ASIC, FPGAs and programmable processors

CMPE 641
Technology Timeline

- Transistors
- ICs (General)
- SRAMs & DRAMs
- Microprocessors
- SPLDs
- CPLDs
- ASICs
- FPGAs
Figure 3-03
PLDs

SPLDs

CPLDs

ASICs

Standard Cell

Full Custom

*Not available circa early 1980s

The GAP

Figure 3-17
FPGAs

- Large array of configurable logic blocks (CLB) connected via programmable interconnects
Features and Specifications of FPGAs

Each PLB can be programmed individually to perform a unique function.

The FF can be triggered by a positive or negative-going clk.

The MUX allows selection of the LUT output or an external input.
Basic Programmable Devices

The LUT can implement any 3-input logic function.

(a & b) | !c

Features and Specifications of FPGAs

MUX- vs. LUT-based Logic Blocks
There are 2 basic flavors of PLBs for medium-grained architectures, multiplexer (MUX) and lookup table (LUT).

In the MUX-based version, each input can be programmed with a logic 0, 1, or the true or inverted version of a variable.
Features and Specifications of FPGAs

MUX- vs. LUT-based Logic Blocks

Most FPGAs today are LUT-based -- here, the input signals are used as a pointer into a lookup table.

Input signals can be decoded using a hierarchy of transmission-gate MUXs.

Transmission gates pass the value on their inputs or are high-impedance.

Note that the diagram does not show the serial connection of the cells (scan chain) for simplicity.
Features and Specifications of FPGAs

Terminology and Hierarchy

The CLB also has some fast interconnect (not shown), that is used to connect neighboring slices.

The organization of LC -> Slice -> CLB is complemented by an equivalent hierarchy in the interconnect.

That is, fast interconnect between LCs in a slice, slightly slower between slices in a CLB, followed by the interconnect between CLBs.
Generic Xilinx FPGA Architecture

- FPGAs provide a highly parallel and flexible implementation platform for DSP... this diagram is representative of Xilinx DSP series devices.
Clock Tree in FPGAs

- Everything is preplaced and routed (there is no space for improvement)
- There is no gate sizing to enhance performance
FPGA vs ASIC
Standard cell based IC vs. Custom design IC

- **Standard cell based IC:**
  - Design using standard cells
  - Standard cells come from library provider
  - Many different choices for cell size, delay, leakage power
  - Many EDA tools to automate this flow
  - Shorter design time

- **Custom design IC:**
  - Design all by yourself
  - Higher performance
Standard cell based VLSI design flow

- Front end
  - System specification and architecture
  - HDL coding & behavioral simulation
  - Synthesis & gate level simulation

- Back end
  - Placement and routing
  - DRC (Design Rule Check), LVS (Layout vs Schematic)
  - dynamic simulation and static analysis
Simple diagram of the front-end design flow

System Specification → RTL Coding → Synthesis → Gate level code

Ex: \( c = !a \land b \)

INV (.in (a), .out (a_inv));
AND (.in1 (a_inv), .in2 (b), .out (c));
Simple diagram of the back-end design flow

- Gate level Verilog from synthesis
- Place & Route
  - Timing information
  - Gate level Verilog
    - Final layout (go for fabrication)
    - Gate level dynamic and/or static analysis
    - DRC (Design rule check)
    - LVS (Layout vs. schematic)
Flow of placement and routing

• Floorplan (place macros, do power planning)
• Placement and in-place optimization
• Clock tree generation
• Routing
Import needed files

- Gate level verilog (.v)
- Geometry information (.lef)
- Timing information (.lib)

INV (.in (a), .out (a_inv));
AND (.in1 (a_inv), .in2 (b), .out (c));

INV: 1um width  AND: 2 um width

INV: 1ns delay;  AND: 2 ns delay

Delay (a->c): 1ns + 2ns = 3ns
Floorplan

- Size of chip
- Location of Pins
- Location of main blocks
- Power supply: give enough power for each gate
- Note: the height of standard cell gates are the same so the GND/VDD lines can connect them all in rows.

Voltage drop equation: $V_2 = V_1 - I \times R$
Floorplan of a single processor
Placement & in-placement optimization

• Placement: place the gates

• In-placement optimization
  – Why: timing information difference between synthesis and layout (wire delay)
  – How: change gate size, insert buffers
  – Should not change the circuit function!!
Placement of a single processor
Clock tree

- Main parameters: skew, delay, transition time
Clock tree of single processor
Routing

• Connect the gates using wires
• Two steps
  – Connect the global signals (power)
  – Connect other signals
Metal Layer Topology Routing
Layout of a single processor

Area:
0.8mm x 0.8mm

Estimated speed:
450 MHz
Chip 1: low logic utilization 30%

384 Check & 2048 Variable processors

Logic Utilization

Application 1 gates
Application 2 gates

Chip 2: high logic utilization 96%
## Summary of the 167 Many-core Chip

<table>
<thead>
<tr>
<th>Single Tile</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Transistors</td>
<td>325,000</td>
<td></td>
</tr>
<tr>
<td>Area</td>
<td>0.17 mm²</td>
<td></td>
</tr>
<tr>
<td>CMOS Tech.</td>
<td>65 nm ST</td>
<td>Microelectronics low-leakage</td>
</tr>
<tr>
<td>Max. frequency</td>
<td>1.19 GHz @ 1.3 V</td>
<td></td>
</tr>
<tr>
<td>Power (100% active)</td>
<td>59 mW @ 1.19 GHz, 1.3 V</td>
<td>47 mW @ 1.06 GHz, 1.2 V</td>
</tr>
<tr>
<td></td>
<td>608 μW @ 66 MHz, 0.675 V</td>
<td></td>
</tr>
<tr>
<td>App. power (802.11a rx)</td>
<td>16 mW @ 590 MHz, 1.3 V</td>
<td></td>
</tr>
</tbody>
</table>

55 million transistors, 39.4 mm$^2$
FPGA vs ASIC summary

• Front-end design flow is almost the same for both

• Back-end design flow optimization is different
  – ASIC design: freedom in routing, gate sizing, power gating and clock tree optimization.
  – FPGA design: everything is preplaced, clock tree is pre-routed, no power gating
  – Designs implemented in FPGAs are slower and consume more power than ASIC
ASIC and FPGA vs DSP
FPGA vs DSP

- **DSP:**
  - Easy to program (usually standard C)
  - Very efficient for complex sequential math-intensive tasks
  - Fixed datapath-width. Ex: 24-bit adder, is not efficient for 5-bit addition
  - Limited resources

- **FPGA**
  - Requires HDL language programming
  - Efficient for highly parallel applications
  - Efficient for bit-level operations
  - Large number of gates and resources
  - Does not support floating point, must construct your own.
Current trend

- Programming flexibility
- High performance
  - Throughput
  - Latency
- High energy efficiency
- Suitable for future fabrication technologies
Target Many-core Architecture

• High performance
  • Exploit task-level parallelism in digital signal processing and multimedia
    – Large number of processors per chip to support multiple applications
• High energy efficiency
  – Voltage and frequency scaling capability per processor
167-processor Multi-voltage Computational Chip

- 164 programmable procs.
- Three dedicated-purpose procs.
- Per processor Dynamic Voltage and Frequency Scaling (DVFS)
  - Selects between two voltages (VDD High and VDD Low)
  - Programmable local oscillator

## Summary of the 167 Many-core Chip

<table>
<thead>
<tr>
<th>Single Tile</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Transistors</td>
<td>325,000</td>
</tr>
<tr>
<td>Area</td>
<td>0.17 mm$^2$</td>
</tr>
<tr>
<td>CMOS Tech.</td>
<td>65 nm ST Microelectronics low-leakage</td>
</tr>
<tr>
<td>Max. frequency</td>
<td>1.19 GHz @ 1.3 V</td>
</tr>
<tr>
<td>Power (100% active)</td>
<td>59 mW @ 1.19 GHz, 1.3 V</td>
</tr>
<tr>
<td></td>
<td>47 mW @ 1.06 GHz, 1.2 V</td>
</tr>
<tr>
<td></td>
<td>608 μW @ 66 MHz, 0.675 V</td>
</tr>
<tr>
<td>App. power (802.11a rx)</td>
<td>16 mW @ 590 MHz, 1.3 V</td>
</tr>
</tbody>
</table>

55 million transistors, 39.4 mm$^2$