# FPGAs Architecture

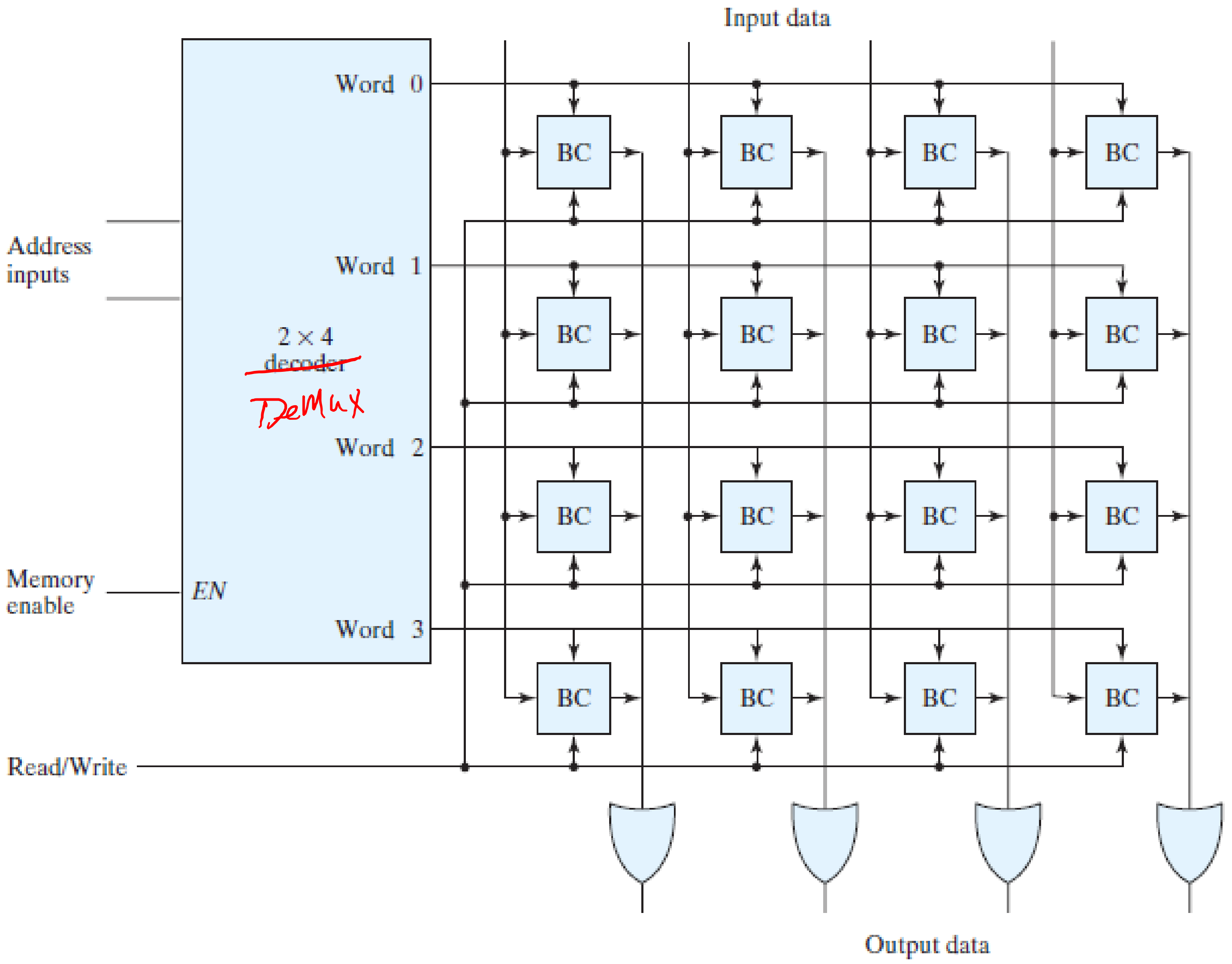
Andrew Lukefahr

Portions borrowed from:
http://www.gstitt.ece.ufl.edu/courses/fall15/eel4720_5721/index.html

## Topics

- FPGA internals
- Synthesis Process

# Review:  RAM


Input data
Address inputs
Word 0
Word 1
Word 2
Word 3
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

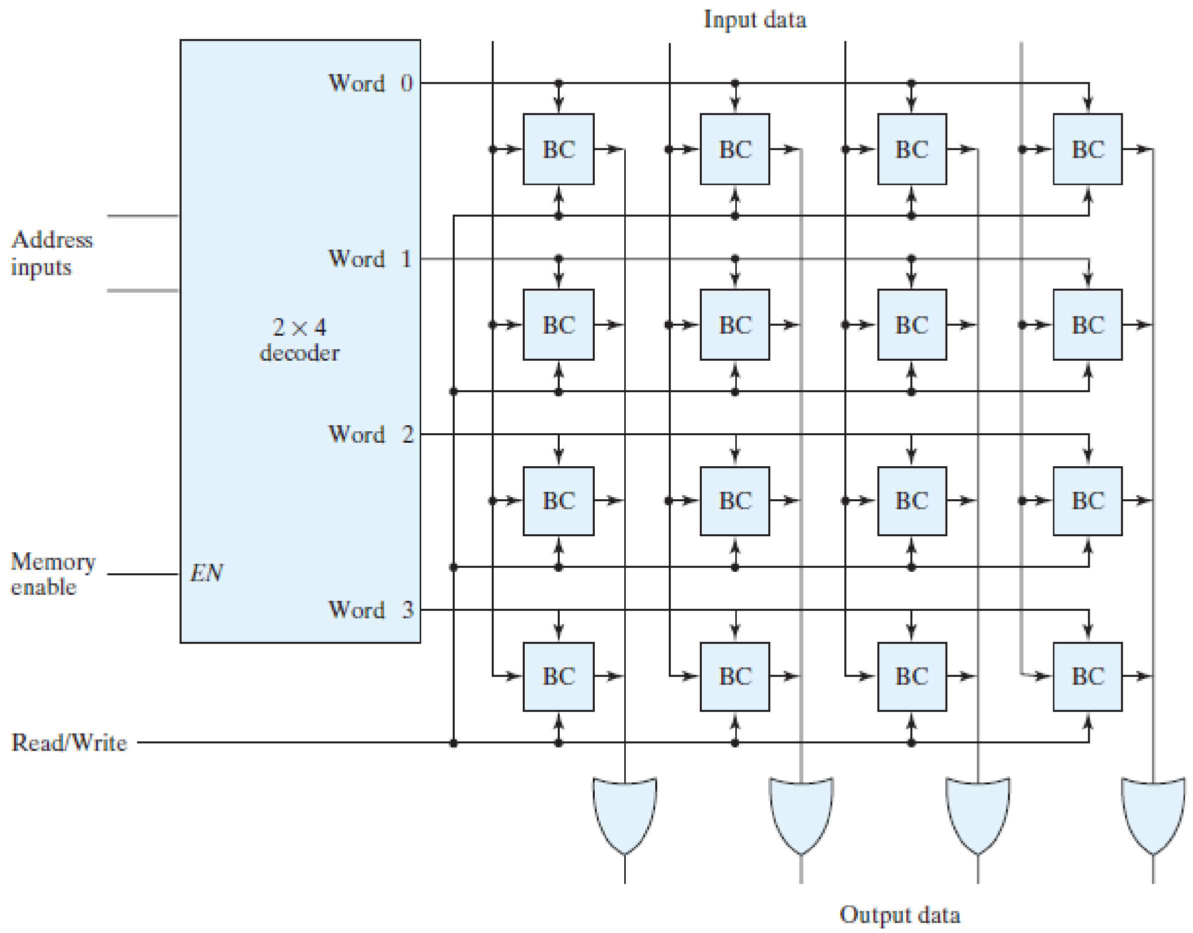
- Examples:

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

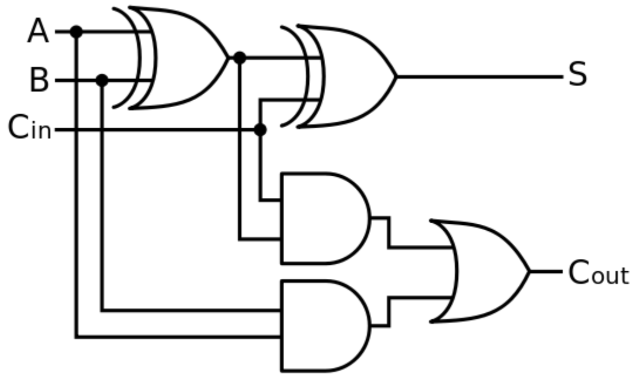
# RAM to LUT

- 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 |

## LUT size

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

- 3 inputs => $2^3$ rows = 8 rows
- 4 inputs => $2^4$ rows = 16 rows
- 5 inputs => $2^5$ rows = 32 rows
- …
- 64 inputs => $2^{64}$ rows = 1.85 x$10^{19}$ rows

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

## 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

# Configurable Logic Block (CLB)

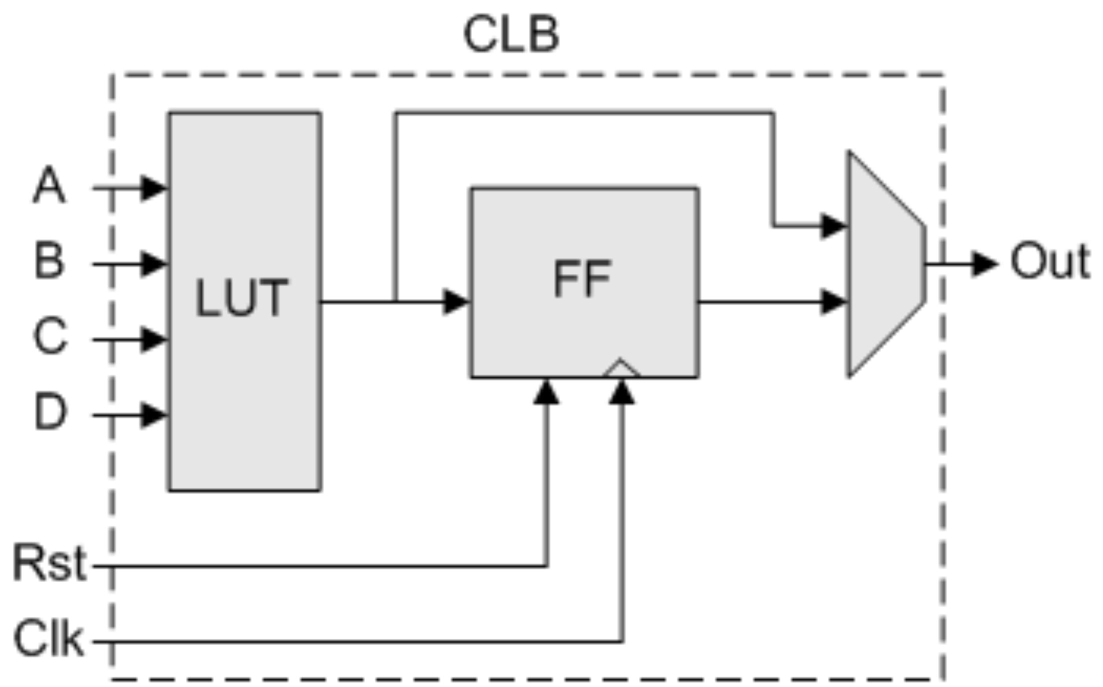


Fig.4 Example of Configurable Logic Cell.

# Configurable Logic Block (CLB)

- What if I only want to store a value?

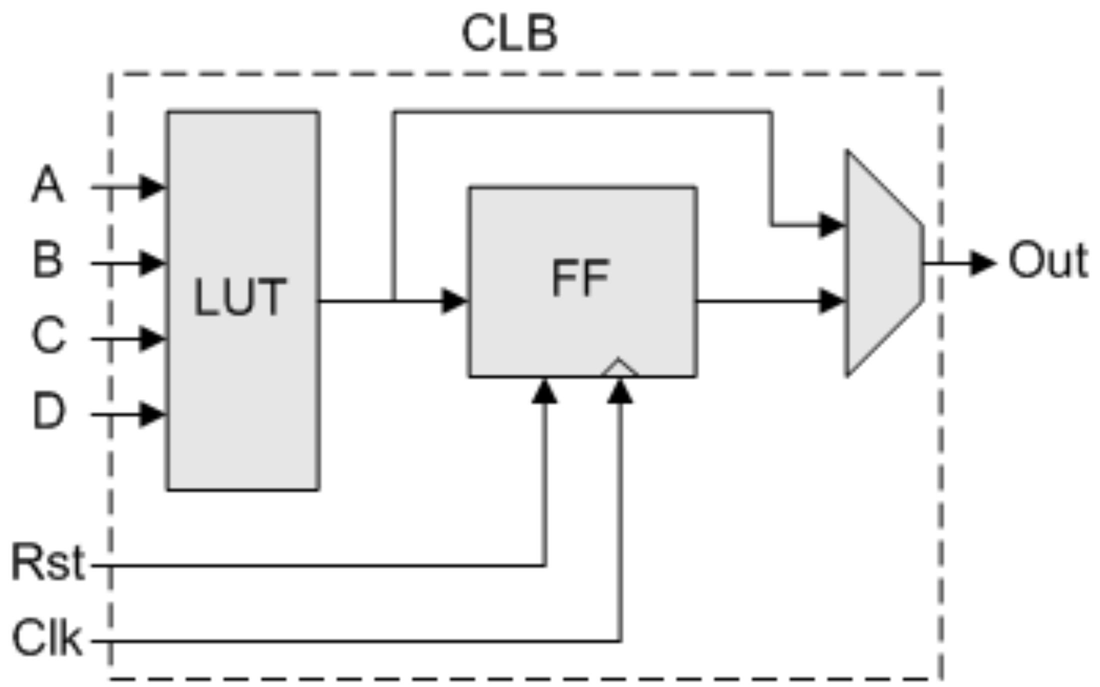
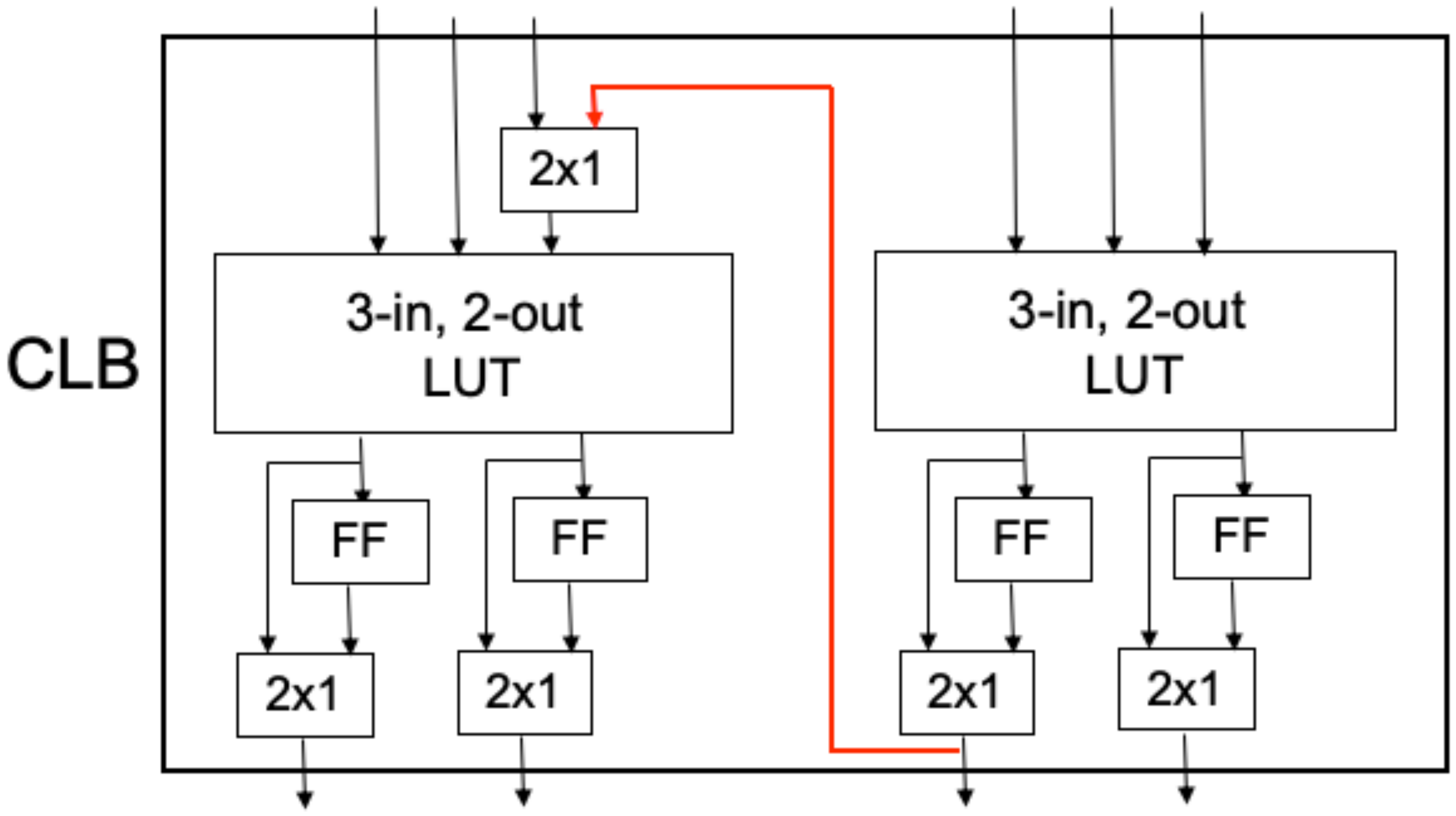
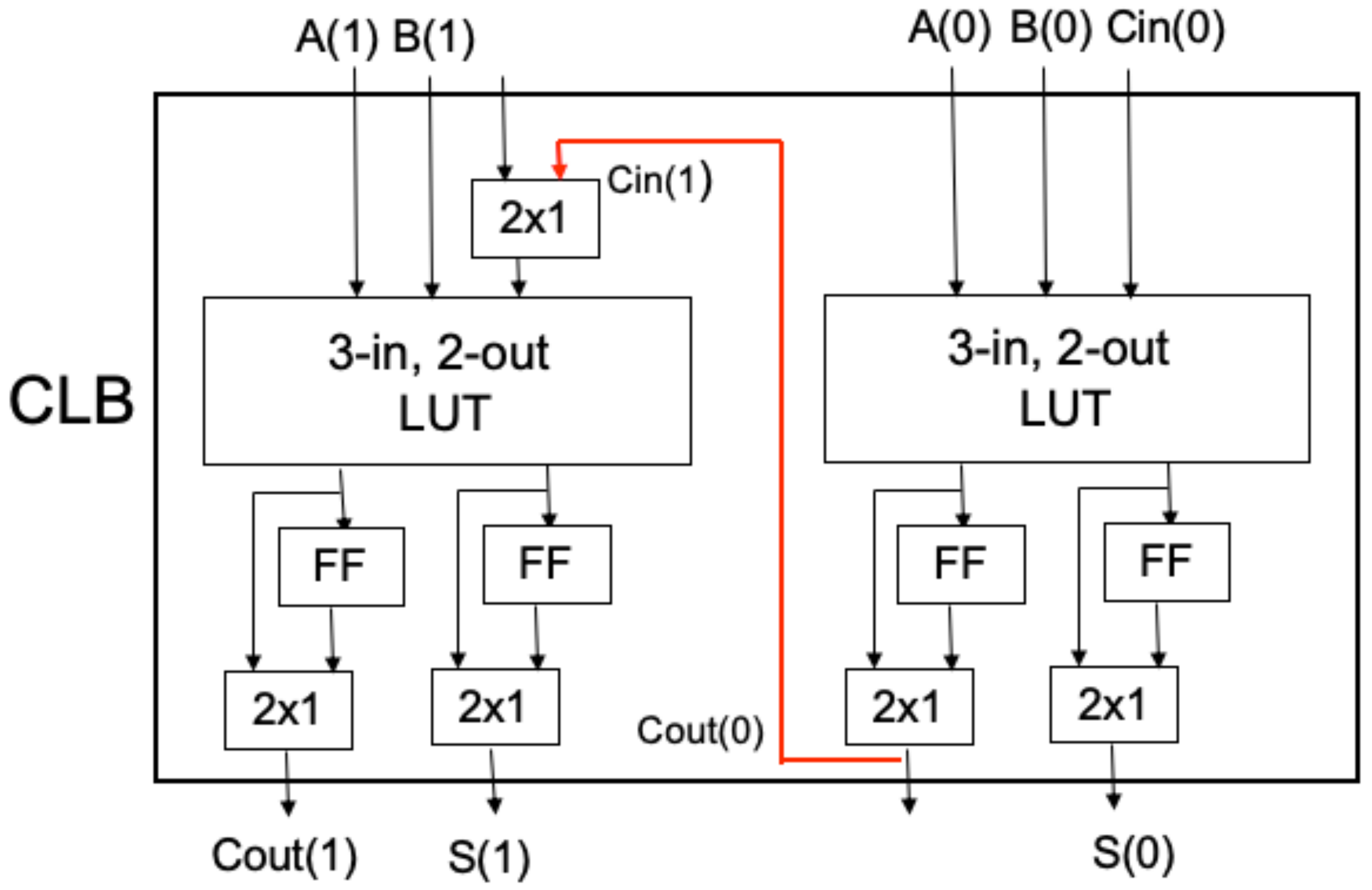


Fig.4 Example of Configurable Logic Cell.

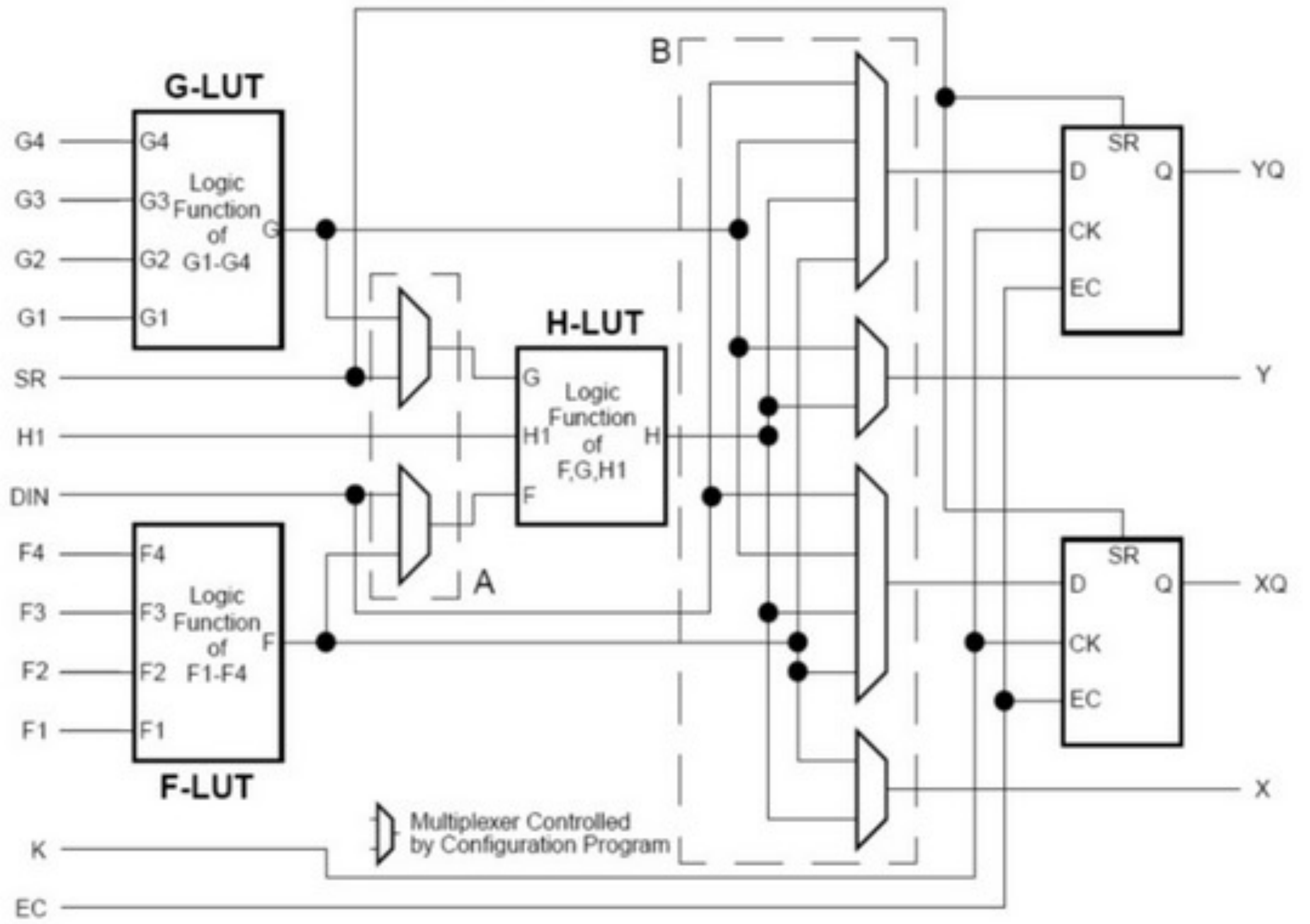
## Improved CLB



# 2-Bit Ripple-Carry w/ CLB

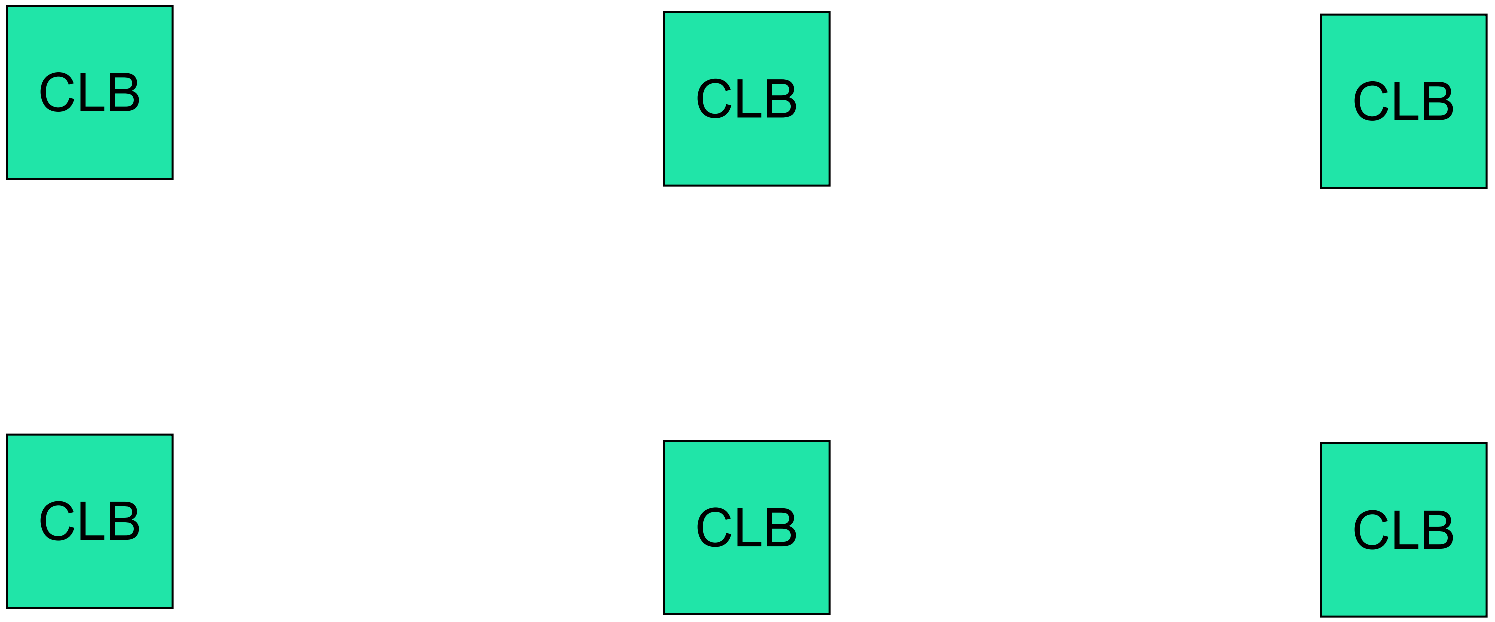


# Realistic CLB: Xilinx



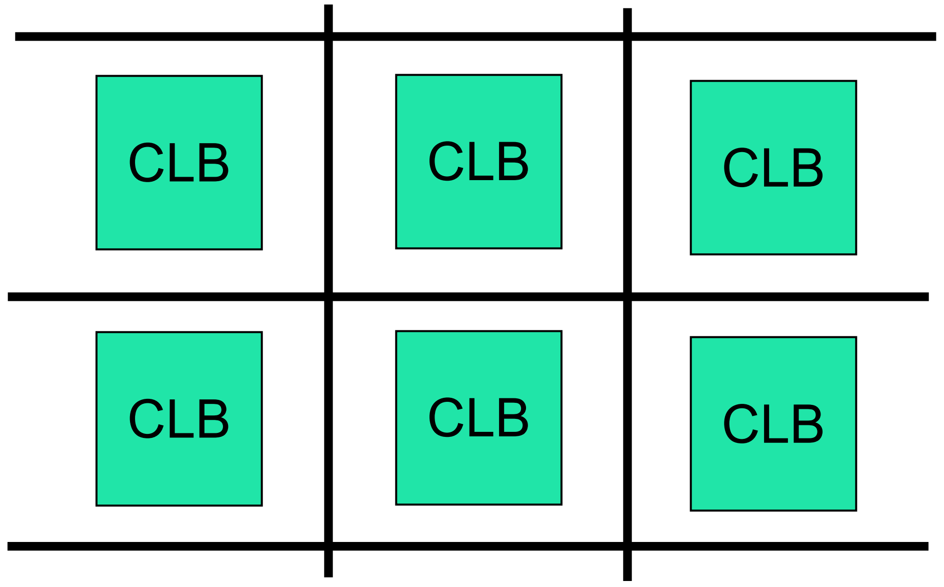
## Connecting CLBs

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



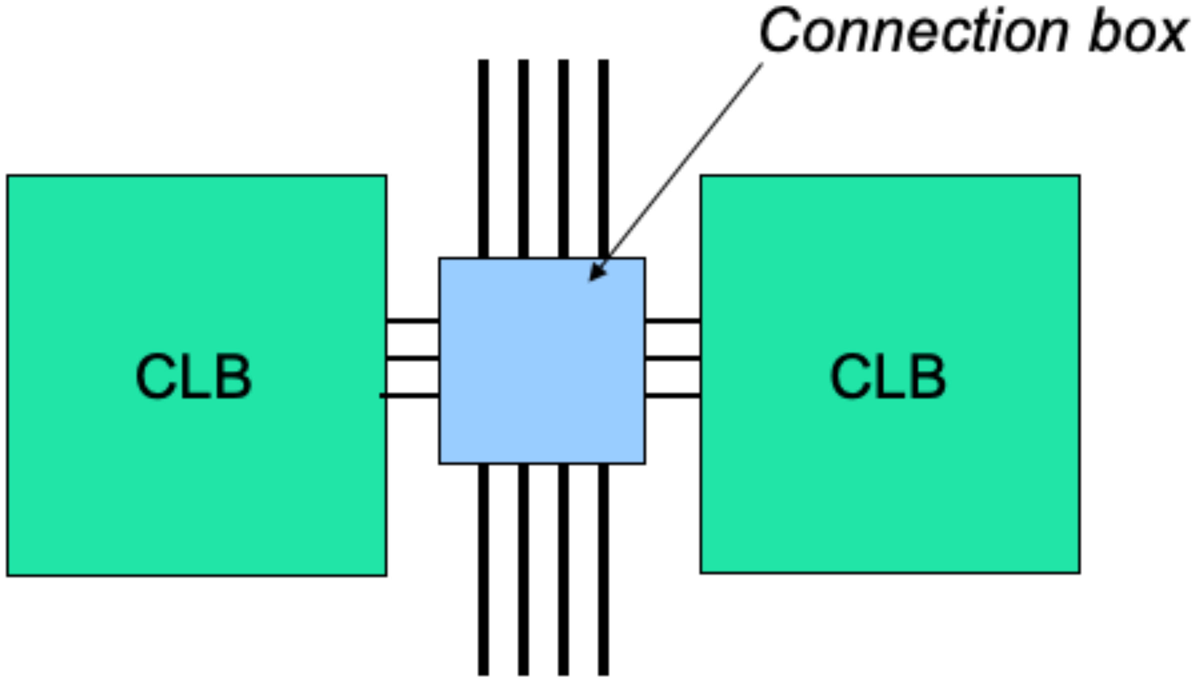
## Connecting CLBs

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

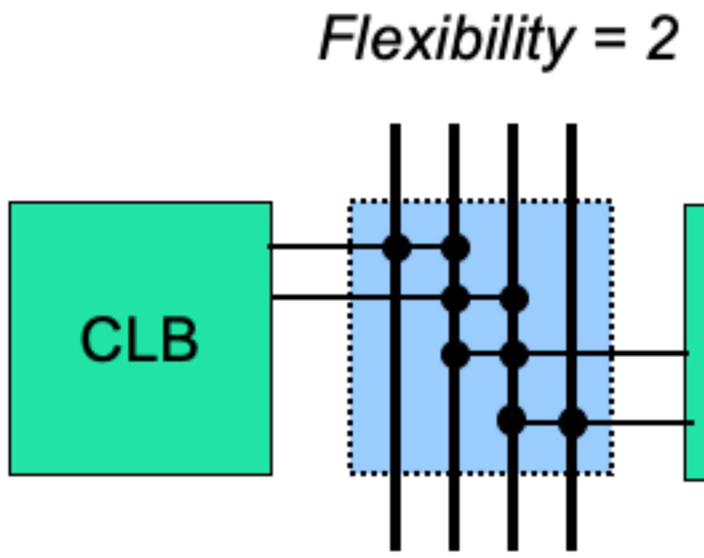


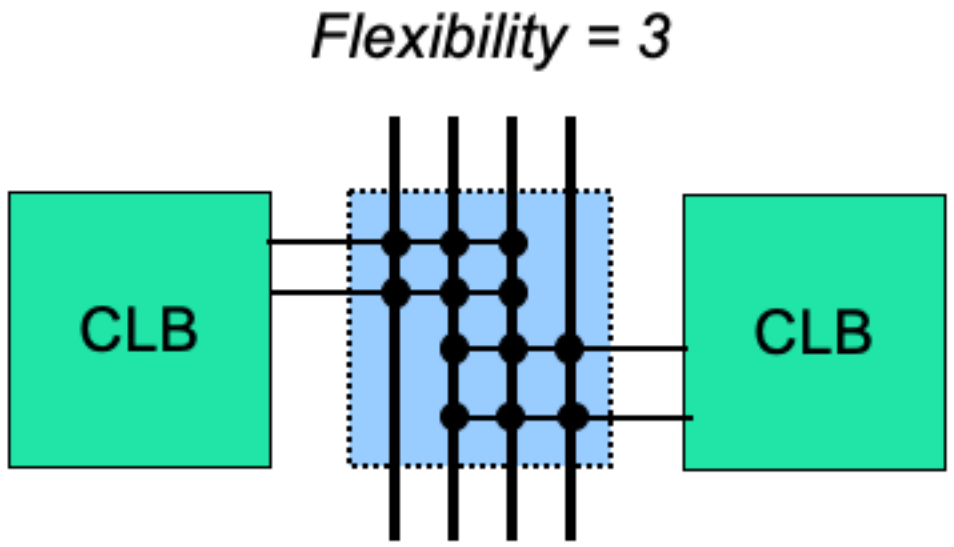
## How to connect CLBs to wires?

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



## Connection Box Flexibility



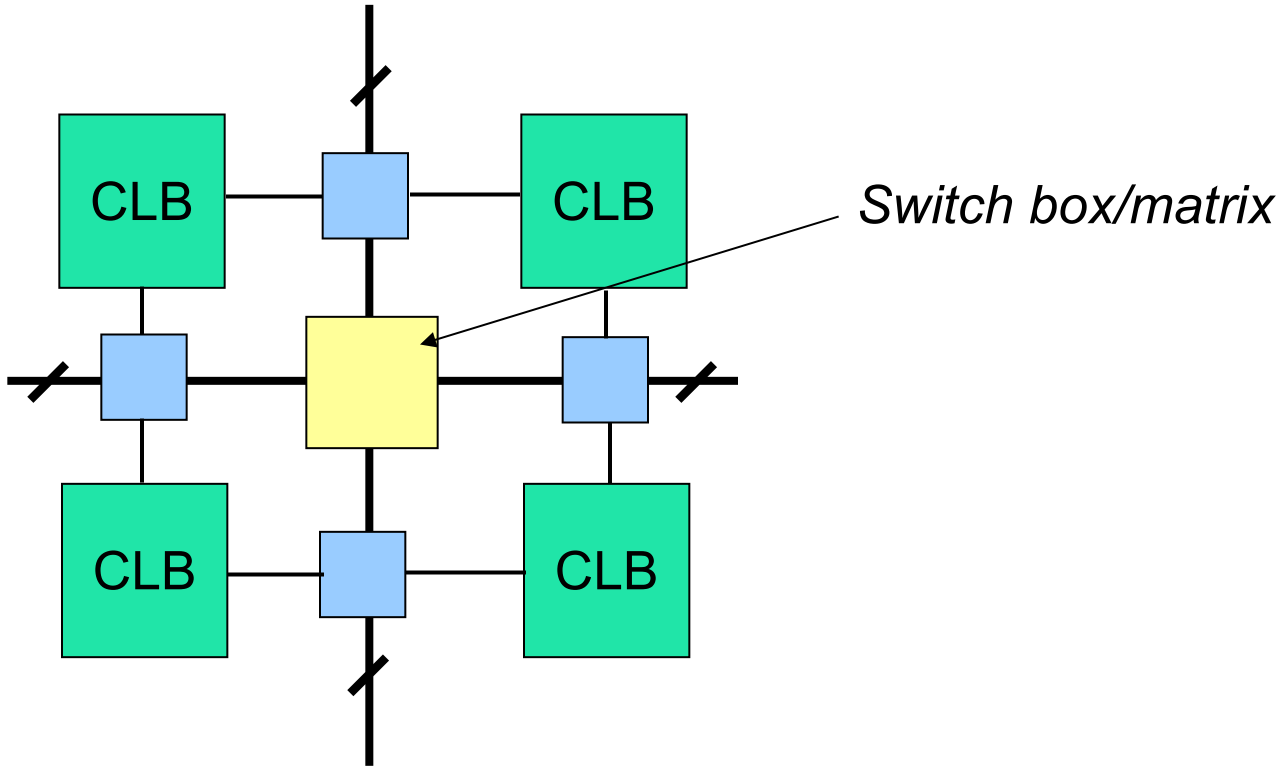


**Dots represent* ***possible*** *connections*

## How to connect wires to each other?

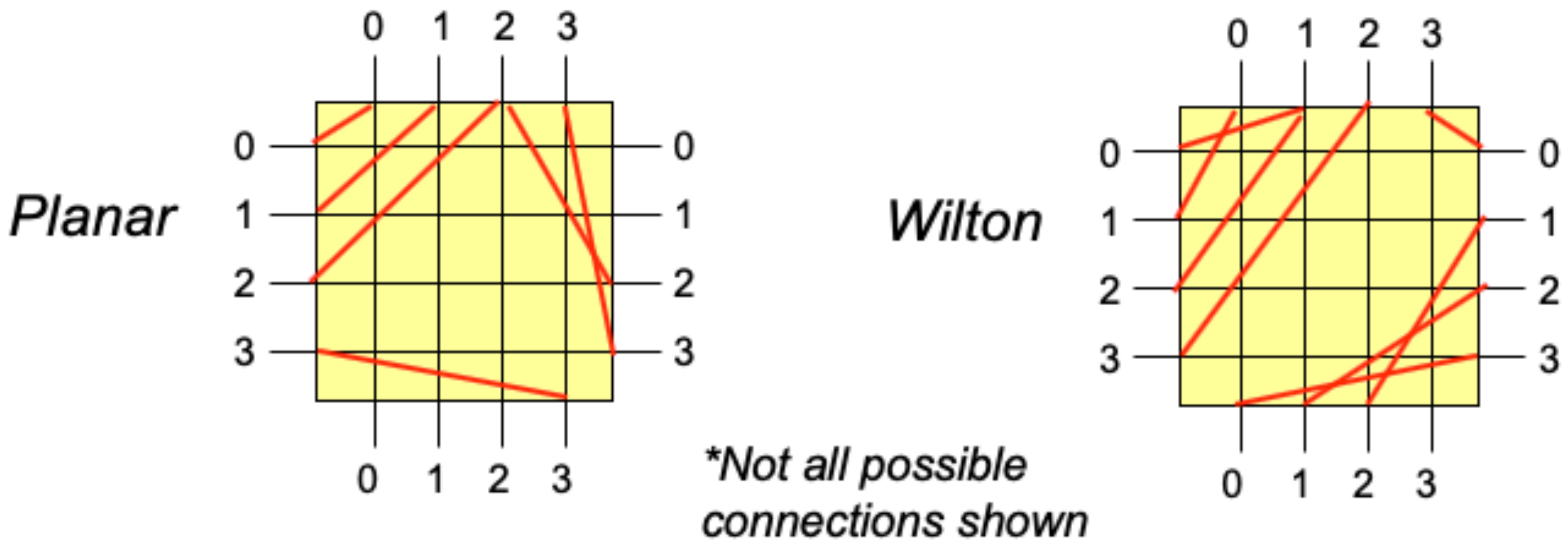
## Switch Box

- Connects horizontal and vertical routing channels

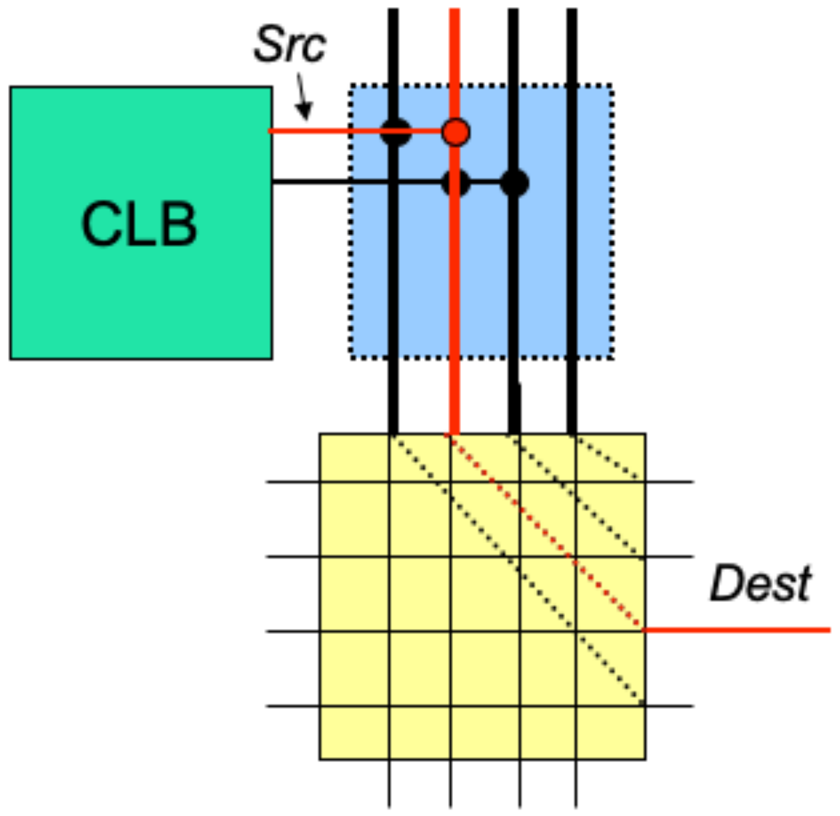


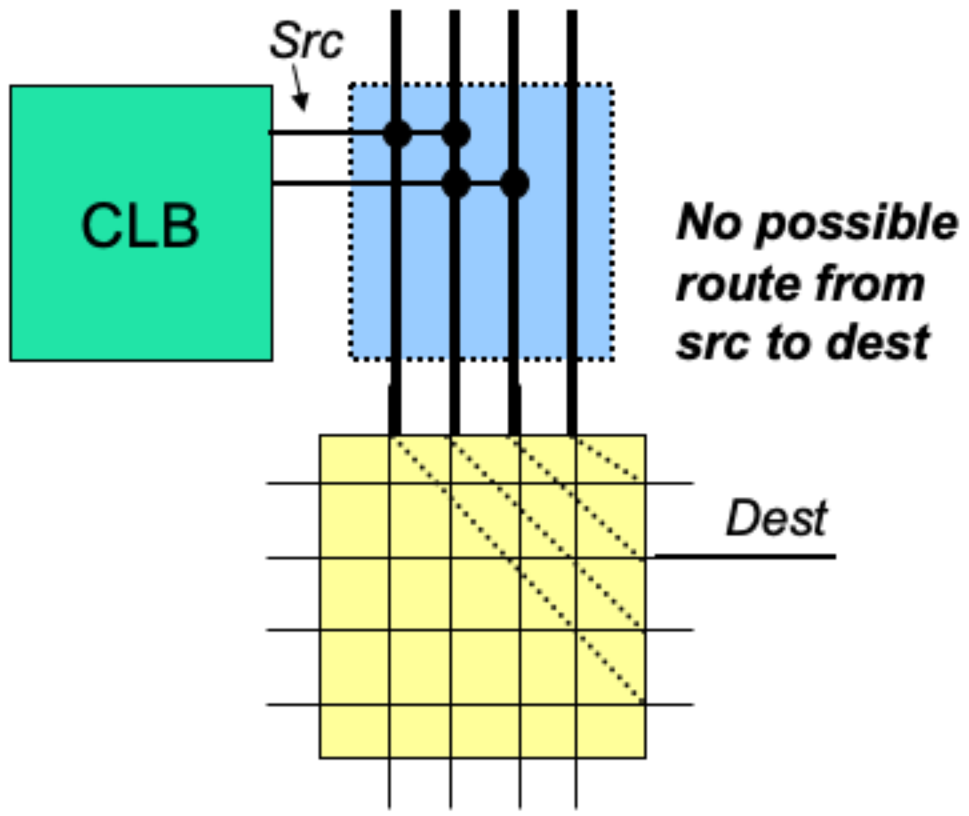
# Switch Box Connections

- Programmable connections between inputs and outputs



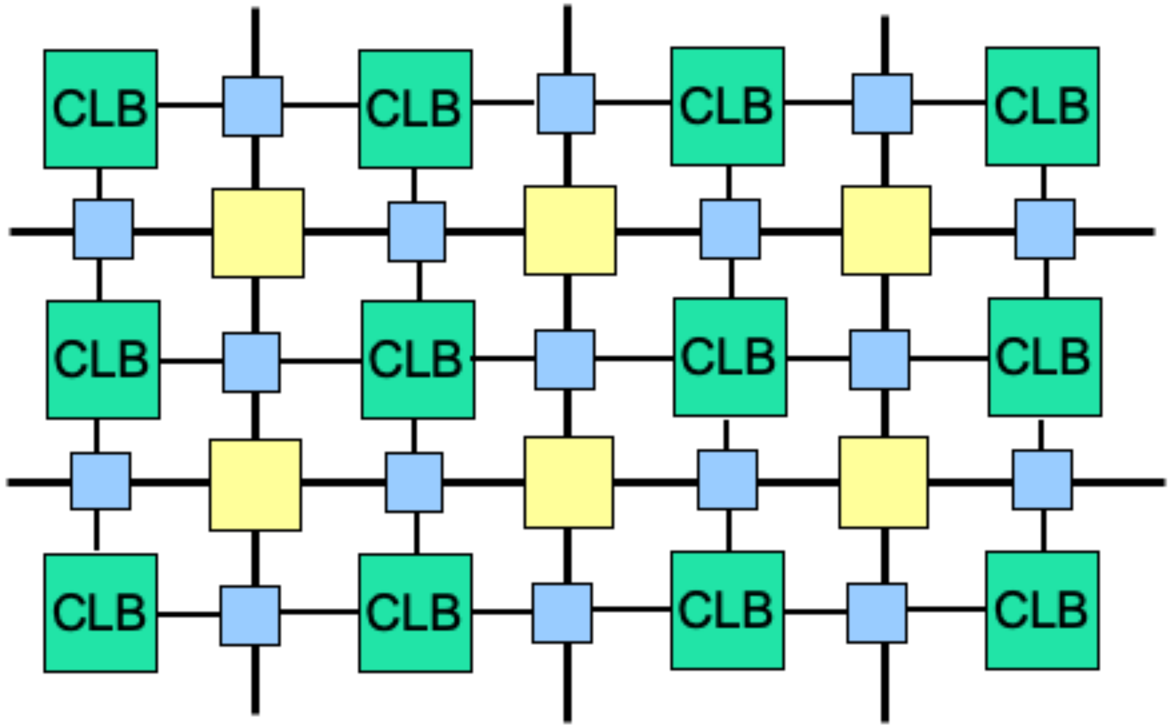
# Switch Box Connections





## FPGA “Fabric”

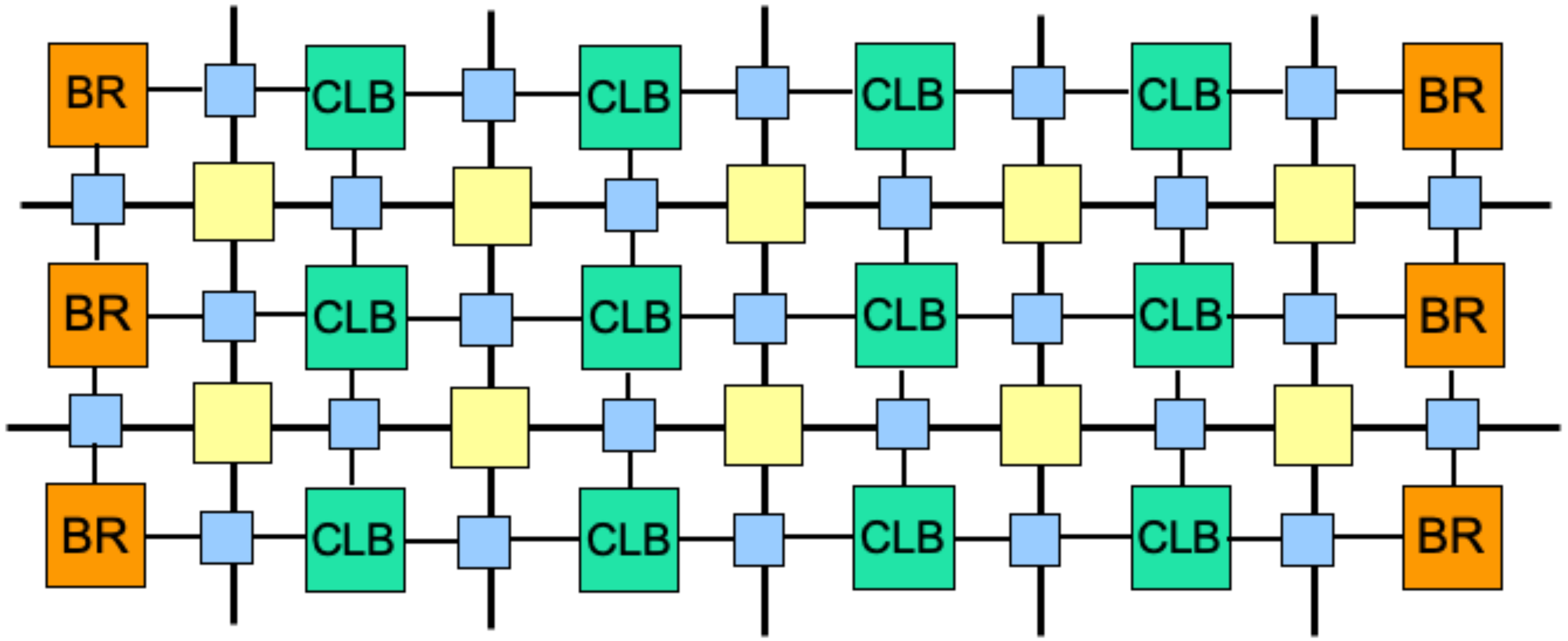
- 2D array of CLBs + interconnects



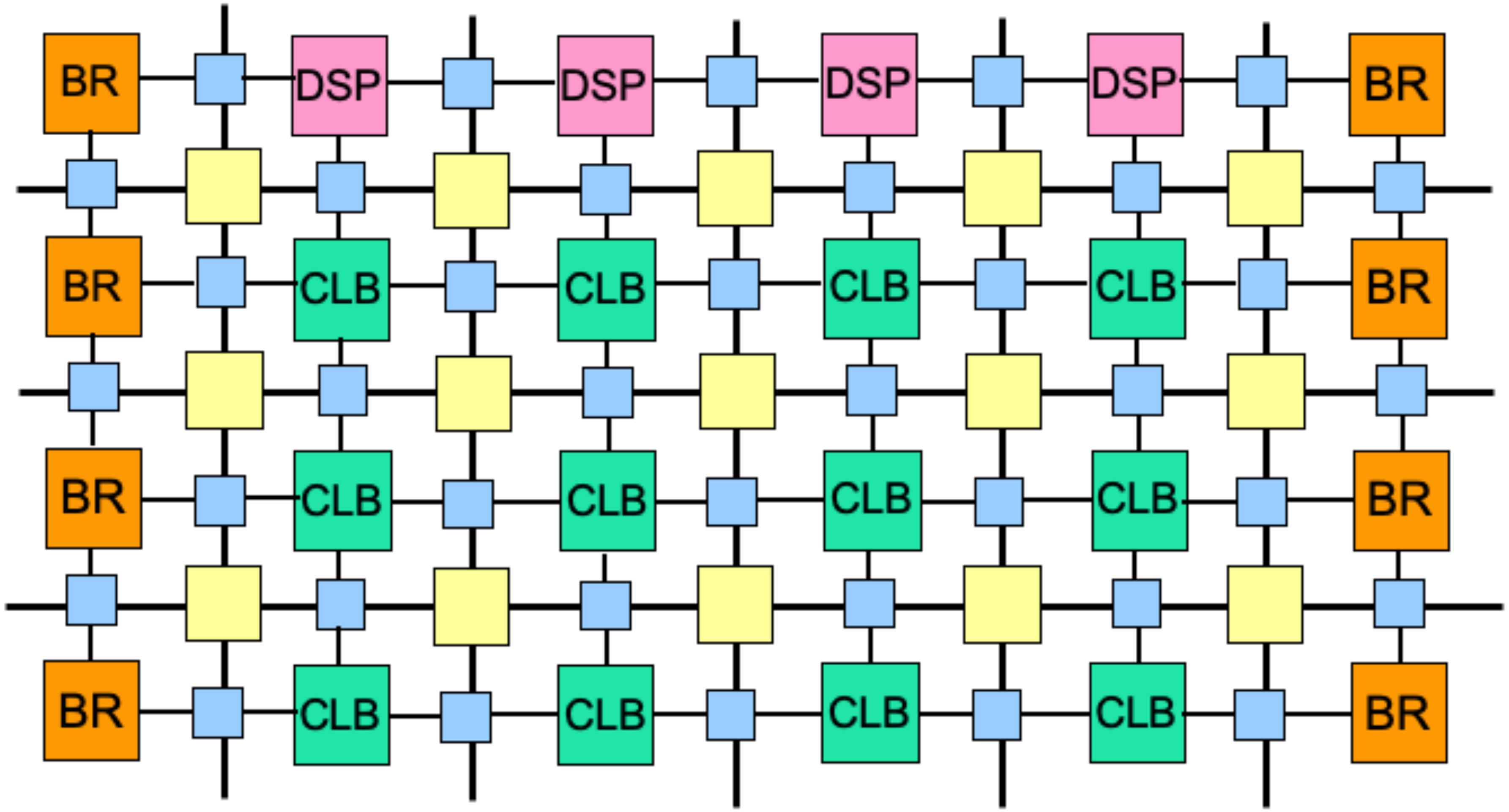
- Am I missing anything?

## Block RAM

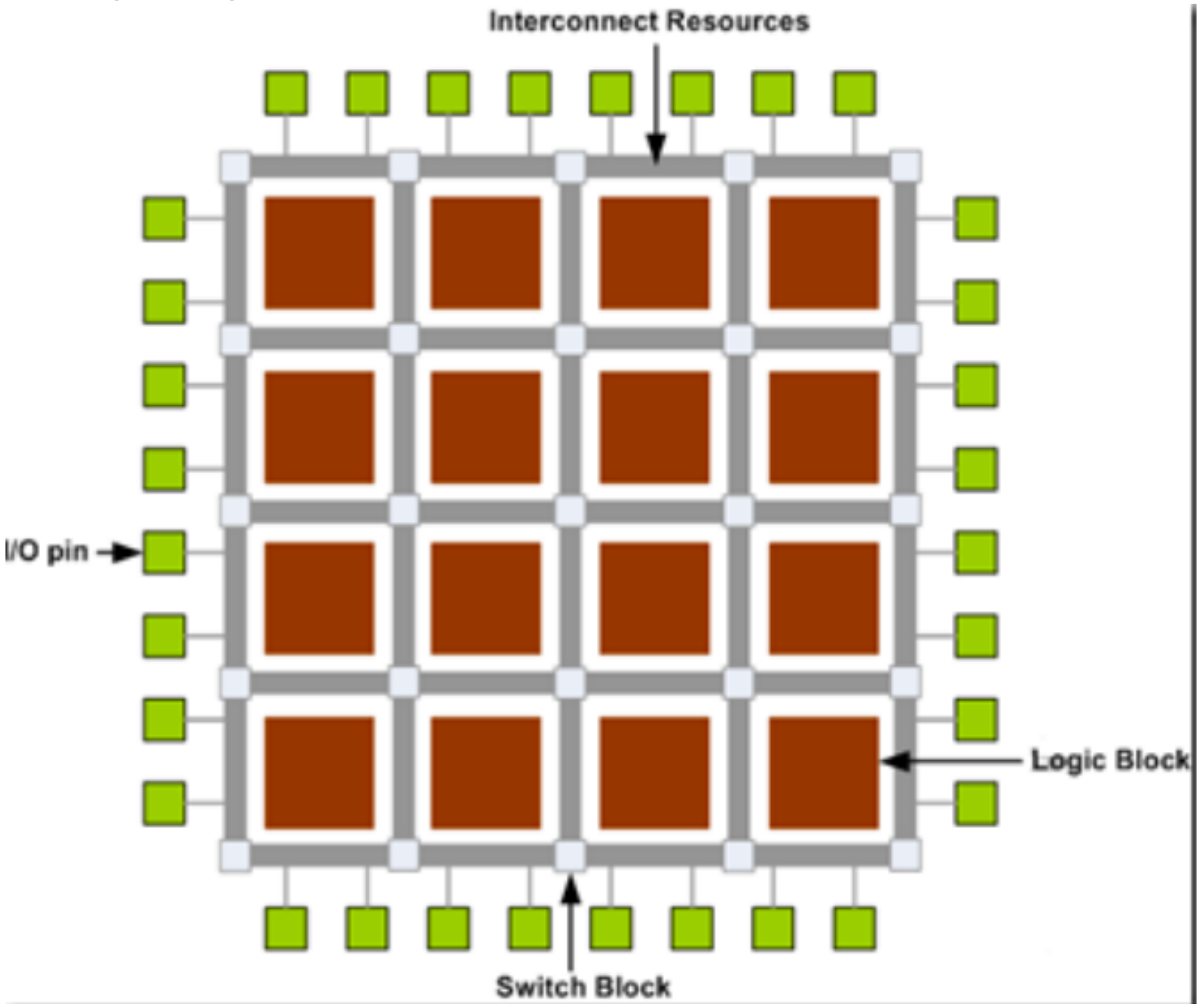
- Special blocks of just RAM
- Big CLBs without LUTs



## DSPs

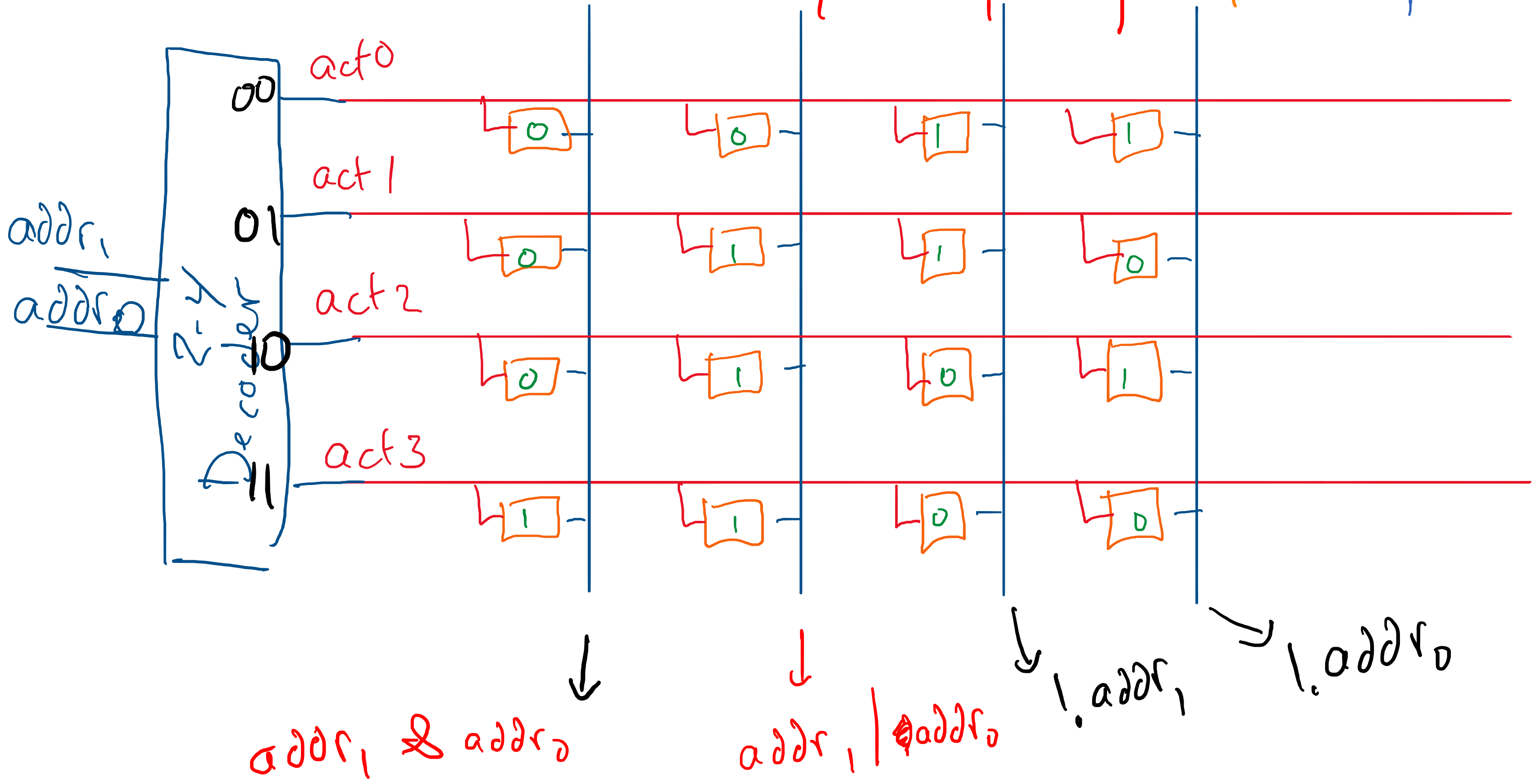

BR
DSP
DSP
DSP
DSP
BR
BR
CLB
CLB
CLB
CLB
BR
BR
CLB
CLB
CLB
CLB
BR
BR
CLB
CLB
CLB
CLB
BR

## Input/Output (IO)

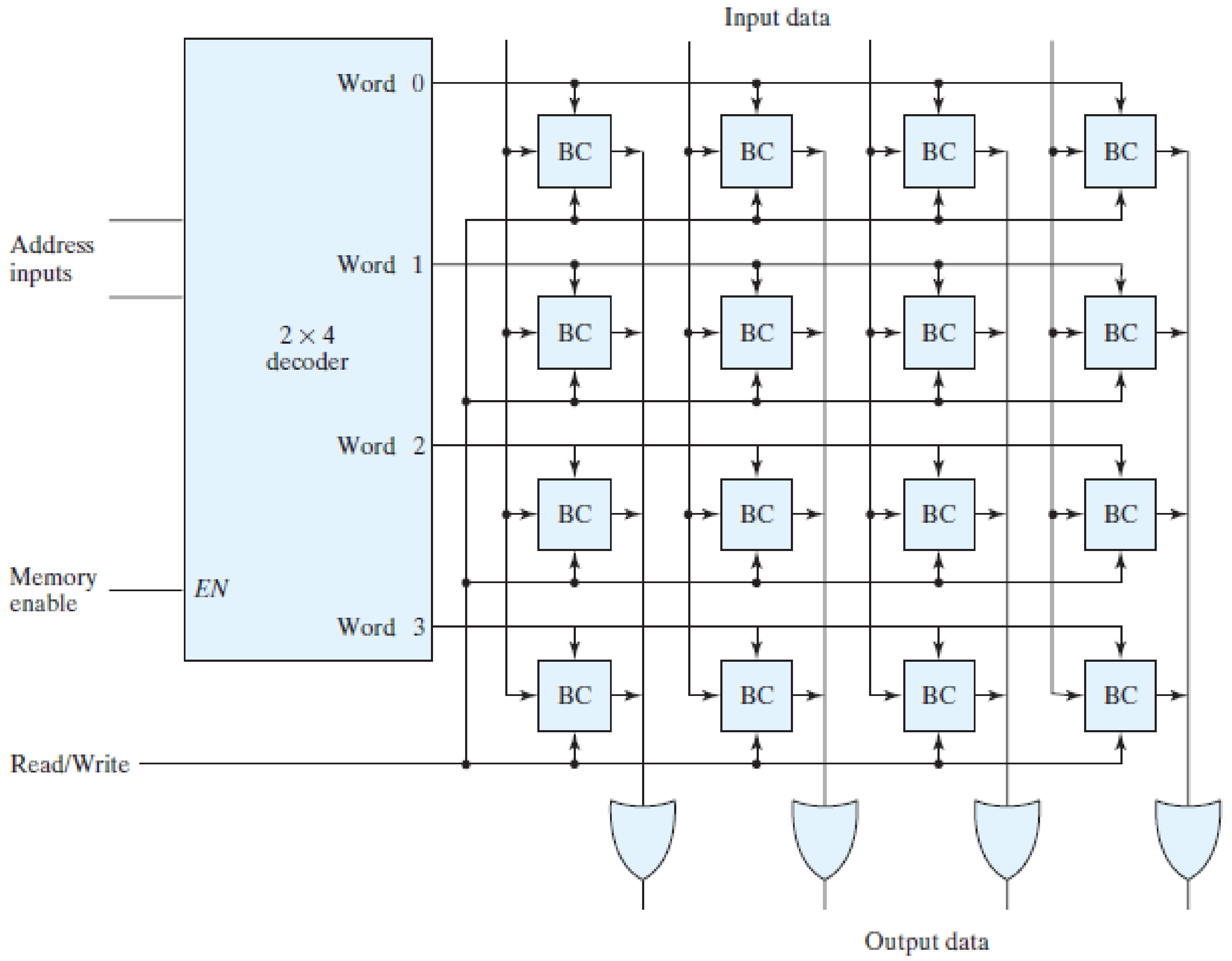


# 2-bit ROM of AND + OR

| $Addr_1$ | $Addr_0$ | AND | OR | $!Addr_1$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |



# 4 x 4 static RAM



# FPGAs

- Field Programmable Gate Arrays

- Tackle in this order:
  - Gate Arrays
  - Field Programmable

Sometimes

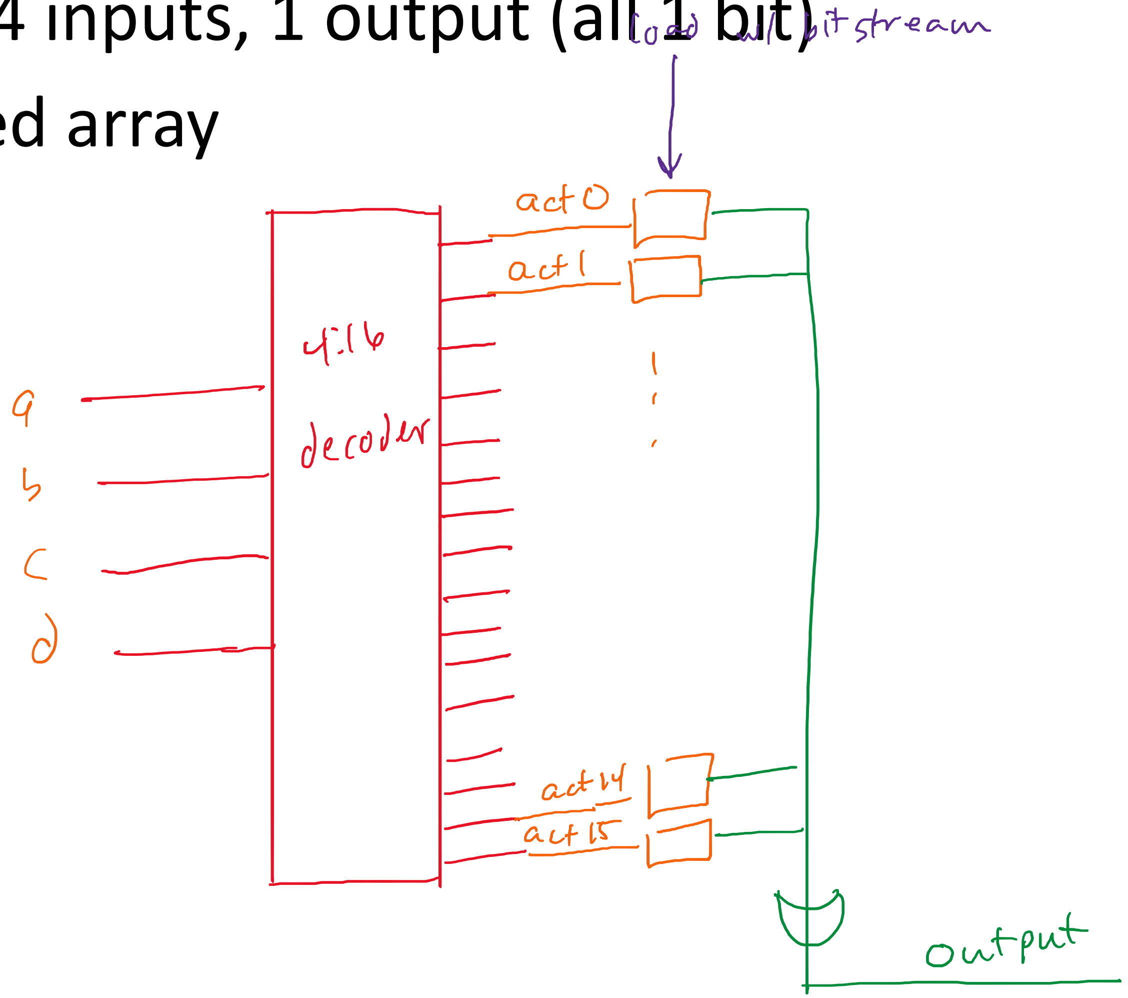
- Older technology / terminology

## Look Up Table (LUT)

- Assume:  4 inputs, 1 output (all 1 bit)
- RAM-based array

# Look Up Table (LUT)

- Assume: 4 inputs, 1 output (all 1 bit)
- RAM-based array



# Configurable Logic Block (CLB)

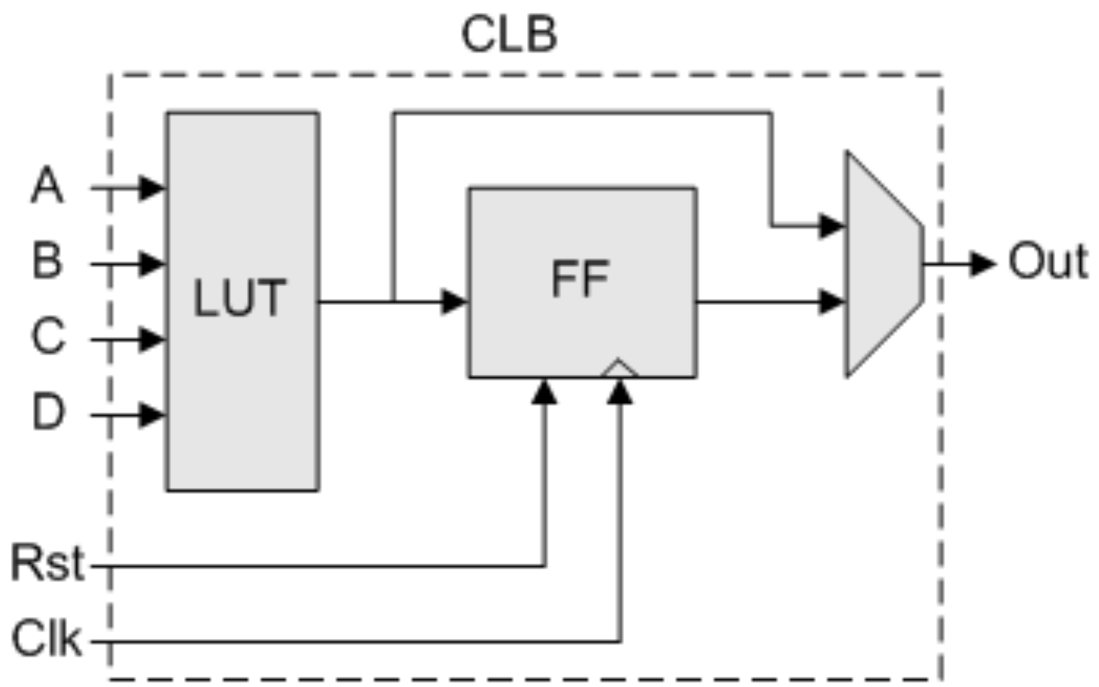


Fig.4 Example of Configurable Logic Cell.

## Configurable Logic Block (CLB)

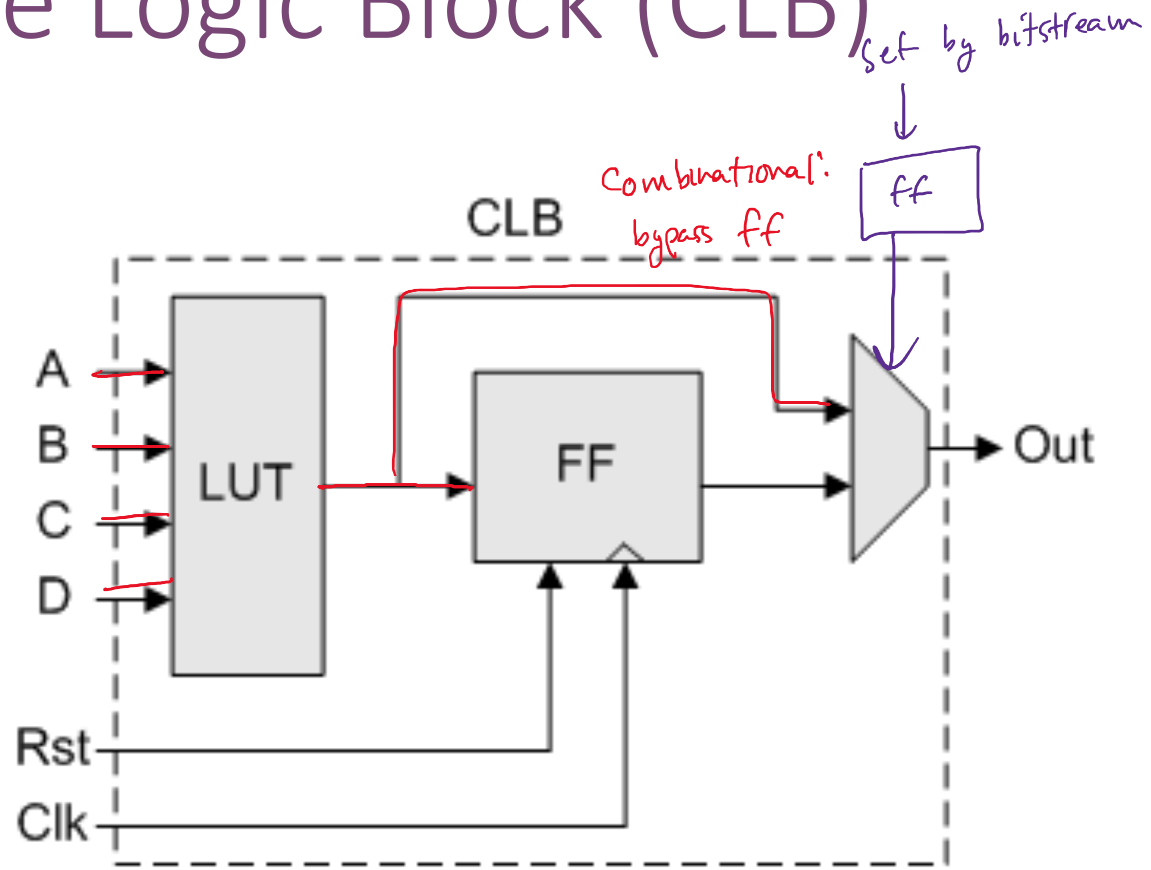


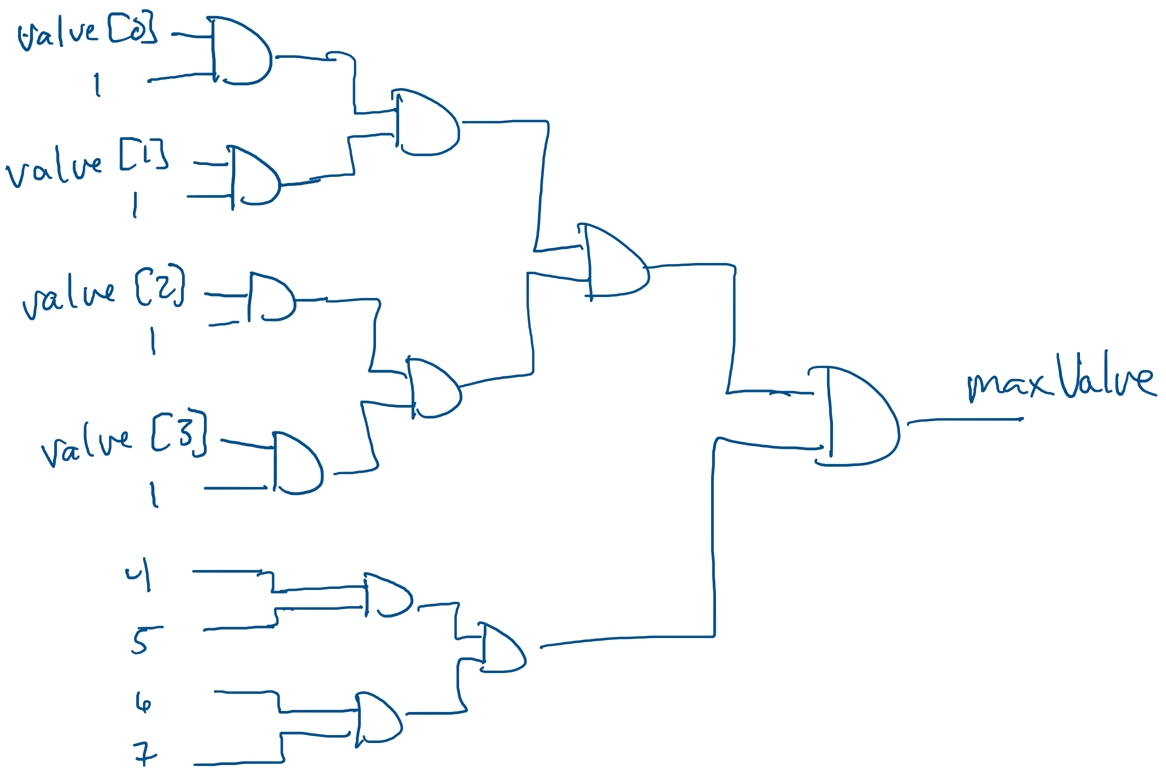
Fig.4 Example of Configurable Logic Cell.

## Connecting CLBs

```
wire [7:0] value;
wire maxValue = ( value == 8'hff );
```

# Connecting CLBs

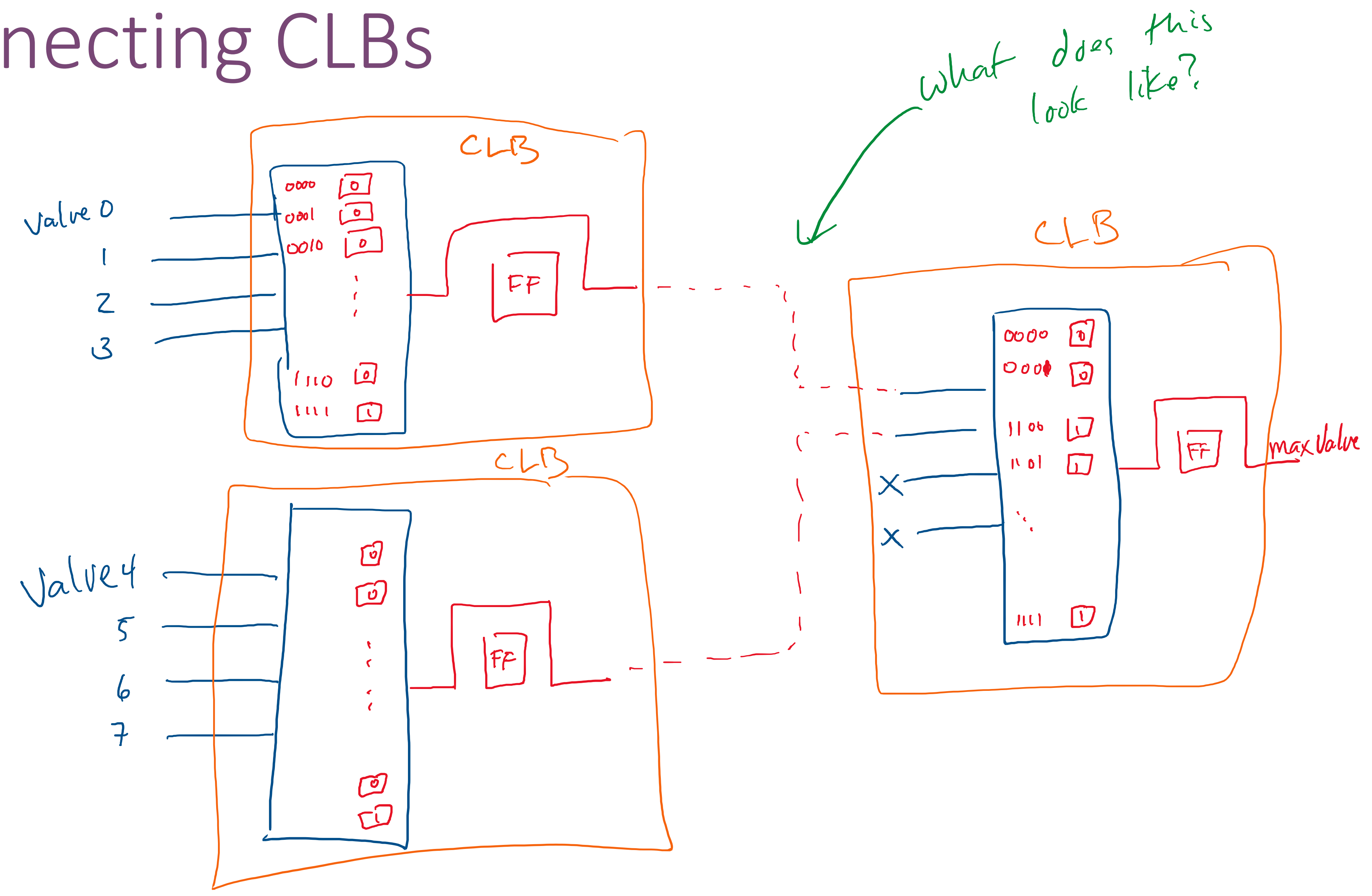
```
wire [7:0] value;
wire maxValue = ( value == 8'hff );
```



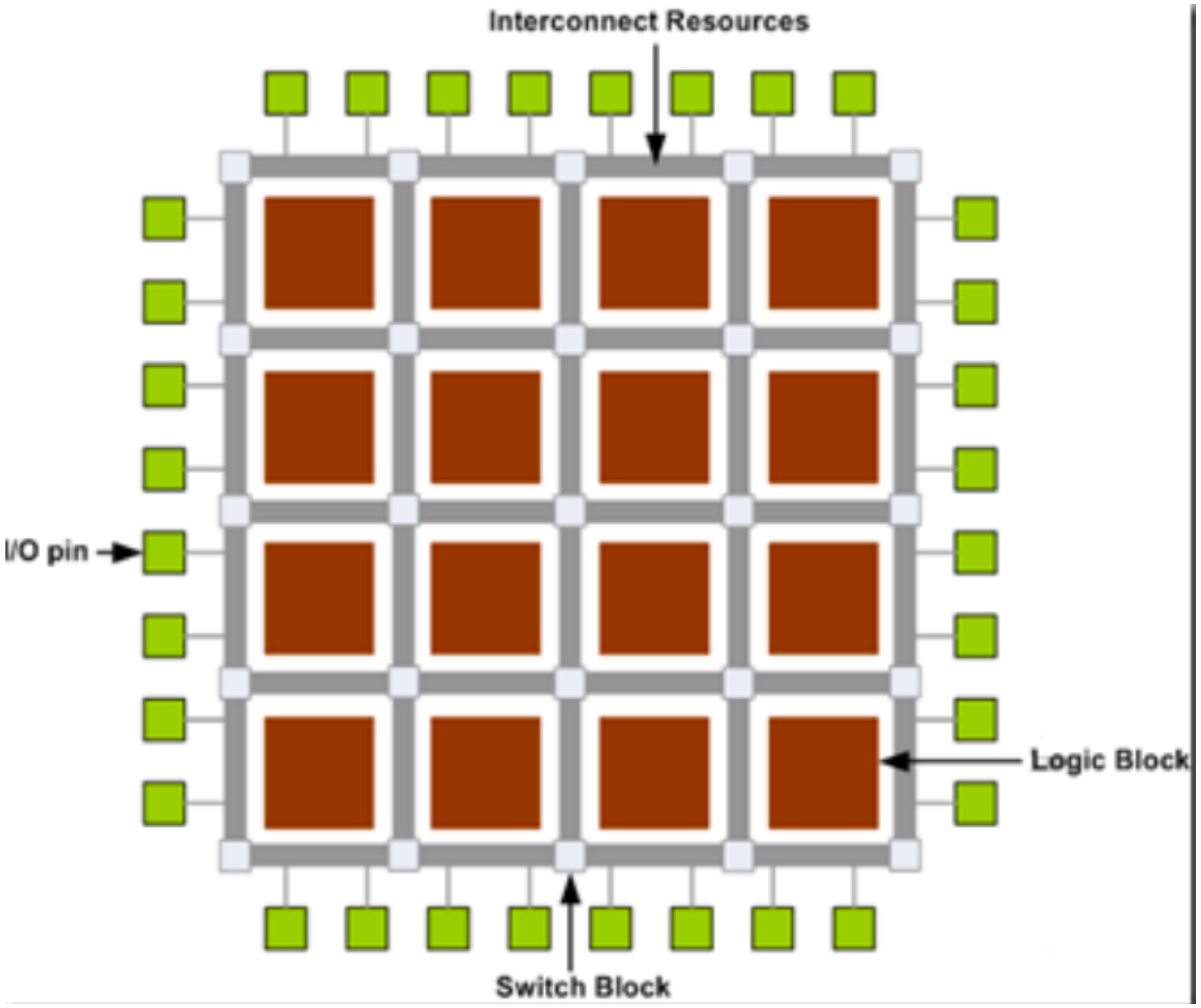
# Connecting CLBs

```
wire [7:0] value;
wire maxValue = ( value == 8'hff );
```

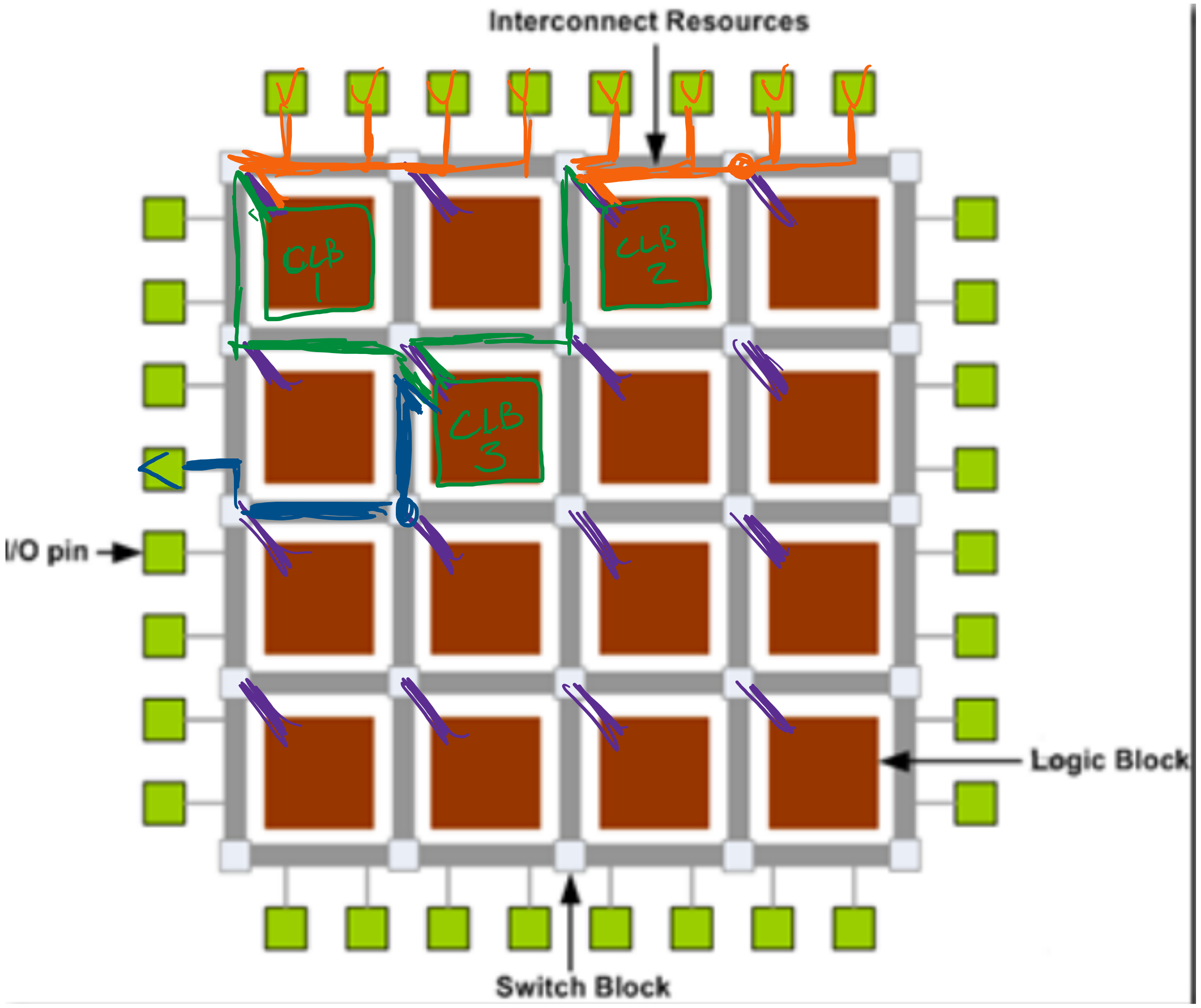
# Connecting CLBs



## CLB Interconnect



## CLB Interconnect



## FPGA w/BRAM

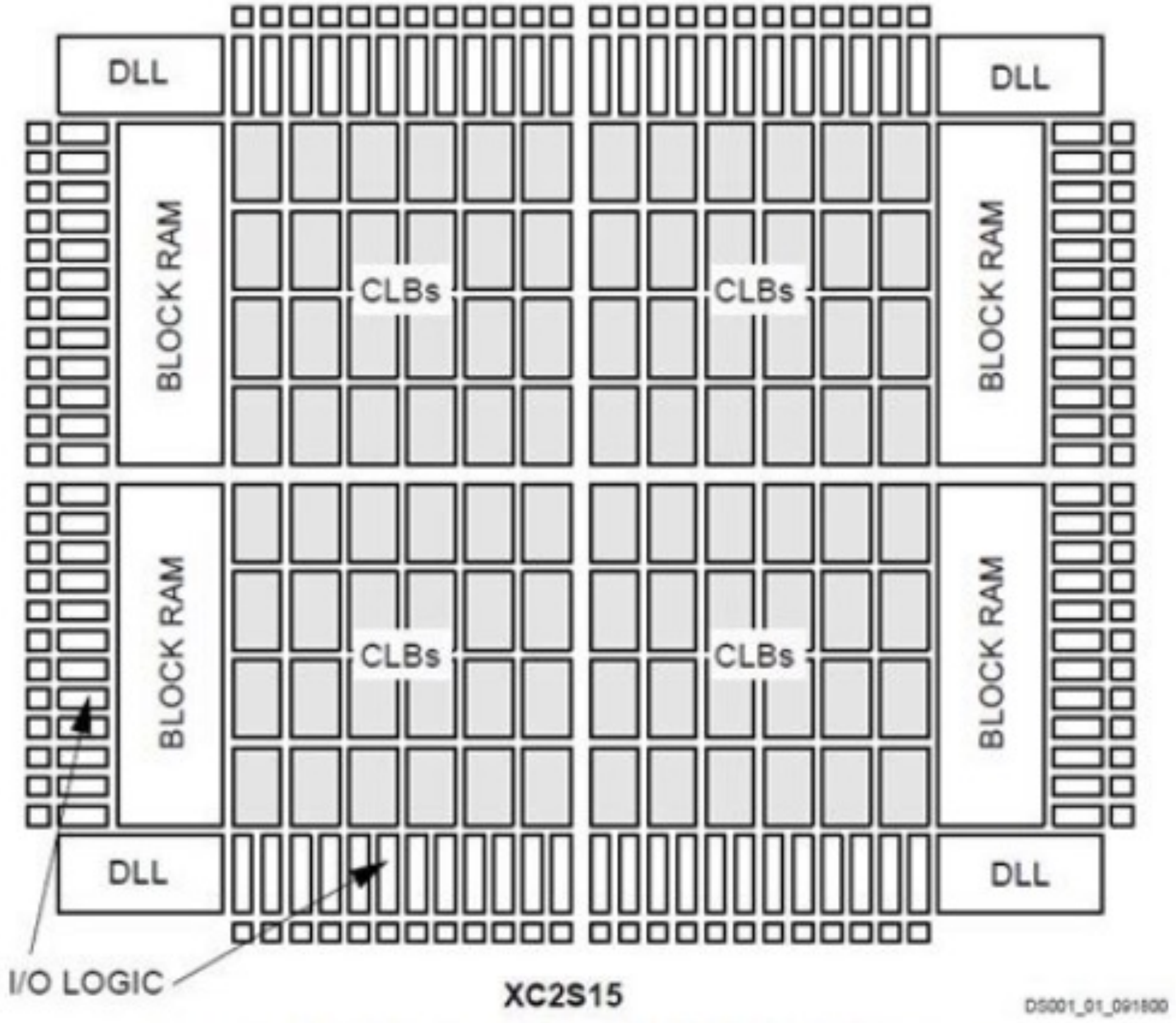


*Figure 1:* **Basic Spartan-II Family FPGA Block Diagram**

## More on FPGAs

- There is a *lot* more we could say about using FPGAs

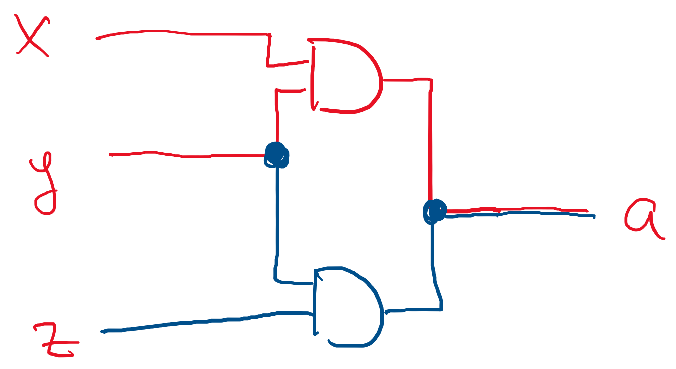
- Why synthesis takes so long: (one hots)
  - Remapping state machines
  - Behavioral Verilog -> Structural Verilog
  - Mapping to LUTs / CLBs
  - Layout of CLBs / IOs
  - Interconnection
  - Generating a configuration bitstream

## 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:

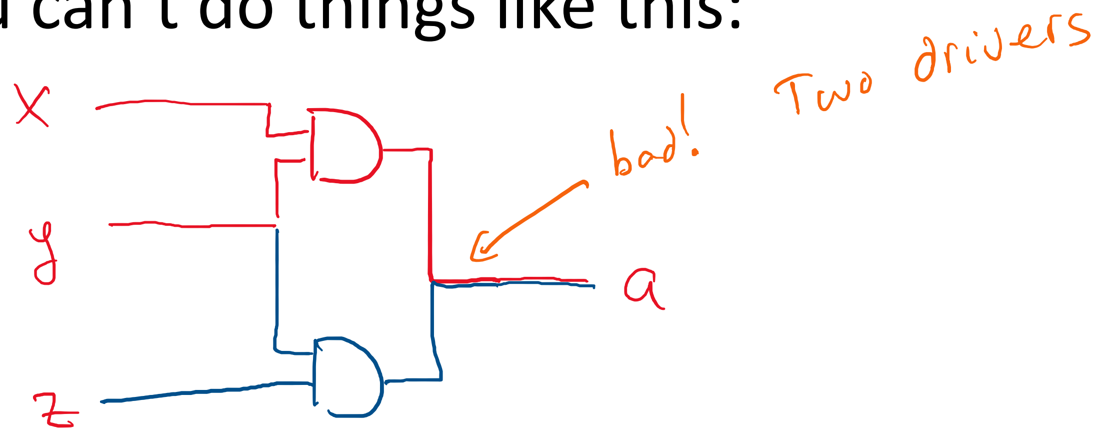


## Busses

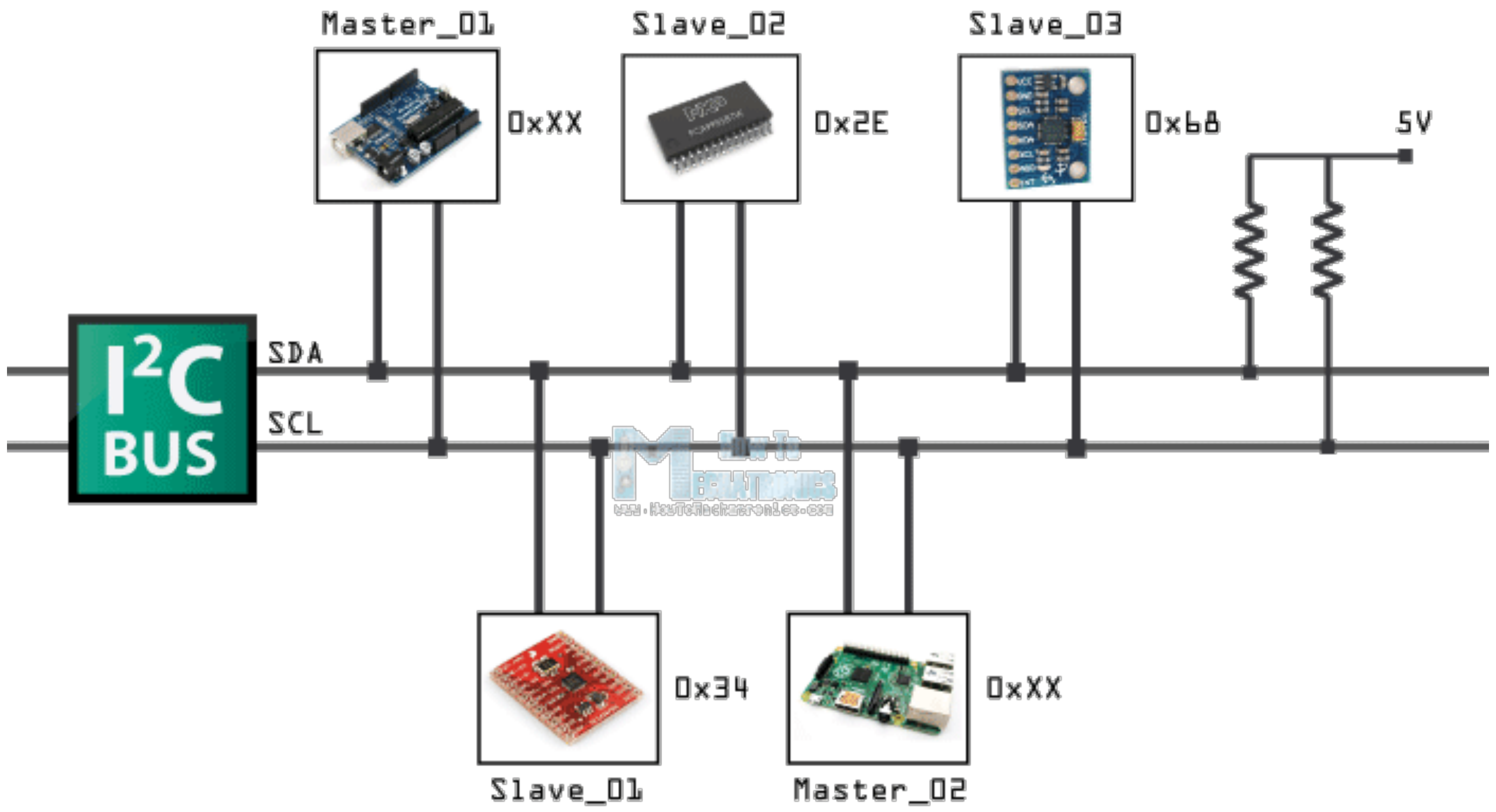
- Boolean Logic is bi-state:
  - 1: logical true
  - 0: logical false

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

- So you can't do things like this:



# Then how does this work?

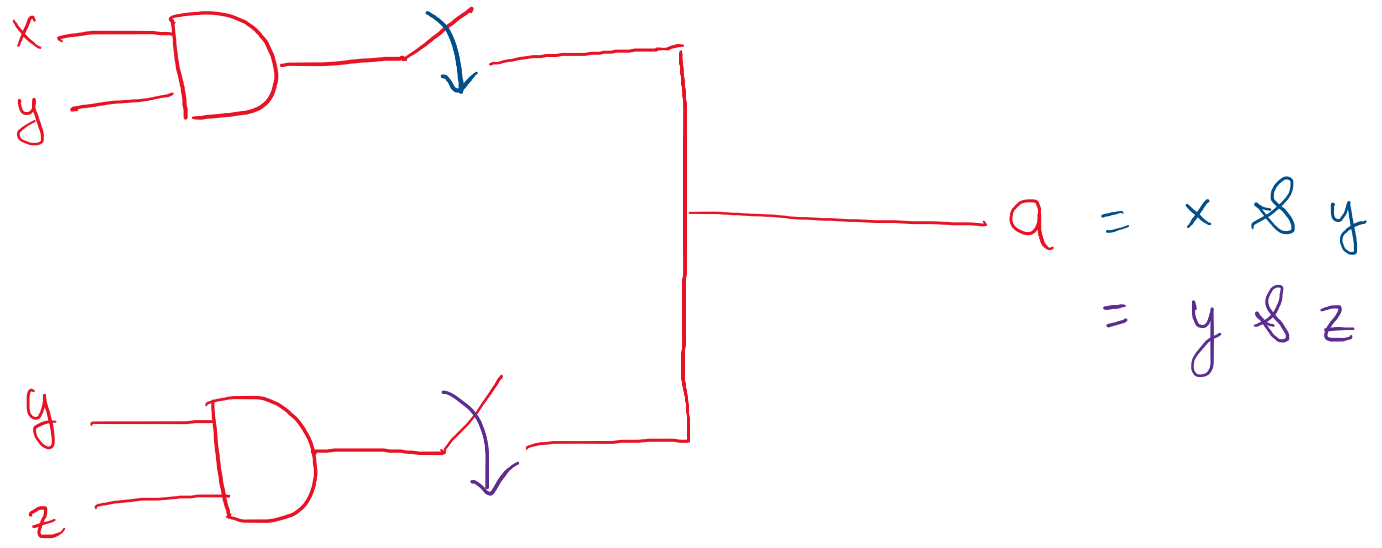


## Answer: A “Tri-State” Bus

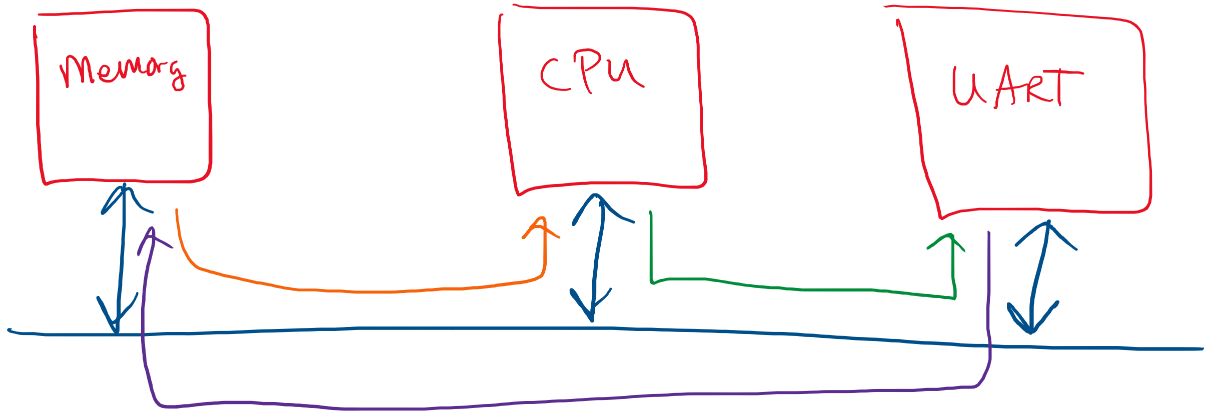
- “Tri-State” signals:
  - 1: this is logical true
  - 0: this is logical false
  - X: The simulation tools don’t know if it’s 1 or 0
  - Z: this is “high impedance”

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

# Tri-State logic



# Tri-State Bus

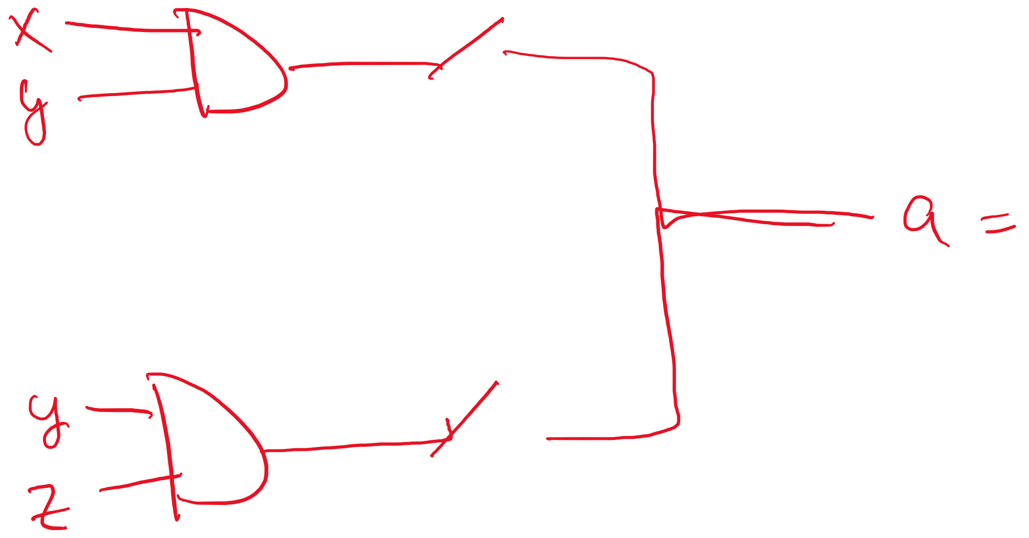


all on same bus

but not at same time

## Problems with Tri-State Logic

- What if two signals "drive" at once?

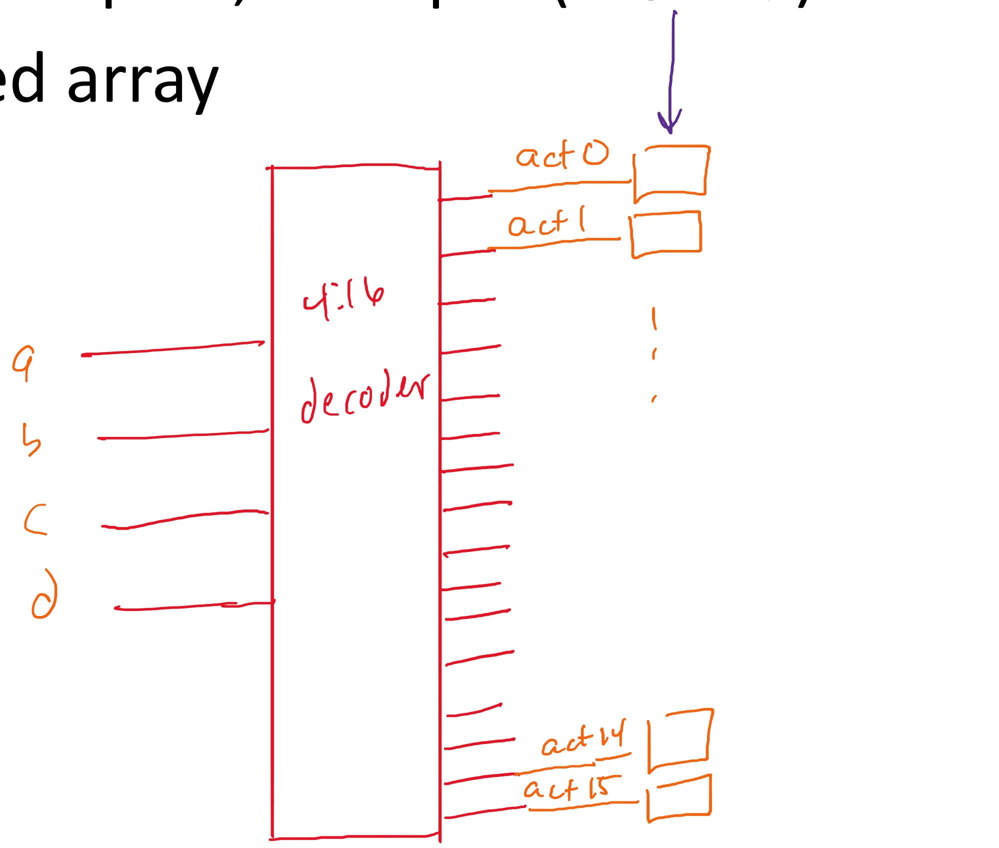


## Solution:  Don’t Do That!

Tri-State

# Look Up Table (LUT)

- Assume: 4 inputs, 1 output (all 1 bit)
- RAM-based array



## Next Time

- We start designing a CPU!

- Specifically:  Control / Datapath