Orange Coast College
Business Division
Computer Science Department

CS 116- Computer Architecture

Logic Design: Part the Last
Sequential Circuits

• Circuits with memory
• Outputs depend on:
  – past sequence
  – possibly the input
• Represented with state diagrams or tables
State Stuff

- **State (Transition) diagram:**
  - Circles represent states
  - Directed line segments represent transitions between the states.
  - One or more actions (outputs) may be associated with each transition.
  - Represents a finite state machine.

- **Finite state machine:**
  - A function which maps an ordered sequence of input events into a corresponding sequence of (sets of) output events.
State Diagram Examples

- Insert pics?
  - Special counter: http://www.utdallas.edu/~frankd/SD1.html
  - Invoice: http://www.dcs.warwick.ac.uk/~ananda/lnotes/node278.html
  - Multiprocesor system: http://www.dca.see.unicamp.br/~leopini/private/sib98/sld037.htm
  - T-Flip-flop: http://ranger.uta.edu/~carroll/cse2341/spring99/chap.htm
Sequential Circuit Models

- **Mealy machine**
  - Output depends on both input & current state
  - Output changes whenever input changes
Sequential Circuit Examples

- State diagram for Mealy machine
Sequential Circuit Models

• Moore machine
  – Output depends on current state only
  – Output changes after clock edge
Moore Machine State Diagram

- State diagram for Moore machine
Memory Elements

- Memory element:
  - Stores values
  - Controlled usually by clock
  - Can be static or dynamic, volatile or non-volatile

![Diagram of memory element](image-url)
Clocks

• Relative execution times of instructions on a computer are usually measured by number of clock cycles rather than seconds.

• Clock rates for various models of the computer may increase as technology improves.

• Definitions:
  – A free-running signal with a fixed cycle time.
  – A processor's clock or one cycle thereof.
Clock Systems

• Synchronous system:
  – A system in which state changes only occur in specific time controlled by a free-running clock.

• Asynchronous system:
  – A system in which state changes occur according to some other events.

• Clock cycle time (period/interval):
  – The time between successive transitions in the same direction, i.e. a complete period in which the signal has one high and one low signal levels.
Clock Definitions

- **Duty cycle**: The percentage that the clock signal is at its asserted level.
- **Clock frequency**: The inverse of the cycle time.
- **Edge-triggered clocking**: State changes occur on a clock edge.
- **Rising edge**: The edge that converts the signal from low to high.
- **Falling edge**: The edge that converts the signal from high to low.
- **Set-up time**: The minimum time that the input must be valid before the clock edge.
- **Hold time**: The minimum time during which the input must be valid after the clock edge.

A clock signal oscillates between high & low values
Clocks

- Clock edge determines when contents of state elements are updated
  - Cycle must be long enough for input values to stabilize.
  - Edge triggered clocking preferred because a state element can be used as both input and output to the same logic circuit.
- Clocks are needed in sequential logic to decide when an element that contains a state should be updated.
- Must have enough period that allows all signals to stabilize.
Bistable Elements

- The most fundamental element from which all flip flops are constructed
- A pair of inverters connected to each other.
  - Circuit is always self consistent
- No way of controlling or changing the element's state.
  - Random state when power is switched on
  - Stays there forever.
- 2 possible stable states
Latch

• Used to store 1 or more bits
  – Has a data input and an output
  – Input is latched (stored)
  – Transferred to output
    • Output retains value until next clock cycle
More Latches

• Simplest type of memory element
  – No clock involved
  – Change can occur any time as long as the input is asserted
  – After the input is applied, the latch remains in its state.
Flip Flop (FF)

- Can be in one of two states
  - Toggles between based on inputs
  - Essentially a 1 bit memory

- Common types
  - SR flip-flop
  - JK flip-flop
  - D-Type flip-flop (latch)
  - T flip-flop
More flip-flops

• Non-transparent
• Reading input and generating output are separate events
• Changes occur only at a clock edge

• Flip-Flop’s and latches are the basic building blocks of most sequential circuits. Their function is to store signals
Set-Reset (S-R) Latches

- Built from a pair of NOR or NAND gates.
- Two-Inputs:
  - S for set & R for reset.
- Two-Outputs:
  - Q output & Q' (inverted output).

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>Q'</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0'</td>
</tr>
</tbody>
</table>

**Implementation Using NANDs**

**Implementation Using NORs**
Set-Reset Latches

- What happens on different inputs?
  - Only S is asserted
    - Latch is set to the state \((Q=1 \text{ and } Q'=0)\).
  - Only R is asserted
    - Latch is reset to the state \((Q=0 \text{ and } Q'=1)\).
  - Both S & R are not asserted
    - Latch stays in its current state.
  - If both S and R are asserted
    - Undefined AND Dangerous !!

- Circuit behaves like a bistable element.
  - Can lead to incorrect operation or oscillate in an unstable state.
Delay (D) Latch/FF

• Stores the value of its data input signal.
• Implementation
  – Using gates
  – Using an inverter with SR flip flop
• Applications:
  – Set or reset flags in response to some condition
  – Build registers and store bits of information Input using an array of D-FFs
Delay/Flip-Flop

• **Output:**
  – The value of the internal state (Q)
  – Complement of internal state (Q')

• **When the input is**
  – Asserted
    • Latch is open and the value of the output becomes the value of the input.
  – Deasserted:
    • Latch is closed and the output holds its value.
Closer look at Delay Flip-flops

Assumption: output initially deasserted

Truth table

<table>
<thead>
<tr>
<th>Clock</th>
<th>D</th>
<th>Q</th>
<th>Q'</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>Q</td>
<td>Q'</td>
</tr>
</tbody>
</table>

Operation of a D-latch

Assumption: output initially deasserted
Master-Slave FFs

- Also called Edge-Triggered or Falling-Edge FF
  - Example: Two D-FFs:
    - The Master is open and follows the input (D) when the clock (C) is asserted.
    - The slave is open when the clock falls, while the master will be closed. It gets the input from the output of the master FF.

\[\text{D} \rightarrow \text{FF} \rightarrow \text{Q} \]

\[\text{D} \rightarrow \text{FF} \rightarrow \text{Q} \]

\[\text{D} \rightarrow \text{FF} \rightarrow \text{Q} \]
Master-Slave Flip-flops

- During each clock period at most one state change will take place
  - Race conditions is avoided
- Especially useful when input of a FF is a function of its own output
- We can also have Master-Slave S-R or J-K FFs

Operation of a D-FF with a falling edge trigger, assuming the output is initially deasserted.
Master-Slave FFs

• Race Conditions:
  – Multiple variables change state due to a single input changing state
  – Non-critical racing
    • Exactly one final state is reached regardless of the order and speed of internal variable changes
  – Critical racing:
    • Two or more final states could be reached
    • Depends on the order and speed of the internal variable changes

• Critical racing should be avoided
  – The behavior of the circuit cannot be predicted
Other Flip-Flops

- **J-K-FFs**
  - A modified SR flip-flop
  - Prevents both inputs from being assigned 1 simultaneously

- **T-FFs**
  - A single-input JK flip-flop
  - Both inputs are tied together
  - The output toggles whenever input is applied
## Characteristics Tables of FFs

### J-K FF

<table>
<thead>
<tr>
<th>J</th>
<th>K</th>
<th>Q(t+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>q(T) No change</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0 Reset</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1 Set</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>q'(T) Complement</td>
</tr>
</tbody>
</table>

### S-R FF

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q(t+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>q(T) No change</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0 Reset</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1 Set</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>? Unpredictable</td>
</tr>
</tbody>
</table>

### D FF

<table>
<thead>
<tr>
<th>D</th>
<th>Q(t+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reset</td>
</tr>
<tr>
<td>1</td>
<td>Set</td>
</tr>
</tbody>
</table>

### T FF

<table>
<thead>
<tr>
<th>T</th>
<th>Q(t+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Q(t) No change</td>
</tr>
<tr>
<td>1</td>
<td>Q'(t) Complement</td>
</tr>
</tbody>
</table>
## Excitation Tables of FFs

<table>
<thead>
<tr>
<th>S-R FF</th>
<th>J-K FF</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Q(t)</strong></td>
<td><strong>Q(t+1)</strong></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D FF</th>
<th>T FF</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Q(t)</strong></td>
<td><strong>Q(t+1)</strong></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

| **Q(t)** | **Q(t+1)** | **D** |
| 0     | 0     | 0    |
| 0     | 1     | 1    |
| 1     | 0     | 1    |
| 1     | 1     | 0    |
Sequential Circuits Design Steps

• State the description of the circuit behavior,
  – State diagram, timing diagram, etc.

• Obtain the state table.
  – Reduce the number of states
  – Assign binary value to each state.

• Determine the number of FFs needed for the state combination.
  – Choose the type of the FFs used.
Designing a circuit Part 2

• From the state table, derive the excitation table.
  – Derive the circuit output functions and FFs input functions,
    • Karnaugh map, or any other simplification method.

• Draw the logic diagram.
Sequential Circuits Analysis Steps

• Determine the value of the input function,
  – In terms of the present state and input variables
  – For each flip-flop in the circuit.

• Use the flip-flop characteristics table to determine the next state.

• Build the State table
  – Use information known
    • Use columns indicating the present state, input, next state, and output.

• Draw the state diagram
 Registers

- High-speed memory locations in a CPU.
- Only a small number of registers available
  - “Register set” or “Register file”
- Typically 32 in a modern processor
  - (Some, e.g. SPARC, have as many as 144)
Registers Cont'd

- May be directly addressed with a few bits
- Fast access;
  - Typically, two registers can be read and a third written -- all in a single cycle
Special Registers

• $0:$
  – Contains always zero.
  – No write logic required for this register.

• $31:$
  – Contains return address (link) for procedure calls.
  – Can only be loaded via ALU
Special Registers

- Shift Registers
  - Serial shift-register:
    - The output of one flip-flop is the input of the next
  - We can add more circuitry to allow the shift in the other direction too

![4-Bit Shift-register Diagram]
Special Registers

- **Parallel-Load register**
  - The input is entered in parallel

- The load input can also be added to the previous register to generate a universal register that allows:
  - Shift left
  - Shift right
  - Parallel load
  - Clear
Register Files (Register Sets)

• Set of 32 registers
  – Indexed by register number

• Input signals:
  – Register numbers (source & destination)
  – Data
  – Write signal

• Output signals
  – Result
Implementation

- Implementation options:
  - Decoder for each read or write port
  - Multiplexor to choose the read port
  - Array of registers
Operations

• Reading from a specific register.
  – Input: Supply the register number.
  – Output: Data contained in the indicated register.
  – Reading a register doesn’t change its state.

• Writing into a register.
  – Input: Supply the data, the register number, clock
  – Output: No output is specified.
Register File Diagram

A register file with 2 read ports & 1 write port
5 inputs & 2 outputs
Implementation of Register Files

- Implementation of 2 read ports
Implementation of Register Files

- Implementation of the Write ports
- Decoder is used to generate clock input to registers
Counters

• Counter:
  – Register that goes through a sequence of states when input applied
    • Input may be a clock pulse or other source
    • Input may be random or at fixed time intervals
Binary Counters

• Clock input
• A number of count outputs which give the number of clock cycles.
  – The output may change either on rising or falling clock edges.
• May also have a reset input
• The counter may be either a synchronous counter or a ripple counter
Other Counters

- **n-bit binary counter:**
  - Counts from 0 to $2^n-1$
  - Needs $n$ FFs
- **Synchronous counter:**
  - All FFs have common clock
  - State change determined from the present state
- **Ripple counter:**
  - The output of one FF is used to trigger another FF
Counters

- Example: 2-bit binary counter
  - 2 Flip-flops are needed
- Exercise
  - Draw the sequential circuit

<table>
<thead>
<tr>
<th>F0 (t)</th>
<th>F1 (t)</th>
<th>f0 (t+1)</th>
<th>f1 (t+1)</th>
<th>F1 (t)</th>
<th>F1 S</th>
<th>F1 R</th>
<th>F2 S</th>
<th>F2 R</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Programmable Logic devices (PLD)

• Programmable Logic:
  – A logic element whose function is not restricted to a particular function.
  – May be programmed at different points of the life cycle

• PLD Compiler:
  – A software tool that converts functional description into a set of interconnections that can be programmed into the PLD.
PLD Details

• A large-scale integration (LSI) chip
  – Contains regular circuit structure
  – Is programmed by the purchaser

• Implementation
  – A programmable AND array followed by a fixed fan-in OR gates that are followed by flip-flops.

• Product lines can be any input combination and flip-flops can be fed back to input.

• Programmable element can be fuse or a transistor.
More PLDs

- Densities range from 1,000 to 10,000 gates.
- Reduced cost for both customer and manufacturer.
- Many modern PLDs are erasable and can be reprogrammed.
Basic PLD Block Diagram
Programmable Logic Array (PLA)

- A combinational circuit,
  - 2-level, AND-OR device programmed
  - Can realize any sum-of-product logic expressions
- Has both programmable AND and OR planes.
  - Programmed by establishing the connections that are actually needed.
Programmable Logic Array (PLA)

- A set of inputs and input complements

- Two stages of logic.
  - First: Array of AND gates (AND-plane)
    - Forms a set of product terms.
  - Array of OR gates (OR-plane),
    - May be a logical sum of any number of the product terms.

- Can directly implement truth table
  - The number of OR gates correspond to the number of truth table entries for which the output is true.
PLA Fun

• Example:
  – PLA for implementing the function:
  \[ D = (A' \cdot B' \cdot C) + (A' \cdot B \cdot C') + (A \cdot B' \cdot C') + (A \cdot B \cdot C) \]

• Exercise:
  – Find the equations for \( E \) and \( F \)

\[
\begin{array}{cccc|ccc}
A & B & C & D & E & F \\
\hline
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 0 \\
\end{array}
\]

Another way to draw the PLA
Read-Only Memories (ROMs)

• Definitions:
  – Nonvolatile memory that can only be read
  – A combinational circuit
    • n-address inputs
    • b-data outputs
    • Encodes logic functions directly from truth tables.
  – Manufactured with fixed contents
  – Internal structure works with diodes or transistors.
    • The presence or absence of a diode or a transistor distinguishes between 0 and 1.
  – Modern ROMs are fabricated as a single IC chip.
ROMs

- Advantages:
  - Low-cost
  - Inherently non-volatile

- Application:
  - Programs for embedded systems (microprocessor controlled) (e.g. washers, microwave, ...)
  - Storage of the lowest level bootstrap software (firmware) in a computer.
  - Permanent storage of computer programs
    - Look-up tables
ROMs

- Characteristics:
  - Input (height):
    - $2^n$ addressable entries (Address lines) & $n$-input lines.
  - Output (Width):
    - Number of bits ($d$) in each addressable entry.
  - Total number of bits = Height x Width.
ROM Variations

- **Programmable ROMs (PROMs):**
  - Can be programmed electronically, when the designer knows their contents.

- **Erasable PROMs (EPROMs):**
  - Requires slow erasure process using ultraviolet light.
  - Special devices are needed for reprogramming

- **Electrically Erasable PROM (EEPROM):**
  - Stored bits may be electrically erased.
## ROMs vs. PLAs

<table>
<thead>
<tr>
<th>ROM</th>
<th>PLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully decoded:</td>
<td>Partially decoded.</td>
</tr>
<tr>
<td>Contains a full output word for each possible input combination.</td>
<td></td>
</tr>
<tr>
<td>Number of entries grows exponentially with number of inputs.</td>
<td>Number of product terms grows more slowly.</td>
</tr>
<tr>
<td>Less efficient for implementing combinational logic.</td>
<td>More efficient for implementing combinational logic.</td>
</tr>
<tr>
<td>Ability to implement any logic function with the matching number of input and outputs.</td>
<td>To implement a different function, modification is required.</td>
</tr>
<tr>
<td>Easy to change ROM's contents if the logic function changes.</td>
<td>To change of content requires modification in the connections and/or number of gates.</td>
</tr>
<tr>
<td>Size doesn't change.</td>
<td>Size might change.</td>
</tr>
</tbody>
</table>
Static RAMs (SRAM, not spam)

- Registers & register files are building blocks for small memories.
- Large memories are built using SRAMs or DRAMs.
SRAM

- SRAM is:
  - Each bit of storage is a bistable flip-flop
  - Will retain a value as long as power is supplied
  - Refresh regularly, unlike dynamic random access memory (DRAM).
  - Will lose its contents when the power is turned off, unlike ROM
  - IC-chips memory arrays with read/write port.
  - The value kept in a cell is kept on a pair of inverting gates.
  - As long as power is applied, the value is kept indefinitely.
Static RAMs (SRAM)

- Type of SRAM is defined by:
  - Width and height:
  - Number of addressable locations
  - Width of each addressable location

- SRAMs have fixed access time (5-25ns).

- Address line depends in the first number
  - To initiate read/write access, the Chip select signal must be active.
  - The Output enable signal allows the output data to be accessed.
SRAM Example

- Example:
  - 256k x 1 SRAM $\equiv 256k$ ($=2^{18}$) entries, each 1-bit wide
    - needs
    - 18 address
    - 1 data input
    - 1 data output lines
  - 32k x 8 SRAM $\equiv 32k$ ($=2^{15}$) entries, each 8 bits wide
    - Needs
    - 15 address
    - 8 data output
    - 8 data input lines are needed
SRAM Operations

- **Reading from a specific location.**
  - input: Register number.
  - output: Data contained in the indicated register.

- **Writing into a specific location.**
  - input: Data, location, write enable signal, and Chip select signals.
  - output: The Chip select & Output enable signals should be activated.

- There are set-up and hold-time requirements for the address & data lines.

- Write enable signal isn't a clock, but a pulse with minimum width requirements.

- Instead of using MUXs, large memories are implemented with a shared output line (Bit line).

- **Bit line:** A shared line that multiple memory cells in a memory array can assert.
Basic Structure of a 4x2 SRAM
Fast SRAM

- Typical organization of a 32x8 SRAM
Dynamic RAMs (DRAMs)

- The value kept in a cell is stored as a charge in a capacitor.
  - A single transistor is used to access the stored charge.
- The value can't be kept indefinitely
  - Must be periodically refreshed.
  - The charge can be kept for few milliseconds.
- Refreshing the cell
  - Read the data and then write it back.
- Two-level decoding structure is used, that allows an entire row to be refreshed.
DRAM

- To save pins & reduce cost, the same address lines are used for both rows & column addresses.
  - The access is slower than SRAMs. Typical DRAM access times range from 60 - 110 ns.
  - Row Access Strobe (RAS): used to signal row addressing.
  - Column Access Strobe (CAS): Used to signal column addressing.
More DRAM

- A single transistor DRAM

- A 4M x 1 DRAM with a 2048 x 2048 array
Error Detection & Correction

• Most computer systems use some sort of error-checking code to detect possible corruption of data.

• A collection of methods to detect errors in transmitted or stored data and to correct them.

• Involves some form of encoding
Error Checking Method

• Single parity:
  – The simplest form of error detection is a single added parity bit or a cyclic redundancy check.
  – Parity code:
    • The number of 1's in a word is counted.
  – Odd parity:
    • If the number of 1's is odd.
  – Even parity:
    • If the number of 1's is even.
  – Parity can only detect, but not correct errors.
  – Only odd number of errors can be detected.
Error Detection & Correction

• Multiple parity:
  – Detect that an error has occurred
  – Which bits have been inverted,
  – The more extra bits are added, the greater the chance that multiple errors will be detectable and correctable.

• Several codes can perform
  – Single Error Correction, Double Error Detection (SECDEC).
  – One of the most common is the Hamming code.

• Other error-correcting codes exist and allow detection as well as correction of errors. More bits are used to encode the data.
Buses

• A set of electrical connections (wires) through which signals (and power) can pass.
  – May be either synchronous or asynchronous

• In a computer bus signals usually synchronous,
  – Controlled by the system clock.
Different Busses

• Data bus :
  – 8, 16, 32, 64, 80 (or more) wires for transmission
  – Bi-directional
  – Parallel

• Address bus :
  – Governs the amount of memory that a computer can address.
  – Unidirectional, only the processor can 'address' memory
  – Parallel
• Control Bus:
  – This includes all the different wires to carry power, earth, clock signals, interrupts, controls for other buses, connections to other processors and chips, logic signals, any other electrical connections at 5V or below.
  – Consists (mostly) of individual wires.
  – Serial
Bus Picture
Bus business

• Bus notation:
  – Usually has its own name
  – Drawn with either a double arrow or a thick line.

• Assumption
  – A bus with no label is assumed to be 32-bits wide
  – If not 32-bits it will be explicitly indicated on the graph.