Overview of Chapter 4

- Part 1:
  - Types of Sequential Circuits
  - Storage Elements
  - Latches and Flip-Flops
- Part 2:
  - Sequential Circuit Analysis
  - State Tables and State Diagrams
- Part 3:
  - Sequential Circuit Design
    - Specification
    - Assignment of State Codes
    - Implementation
- Part 4:
  - HDL Representation

Sequential Circuits

- A Sequential circuit contains:
  - Combinatorial Logic:
    - Next state function:
      \[ \text{Next State} = f(\text{Inputs, State}) \]
    - Output function (Mealy):
      \[ \text{Outputs} = g(\text{Inputs, State}) \]
    - Alternate output function (Moore):
      \[ \text{Outputs} = h(\text{State}) \]
  - Type of output function heavily influences the design

Types of Sequential Circuits

- Depends on time at which inputs are observed by storage elements and state of storage elements change
- Synchronous:
  - Behavior defined from knowledge of its signals at discrete instances of time
  - Storage elements affected by inputs and can change state only in relation to a timing signal (clock pulses) from a clock
- Asynchronous:
  - Behavior defined from knowledge of inputs at any instant of time and the order (in continuous time) in which inputs change
  - If clock just regarded as another input, all circuits are asynchronous!
  - Synchronous abstraction makes complex designs tractable!

Discrete Event Simulation

- In order to understand the time behavior of a Sequential Circuit we use discrete event simulation.
- Rules:
  - Gates modeled by an ideal (instantaneous) function and a fixed gate delay
  - Any change in input values is evaluated to see if it causes a change in output value
  - Changes in output values are scheduled for after the fixed gate delay
  - At the time for a scheduled output change, the output value is changed along with any signals/lines connected to it
Simulated NAND Gate

- Example: A 2-Input NAND gate with a 5 ns. delay:
- Assume A and B have been 1 for a long time
- At time t=0, A changes to a 0 at t=8 ns, back to 1.

<table>
<thead>
<tr>
<th>t (ns)</th>
<th>A</th>
<th>B</th>
<th>F(I)</th>
<th>F Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0 A=B=1 for a long time</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>F(I) changes to 1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>F changes to 1 after a 5 ns delay</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>F changes to 0 after a 5 ns delay</td>
</tr>
</tbody>
</table>

Gate Models

- Suppose we represent gates with delay n ns as follows:

Storing State

- Consider a simple 2-input multiplexer:
- With function:
  - Y = A for S = 1
  - Y = B for S = 0
- What would happen if we connect output signal Y to input signal A?

Simulation example as input signals change with time. Changes occur every 100 ns, so that the 0.2 ns delays are negligible.

<table>
<thead>
<tr>
<th>B</th>
<th>S</th>
<th>Y</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Y=A when S=1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Y=B when S=0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Y “remembers” B for S=0, even after B changes</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Y “remembers” B for S=0, even after B changes</td>
</tr>
</tbody>
</table>

Storing State (Continued)

- The circuit becomes:
- With function:
  - Y = B for S = 1, and
  - Y(t) dependent on Y(t-2) for S = 0
- The simple combinational circuit has now become a sequential circuit because its output is a function of a time sequence of input signals!

Storing State (Continued)

- Suppose we place an inverter in the “feedback path.”
- The following behavior results:
- The circuit is said to be unstable.
- For S = 0 it is an oscillator!
Basic (NAND) S – R Latch

- "Cross-Coupling" two NAND gates gives the S – R Latch:

- Which has the time sequence behavior:

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>Q'</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>We don't know</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>&quot;Set&quot; Q to 1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Now Q &quot;remembers&quot; 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>&quot;Reset&quot; Q to 0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Now Q &quot;remembers&quot; 0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Both go high</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>?</td>
<td>INSTABILITY</td>
</tr>
</tbody>
</table>

Basic (NOR) S – R Latch

- Cross-coupling two NOR gates gives the S – R Latch:

- Which has the time sequence behavior:

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>Q'</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>We don't know</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>&quot;Set&quot; Q to 1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Now Q &quot;remembers&quot; 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>&quot;Reset&quot; Q to 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Now Q &quot;remembers&quot; 0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Both go low</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>?</td>
<td>?</td>
<td>INSTABILITY</td>
</tr>
</tbody>
</table>

Clocked (or Gated) S – R Latch

Adding two NAND gates to the basic S - R NAND Latch, we arrive at the Clocked S – R Latch:

This has a time sequence behavior similar to the Basic S – R Latch except that:
* S and R are now active high signals (i.e. -- a "1" signal on S sets Q to 1) and
* The S and R inputs are only observed when the line C is high.
* C has the meaning "Clock" or "Clock Pulse".

Clocked S - R Latch (Continued)

The Clocked S-R Latch can be described by a table:

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q0</th>
<th>Q(t+1)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>No change</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Clear Q</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Set Q</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>?</td>
<td>Indeterminate</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>No change</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Clear Q</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Set Q</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>?</td>
<td>?</td>
<td>Indeterminate</td>
</tr>
</tbody>
</table>

The table describes what happens after the clock [at time (t+1)] based on: current inputs (S,R) and current state Q(t).

Characteristic Equation for S - R Latch

We can describe the behavior of output Q at time (t+1) (immediately after one clock pulse) using a K-Map:

We can see that:
Q(t+1) = S + R' Q

The Clock Latch S – R Latch has the symbol:

D Latch

Adding an inverter to the S-R Latch, gives the D Latch:

Note that there are no "Indeterminate" states!

<table>
<thead>
<tr>
<th>Q</th>
<th>Q'</th>
<th>D</th>
<th>D'</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>No change</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Set Q</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Clear Q</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>No Change</td>
</tr>
</tbody>
</table>

The graphic symbol for a D Latch is:
Latch and Flip-Flop Triggering

So far, the latches we have talked about are "clocked" with an input pulse. Here are some possible waveforms:

- Positive Clock Pulse
- Negative Clock Pulse
- W = Pulse Width
- Clock Period = Time between referenced edges.
- Reference level is generally 50%.
- Rise and Fall times may be important as well.

System Level Clocking

Consider a system comprised of ranks of latches or flip-flops connected by logic:

If the Clock Period is TOO SHORT, some data changes will not propagate through the network.
If the Clock Pulse Width is TOO LONG, some data will propagate through the second rank of latches!

Master-Slave Flip-Flop

One way to solve the Clocking Problem is with a master-slave organization:

- The complement of the clock is used to change the outputs.
- Now outputs change on C only.
- Problem: One's catching in Master can result into instability.
- Another solution: Use D-FF's or Edge Triggering

Edge Triggered Flip-Flops

Edge triggered Flip-Flops are sensitive to a small window for data changes around the time of a clock edge.

- Setup Time: The time required for input data to be stable before the clock edge.
- Hold Time: The time data must remain stable after the clock edge.

Flip-Flop Characteristic Tables

The Characteristic Tables:

- Show current inputs.
- Show current state implicitly.
- Predict flip-flop state AFTER CLOCKING.

Clocking conditions are:

- Positive level triggered.
- Negative level triggered.
- Positive edge triggered.
- Negative edge triggered.

NOTE: Proper clocking or flip-flop operation may be subject to conditions such as:

- Set-up and hold times are met.
- Simultaneous SR changes disallowed.

Characteristic Tables

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

<table>
<thead>
<tr>
<th>S</th>
<th>D</th>
<th>Q(t+1)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Q(t)</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Q'(t)</td>
<td>Complement Q</td>
</tr>
</tbody>
</table>

Positive Edge Triggered FFs

Negative Edge Triggered FFs
J-K Master Slave Flip Flop

Two SR Latches driven by inverted clocks form a master-slave configuration. 

Input logic forms the J-K logic transition:
* Set (master) = J'Qs'
* Reset (master) = KQs

Master Set is possible if the slave Qs is currently "0" any time the clock CP is high!

Master Reset is possible if the slave Qs is currently "1" any time the clock CP is high!

This is referred to as One's Catching.

Master Slave Symbols

Master-Slave Flip Flops are denoted by a line near the outputs.

This highlights the fact that the slave changes AFTER the master clocking condition is deasserted.

Flip-Flop Conventions

A Bubble near a clock input denotes an active low assertion.

A Triangle near the clock input denotes edge sensitive.

A L-Shaped Line near the output denotes Master/Slave.

Calculating Clock Frequency

Given the network below, assume signal A is changing from "1" to "0":

Clock Period = tsu (Flip-Flop) + 3*tpLH (Logic) + tsu (Flip-Flop)

Frequency = 1/(Clock Period)

Calculating Clock Frequency (Continued)

Given the network below, assume signal A is changing from "0" to "1":

Clock Period = tsu (Flip-Flop) + 3*tpLH (Logic) + tsu (Flip-Flop)

Frequency = 1/(Clock Period)

We usually pick MAX(tsu, tsu, tsu).