CSE372
Digital Systems Organization and Design Lab

Prof. Milo Martin

Unit 2: Field Programmable Gate Arrays (FPGAs)

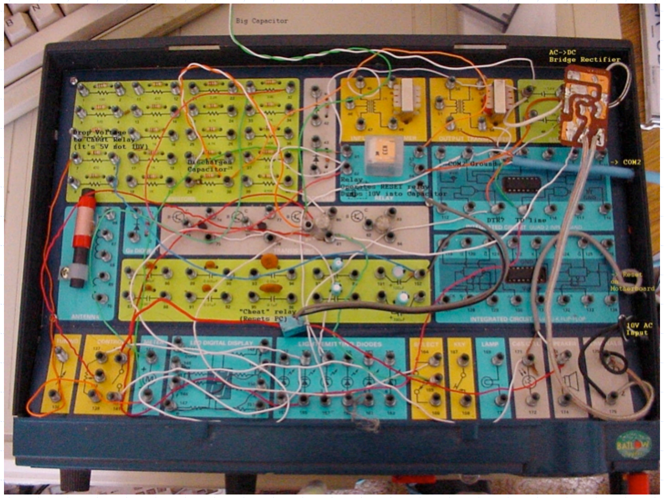
Announcements

- Lab 1 due in one week
  - Questions/comments?
  - Testbench coming soon (according to the TAs)
- Today's lecture:
  - How FPGAs work

Field Programmable Gate Array (FPGA)

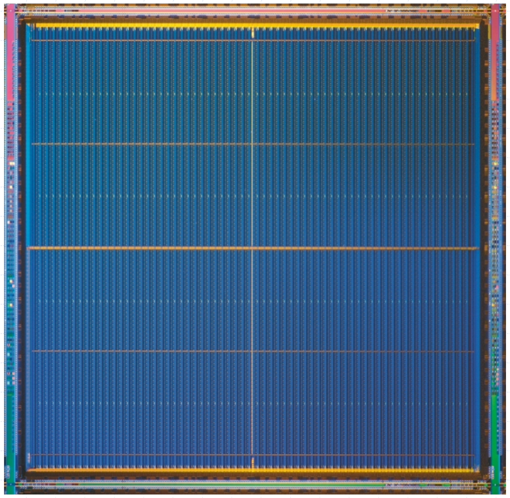
- An alternative to a "custom" design
  - A high-end custom design "mask set" is expensive (millions of $!)
- Advantages
  - Simplicity of gate-level design (no transistor-level design)
  - Fast time-to-market
    - No manufacturing delay
  - Can fix design errors over time (more like software)
- Disadvantages
  - Expensive: unit cost is higher
  - Inefficient: slower and more power hungry
- Result: good for low-volume or initial designs

Early Programmable Logic Device…

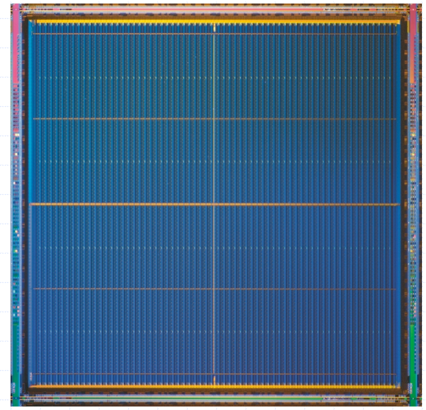


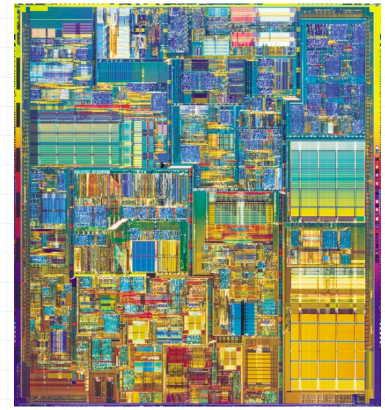
From UC-Berkeley CS152 slides

Modern FPGA: Xilinx Vertex II



For Comparison: FGPA vs Pentium 4





Not to scale

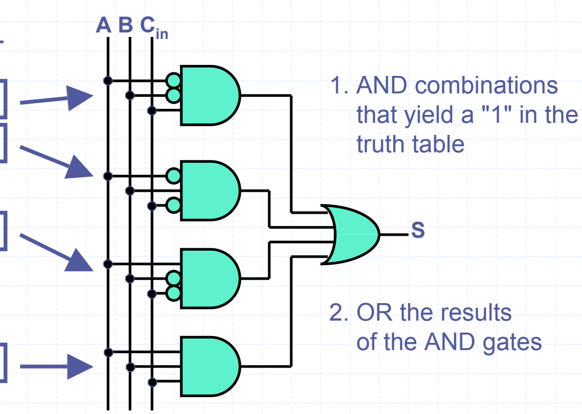
FPGA Design Flow

- Synthesis
  - Break design into well-define logic blocks
  - Examples:
    - 2-input gates
    - Only NANDs
    - Limited set of "standard cells" with three-inputs, one output
- Place and route
  - Custom: position the devices and wires that connect them
  - FPGA: configure logic blocks and interconnect
- Goals:
  - Reduce latency (performance)
  - Reduce area (cost)
  - Reduce power (performance and/or cost)

Review: Logical Completeness

- AND, OR, NOT can implement <u>ANY</u> truth table

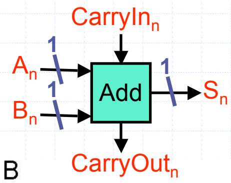
A	B	C_{in}	S
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



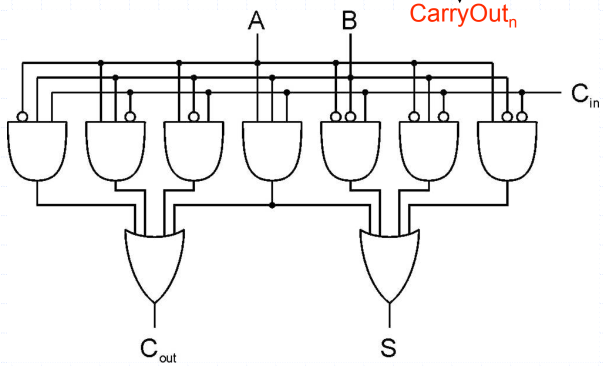
Mechanical process, but many optimizations

Our Old Friend, The Full Adder

- Add two bits and carry-in produce one-bit sum and carry-out



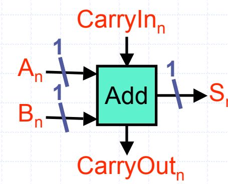
A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

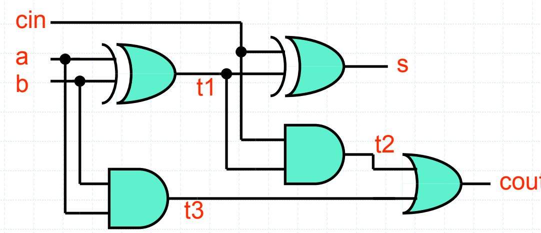


A Better Full Adder

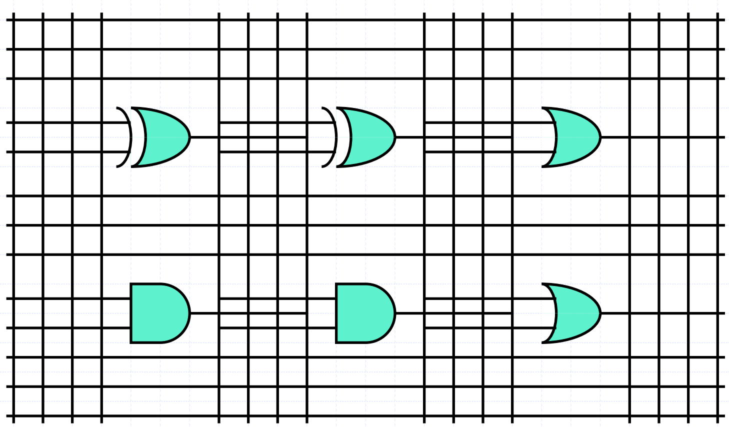
```
module full_adder(s, cout, a, b, cin);
    output s, cout;
    input a, b, cin;

    xor (t1, a, b);
    xor (s, t1, cin);
    and (t2, t1, cin);
    and (t3, a, b);
    or  (cout, t2, t3);
endmodule
```



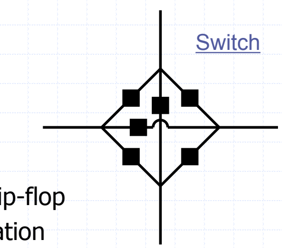


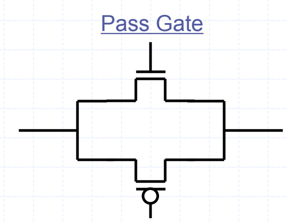
A Simple (Fake) FPGA Substrate



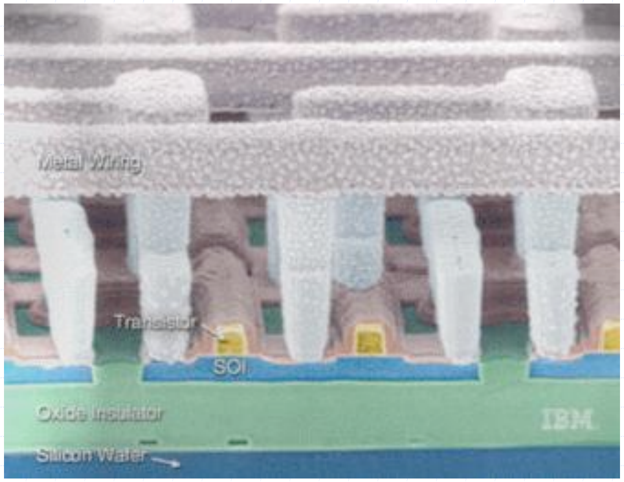
How Do We "Route" Signals?

- Switch matrix
  - Each junction has 6 "switches"
  - Each switch is a pass gate
- Programming
  - Each pass gate controlled by 1-bit flip-flop
  - 0/1 value of flip-flop set at configuration
- Programmable "interconnect"
  - Allows for arbitrary routing of signals
  - Each segment adds delay
  - Takes up lots of chip area



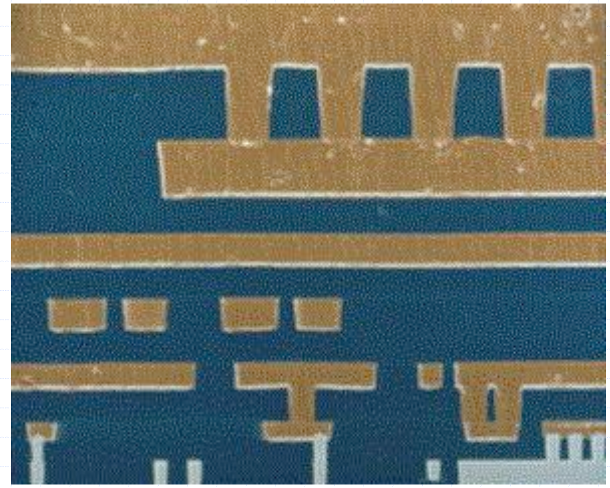


On-Chip Wires



©IBM

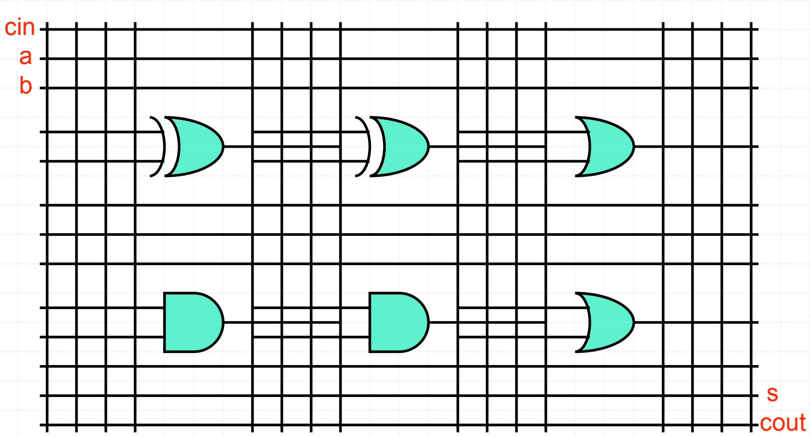
More Wires



©IBM

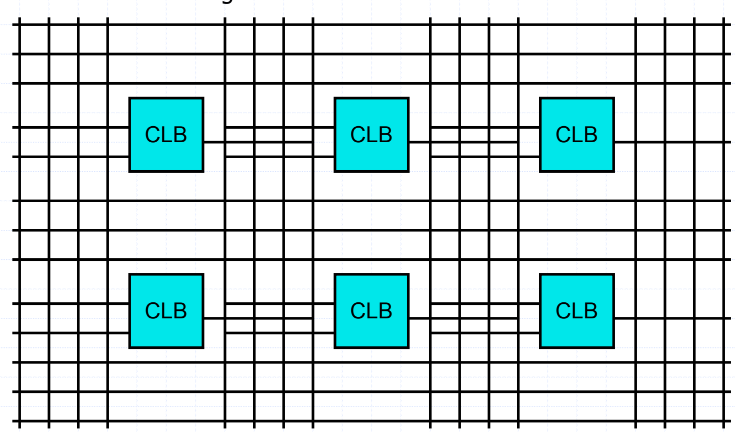
IBM CMOS7, 6 layers of copper wiring

Configure This As a Full Adder



A Better FPGA

- Replace gates with general "CLB"
  - Combinational logic block



Combinational Logic Block

- Simple example CLB
  - Configure as any two-input gate
  - Use 4-bit RAM to implement function
    - LUT - Lookup Table
  - Simple lookup operation
- Add sequential state
  - Add a latch/flipflop or two

A Standard Xilinx CLB

- Two 4-input LUTs
  - Any 4-input function
  - Limited 5-input functions
- Two flip-flops
- Fast carry logic (direct connect from adjacent CLBs)
- LUTs can be be configured as RAM:
  - 2x16 bit or 1x32 bit, single ported
  - 1x16 bit dual ported
- Routing
  - Short and long wires (skip some CLBs)
  - Clocks have dedicated wires
- Also has IOBs (input/output blocks)
  - Specialized for off-chip signals, one per pin on package

The Xilinx 4000 CLB

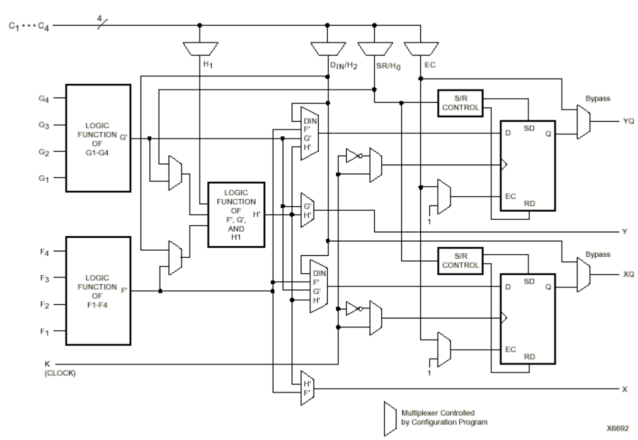


Figure 1: Simplified Block Diagram of XC4000 Series CLB (RAM and Carry Logic functions not shown)

Two 4-input functions, registered output

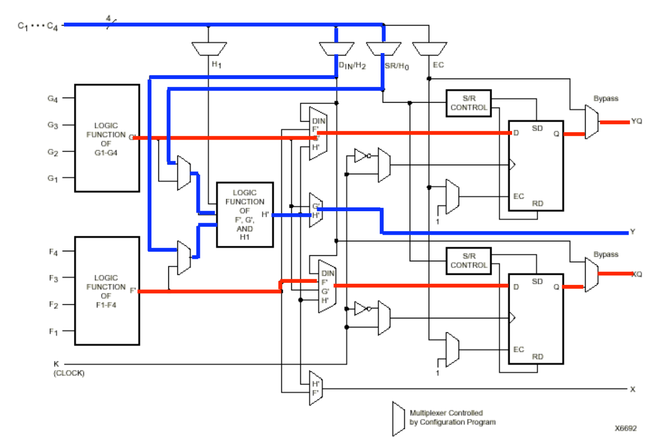


Figure 1: Simplified Block Diagram of XC4000 Series CLB (RAM and Carry Logic functions not shown)

5-input function, combinational output

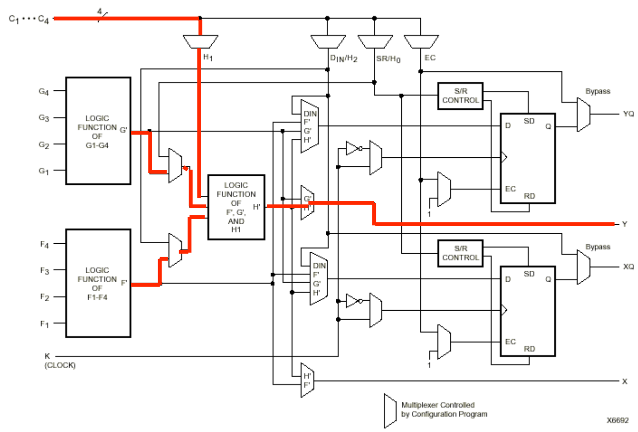


Figure 1: Simplified Block Diagram of XC4000 Series CLB (RAM and Carry Logic functions not shown)

CLB Used as RAM

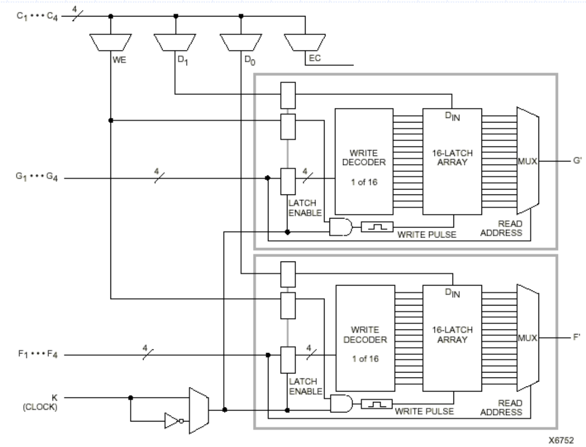
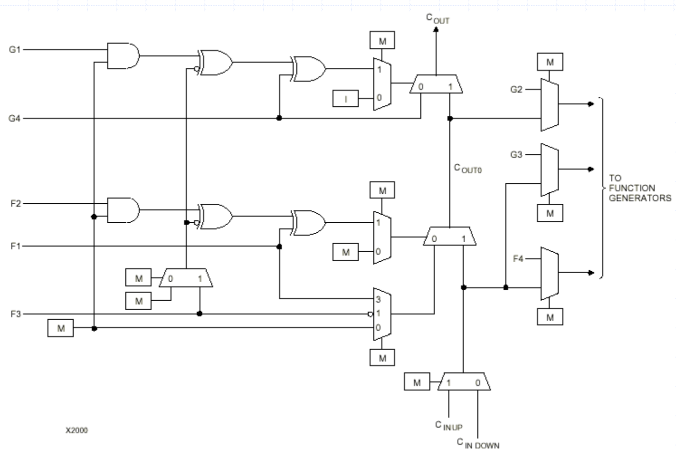


Figure 4: 16x2 (or 16x1) Edge-Triggered Single-Port RAM

Fast Carry Logic



Xilinx 4000 Interconnect

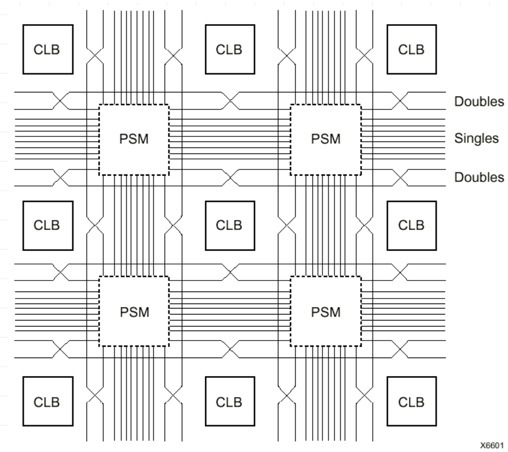


Figure 28: Single- and Double-Length Lines, with Programmable Switch Matrices (PSMs)

Switch Matrix

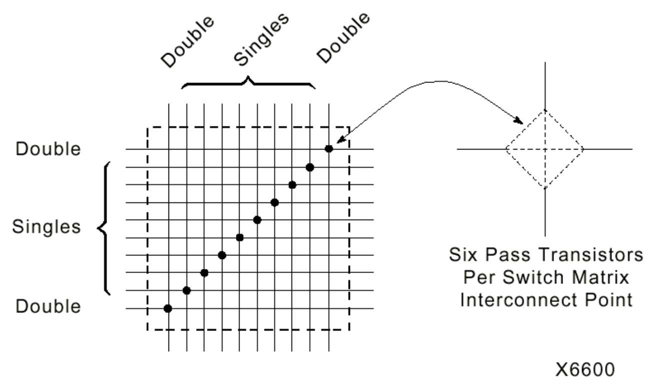
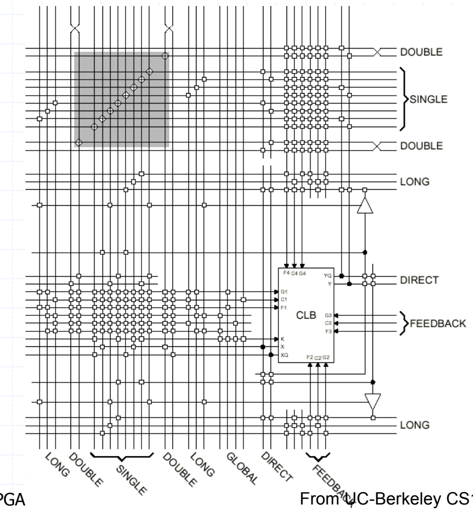


Figure 26: Programmable Switch Matrix (PSM)

Xilinx 4000 Interconnect Details



FPGA Design Issues

- How large should a CLB be?
  - How many inputs?
  - How much logic and state?
  - Example: two full-adders plus two latches in each Xilinx CLB
    - N-bit counter uses N/2 CLBs
- Routing resources
  - Faster, better routing
- Other imbedded hardware structures
  - RAM blocks
  - Multipliers
  - Processors

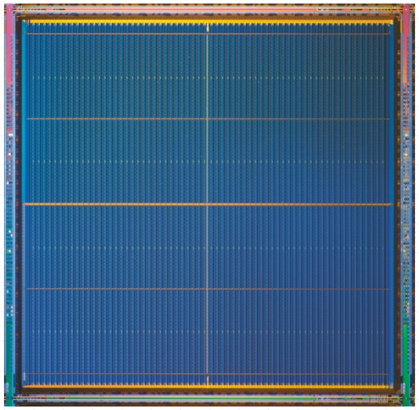
Our FPGAs: Virtex-2 Pro XC2VP30

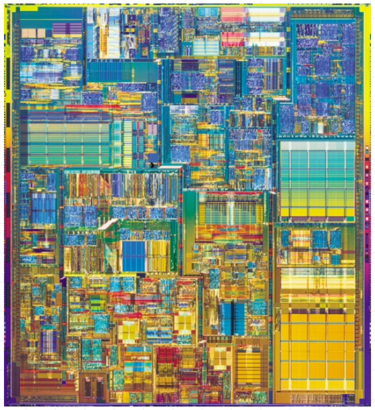
- Viertex-2 Pro
  - More powerful CLBs
  - More routing resources
  - Embedded PowerPC core
- XC2VP30
  - 30,816 CLBs
  - 136 18-bit multipliers
  - 2,448 Kbits of block RAM
  - Two PowerPC processors
  - 400+ pins

FPGA vs Custom Designs

- Downside of configurability
  - Wires are much slower on FPGAs
  - Logic is much slower on FPGAs
- However, FPGAs are "real" logic (not software)
  - Great for our prototyping
- "Synthesis to chip" an option ($$$)
  - Standard cell design
  - Hard coded, but based on synthesis design flow
  - Not as good as "full custom" as used by Intel, AMD, IBM

FPGA vs Custom Designs





Not to scale