• Characteristics of digital systems
  ‣ Synchronous vs. Asynchronous
  ‣ Sequential vs. Combinational
  ‣ Design parameters
Synchronous vs. Asynchronous

• Synchronous: elements change their values at certain specific times (clocked)
  ‣ Period reference signal or clock causes the storage elements to accept new values and to change state

• Asynchronous: outputs can change at any instant time
  ‣ No single indication of when to change state
Types of Digital Circuits

- Combinatorial logic
  - A combinational circuit consists of logic gates whose outputs, at any time, are determined by combining the values of the inputs

\[ y_i = f_i(x_1, \ldots, x_n) \]
Types of Digital Circuits

- Sequential logic
  - Output depends not only on the present value of its input signals but on the sequence of past inputs
  - We now have a memory requirement!

\[
y_i^t = f_i(x_1^t, \ldots, x_n^t, s_1^t, \ldots, s_m^t)\\
S_i^{t+1} = g_i(x_1^t, \ldots, x_n^t, s_1^t, \ldots, s_m^t)
\]
Memory & Digital Design

- A sequential circuit combinational circuit with feedback through memory
  - The stored information at any time defines a state
- Outputs depends on inputs and previous inputs
  - Previous inputs are stored as binary information into memory
- Next state depends on inputs and present state

![Diagram of combinational circuit with memory](image)

- Inputs x
- Present state
- Outputs
- Next state
Types of Digital Circuits

- **Sequential logic**
  - Storage elements observe inputs and can change state only in relation to a timing signal.
  - Need for discrete instances of time.
  - We now have a clock requirement!
A synchronous system is synchronized according to a clock.

A clock cycle or cycle time or clock period is the duration between two consecutive rising or falling edges.

4 GHz = clock speed = \( \frac{1}{\text{cycle time}} \) = \( \frac{1}{250 \text{ ps}} \)
A storage element can maintain a binary state (0,1), until directed by an input signal to switch state.

Main difference between storage elements:
- Number of inputs they have
- How the inputs affect the binary state
A storage element can maintain a binary state (0,1), until directed by an input signal to switch state.

Two main types:
- Latches (level-sensitive)
- Flip-Flops (edge-sensitive)

Latches are useful in asynchronous sequential circuits.

Flip-Flips are built with latches.
Latches & Flip Flops

- A latch is binary storage element
- Can store a 0 or 1
- The most basic memory
- Easy to build
  - Built with gates (NORs, NANDs, NOT)

- SR Latch
- S'R' Latch
- SR Latch with Clock
- D Latch
Latches & Flip Flops

• A flip flop can be built using two latches in a master-slave configuration

• A master latch receives external inputs

• A slave latch receives inputs from the master latch

• Depending on the clock signal, only one latch is active at any given time
  - If clock=1, the master latch is enabled and the inputs are latched
  - if clock=0, the master is disabled and the slave is activated to generate the outputs
Computer Hardware Elements

- Combinational circuits
  - Mux, Demux, Decoder, ALU, ...

- Synchronous state elements
  - Flipflop, Register, Register file, SRAM, DRAM
    - Edge-triggered: Data is sampled at the rising edge
A Simple Memory Model

- Reads and writes are always completed in one cycle
  - A Read can be done any time (i.e. combinational)
  - A Write is performed at the rising clock edge
- If it is enabled Then the write address and data must be stable at the clock edge
In this class we will focus on FPGA-based design as a sub-domain of digital design.
In this class, we will learn the principles of RTL (register level transfer) coding for synthesis tools through the Verilog hardware description language (HDL) for the design and documentation of our electronic systems.

- Verilog allows designers to design at various levels of abstraction.
- It is the most widely used HDL.
Digital Design Flow

**Design Stage** | **Tools**
--- | ---
HDL Design (Verilog, VHDL, Bluespec) | Text Editor Emacs, Nedit, Vi
Verification | Mentor - ModelSim SE Synopsys - Leda
Synthesis | Synopsys - Design Compiler
Test Insertion | Synopsys - TetraMax Mentor - Fastscan
Static Timing Anal. | Synopsys - Primetime
Place & Route | Cadence - Senseless/SOC Encounter Synopsys - Apollo
Clock Tree Insertion | Cadence - CTgen
Timing Extraction | Synopsys - StarRXT Cadence - Pearl
DRC/ANT Checking | Cadence - Assura, Dracula Mentor – Calibre
LVS | Cadence - Assura, Dracula Mentor – Calibre
Programmable Logics

- Field Programmable Gate Arrays (more on it later)
  - Each cell in array contains a programmable logic function
Programmable Logics

• Field Programmable Gate Arrays (more on it later)

  › Array has programmable interconnect between logic functions
Class Laboratories

• Labs will be built around:
  • Single MIPS core for undergraduate students
  • 4-Core MIPS system for graduate students
Laboratory 0

- Half-Adder & Full Adder Designs
Binary Adder

- Binary adder

As a single module – Full Adder (FA)
Binary Adder

\[
\begin{array}{c}
\text{Adder} \\
A_3A_2A_1A_0 + B_3B_2B_1B_0 \\
\downarrow \quad \downarrow \\
S_3S_2S_1S_0 \\
\hline \\
C_{out} \quad \quad \quad \quad C_{in}
\end{array}
\]

\[
\begin{array}{c}
c_3 \quad c_2 \quad c_1 \quad c_{in} \\
+ \quad A_3 \quad A_2 \quad A_1 \quad A_0 \\
+ \quad B_3 \quad B_2 \quad B_1 \quad B_0 \\
\hline \\
C_{out} \quad S_3 \quad S_2 \quad S_1 \quad S_0
\end{array}
\]
Laboratory 1

- Design of register file and ALU
MIPS Instructions

- There 3 types of instruction in MIPS

1. **R-Type**

   - Format:
     - `op` (6 bits)
     - `rs` (5 bits)
     - `rt` (5 bits)
     - `rd` (5 bits)
     - `shamt` (5 bits)
     - `funct` (6 bits)

2. **I-Type**

   - Format:
     - `op` (6 bits)
     - `rs` (5 bits)
     - `rt` (5 bits)
     - `Immediate` (16 bits)
     - `Displacement` (16 bits)

3. **J-Type**

   - Format:
     - `op` (6 bits)
     - `target` (26 bits)
Arithmetic Logic Unit (ALU)

- Arithmetic circuits is built in a hierarchical fashion
- Input: data and operation to perform
- Output: result of operation and status information

```
+-------------------+
|  ALU Control      |
+-------------------+
| 4                 |
+-------------------+
| 32                |
+-------------------+
| A                 |
+-------------------+
| 32                |
| B                 |
+-------------------+
| 32                |
+-------------------+
| ALU               |
|                   |
|                   |
| Result            |
| 32                |
| Overflow          |
| Zero              |
| CarryOut          |
```

```
Arithmetic Logic Unit (ALU)

- Operation Examples

<table>
<thead>
<tr>
<th>ALU control lines</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>AND</td>
</tr>
<tr>
<td>0001</td>
<td>OR</td>
</tr>
<tr>
<td>0010</td>
<td>ADD</td>
</tr>
<tr>
<td>0110</td>
<td>SUB</td>
</tr>
<tr>
<td>0111</td>
<td>SLT</td>
</tr>
<tr>
<td>1100</td>
<td>NOR</td>
</tr>
</tbody>
</table>
Register File

- Read register number 1
- Read register number 2
- Write register
- Write data
- Read data 1
- Read data 2
- Write
Connecting Register File & ALU

Instruction

rs 5
rt 5
rd 5
32

read register 1
Read register 2
Write register
Write data

Write
go to ALU Control

32
read data 1
32
read data 2

ALU

Zero
Overflow

CarryOut

4
Laboratory 2

• Design of memory module
Memory Module

Address

Write Data

MemWrite

Data Memory

Read Data

MemRead
Laboratory 3

- CPU Design
Next Class

• Verilog Fundamentals