Optimizations and Tradeoffs

Sequential Logic Optimization, Datapath Component Tradeoffs
State Reduction/Minimization

- A process of reducing the number of FSM states without changing behavior.

- Equivalent States can be removed
- States are equivalent if
  1. Both states assign the same values to outputs
  2. For all possible sequences of inputs, FSM outputs will be the same starting from either state.

- To remove equivalent states:
  - Remove one of the states from the FSM
  - Transfer transitions pointing to the removed state to the other state.

- Visual Inspection of large FSMs is not feasible
Partitioning Method

- The partitioning method can be used to detect groups of equivalent states.
  - Method contains two set of groups
    1. Groups cannot be equivalent.
    2. Groups might be equivalent.
Partitioning Method Process

1. Partition states into groups based on the values they assign to outputs.
   - States assigning the same values are grouped together.
2. List next state values for each state in a group for all input values.
3. Compare states in the group with the same input values
   1. If for the same input value, two states transition to states in different groups, they cannot be equivalent.
   2. Partition states that are not equivalent into sub groups and repeat.
Partitioning Method Example

- **G1 = {A,D}**
  - **x = 0**
    - A goes to A (G1)
    - D goes to D (G1)
  - **x = 1**
    - A goes to B (G2)
    - D goes to B (G2)

- **G2 = {B,C}**
  - **x = 0**
    - B goes to D (G1)
    - C goes to B (G2)
    - **Different!**
  - **x = 1**
    - B goes to C (G2)
    - C goes to B (G2)
Partitioning Method Example

Repartition

- **G1 = \{A,D\}**
  - x = 0
    - A goes to A (G1)
    - D goes to D (G1)
  - x = 1
    - A goes to B (G2)
    - D goes to B (G2)

- **G2 = B**
  - Only one state

- **G3 = C**
  - Only one state
Partitioning Method Example

- A & D are equivalent
- Rework the FSM
Partition Example 2
State Encoding Revisited

- **Review:**
  - State Encoding involves choosing the unique binary representation for a state.
  - The choice of encoding can effect the size and performance of a circuit.

- Previously, we have covered general guidelines for state encoding, but there are several standard encodings that may be useful.
Standard Encodings

*Minimum-bitwidth binary encoding*

- What we have done so far
- Utilizes the minimum number of bits
- Many combinations of encodings are possible
- $2^N$ states means $N$ bits in the encoding

- Example: List two different ways of encoding a state machine with six states, $s_0$ through $s_5$:
  1. Using 3 bits, each state’s number if encoded in binary
     1. Means that $b'110$ and $b'111$ are unused
  2. Use the inversion of the previous mapping
     1. Now 001 and 000 are unused

Both of these are equally valid, though maybe not optimal.
Standard Encodings

One-Hot Encoding

- A different bit is used to represent each state
- N states means N bits.
- Tradeoff: Speed vs. size
  - One-hot is fast because a state can be detected from just one bit
- Useful for generating outputs based off state

Example: Create a One-Hot encoding for a state machine with six states, s0 through s5:

Use 6 bits, where the weight of the bit is equivalent to the state number:

- s0 \(\rightarrow\) 000001
- s1 \(\rightarrow\) 000010
- s2 \(\rightarrow\) 000100
- ...
- s5 \(\rightarrow\) 100000
Standard Encodings

Output Encoding
- Uses the output values of a state as it’s encoding
  - Reduces output combinational logic
  - Each bit in the encoding is a decomposition of desired output.

- Requires two conditions
  - FSM has at least as many outputs as needed for a binary encoding of states
    - Otherwise, “dummy” outputs may be created
  - The output values assigned by each state are unique

- Example: Create an output encoding for the following FSM
Types of FSMs

- So far we have only considered one type of FSM, but there is a second.

- Moore FSM
  - What we have studied so far
  - Outputs are only dependant on the current state

- Mealy FSM
  - Outputs are dependant on the current state AND INPUTS!
  - Allows events to be detected one clock cycle sooner.
  - May result in fewer states.
Mealy Standard Controller Architecture

- Note that the inputs now also go to the output logic block.
Differences in Drawing Mealy State Diagrams

**Moore**
- Outputs are defined in the state
  - Effective during the entire time spent in the state.

**Mealy**
- Outputs are defined on transitions
  - Effective as soon as the transition is detected.
  - Can have fewer states due to this.
  - Outputs may be glitchy!
Mealy FSM Example

Revisit the sequential bit sequence detector design using a Mealy FSM:
1 – 0 – 1 – 1 with overlap

- Overlap indicates that a sequence can be detected within another valid sequence.
- The input X changes on each clock tick.
- The output F is 1 the cycle after the sequence is detected, 0 otherwise.

clk:

X:

F:

F:

Moore

Mealy
Mealy FSM Example

The Mealy version requires one fewer state than the Moore version:
Mealy FSM Example

Let’s use the following encoding so we can generate a truth table

<table>
<thead>
<tr>
<th>State</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
</tr>
<tr>
<td>1-0</td>
<td>10</td>
</tr>
<tr>
<td>1-0-1</td>
<td>11</td>
</tr>
</tbody>
</table>
Mealy FSM Example

- Truth Table

<table>
<thead>
<tr>
<th>S1</th>
<th>S0</th>
<th>X</th>
<th>N1</th>
<th>N0</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<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>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Timing Issues with Mealy FSMs

- Because outputs change as inputs change, they are no longer synchronous with clock edges.
  - In our example, the value F can change at any time while we are in state 1-0-1 if X changes.
  - This may cause glitches!
- To be safe, you may want to consider using registered outputs for Mealy outputs.

Example: Draw the output for the following timing input:

```
clk    state   X      F
-------+---------+--------+
  | 1-0    | 1-0-1   | X      |
  | 1-0-1  | 1-0     | X      |
  | 1-0    | 0       | X      |
  | 0      | 1       | X      |
```
Combining Moore & Mealy FSMs

- Mealy and Moore types can be merged into a single FSM
  - Allows some actions to be associated with states
  - Others are associated with transitions