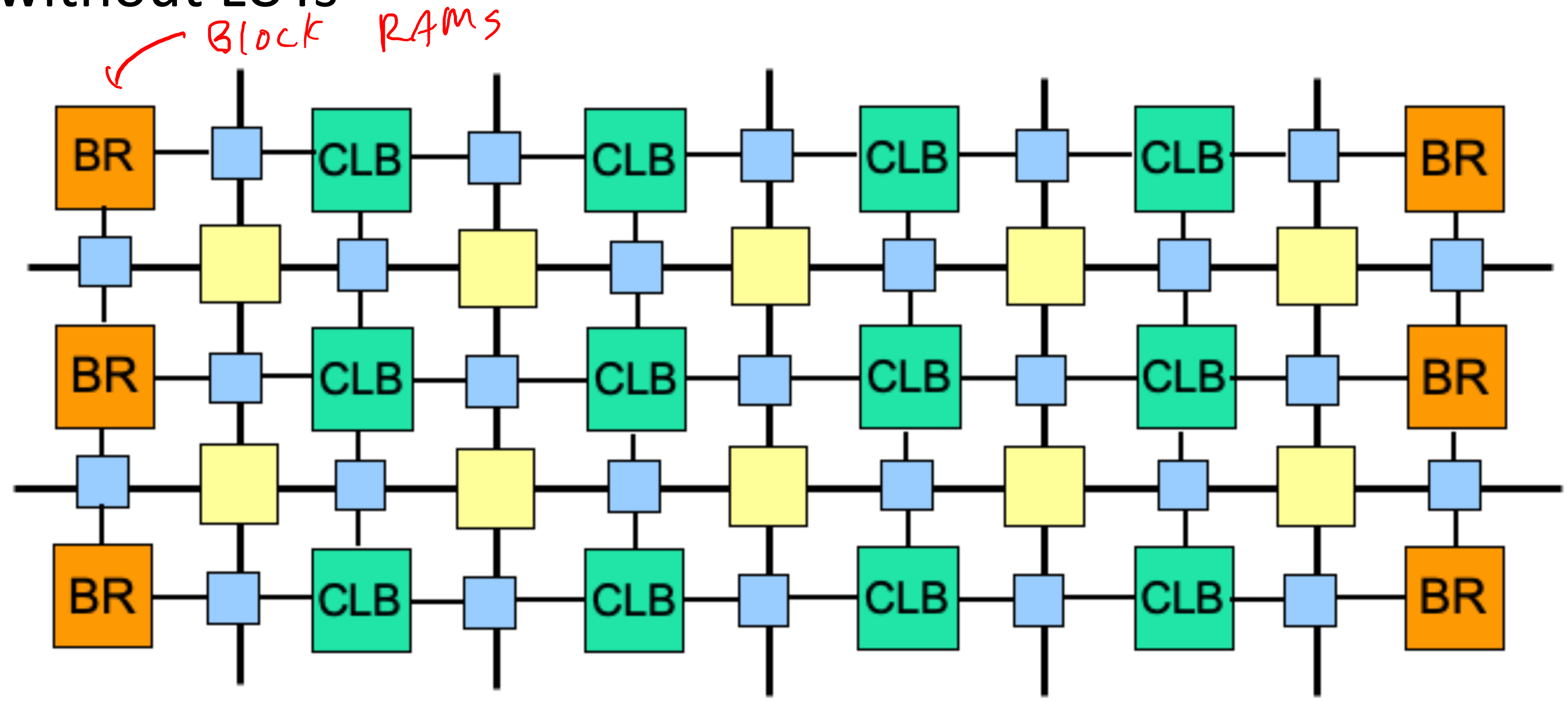
- 2D array of CLBs + interconnects



- Am I missing anything?

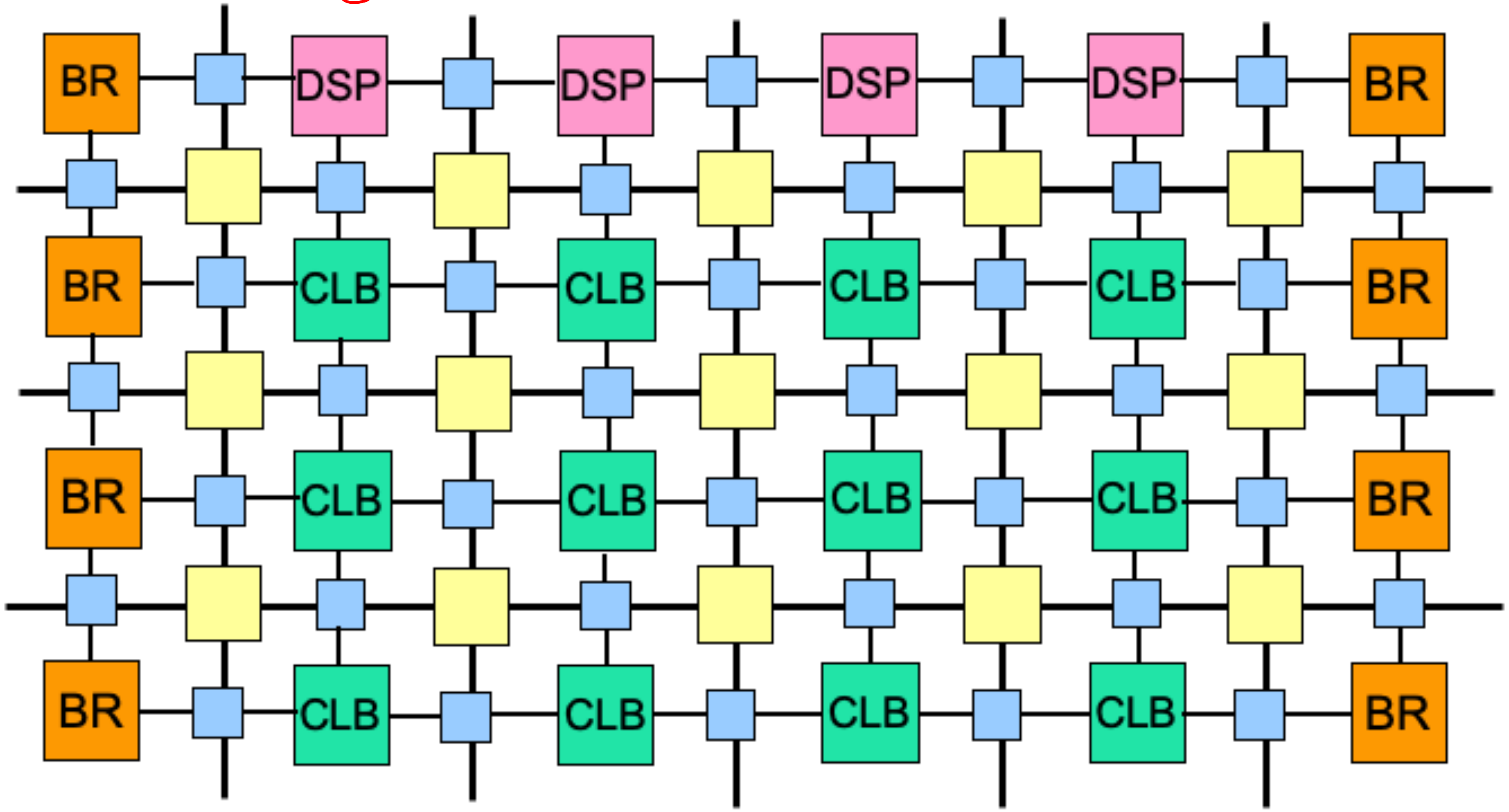
Block RAM

- Special blocks of just RAM
- Big CLBs without LUTs

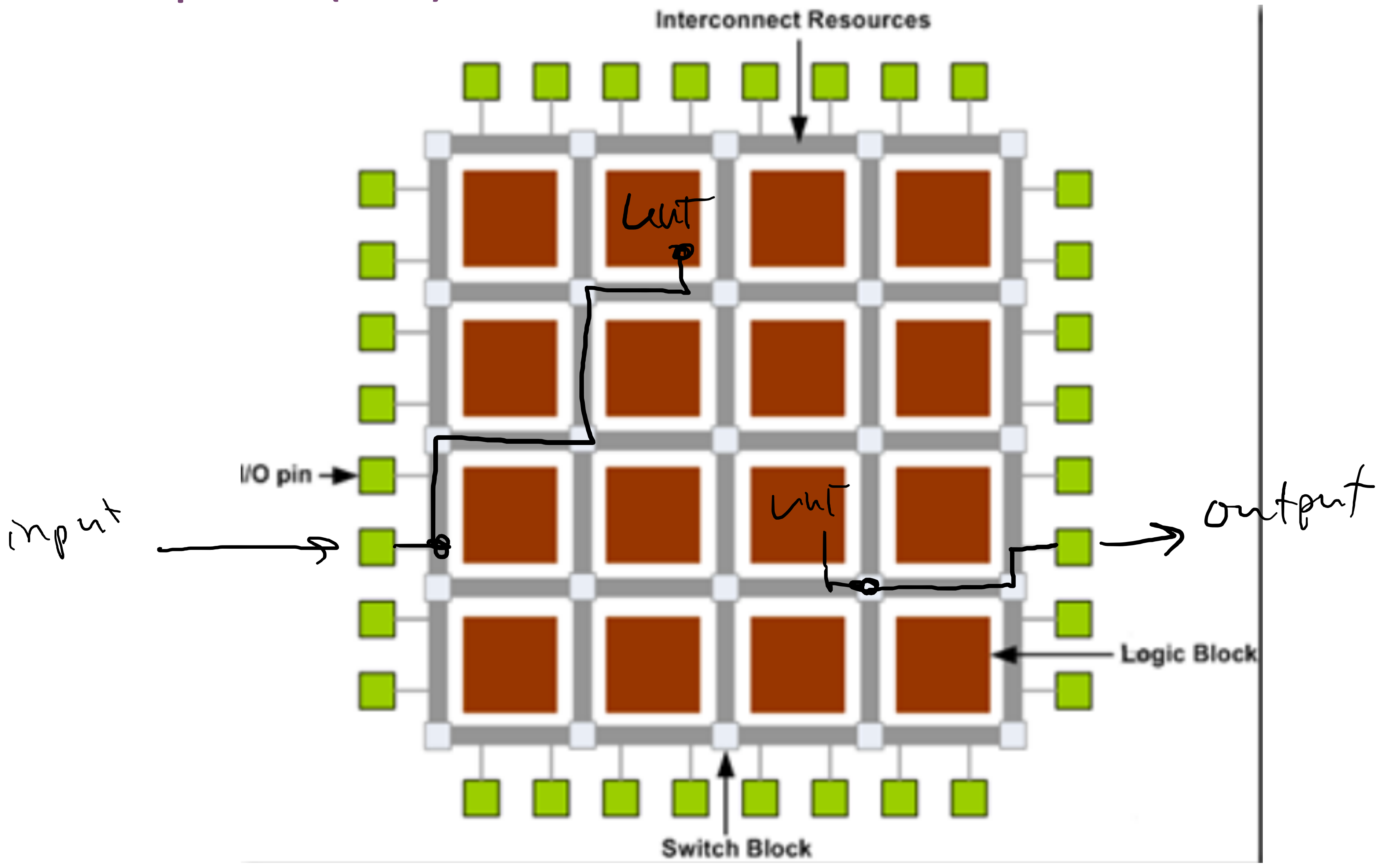


DSPs → Digital Signal Processor → dedicated math units

multiply



Input/Output (IO)


Interconnect Resources
I/O pin
Logic Block
Switch Block
input
output
LUT
LUT