Sequential Logic (2)

ENGG1015
1st Semester, 2012
Dr. Hayden So
Department of Electrical and Electronic Engineering
http://www.eee.hku.hk/~engg1015

Synchronous vs Asynchronous Sequential Circuit

- In synchronous sequential circuits, all state elements are updated synchronously according to a single clock signal
- In asynchronous sequential circuits, state elements may be updated with multiple clocks, no clock signal, or any other schemes.

Synchronous Sequential Circuits
- A synchronous sequential circuit contains exactly 1 clock signal
- All state elements are connected to the same clock signal
  - the state of the entire circuit is updated at the same time
- Common form of synchronous sequential circuits:

![Synchronous Sequential Circuit Diagram](image)

Clock Signal
- A clock signal is particularly important signal in a synchronous sequential circuit
  - It controls the action of all DFFs
- A clock signal toggles between ‘0’ and ‘1’ periodically
- The frequency of the toggling determines the maximum speed of the circuit
  - E.g.: in the accumulator example earlier, the output S cannot change faster than the clock frequency

Finite State Machine (FSM)
- Finite State Machine (FSM) is an abstraction of computation
  - Can be used to model many computing tasks, both in software and in hardware
- Very useful abstraction to help design sequential circuits
  - It is systematic, and can be analyzed mathematically
- Used to describe very complex behavior of circuits and systems
  - Decision making
  - Network communication
  - Microprocessor control...
Defining Finite State Machines

- Each FSM defines:
  - Finite number of states that the machine can be in
  - The conditions under which it will transition from one state to another
- At any moment in time, an FSM can only exist in 1 of the defined states
- The output of an FSM depends on the state that the FSM
  - Optionally depend on the input to the FSM

Quick Quiz

Which of the following is/are possible sequence(s) of states that the FSM may go through?

1. S0 S1 S3 S2 S0
2. S0 S1 S1 S2 S2 S0
3. S0 S0 S1 S3 S3 S3
4. S2 S2 S0 S0 S0 S0

State Transition Diagram

- A graphical tool to describe the behavior of an FSM
- Represent states as blocks
  - Labeled: name of the state
- Represent transitions as directed edge
  - Direction of an edge represents the direction of state transition
- All possible states & transitions are included

State Transitions

- Each state transition is labeled with:
  - 1. Condition that the transition should take place
  - 2. Output of the FSM during the transition
- There should only be 1 active transition at any one time
  - The input conditions of all transitions in a FSM should be mutually exclusive

Ex: Ticket Gate at MTR

- The gate should only open after a valid Octopus card is scanned.
- It should close the gate after a person has passed through the gate.

FSM in Hardware

- FSM can be efficiently implemented in hardware using synchronous sequential circuits
  - FSM states can be implemented by registers
  - State transition conditions can be implemented by combinational function on input signals and the states
  - FSM outputs are simply output signals of the circuit
- Transition condition is checked on every clock edge
Ticket Gate Control (1)

Step 1: Define the input/output signals
- We use the following input signals:

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>valid</td>
<td>'1' if a valid Octopus card is presented, '0' otherwise</td>
</tr>
<tr>
<td>input</td>
<td>passed</td>
<td>'1' if a passenger has completely passed through the gate, '0' otherwise</td>
</tr>
<tr>
<td>output</td>
<td>motor</td>
<td>'1' close the ticket gate by turning on a motor , '0' otherwise</td>
</tr>
</tbody>
</table>

Ticket Gate Control (2)

Step 2: Determine how the FSM states will be represented in hardware
- 2 FSM states → 1 DFF needed
- Encode the state as follows:
  - '0' → WAIT_CARD state
  - '1' → WAIT_PASS state

Ticket Gate Control (3a)

Step 3: Implement the state transition logic
- At each cycle, determine what is the next state this FSM should be in in the next cycle
  - Determine which transition is active by checking all the transition conditions
- The next state logic is a combinational function of the current state and the input signals
  - the input to the state register
  - Can be found using a truth table

Ticket Gate Control (3b)

<table>
<thead>
<tr>
<th>s</th>
<th>valid</th>
<th>passed</th>
<th>ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Ticket Gate Control (4a)

Step 4: Determine the output logic
- Can be performed similar to the way the next state logic is obtained

<table>
<thead>
<tr>
<th>s</th>
<th>valid</th>
<th>passed</th>
<th>ns</th>
<th>motor</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Ticket Gate Control (4b)
Ticket Gate Control (5)

Step 4: Implement the circuit

Next state (ns) is a combinational function on current state (s) and current input.

Output depends on current state, but not the input.

Moore Machine vs Mealy Machine

- In a **Mealy machine**, the output depends on both input and the current state of the machine.
- In a **Moore machine**, the output depends only on the current state, but not the input.

Moore machine avoids combinational path between input and output of a state machine.

However, in general, Moore machine requires more states to implement the same function than a Mealy machine.

State Encoding

- State encoding refers to the way the abstract FSM states are represented in hardware.
- In a **binary encoding scheme**, each state is encoded using an n-bit binary number.
  - 2^n possible states
  - The ticket gate example is the simplest binary encoding with 1 bit = 2 states.
- In a **one-hot encoding scheme**, each FSM state is encoded using 1 bit at a unique bit position.
  - n DFFs are used to encode n FSM states.

Observations:
- One-hot encoding scheme requires more DFFs for large FSMs.
- But generally simpler next state/output logic.
- The mapping between state and its encoding can be quite arbitrary.
  - E.g. No reason why State_C cannot be “00”
  - Non-trivial to define the best encoding.

State Encoding Example:

<table>
<thead>
<tr>
<th>State</th>
<th>Binary</th>
<th>One-Hot</th>
</tr>
</thead>
<tbody>
<tr>
<td>State_A</td>
<td>00</td>
<td>0001</td>
</tr>
<tr>
<td>State_B</td>
<td>01</td>
<td>0010</td>
</tr>
<tr>
<td>State_C</td>
<td>10</td>
<td>0100</td>
</tr>
<tr>
<td>State_D</td>
<td>11</td>
<td>1000</td>
</tr>
</tbody>
</table>

In either encoding scheme, there will be codes that do not correspond to any FSM state.
- E.g. “000110” is not a valid one-hot encoding.
- E.g. “111” may not be valid if there are only 5 states in the FSM.

How to handle invalid states?
- Ignore them – since in theory your state machine will never end up in that state.
- Flag an error, stop the machine.
- Reset the machine/Jump to a default state.
- Take advantage of them to simplify the design of next state/output logic.

CIRCUIT TIMING
Timing of Circuits

- So far, we have assumed:
  - Output of a combinational circuit changes instantaneously w.r.t. input
  - Output of a FF changes instantaneously w.r.t. clock edge
- In reality, it takes finite amount of time for a signal to travel through a circuit.
- The timing of different parts of a circuit
  - May cause glitches in output,
  - Limit the maximum speed of a design

Propagation Delay

- Each logic gate incurs delay between the input and output to allow signal to propagate
  - Such delay is referred as propagation delay
- Exact value is technology-dependent
- In this class, we assume all gates have the same unit propagation delay.
  - The speed of a circuit is always limited by the slowest path

Glitches

- Glitches refer to any momentarily change in a signal value
- One common cause is due to race condition between signal paths
  - Imbalanced propagation delay
- May cause incorrect output if a signal is used during the glitch

Timing in Synchronous Circuits

- In general, the propagation delay through the combinational logic between any two registers must be shorter than the clock period
- The longest such path is called the critical path of the circuit
- The critical path determines the maximum clock speed

Timing in Synchronous Circuits Tradeoff

- In a synchronous sequential circuit, signal changes occur only during clock edge
- All signals are therefore synchronized to change values right after a clock edge
- In the above example, need to make sure correct value of y available BEFORE next clock edge

Synchronous Circuits Timing Tradeoff

- Since only values right before the clock edge in the input port are captured, all the glitches within the circuit are ignored
- A short period of time before a clock edge must be allocated to ensure stable inputs
  - Too small \(\rightarrow\) Chance of failing circuit
  - Too big \(\rightarrow\) Wasted idle time
- Since all circuit runs on the same clock, clock frequency limited by the longest critical path
Summary

- Synchronous sequential circuits contain one single clock
  - All FFs connected to the same clock
  - All signal change synchronized to the same clock edge
- State machine is an important abstraction for computation
  - Straight forward implementation as synchronous sequential logic
- Propagation delay of logic gates determines the highest frequency a synchronous sequential circuit can run