Lecture 13:
ALU & Sequential Logic
Two Concepts

- Combinational Logic
  - Output of the logic only depends on the value of the input
  - e.g.: adder, mux

- Sequential Logic
  - Output of the logic depends on not only the input but also the previous state of the logic
  - e.g. register, memory, traffic light control
An Abstract View of the CPU

Datapath

Control

Control Signals

Conditions

Ideal
Instruction
Memory

Instruction Address

Next Address

PC

Clk

Rd

Rs

Rt

32 32-bit Registers

32

32

32

A

B

Data

Address

Data

In

Clk

Ideal

Data

Memory

Data Out

Clk Out

An Abstract View of the CPU

Instruction

Fabrication
MUX & ALU
Data Multiplexor (here 2-to-1, n-bit-wide)

C = A if S = 0
C = B if S = 1
\[ c = \overline{s}ab + \overline{s}ab + s\overline{ab} + sab \]
\[ = \overline{s}(a\overline{b} + ab) + s(\overline{a}b + ab) \]
\[ = \overline{s}(a(\overline{b} + b)) + s((\overline{a} + a)b) \]
\[ = \overline{s}(a(1) + s((1)b)) \]
\[ = \overline{s}a + sb \]
Arithmetic and Logic Unit

- Most processors contain a special logic block called “Arithmetic and Logic Unit” (ALU)
- We’ll show you an easy one that does ADD, SUB, bitwise AND, bitwise OR

when $S=00$, $R=A+B$
when $S=01$, $R=A-B$
when $S=10$, $R=A \text{ AND } B$
when $S=11$, $R=A \text{ OR } B$
Our simple ALU
A 1-bit ALU.
Eight 1-bit ALU slices connected to make an 8-bit ALU. The enables and invert signals are not shown for simplicity.
Combinational Logic Elements
(Building Blocks)

Adder

- CarryIn
- A: 32
- B: 32
- Sum: 32
- CarryOut

MUX

- Select
- A: 32
- B: 32
- Y: 32

ALU

- OP
- A: 32
- B: 32
- Result: 32

The diagram illustrates the basic combinational logic elements used in digital systems, including an adder, a multiplexer (MUX), and an ALU (Arithmetic Logic Unit).
An Abstract View of the Implementation
Sequential logic

Memory element: latches, flip-flops
R-S Latch
S=0, R=0; Q holds its original value
\[ S = 0, R = 1: \]

**RESET** \( Q = 0 \)

\[ Q(t-1) = 0 \]
\[ Q(t) = 0 \]

\[ Q(t-1) = 1 \]
\[ Q(t) = 0 \]
$S=1, R=0$:

Set $Q=1$
R-S Latch

- Has feedback
  - The outputs are fed back to the inputs
  - This means that the state of the latch depends on the previous state of the latch ("sequential" rather than "combination")

- Quiescent state
  - Set: $S = 1$ (sets $Q = 1$)
  - Reset: $R = 1$ (sets $Q = 0$)
  - Hold $S=0$ and $R=0$ (hold value of $Q$)
Clocked R-S Latch
Clocked D Latches

Truth table:

<table>
<thead>
<tr>
<th>D</th>
<th>CK</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>( Q_0 )</td>
</tr>
</tbody>
</table>

Clock

\[ D_{(input)} \]

\[ Q_{(output)} \]
Flip-Flops (1)

(a) A pulse generator.
(b) Timing at four points in the circuit.
Flip-Flops (2) – D flip flop

Truth table

<table>
<thead>
<tr>
<th>D</th>
<th>CK</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>↑</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>↑</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>0,1</td>
<td>Q₀</td>
</tr>
</tbody>
</table>

Clock

D_{input}

Q_{output}
D-Flip Flop vs. D-Latch

**Truth table**

<table>
<thead>
<tr>
<th>D</th>
<th>CK</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>↑</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>↑</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>0,1</td>
<td>Q₀</td>
</tr>
</tbody>
</table>

**Truth table**

<table>
<thead>
<tr>
<th>D</th>
<th>CK</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>Q₀</td>
</tr>
</tbody>
</table>
An Abstract View of the Implementation
Storage Element: Idealized Memory

- Memory (idealized)
  - One input bus: Data In
  - One output bus: Data Out
- Memory word is selected by:
  - Address selects the word to put on Data Out
  - Write Enable = 1: address selects the memory word to be written via the Data In bus
- Clock input (CLK)
  - The CLK input is a factor ONLY during write operation
  - During read operation, behaves as a combinational logic block:
    - Address valid $\Rightarrow$ Data Out valid after “access time.”
Storage Element: Register (Building Block)

- Similar to D Flip Flop except
  - N-bit input and output
  - Write Enable input
- Write Enable:
  - negated (or deasserted) (0): Data Out will not change
  - asserted (1): Data Out will become Data In
Storage Element: Register File

- Register File consists of 32 registers:
  - Two 32-bit output busses: busA and busB
  - One 32-bit input bus: busW

- Register is selected by:
  - RA (number) selects the register to put on busA (data)
  - RB (number) selects the register to put on busB (data)
  - RW (number) selects the register to be written via busW (data) when Write Enable is 1

- Clock input (CLK)
  - The CLK input is a factor ONLY during write operation
  - During read operation, behaves as a combinational logic block:
    - RA or RB valid => busA or busB valid after “access time.”
Instruction Fetch Unit

- The common RTL operations
  - Fetch the Instruction: \text{mem}[\text{PC}]
  - Update the program counter:
    - Sequential Code: \( \text{PC} = \text{PC} + 4 \)
    - Branch and Jump: \( \text{PC} = \text{“something else”} \)