

# L12: Reconfigurable Logic Architectures



## Acknowledgements:

- Lecture material adapted from R. Katz, G. Borriello, “Contemporary Logic Design” (second edition), Copyright 2005 Prentice-Hall/Pearson Education.
- Frank Honore

- **Discrete devices: relays, transistors (1940s-50s)**
- **Discrete logic gates (1950s-60s)**
- **Integrated circuits (1960s-70s)**
  - e.g. TTL packages: Data Book for 100's of different parts
- **Gate Arrays (IBM 1970s)**
  - Transistors are pre-placed on the chip & Place and Route software puts the chip together automatically – only program the interconnect (mask programming)
- **Software Based Schemes (1970's- present)**
  - Run instructions on a general purpose core
- **Programmable Logic (1980's to present)**
  - A chip that be reprogrammed after it has been fabricated
  - Examples: **PALs**, EPROM, EEPROM, PLDs, **FPGAs**
  - Excellent support for mapping from Verilog
- **ASIC Design (1980's to present)**
  - Turn Verilog directly into layout using a library of standard cells
  - Effective for high-volume and efficient use of silicon area

- **Logic blocks**

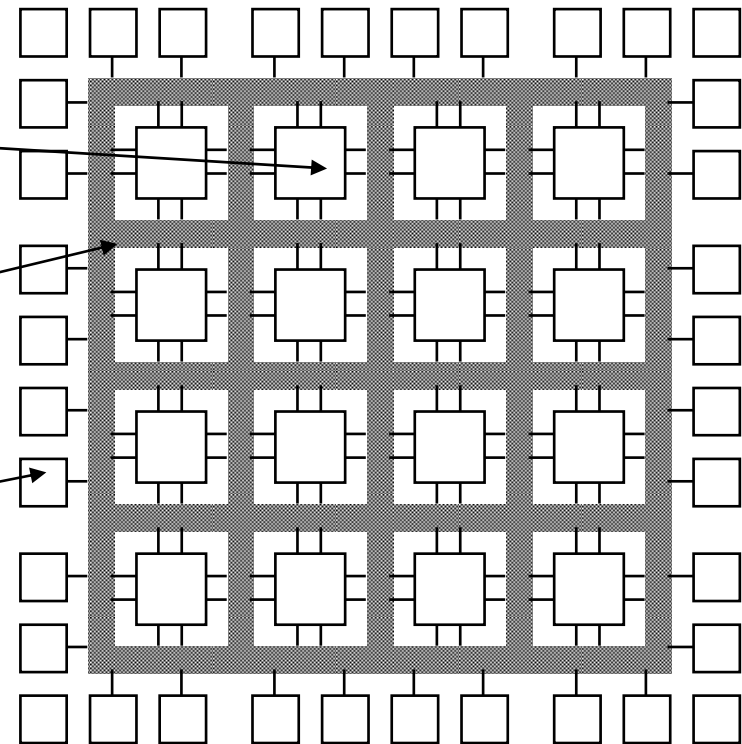
- To implement combinational and sequential logic

- **Interconnect**

- Wires to connect inputs and outputs to logic blocks

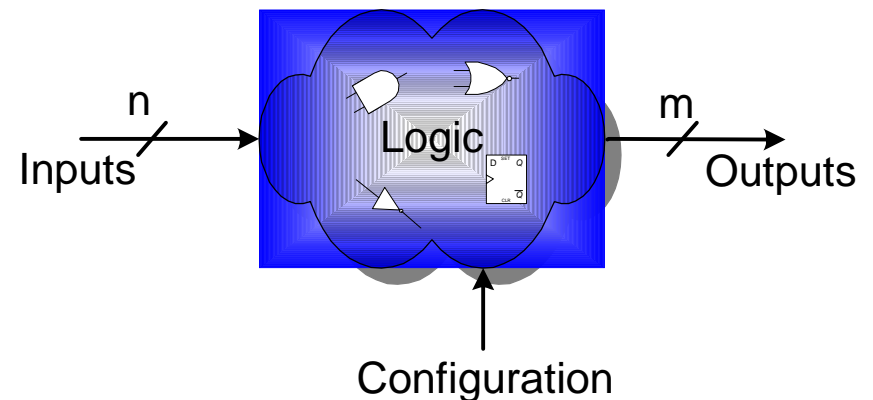
- **I/O blocks**

- Special logic blocks at periphery of device for external connections

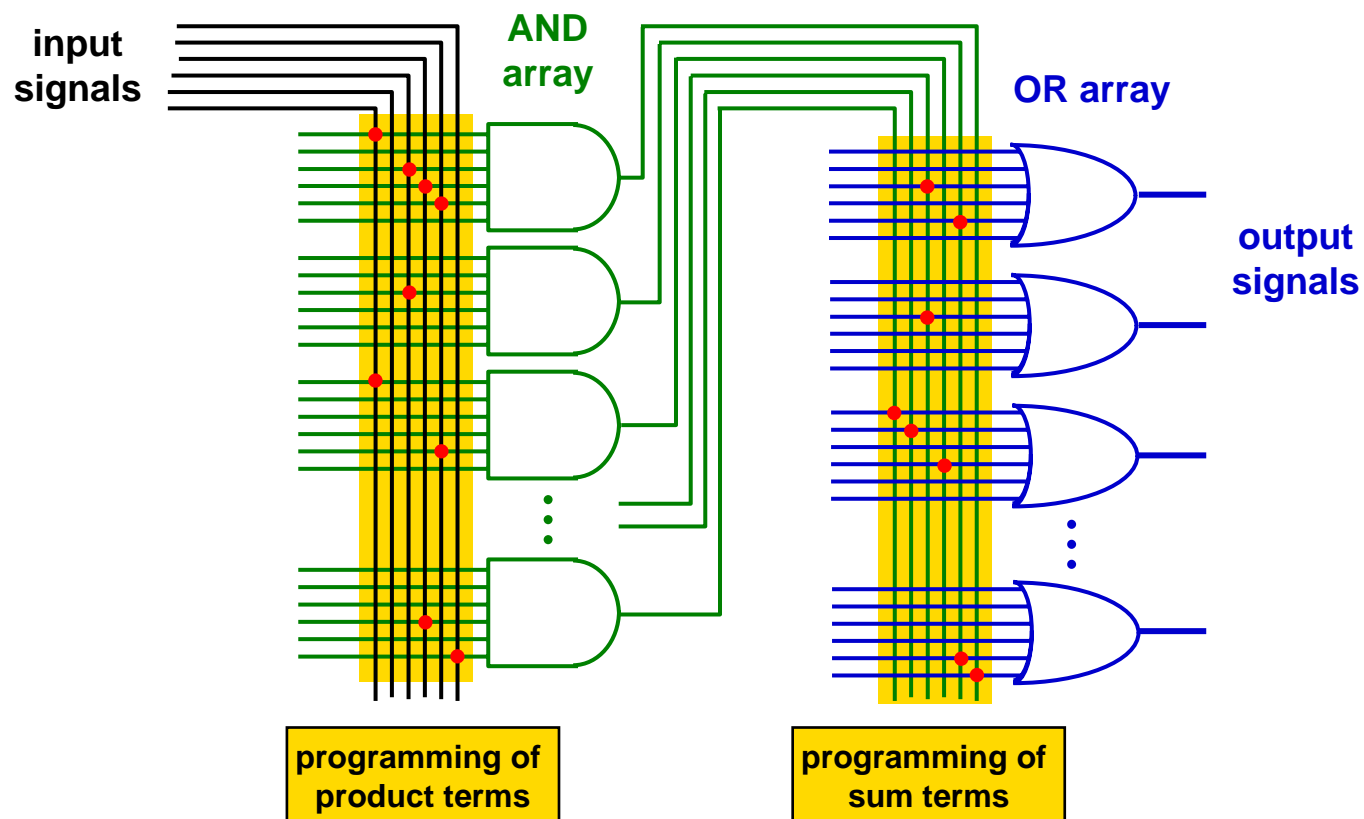


- **Key questions:**

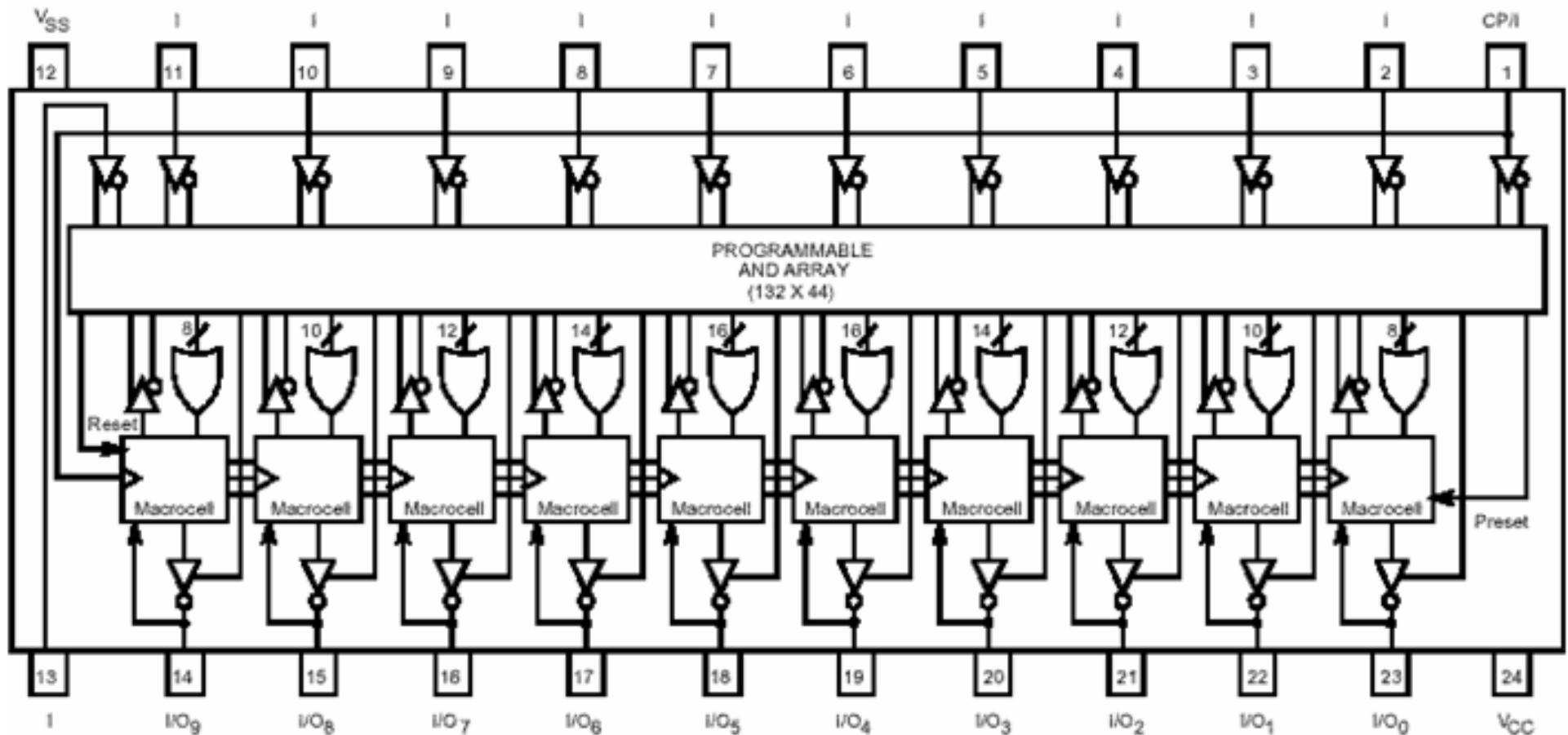
- How to make logic blocks programmable? (after chip has been fabbed!)
- What should the logic granularity be?
- How to make the wires programmable? (after chip has been fabbed!)
- Specialized wiring structures for local vs. long distance routes?
- How many wires per logic block?



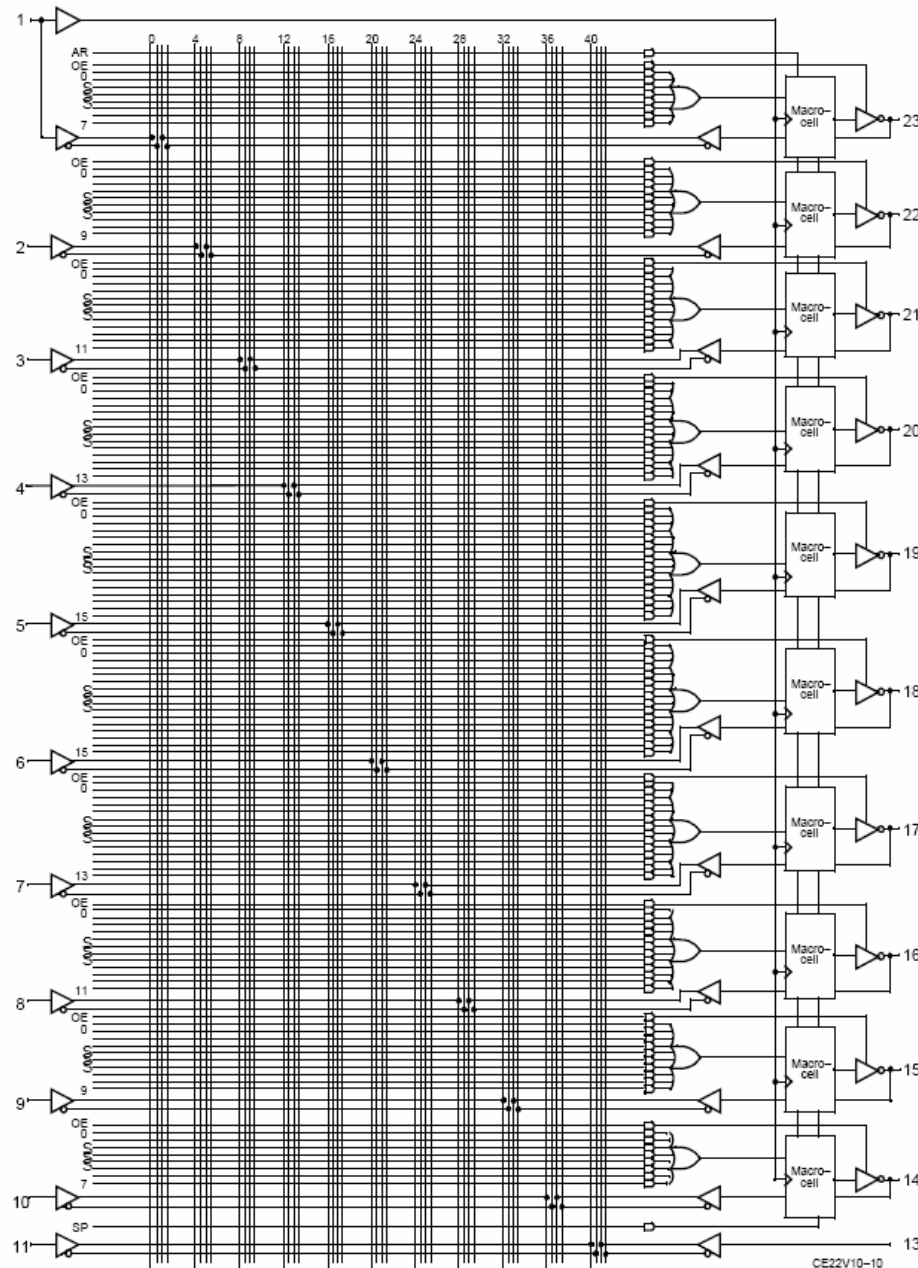
- Based on the fact that any combinational logic can be realized as a sum-of-products
- PALs feature an array of AND-OR gates with programmable interconnect



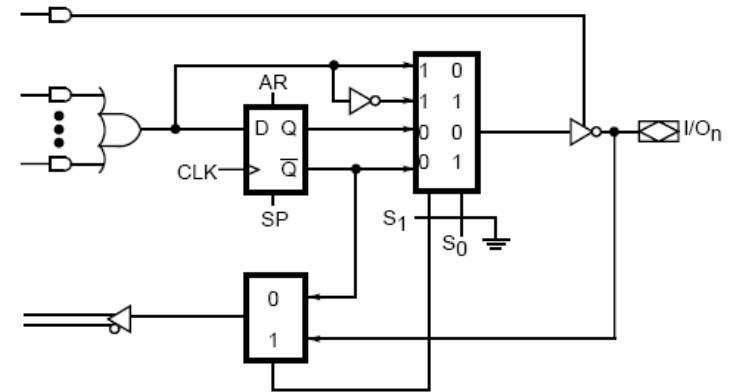
- Each input pin (and its complement) sent to the AND array
- OR gates for each output can take 8-16 product terms, depending on output pin
- “Macrocell” block provides additional output flexibility...



Functional Logic Diagram for PALCE22V10 **From Cypress**



**From Lattice Semiconductor**



S <sub>1</sub>	S <sub>0</sub>	Output Configuration
0	0	Registered/Active Low
0	1	Registered/Active High
1	0	Combinational/active low
1	1	Combinational/active high

0 = Programmed EE bit  
1 = Erased (charged) EE bit

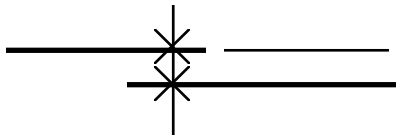
- **Outputs may be registered or combinational, positive or inverted**

Rows of programmable logic building blocks

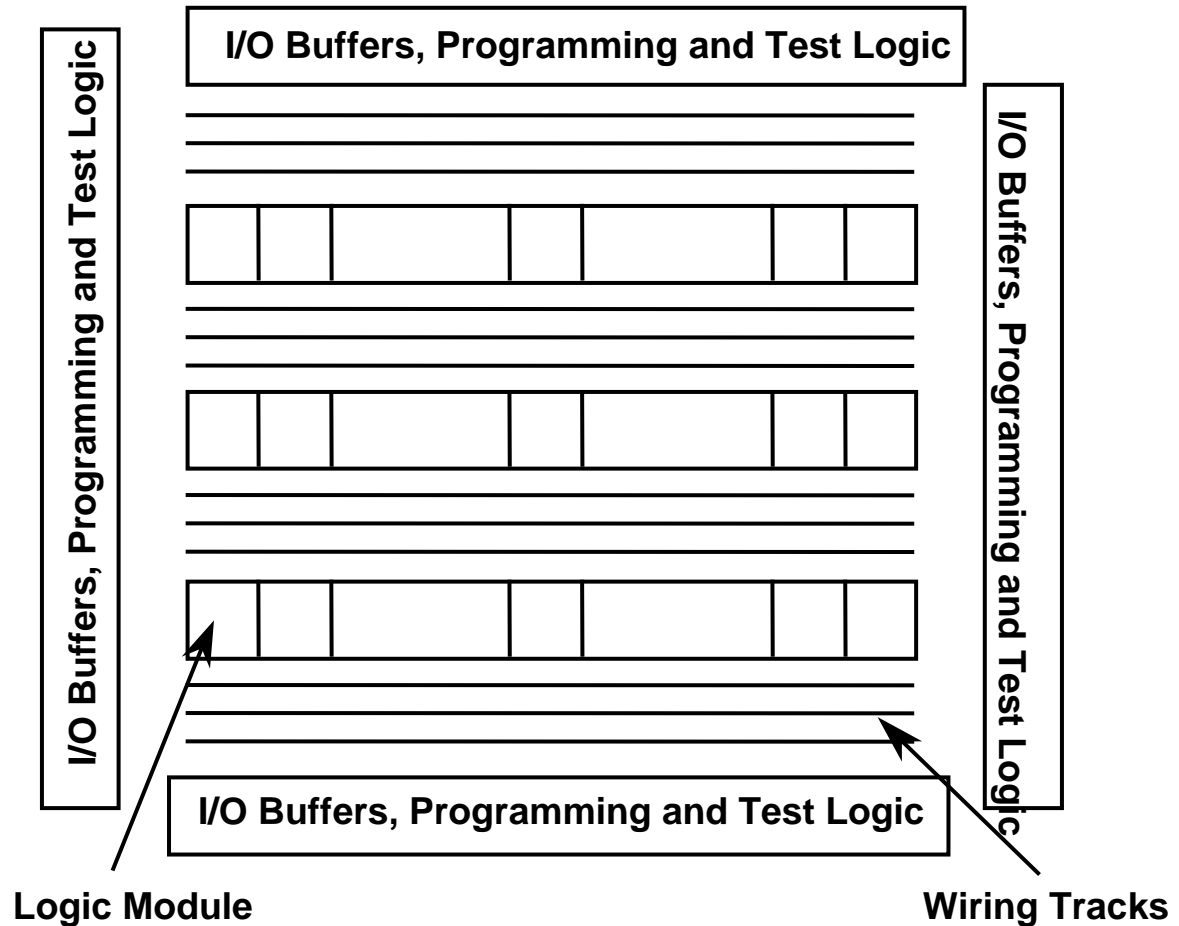
+

rows of interconnect

Anti-fuse Technology:  
Program Once



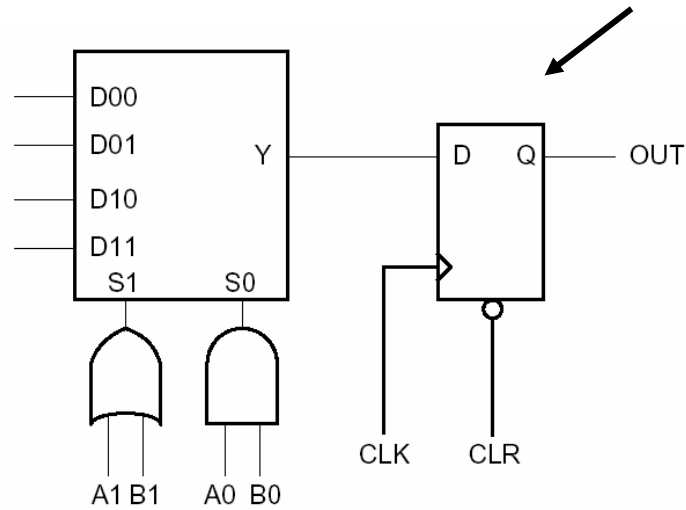
Use Anti-fuses to build up long wiring runs from short segments



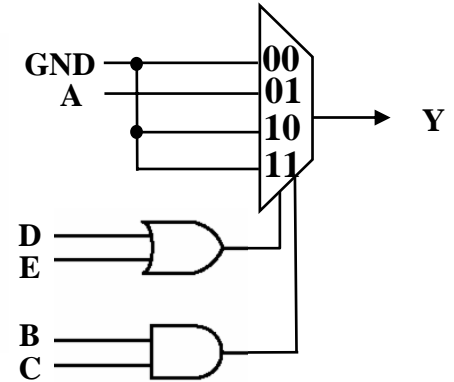
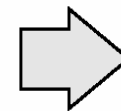
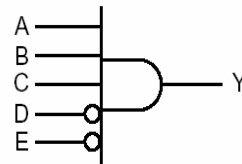
8 input, single output combinational logic blocks

FFs constructed from discrete cross coupled gates

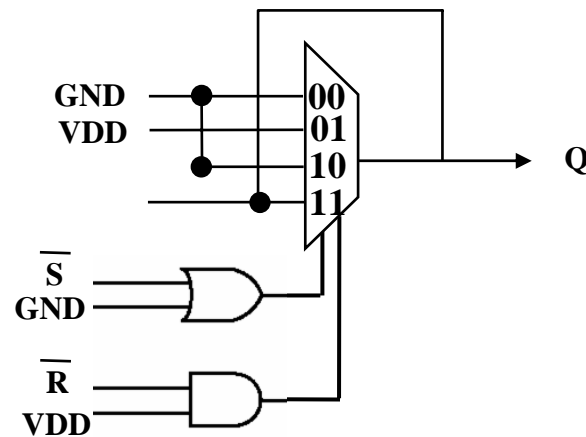
Combinational block does not have the output FF



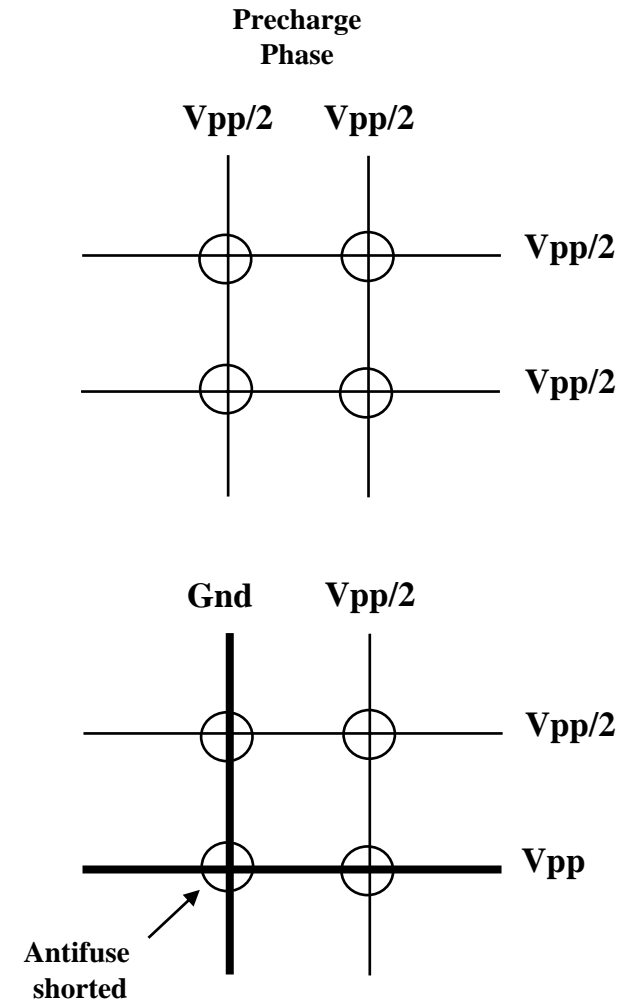
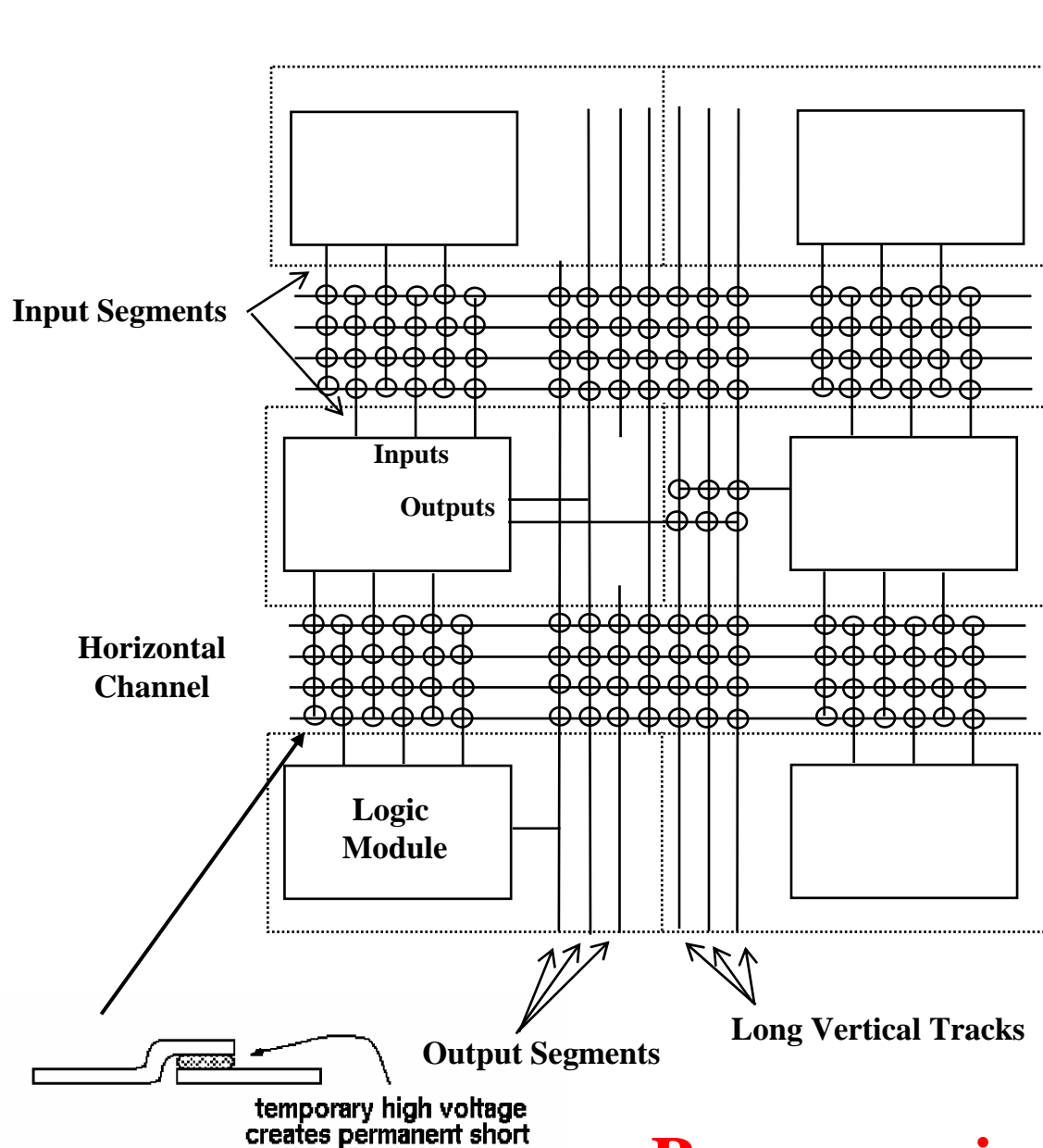
Example Gate Mapping



S-R Flip-Flop



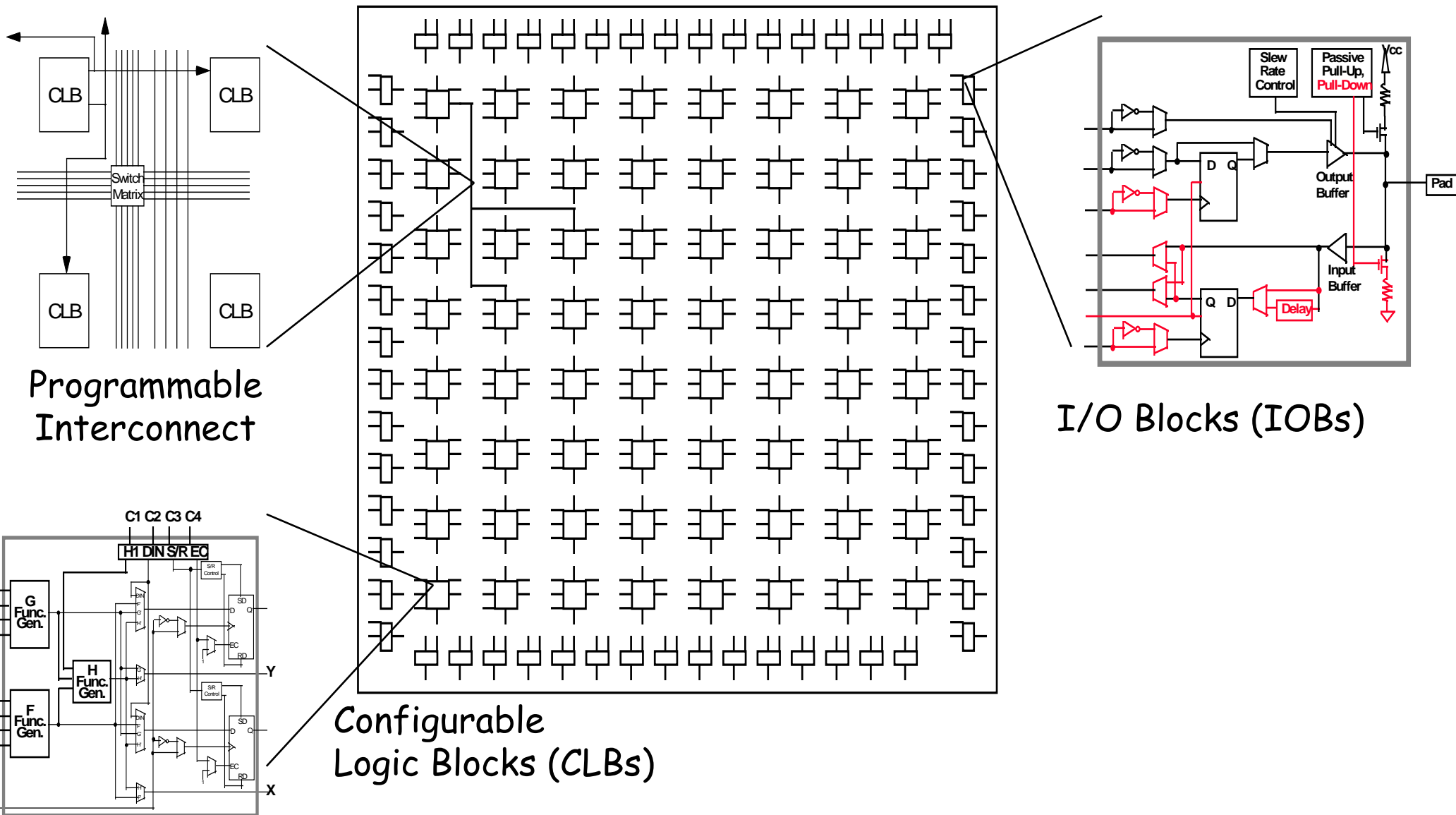


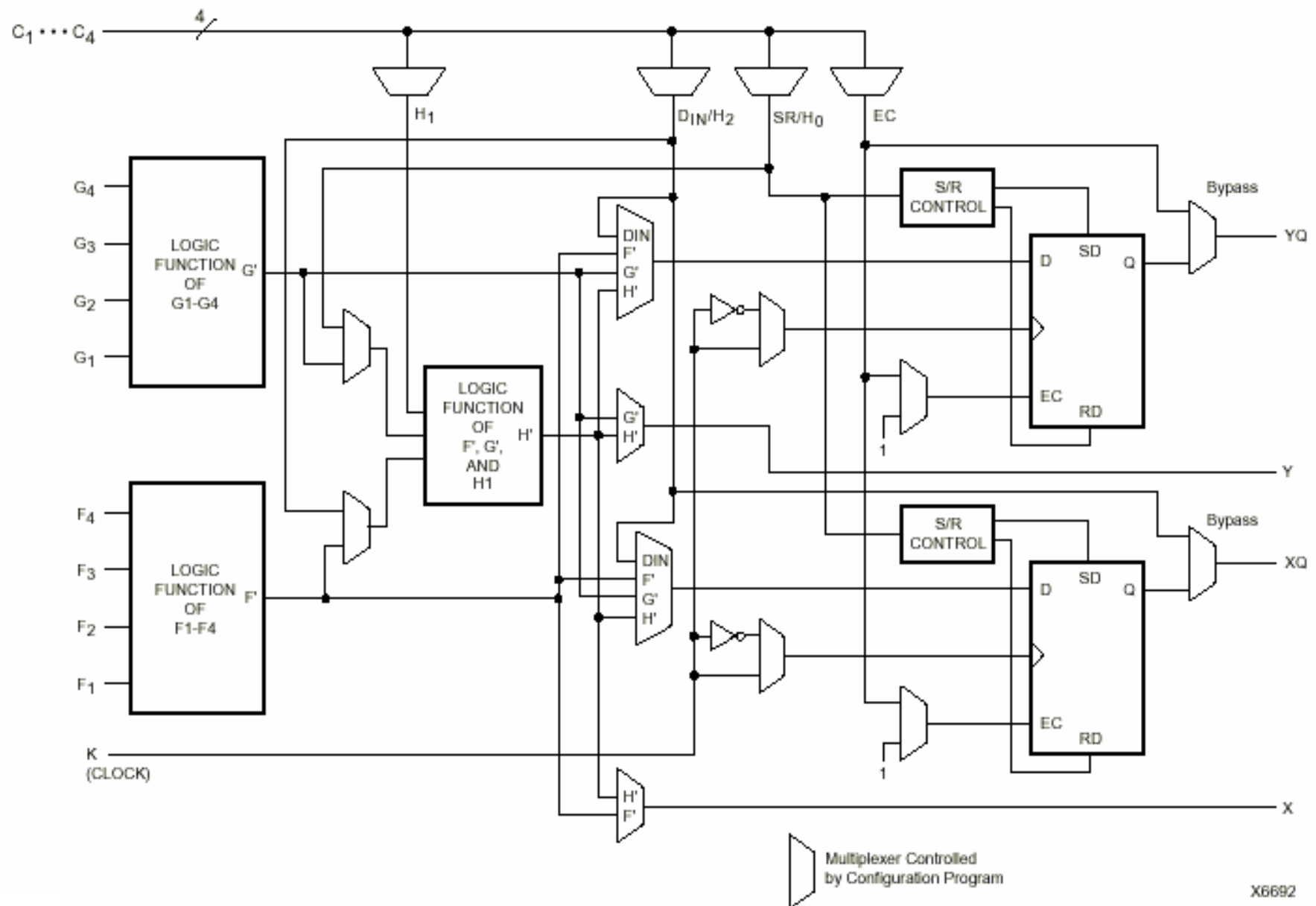


Programming an Antifuse

**Programming is Permanent (one time)**

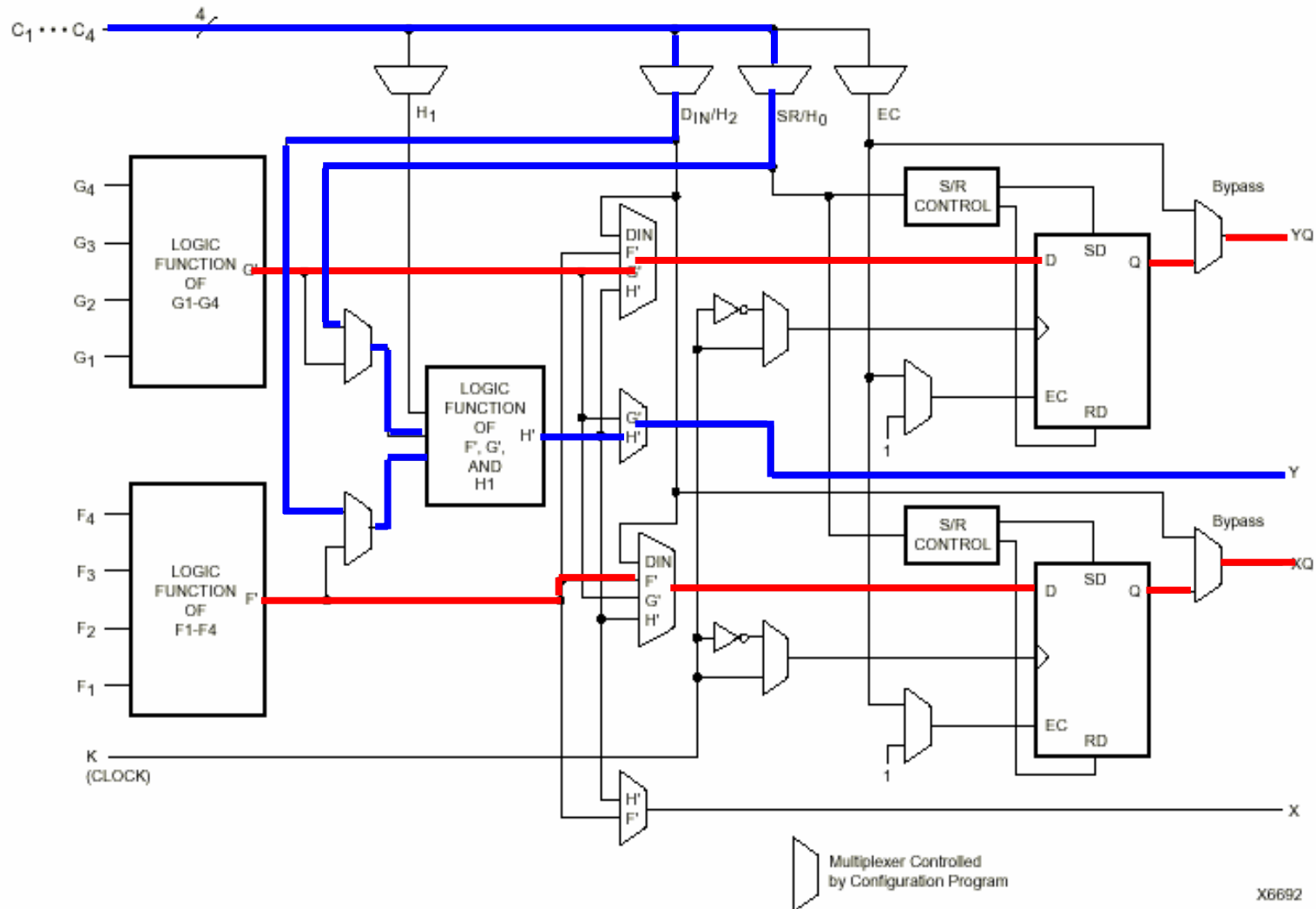
# RAM Based Field Programmable Logic - Xilinx



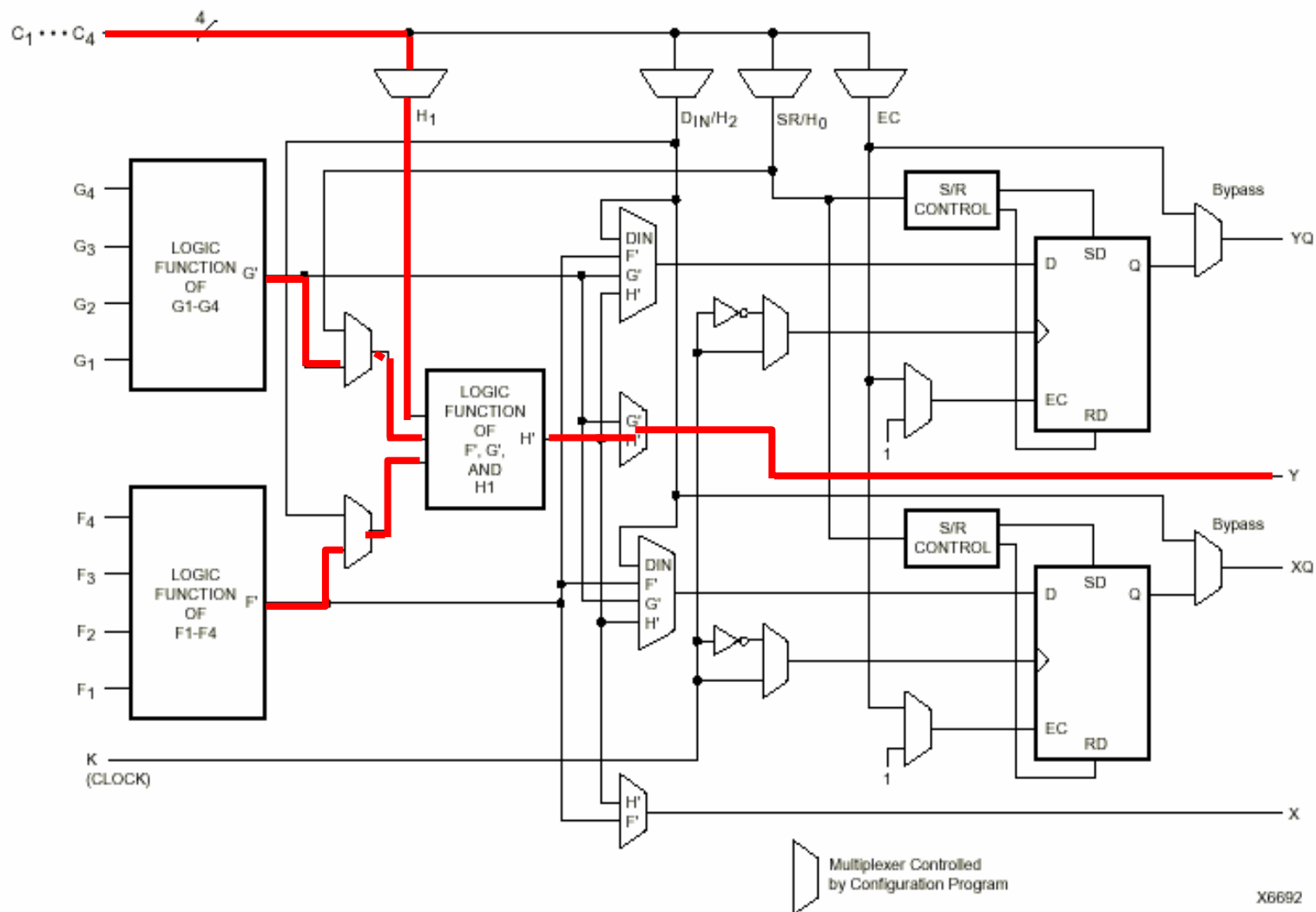


Simplified Block Diagram of XC4000 Series CLB (RAM and Carry Logic functions not shown)

# Two 4-input Functions, Registered Output and a Two Input Function

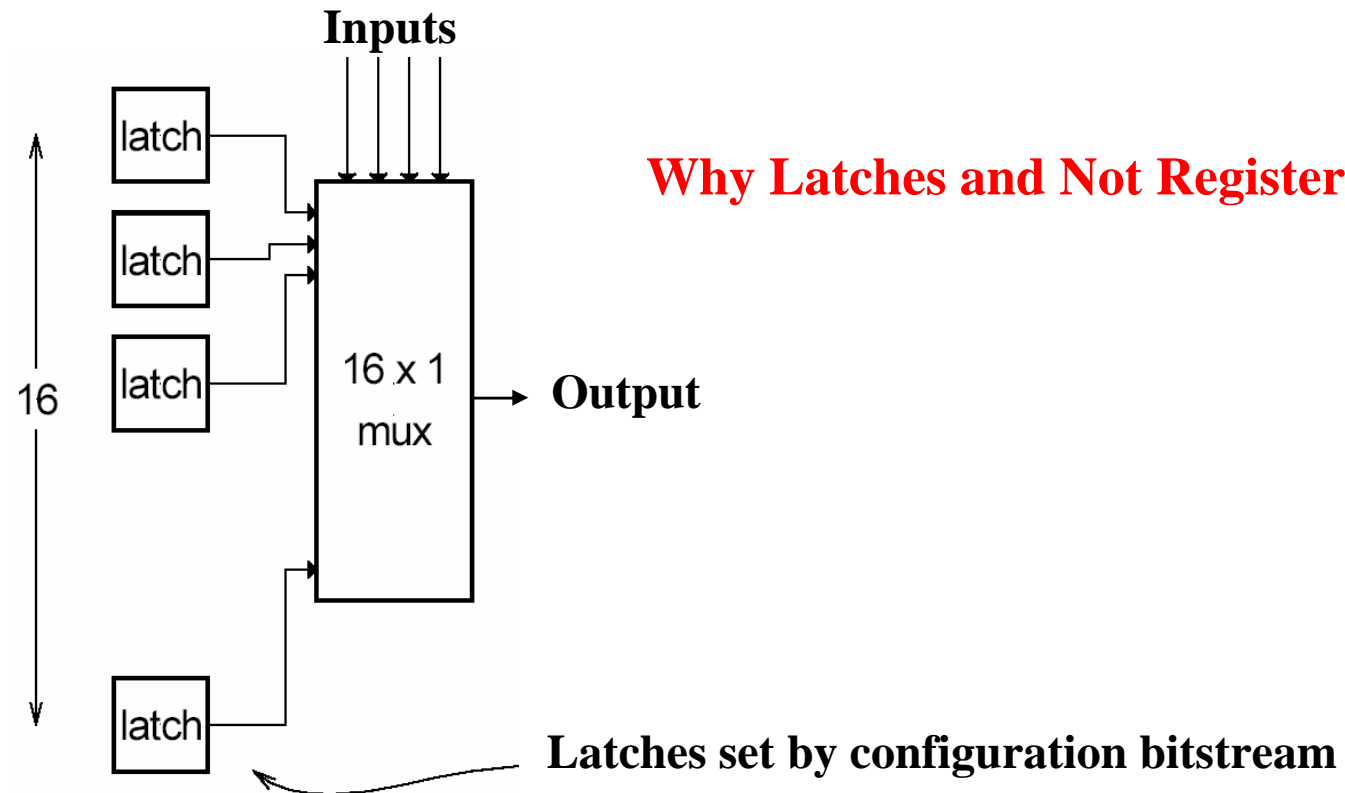


Simplified Block Diagram of XC4000 Series CLB (RAM and Carry Logic functions not shown)



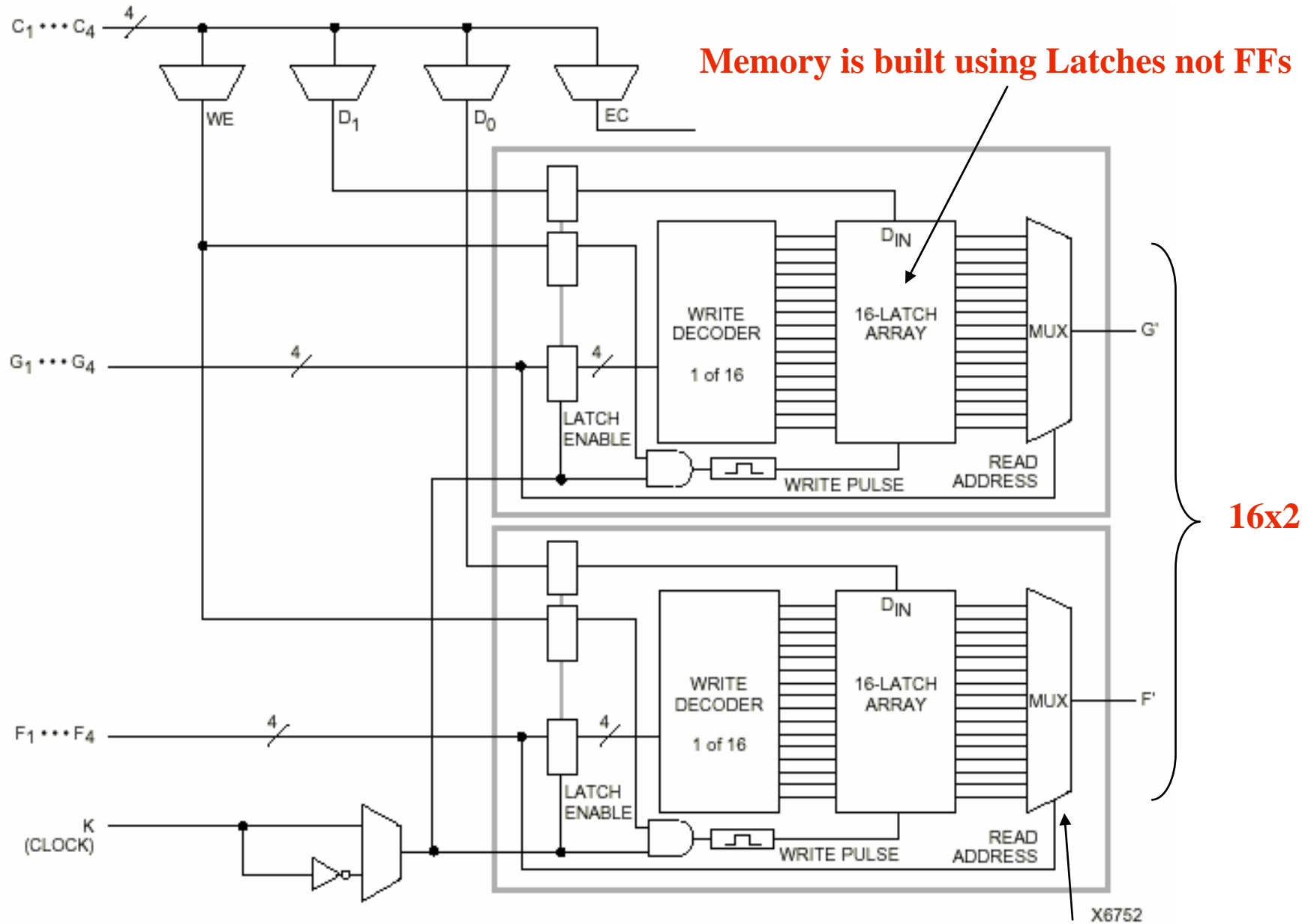
Simplified Block Diagram of XC4000 Series CLB (RAM and Carry Logic functions not shown)

- N-LUT direct implementation of a truth table: any function of n-inputs.
- N-LUT requires  $2^N$  storage elements (latches)
- N-inputs select one latch location (like a memory)

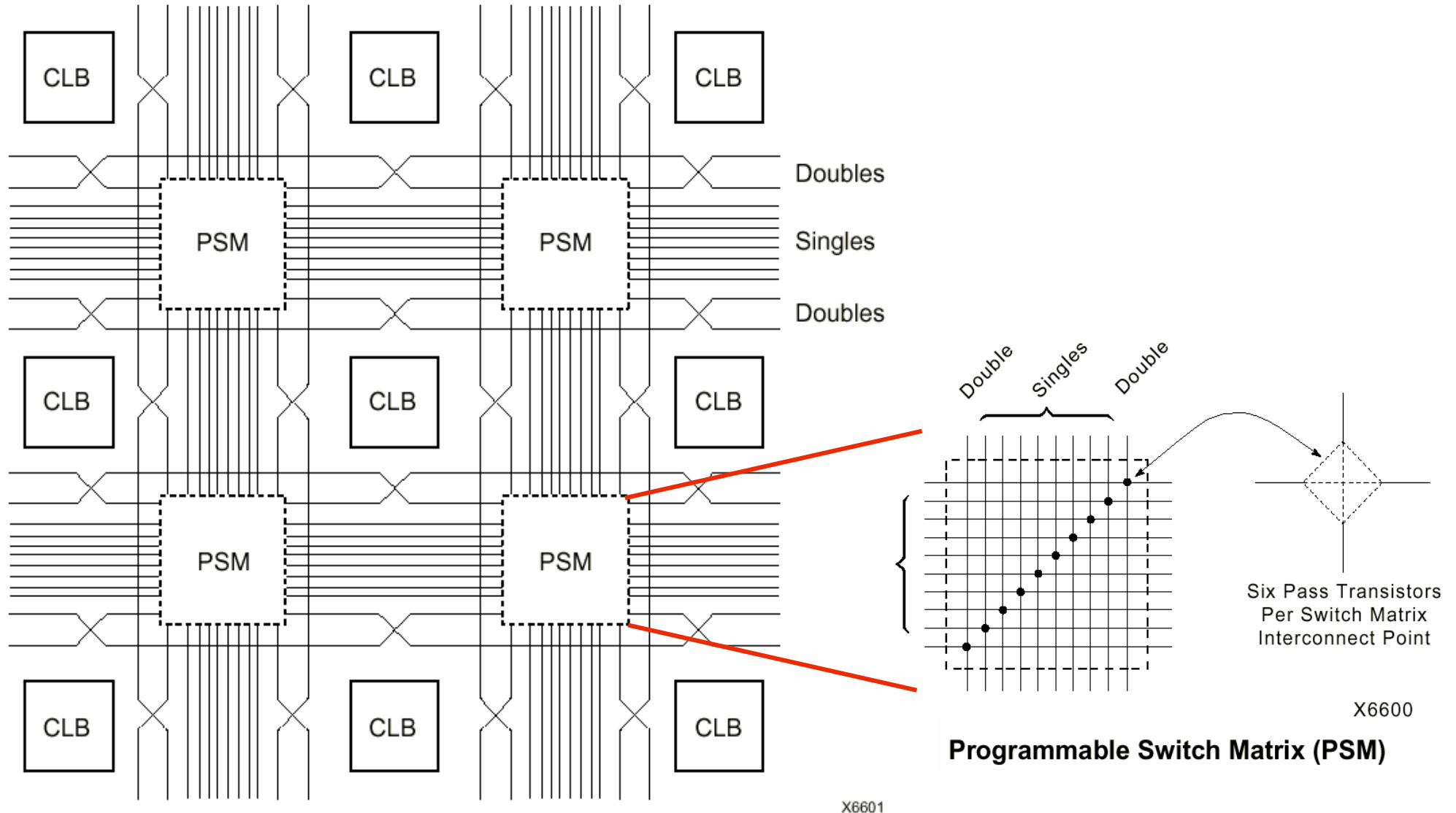


**Why Latches and Not Registers?**

**4LUT example**

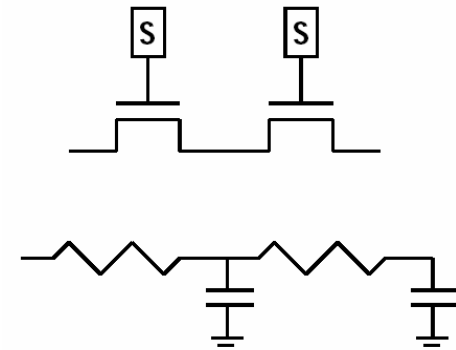
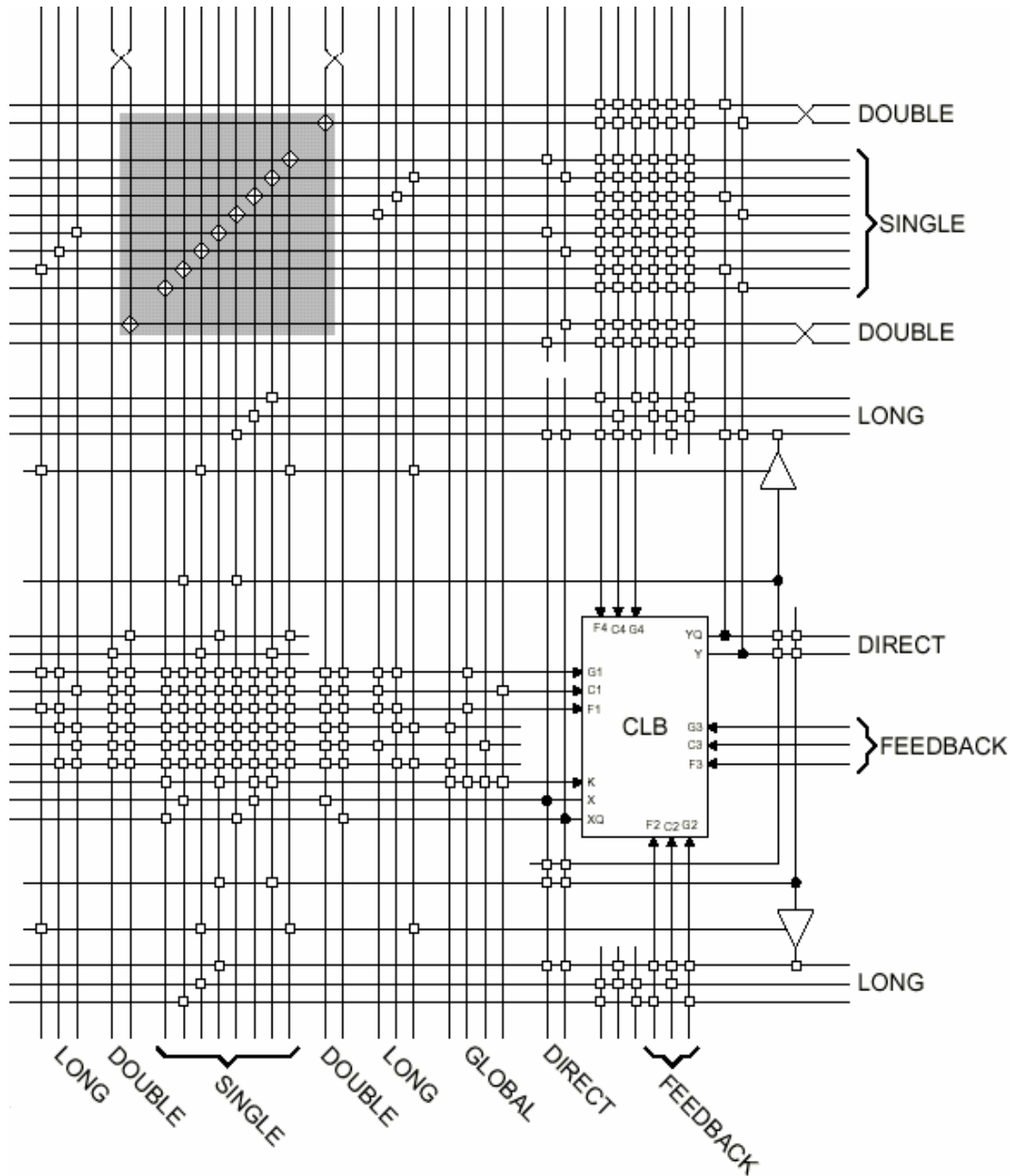


**Read is same a LUT Function!**

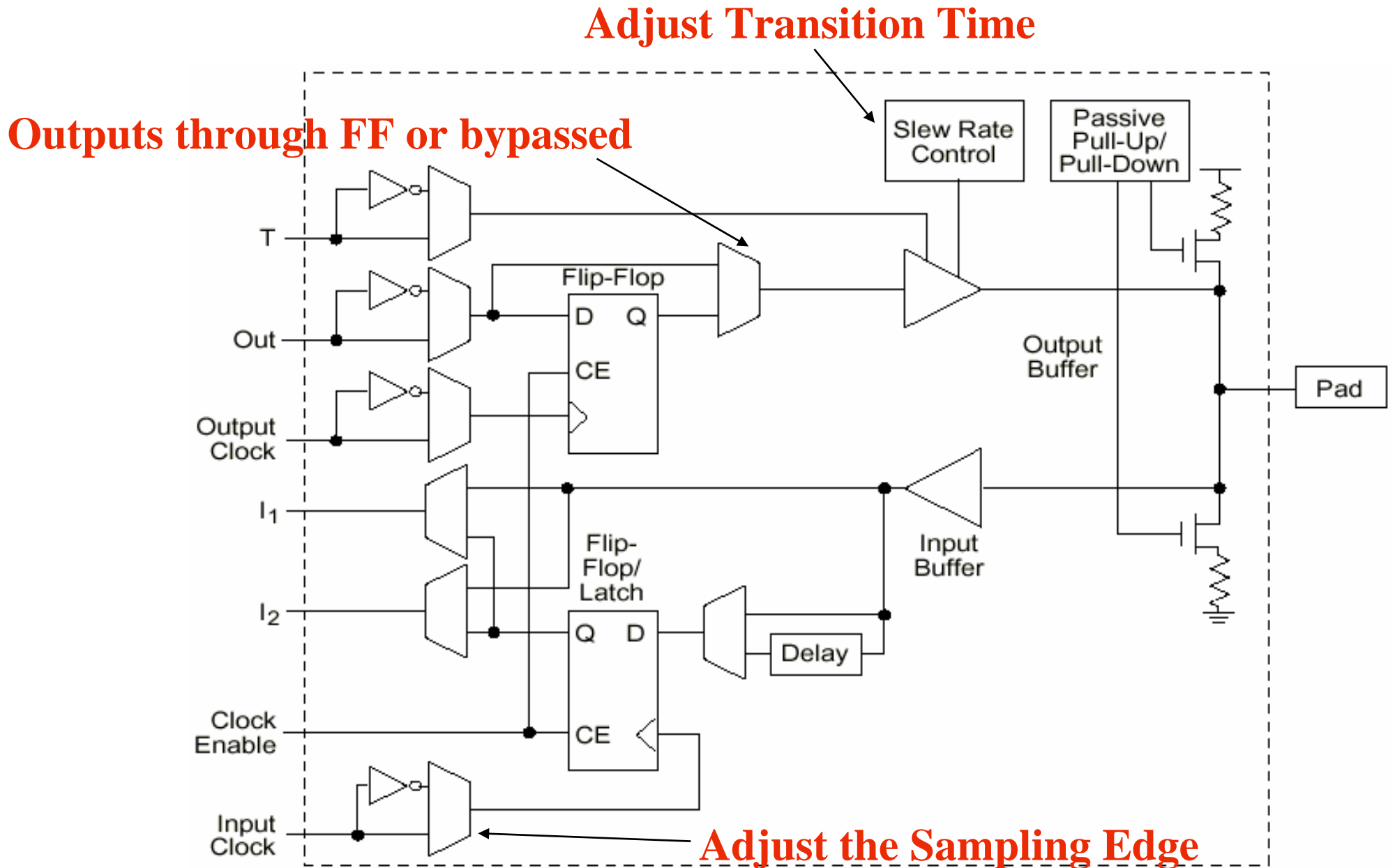


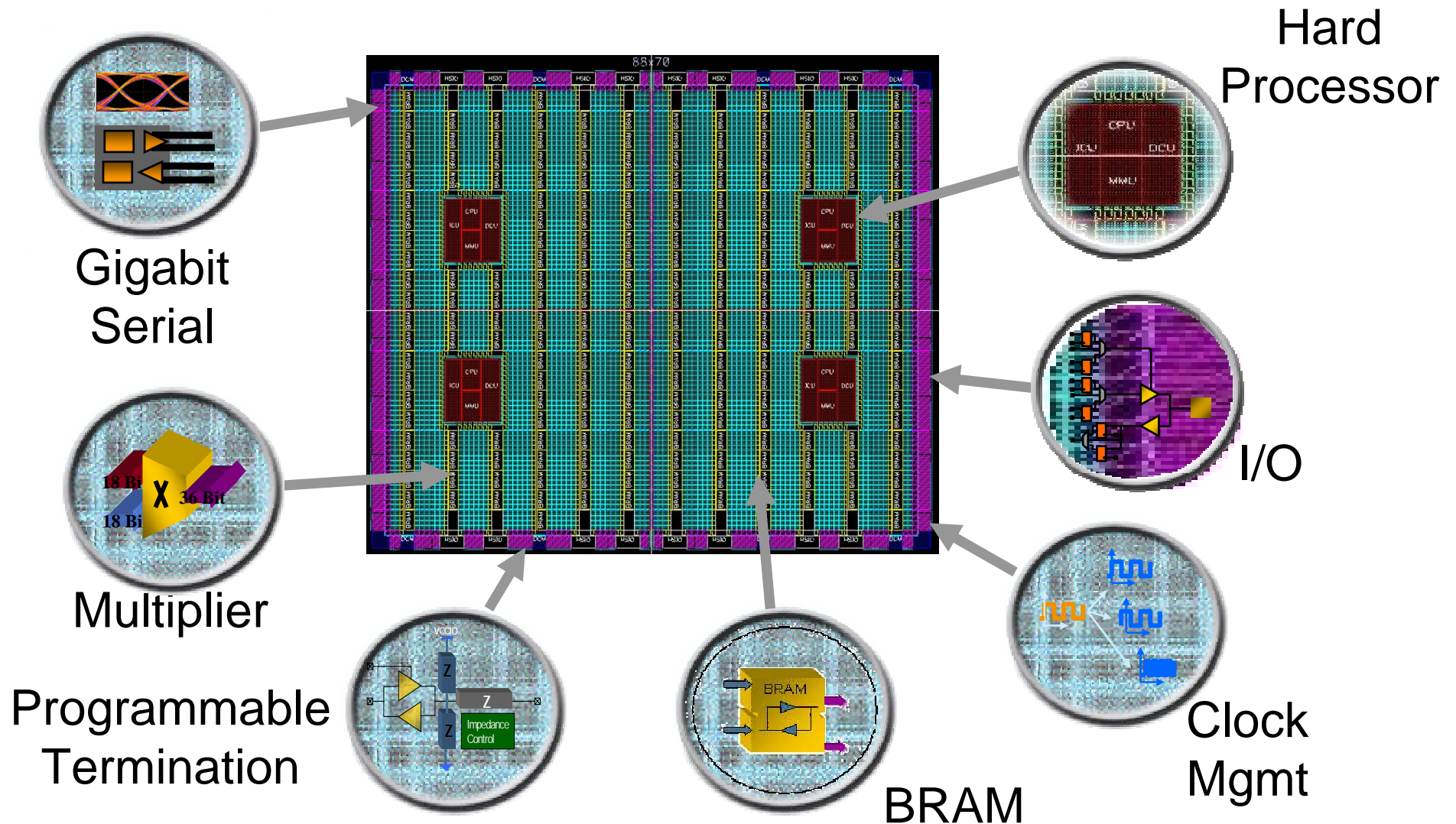
## Single- and Double-Length Lines, with Programmable Switch Matrices (PSMs)



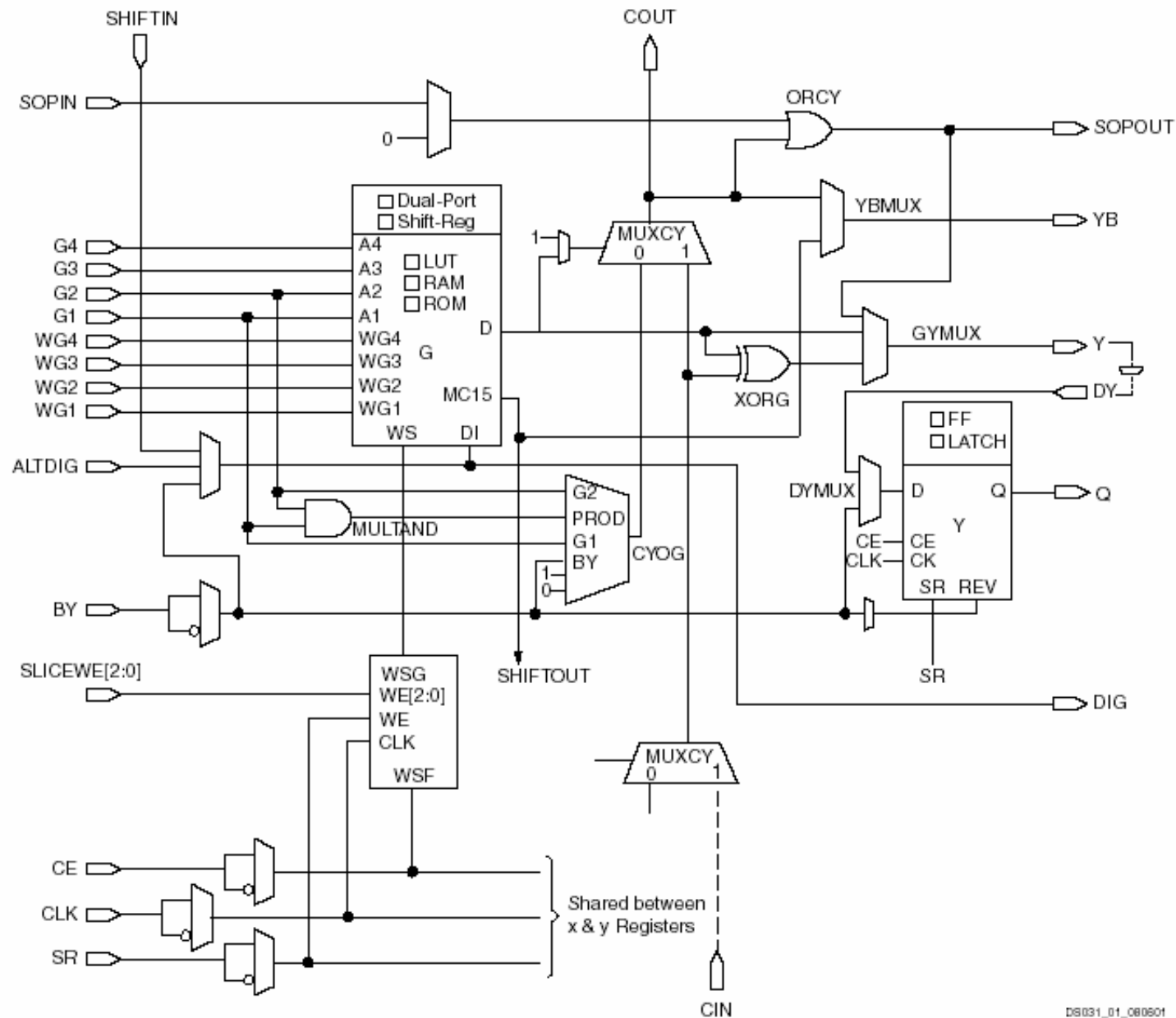


**Wires are not ideal!**

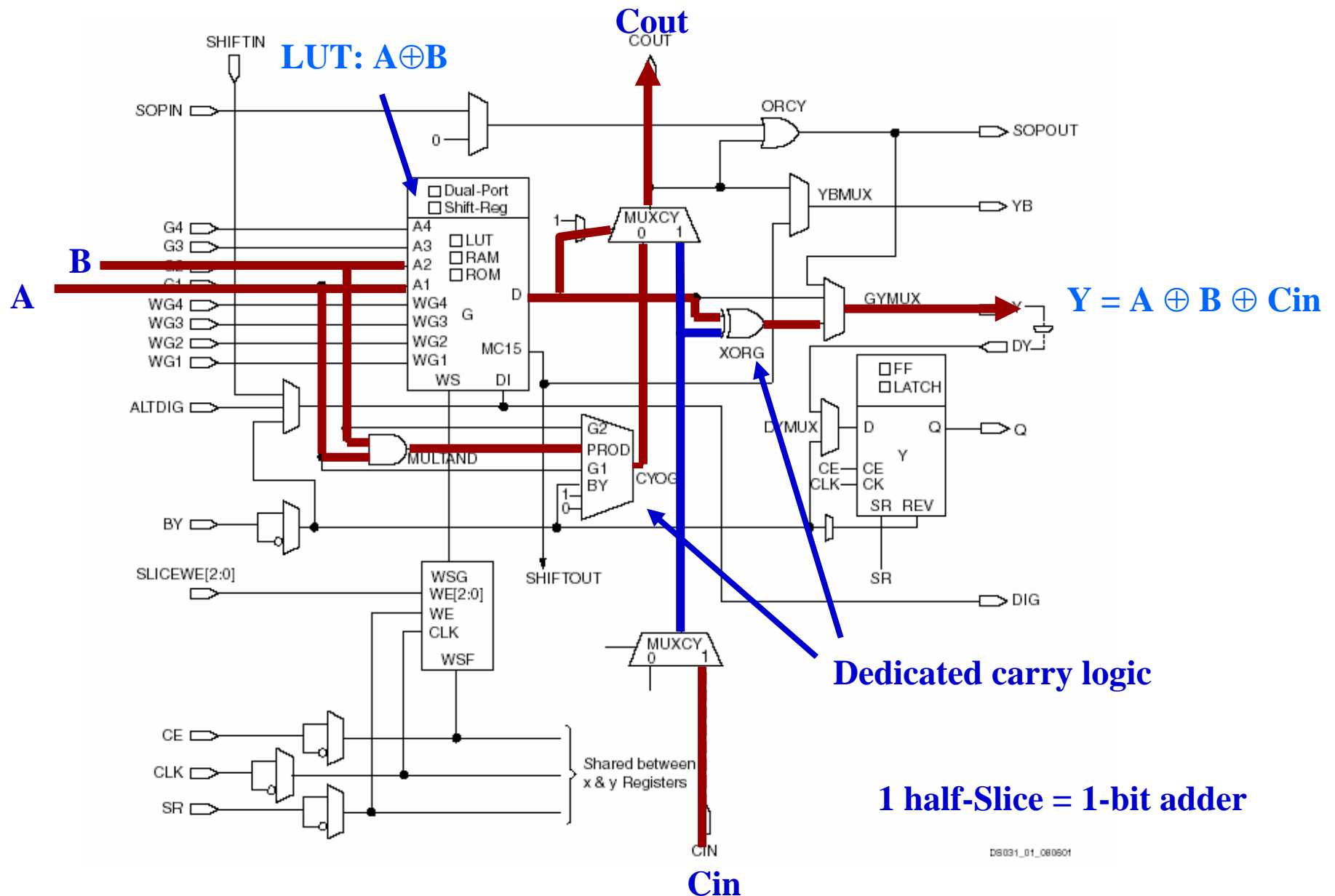




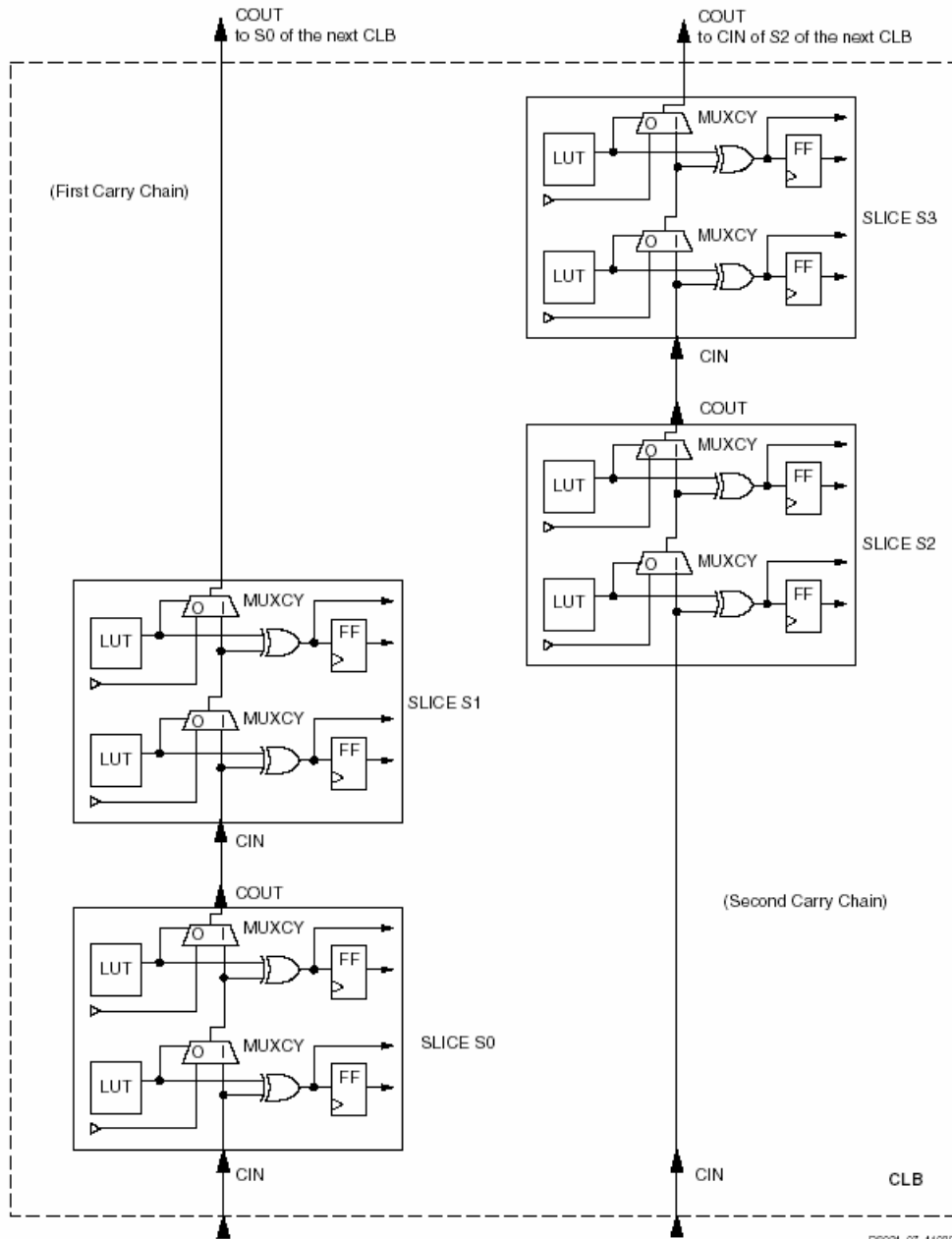
Courtesy of David B. Parlour, ISSCC 2004 Tutorial,  
“The Reality and Promise of Reconfigurable Computing in Digital Signal Processing”



DS031\_01\_08001

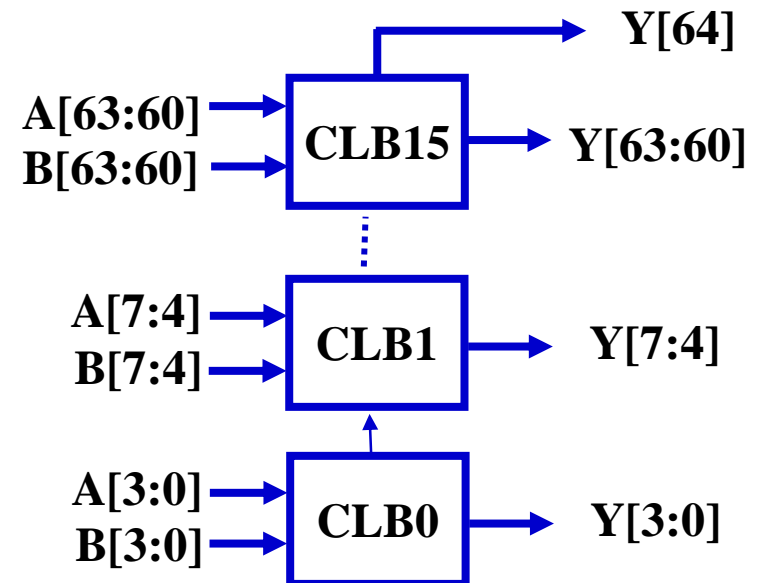
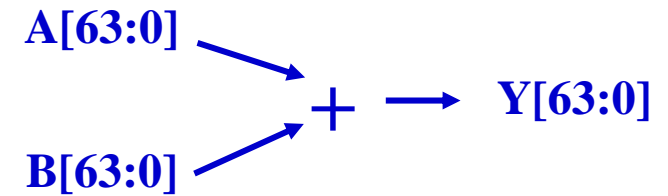


D6031\_01\_080601

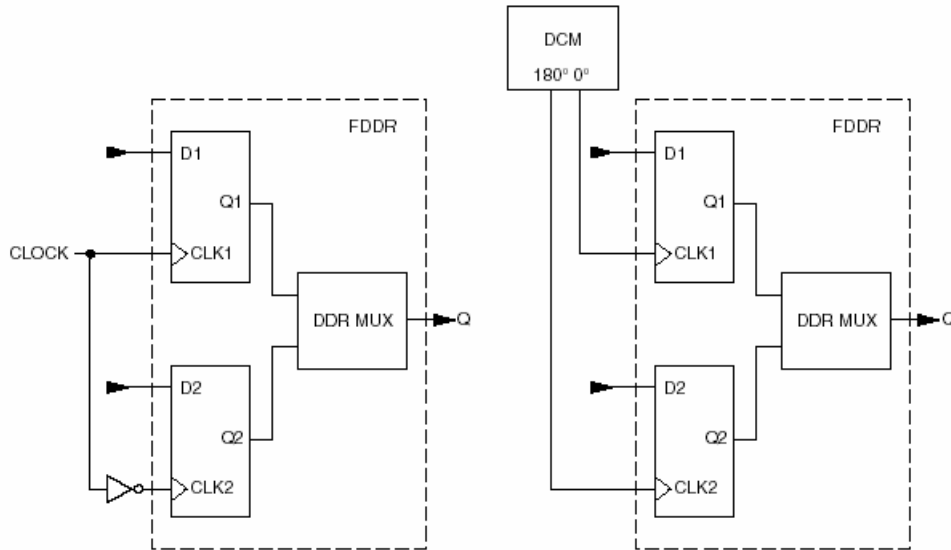


1 CLB = 4 Slices = 2, 4-bit adders

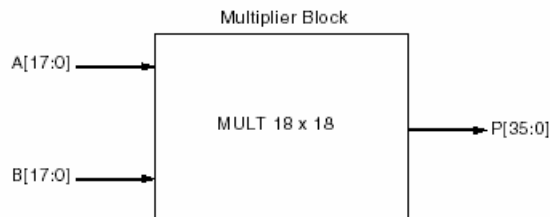
64-bit Adder: 16 CLBs



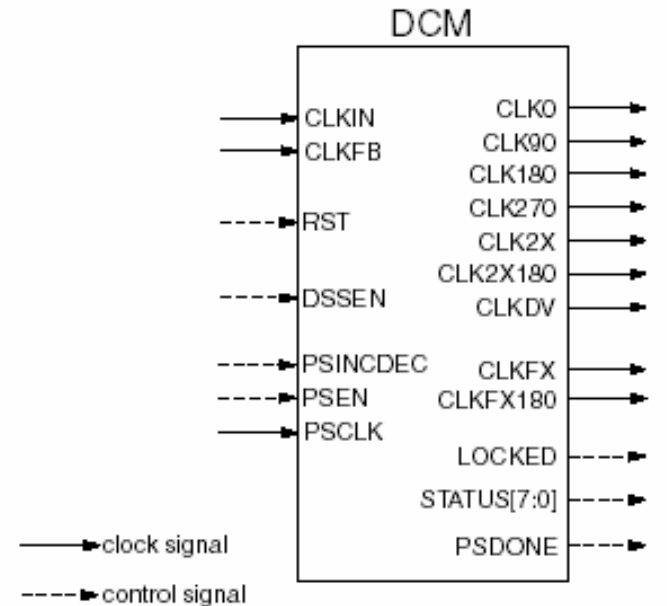
CLBs must be in same column



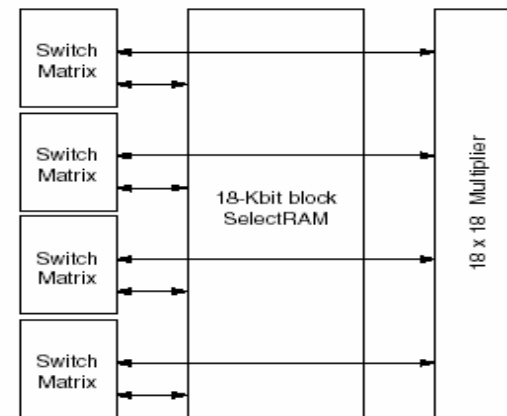
**Double Data Rate registers**



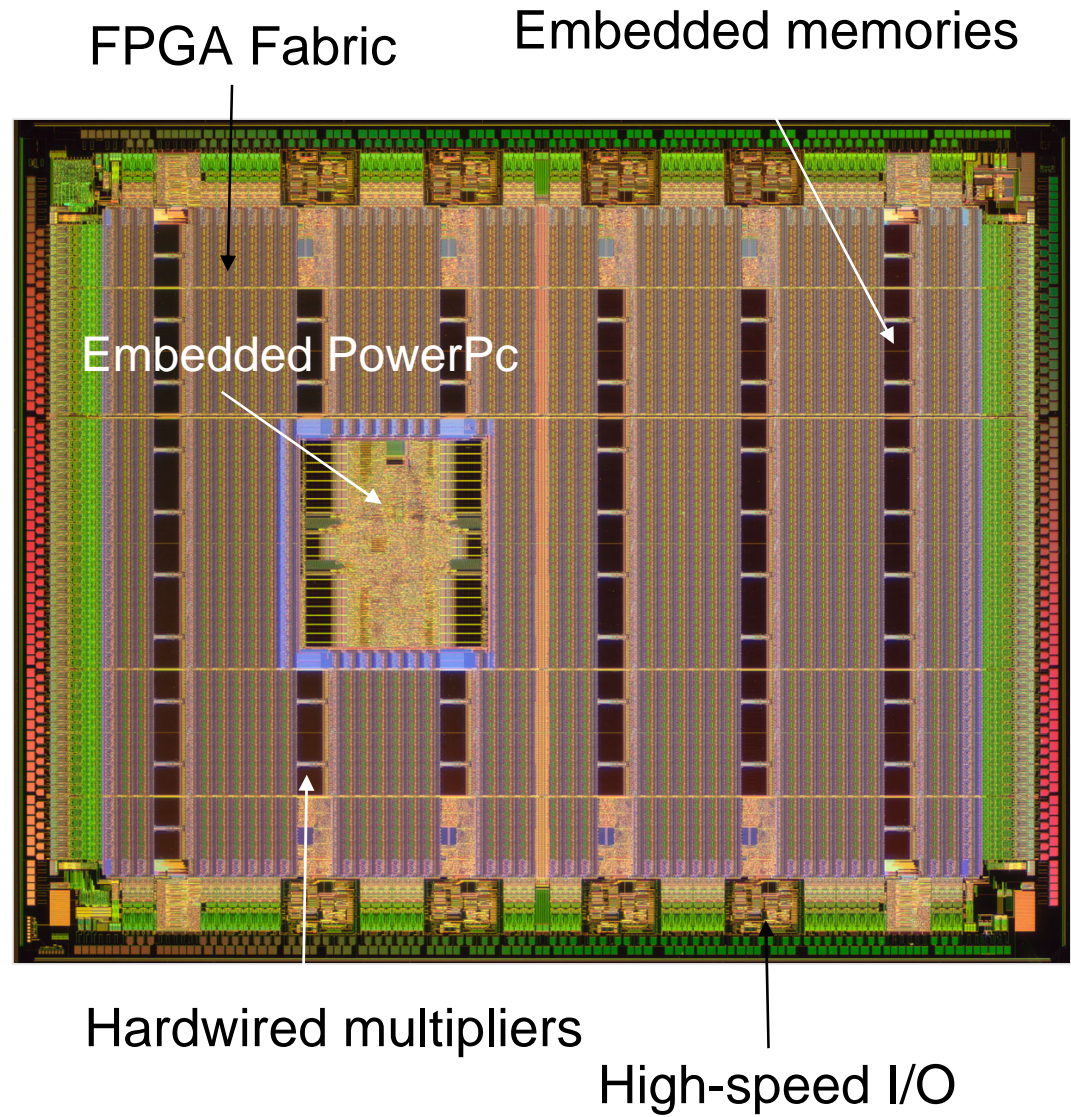
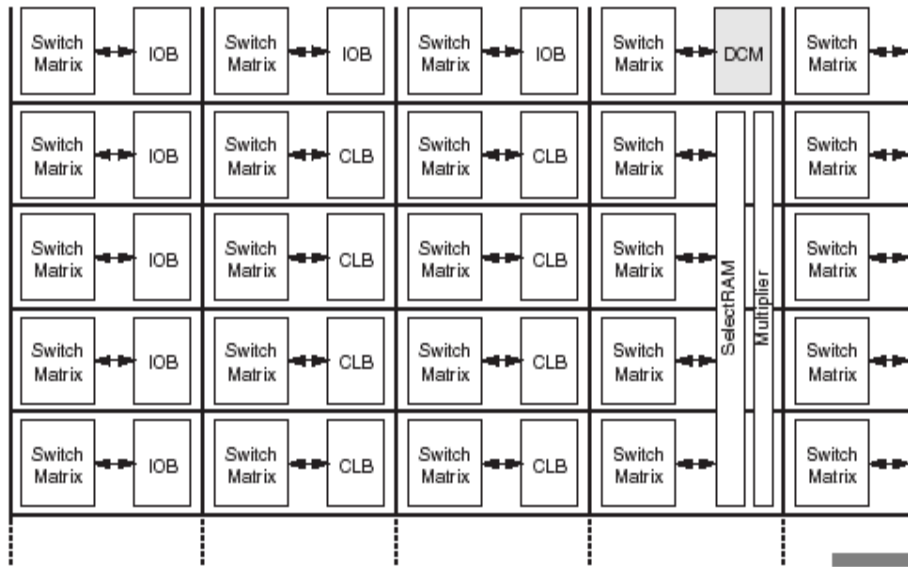
**Embedded Multiplier**



**Digital Clock Manager**

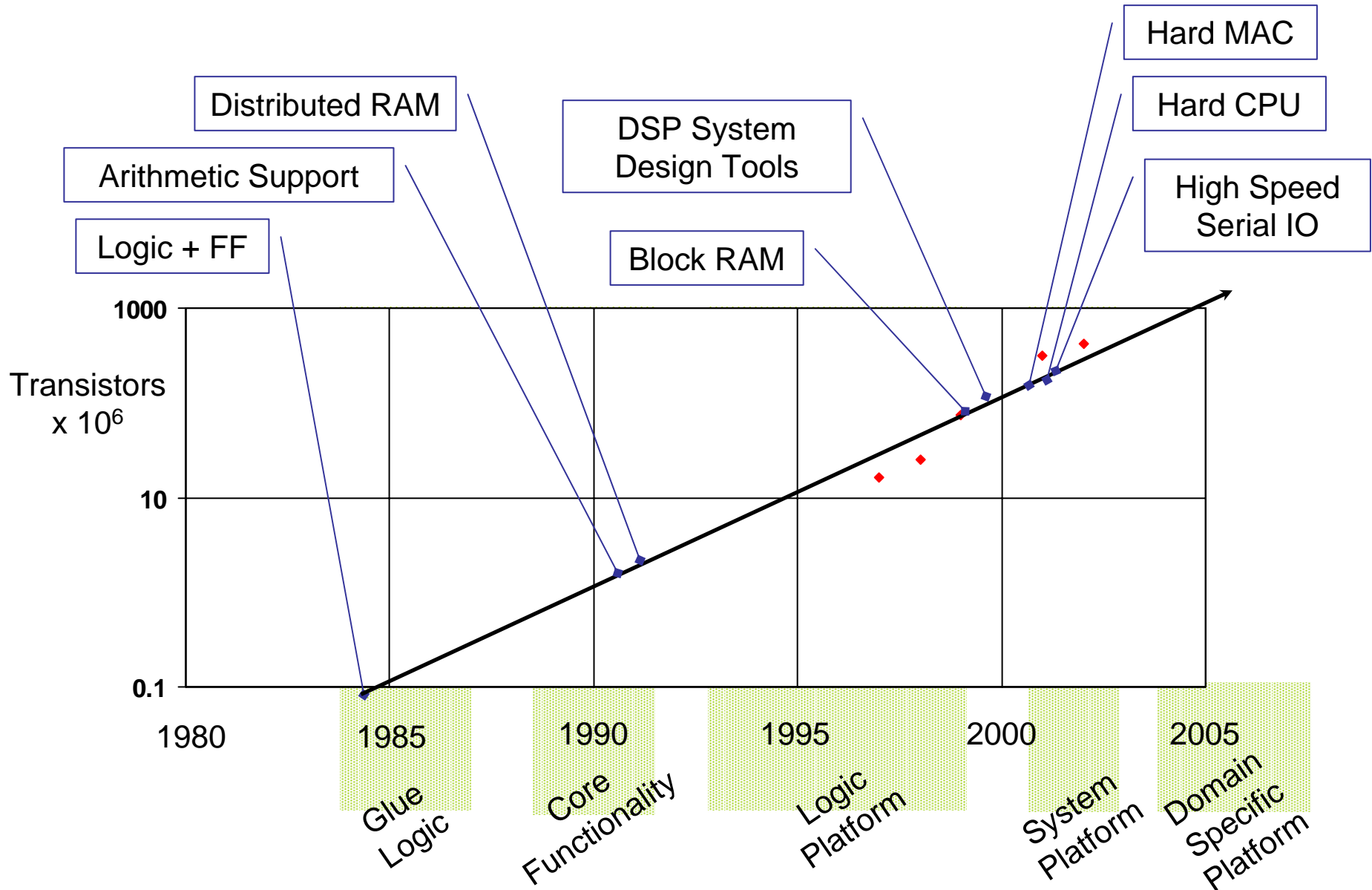


**Block SelectRAM**

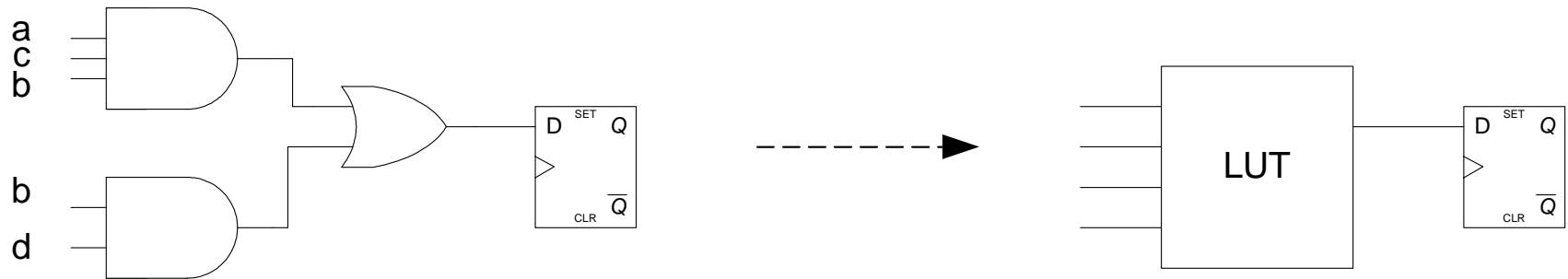


Courtesy Xilinx





- **Technology Mapping: Schematic/HDL to Physical Logic units**
- **Compile functions into basic LUT-based groups (function of target architecture)**

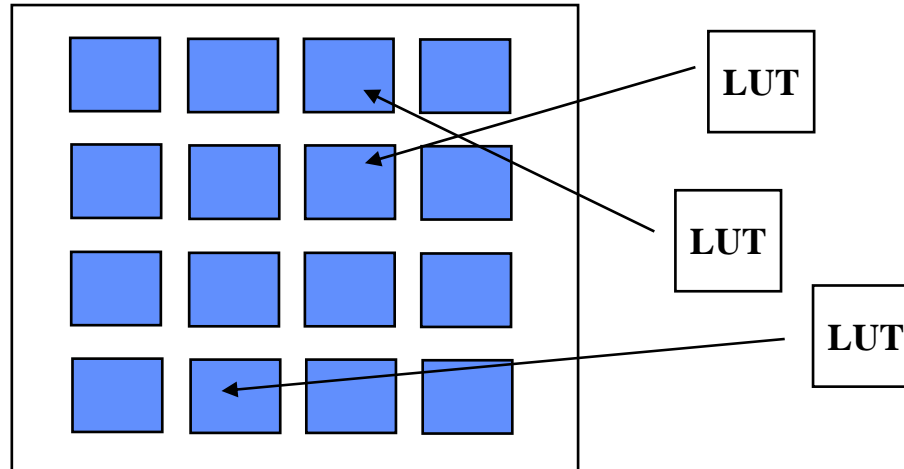


```

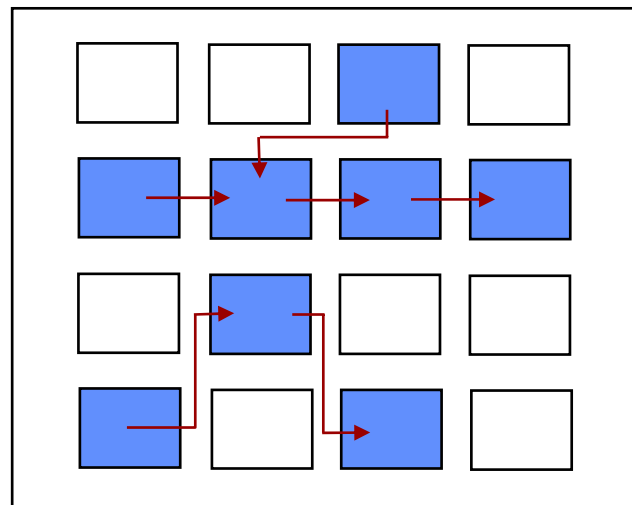
always @(posedge Clock or negedge Reset)
begin
  if (! Reset)
    q <= 0;
  else
    q <= (a & b & c) | (b & d);
end

```

- **Placement** – assign logic location on a particular device



- **Routing** – iterative process to connect CLB inputs/outputs and IOBs. Optimizes critical path delay – can take hours or days for large, dense designs



Iterate placement if timing not met

Satisfy timing? → Generate Bitstream to config device

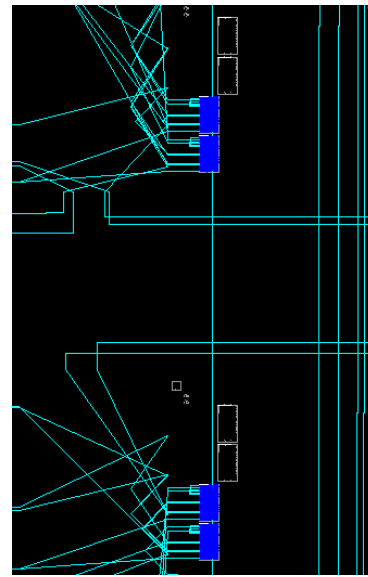
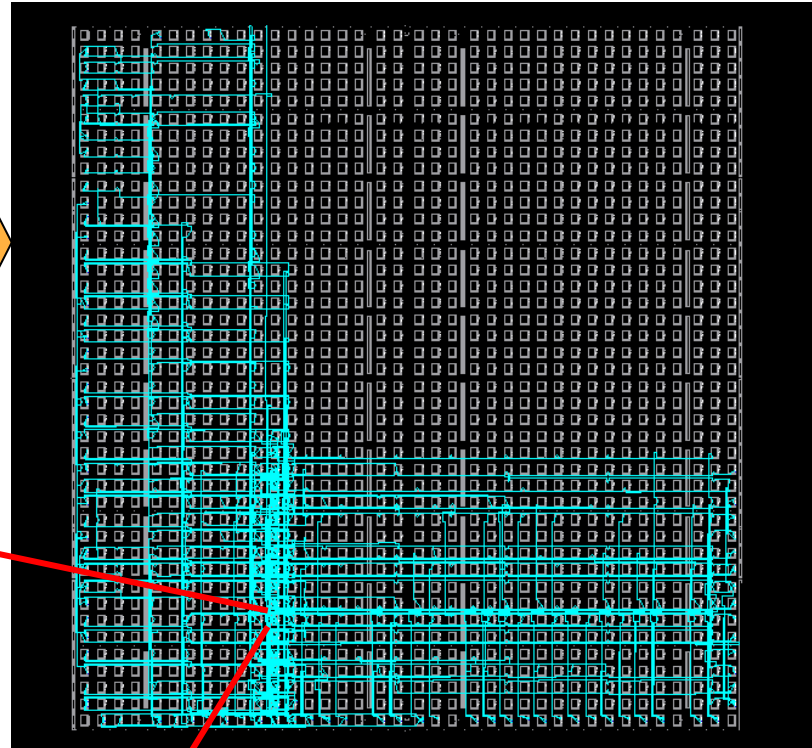
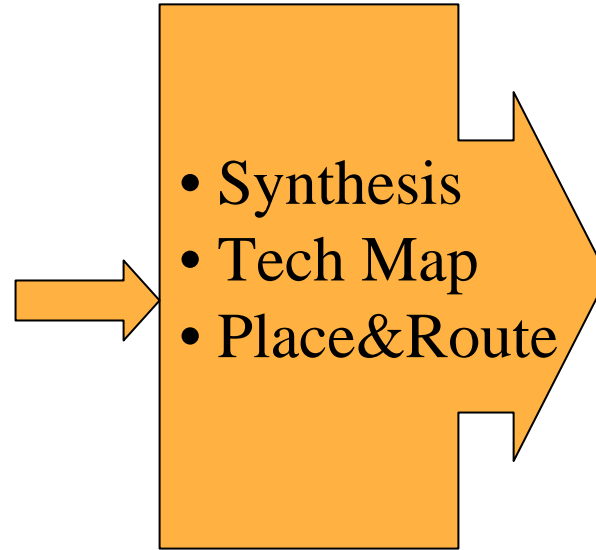
**Challenge!** Cannot use full chip for reasonable speeds (wires are not ideal).

**Typically no more than 50% utilization.**

```

module adder64 (a, b, sum);
  input [63:0] a, b;
  output [63:0] sum;

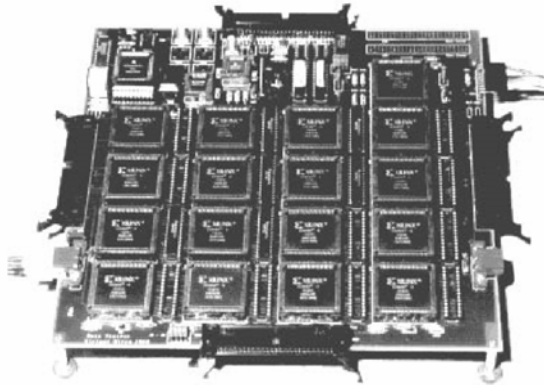
  assign sum = a + b;
endmodule
  
```



## 64-bit Adder Example

Virtex II – XC2V2000

## Logic Emulation



### ■ Prototyping

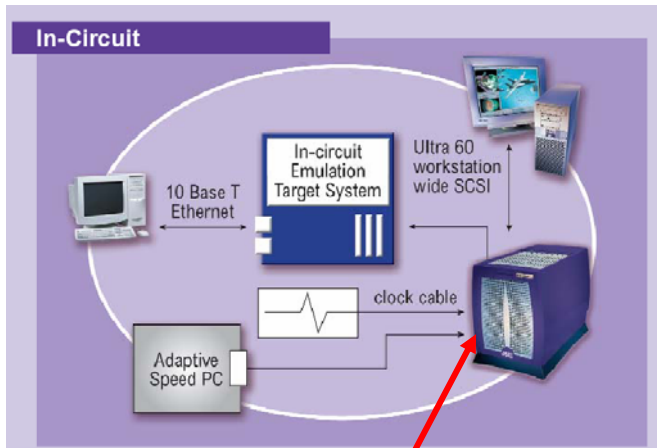
- Ensemble of gate arrays used to emulate a circuit to be manufactured
- Get more/better/faster debugging done than with simulation

### ■ Reconfigurable hardware

- One hardware block used to implement more than one function

### ■ Special-purpose computation engines

- Hardware dedicated to solving one problem (or class of problems)
- Accelerators attached to general-purpose computers (e.g., in a cell phone!)



**FPGA-based Emulator**  
(courtesy of IKOS)

- **FPGA provide a flexible platform for implementing digital computing**
- **A rich set of macros and I/Os supported (multipliers, block RAMS, ROMS, high-speed I/O)**
- **A wide range of applications from prototyping (to validate a design before ASIC mapping) to high-performance spatial computing**
- **Interconnects are a major bottleneck (physical design and locality are important considerations)**

**“College students will study concurrent programming instead of “C” as their first computing experience.”**

**-- David B. Parlour, ISSCC 2004 Tutorial**