Introduction

• A digital circuit design is just an idea, perhaps drawn on paper
• We eventually need to implement the circuit on a physical device
  – How do we get from (a) to (b)?

Manufactured IC Technologies

• We can manufacture our own IC
  – Months of time and millions of dollars
  – (1) Full-custom or (2) semicustom
    • (1) Full-custom IC
      – We make a full custom layout
      – Using CAD tools
      – Layout describes the location and size of every transistor and wire
      – A fab (fabrication plant) builds IC for layout
      – Hard!
        • Fab setup costs (“non-recurring engineering”, or NRE, costs) high
        • Error prone (several “respins”)
        • Fairly uncommon
          – Reserved for special ICs that demand the very best performance or the very smallest size/power

Manufactured IC Technologies – Gate Array ASIC

• (2) Semicustom IC
  – “Application-specific IC” (ASIC)
  – (a) Gate array or (b) standard cell
• (2a) Gate array
  – Series of gates already laid out on chip
  – We just wire them together
  – Using CAD tools
  – Vs. full-custom
    • Cheaper and quicker to design
    • But worse performance, size, power
  – Very popular

Half-adder equations:
\[ s = a' b + a b' \]
## Manufactured IC Technologies – Standard Cell ASIC

- **(2) Semicustom IC**
  - "Application-specific IC" (ASIC)
  - (a) Gate array or (b) standard cell
- **(2b) Standard cell**
  - Pre-laid-out "cells" exist in library, not on chip
  - Designer instantiates cells into pre-defined rows, and connects
  - Vs. gate array
    - Better performance/power/size
    - A bit harder to design
  - Vs. full custom
    - Not as good of circuit, but still far easier to design

## Programmable IC Technology – FPGA

- Manufactured IC technologies require weeks to months to fabricate
  - And have large (hundred thousand to million dollar) initial costs
- Programmable ICs are pre-manufactured
  - Can implement circuit today
  - Just download bits into device
  - Slower/faster/more-power than manufactured ICs
  - But get it today, no fabrication costs
- Popular programmable IC – FPGA
  - "Field-programmable gate array"
    - Though no "gate array" inside
    - Named after gate strings were very popular in the 1960s
    - Programmable in seconds

## FPGA Internals: Lookup Tables (LUTs)

- **Example:** Mapping a half-adder to standard cells

- **Basic idea:** Memory can implement combinational logic
  - e.g., 2-address memory can implement 2-input logic
  - 1-bit wide memory – 1 function; 2-bits wide – 2 functions
- **Such memory in FPGA known as Lookup Table (LUT)**

- **FPGA Internals: Lookup Tables (LUTs)**

- **Example:** Seat-belt warning light

- **FPGA Internals: Lookup Tables (LUTs)**

  - Lookup tables become inefficient for more inputs
    - 3 inputs → only 8 words
    - 8 inputs → 256 words: 16 inputs → 65,536 words
  - FPGAs thus have numerous small (3, 4, 5, or even 6-input) LUTs
    - If circuit has more inputs, must partition circuit among LUTs
    - Example: Extended seat-belt warning light system.
FPGA Internals: Lookup Tables (LUTs)

- Partitioning among smaller LUTs is more size efficient
  - Example: 9-input circuit

Original 9-input circuit

Partioned among 3x1 LUTs

Requires only 4 3-input LUTs (8x1 memories) - much smaller than a 9-input LUT (9x1 memory)

Example: Mapping a 2x4 decoder to 3-input 2-output LUTs

First column unused;
Second column unused;
First column implements AND
Second column implements AND/OR sub-circuit

Example: Partitioning a circuit among 3-input 2-output lookup tables

FPGA Internals: Switch Matrices

- Previous slides had hardwired connections between LUTs
- Instead, want to program the connections too
- Use switch matrices (also known as programmable interconnect)
- Simple mux-based version - each output can be set to any of the four inputs just by programming its 2-bit configuration memory

Example: Mapping the extended seatbelt warning light onto an FPGA with a switch matrix

- Recall earlier example (let's ignore d input for simplicity)
**FPGA Internals: Configurable Logic Blocks (CLBs)**

- LUTs can only implement combinational logic.
- Need flip-flops to implement sequential logic.
- Add flip-flop to each LUT output.
  - Configurable Logic Block (CLB)
    - LUT = flip-flops
    - Can program CLB outputs to come from flip-flops or from LUTs directly.

**FPGA Internals: Overall Architecture**

- Consists of hundreds or thousands of CLBs and switch matrices (SMs) arranged in regular pattern on a chip.

**FPGA Internals: Sequential Circuit Example using CLBs**

- Connections can be "programmed" to create custom circuit.

**FPGA Internals: Programming an FPGA**

- All configuration memory bits are connected as one big shift register.
- Known as scan chain.
- Shift in "bit file" of desired circuit.

**Other Technologies**

- Simple Programmable Logic Devices (SPLDs)
  - Developed 1970s (thus, pre-dates FPGAs).
  - Prefabricated IC with large AND-OR structure.
  - Connections can be "programmed" to create custom circuit.
    - Circuit shown can implement any 3-input function of up to 3 terms.
      - e.g., \( F = abc + a'c' \).

**Programmable Nodes in an SPLD**

- Fuse based – "blown" fuse removes connection.
- Memory based – 1 creates connection.
**PLD Drawings and PLD Implementation Example**

- Common way of drawing PLD connections:
  - Uses one wire to represent all inputs of an AND
  - Uses “x” to represent connection
  - Crossing wires are not connected unless “x” is present

- Example: Seat belt warning light using SPLD

**More on PLDs**

- Originally (1970s) known as Programmable Logic Array – PLA
  - Had programmable AND and OR arrays
- AMD created “Programmable Array Logic” – “PAL” (trademark)
  - Only AND array was programmable (fuse based)
- Lattice Semiconductor Corp. created “Generic Array Logic” – “GAL” (trademark)
  - Memory based
- As IC capacities increased, companies put multiple PLD structures on one chip, interconnecting them
  - Became known as Complex PLDs (CPLD), and older PLDs became known as Simple PLDs (SPLD)
- GENERALLY SPEAKING, difference of SPLDs vs. CPLDs vs. FPGAs:
  - SPLD: tens to hundreds of gates, and usually non-volatile (saves bits without power)
  - CPLD: thousands of gates, and usually non-volatile
  - FPGA: tens of thousands of gates and more, and usually volatile (but no reason why couldn’t be non-volatile)

**Technology Comparisons**

<table>
<thead>
<tr>
<th>Technology</th>
<th>1. More optimized</th>
<th>2. Easier design</th>
<th>3. Faster performance</th>
<th>4. Lower design cost</th>
<th>5. Lower need for extra components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom processor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FPGA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gate array (semicustom)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard cell</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPLD/FPGA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE 7.3: Example % of choice implementations in various technologies. Total is more than 100% due to overlap among categories**

- Technology (%)
  - Standard cell: 55%
  - Gate array: 35%
  - System-on-a-Chip: 10%
  - Full-custom: 20%
  - CPLD/FPGA: 10%
  - Other: 5%

**Technology Comparisons**

(1): Custom processor in full-custom IC
Highly optimized
(2): Custom processor in FPGA
Parallelized circuit, slower IC technology but programmable
(3): Programmable processor in standard cell IC
Program runs mostly sequentially on moderate-costing IC
(4): Programmable processor in FPGA
Not only can processor be programmed, but FPGA can be programmed to implement multiple processors/coprocessors