Introduction to Structured VLSI Design

- Digital systems and recap

Joachim Rodrigues

Overview

- Why digital?
- Some Applications
- Device Technologies
- Digital Components
- Timing
- DFF, Latches
- Registers

Advantages

- Advantage of digital devices
  - Reproducibility of information
  - Flexibility and functionality: easier to store, transmit and manipulate information
  - Economy: cheaper device and easier to design

- Moore’s law
  - Transistor geometry
  - Chips double density (number of transistor) every 18 months
  - Devices become smaller, faster and cheaper
  - Now a chip consists of hundreds of million gates
  - And we can have a “wireless-PDA-MP3-player-camera-GPS-cell-phone” gadget very soon (statement after 2005)

Applications of digital systems

- “Digitization” has spread to a wide range of applications, including information (computers), telecommunications, control systems etc.

- Digital circuitry replaces many analog systems:
  - Audio recording: from tape to music CD to MP3 (MPEG Layer 3) player
  - Image processing: from silver-halide film to digital camera
  - Telephone switching networks
  - Combustion control in car engines
Challenge

Implement the best HW realization. Best??

Different applications, different demands...
Thus, "just good enough" is the best in engineering.

Try to find a balance between effort and cost.

Cost
- Processors
- FPGAs

Flexibility
- Processors
- Dedicated HW

Complexity
- Low power
- Lower power

Low power
- Low cost
- Lower cost

Flexibility

Application Domain
Specific Computing

Power

Reconfiguring of hardware

Std. Proc.

DSP

ASIC

Specialization of processor

Flexibility

Chip power density

Source: Borkar, De Intel

Chips might become hot...

Power Density (W/cm²)

Year


10000

1000

100

10

1

Intel 4004: 1972

- First micro-processor on a single chip
- 2300 transistors
- 0.3 mm x 0.4 mm
- 4 bit words
- Clock: 0.108 MHz

You will have the possibility to design a more powerful processor in one of our courses
Intel Pentium 4

- 42 000 000 transistors.
- 0.18 micron CMOS
- Clock: 1.5 GHz
- Die: 20 mm²

Baseband ASIC of a modern mobile phone has easily 10 times more transistors.

SandyBridge

- 32 nm – 64 bit
- 4,995,000,000 Transistors
- ~3.5 GHz
- 216 mm² (10x Pentium 4)

SandyBridge on a wafer

Typical well know processor

- 90 nm process
- 8 processors
- Playstation
**Cardiac Pacemaker**

- **1958**
  - Recharge after 14 days

- **2000**
  - Replacement after 7-20 years
  - Programmable

**Battery**

**Chest X-Rays**

- *Dual chamber pacemaker*
- *Single chamber pacemaker*

**Bionic Ear**

- **1977**
  - 1977
- **2006**
  - 2006

**Skull X-Rays**
How to implement a digital system

- No two applications are identical and every one needs a certain amount of customization
- Basic methods for customization
  - “General-purpose hardware” with custom software
    - General purpose processor: e.g., performance-oriented processor (e.g., Pentium), cost-oriented processor (e.g., PIC micro-controller)
    - Special purpose processor: with architecture to perform a specific set of functions: e.g., DSP processor (to do multiplication-addition), network processor (to do buffering and routing), “graphic engine” (to do 3D rendering)

Digital systems

- Custom hardware
- Custom software on a custom processor (known as hardware-software co-design)
- Trade-off between Programmability, Coverage, Cost, Performance, and Power consumption
- A complex application contains many different tasks and use more than one customization methods

Classification of device technologies

- Classification:
  - Full-custom ASIC
  - **Standard cell ASIC**
  - Gate array ASIC (80’s)
  - **Complex field programmable logic device (FPGA)**
  - Simple field programmable logic device
  - Off-the-shelf SSI (Small Scaled IC)/MSI (Medium Scaled IC) components

Standard-Cell ASIC

- Circuit made of a set of pre-defined logic, known as standard cells
- E.g., basic logic gates, 1-bit adder, D FF etc
- Layout of a cell is pre-determined, but layout of the complete circuit is customized
- Masks needed for all layers
- Processors are build as standard cell ASICS
Complex Field Programmable Device

- Device consists of an array of generic logic cells and general interconnect structure
- Logic cells and interconnect can be “programmed” by utilizing “semiconductor fuses or “switches”
- Customization is done “in the field”
- Two categories:
  - CPLD (Complex Programmable Logic Device)
  - FPGA (Field Programmable Gate Array) will be used in the Lab
- No custom mask needed

Comparison of technology

- Area (Size): silicon “real-estate”
  - Standard cell is the smallest since the cells and interconnect are customized
  - FPGA is the largest
    - Overhead for “programmability”
    - Capacity cannot be completely utilized
- Speed (Performance)
  - Time required to perform a task
- Power
- Cost

Cost

- Types of cost:
  - NRE (Non-Recurrent Engineering) cost: one-time, per-design cost
  - Part cost: per-unit cost
  - Time-to-market “cost” loss of revenue
- Standard cell: high NRE, small part cost and large lead time (up to several years)
- FPGA: low NRE, large part cost and small lead time

Graph of per-unit cost
Summary of technology

<table>
<thead>
<tr>
<th></th>
<th>FPGA</th>
<th>Gate array</th>
<th>Standard cell</th>
</tr>
</thead>
<tbody>
<tr>
<td>tailored masks area</td>
<td>T</td>
<td>3 to 5</td>
<td>15 or more</td>
</tr>
<tr>
<td>speed</td>
<td></td>
<td></td>
<td>best (smallest)</td>
</tr>
<tr>
<td>power</td>
<td></td>
<td></td>
<td>best (fastest)</td>
</tr>
<tr>
<td>NRE cost</td>
<td>best (smallest)</td>
<td>best (smallest)</td>
<td></td>
</tr>
<tr>
<td>design cost</td>
<td>best (easiest)</td>
<td>best (smallest)</td>
<td></td>
</tr>
<tr>
<td>time to market</td>
<td>best (shortest)</td>
<td>depend on volume</td>
<td></td>
</tr>
<tr>
<td>per unit cost</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Trade-off between optimal use of hardware resource and design effort/cost
- No single best technology
- Application (medical, mobile, ...)

"Digital" is an Abstraction

Only if we guarantee to meet the timing requirements ... do the components guarantee to behave as intended.

Recap

Following slides should fresh up your memory

"Digital" is an Abstraction

Only if we guarantee to meet the timing requirements ... do the components guarantee to behave as intended.

Digital Circuits

- Binary numbers: "0" and "1" or "False" and "True"
- Represented by voltages:
  - "0" / "False" is 0 V
  - "1" / "True" is e.g. 2.5 V (much lower (<1V) in newer technologies)
- Digital is an abstraction
  - Discrete values: 0 or 1
  - Discrete time (clock defines when valid)
Two Basic Digital Components

**Combinatorial Logic**

\[ F \]

D \rightarrow z

Always:

\[ z \leq F(a, b, c); \]

i.e. a function that is always evaluated when an input changes.

Can be expressed by a truth table.

**Register**

\[ Q \]

Rising clock edge

En: Enable

If \( \text{clk'event} \) and \( \text{clk='1'} \) then

\[ Q \leq D; \]

i.e. a stored variable,

Edge triggered D Flip-Flop

with enable.

Register timing

- **Propagation delay** (clk\rightarrow Q): Worst case (maximum) delay \( \text{after clk}\uparrow \) before new output data is valid on \( Q \).

- **Contamination delay** (cont\_clk\rightarrow Q): Minimum guaranteed time old output remains valid \( \text{after clk}\uparrow \).

\[ \begin{align*}
\text{Propagation delay (clk\rightarrow Q)} & : 250\text{ps} \\
\text{Contamination delay (cont\_clk\rightarrow Q)} & : 150\text{ps}
\end{align*} \]

A small exercise/problem

- What is the maximum clock frequency?

\[ \begin{align*}
\text{Reg} & \quad \& \\
\text{AND-gate} & \quad \& \\
\text{Reg} & \quad \& \\
\text{clk} & \quad \& \\
\text{Reg} & \quad \& \\
\text{Propagation delay (Tckl-Q)} & : 250\text{ps} \\
\text{Contamination delay (Tcont-clk-Q)} & : 150\text{ps} \\
\text{Setup time (Tsu)} & : 200\text{ps} \\
\text{Hold time (Th)} & : 100\text{ps} \\
\text{AND-gate} & \\
\text{Propagation delay (Tprop)} & : 250\text{ps} \\
\text{Contamination delay (Tcont)} & : 150\text{ps}
\end{align*} \]
Critical path

- ...begin to explore the construction of digital systems with complex behavior
  - Example: $K = (A +_1 B) *_1 (C +_2 D *_2 E)$
- Combinatorial circuit:
  - Fast (low latency)
  - Not flexible
  - Expensive
  - Latency of MUL?
  - Throughput?

Pipelining

- ...begin to explore the construction of digital systems with complex behavior
  - Example: $K = (A +_1 B) *_1 (C +_2 D *_2 E)$
- Combinatorial circuit:
  - Fast (low latency)
  - Not flexible
  - Expensive
  - Latency of MUL?
  - Throughput?

In the Laundry Processing Plant

60 min

8 h (critical path)

But you only have 4 hours!!

What is he talking about, either of us is in the wrong lecture...

Will be taught in the coming weeks

Combinational Logic

<table>
<thead>
<tr>
<th>$F$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>$A + B$</td>
</tr>
<tr>
<td>001</td>
<td>$A - B$</td>
</tr>
<tr>
<td>010</td>
<td>$A - 1$</td>
</tr>
<tr>
<td>011</td>
<td>$A \text{ and } B$</td>
</tr>
<tr>
<td>100</td>
<td>$A$</td>
</tr>
<tr>
<td>101</td>
<td>$A \text{ or } B$</td>
</tr>
<tr>
<td>110</td>
<td>$A \times B$</td>
</tr>
<tr>
<td>111</td>
<td>etc.</td>
</tr>
</tbody>
</table>

Pipelining
Selection

MUX

<table>
<thead>
<tr>
<th>sel[1:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

Shared bus with multiple drivers, only one driver may drive the bus, the others must be “disconnected” (high impedance / 'Z')

Registers (Flip-Flop’s)

BUS

Din → Dout
En → CLK

D Flip-Flop

• Flip-flop: Bit storage that stores on clock edge, not level
  • One design — master-servant
    – Two latches, output of first goes to input of second, master latch has inverted clock signal
    – So master loaded when C=0, then servant when C=1
    – When C changes from 0 to 1, master disabled, servant loaded with value that was at D just before C changed — i.e., value at D during rising edge of C

D Latch vs. D Flip-Flop

• Latch is level-sensitive: Stores D when C=1
• Flip-flop is edge triggered: Stores D when C changes from 0 to 1
  – Saying “level-sensitive latch,” or “edge-triggered flip-flop,” is redundant
  – Two types of flip-flops — rising or falling edge triggered.
• Comparing behavior of latch and flip-flop:
**D Flip-Flop**

- Solves problem of not knowing through how many latches a signal travels when $C=1$
  - In figure below, signal travels through exactly one flip-flop, for Clk_A or Clk_B
  - Why? Because on rising edge of Clk, all four flip-flops are loaded simultaneously -- then all four no longer pay attention to their input, until the next rising edge. Doesn’t matter how long Clk is 1.

![D Flip-Flop Diagram](image)

We will use positive edge triggered FFs

---

**Basic Register**

- Typically, we store multi-bit items
  - e.g., storing a 4-bit binary number
- **Register**: multiple flip-flops sharing clock signal
  - From this point, we’ll use registers for bit storage
    - No need to think of latches or flip-flops
    - But now you know what’s inside a register

![Basic Register Diagram](image)

---

**Programmable Structure**

- Scheduling / ordering / sequencing of operations
- Mapping / allocation:
  - Variables -> {Reg1, ... ,RegN}
  - Operations -> {MUL, ADD, ALU, ... ,}
- Concurrency?
- Data dependent latency?
Questions?