Lab 2: Wristwatch Example

Digital Design Lab
Overview

- Lab 2 methodology
- Wristwatch example
  - Lab procedure walkthrough
Methodology

1. Understand the Specification
2. Create the FSM
3. State Minimization
4. Write 2-Process Behavioral Model
5. Write the Testbench
6. Simulate and Verify in ISE Tool
7. State Encoding
8. Develop Minimal Boolean Equation for the Combinatorial Logic
9. Optimize Boolean Equations for the Given Gate Library and Estimate Timing
10. Estimate Timing for Combinatorial Part
11. Write Structural Model
12. Simulate with the Testbench
13. Report Cost and Performance Numbers
1. Understand the Specification

- Wristwatch Display

- A wristwatch can display one of the three items: the time, the alarm and the stopwatch. The watch has a menu that user can choose between these three items by pressing a button.
- There are three buttons available for each item on the menu and by pressing each button, it will display the requested item on the screen.
- The user can go back to menu from each item by pressing the back button.
Wristwatch

- There are four buttons in the watch:
  - time, alarm, stopwatch, and back
- We need a two-bit input to distinguish the pressed button.
  - Input: Button (B₁, B₀)

<table>
<thead>
<tr>
<th>Buttons</th>
<th>Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>00</td>
</tr>
<tr>
<td>Alarm</td>
<td>01</td>
</tr>
<tr>
<td>Stopwatch</td>
<td>10</td>
</tr>
<tr>
<td>Back</td>
<td>11</td>
</tr>
</tbody>
</table>

- Also there are 4 different view on screen for:
  - menu, time, alarm, and stopwatch
- Therefore output needs two bit to describe the screen mode.
  - Outputs: Mode (M₁, M₀)

<table>
<thead>
<tr>
<th>Screen Mode</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>01</td>
</tr>
<tr>
<td>Alarm</td>
<td>10</td>
</tr>
<tr>
<td>Stopwatch</td>
<td>11</td>
</tr>
<tr>
<td>Menu</td>
<td>00</td>
</tr>
</tbody>
</table>
2. Create the FSM

![Finite State Machine Diagram]

- **Start**
  - Mode = Menu
  - Button = Back
- **Time**
  - Mode = Time
  - Button = Time
  - Button = Time, Alarm, Stopwatch
- **Alarm**
  - Mode = Alarm
  - Button = Alarm
  - Button = Time, Alarm, Stopwatch
- **Stopwatch**
  - Mode = Stopwatch
  - Button = Time, Alarm, Stopwatch
  - Button = Back
- **Menu**
  - Mode = Menu
  - Button = Back

Button = Back
2. Create the FSM

- **Start** Mode = 00
- **Time** Mode = 01
- **Alarm** Mode = 10
- **Stopwatch** Mode = 11
- **Menu** Mode = 00

### Buttons

<table>
<thead>
<tr>
<th>Buttons</th>
<th>Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1 B0</td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>00</td>
</tr>
<tr>
<td>Alarm</td>
<td>01</td>
</tr>
<tr>
<td>Stopwatch</td>
<td>10</td>
</tr>
<tr>
<td>Back</td>
<td>11</td>
</tr>
</tbody>
</table>

### Screen Mode

<table>
<thead>
<tr>
<th>Screen Mode</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1 M0</td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>01</td>
</tr>
<tr>
<td>Alarm</td>
<td>10</td>
</tr>
<tr>
<td>Stopwatch</td>
<td>11</td>
</tr>
<tr>
<td>Menu</td>
<td>00</td>
</tr>
</tbody>
</table>
3. State Minimization

**Condition 1:** Both states produce the same output for every input

- **State** | **Output**
- Start     | 00  
- Time     | 01  
- Alarm    | 10  
- Stopwatch | 11  
- Menu     | 00  

**Condition 2:** Both states have equivalent next states for every input

<table>
<thead>
<tr>
<th>State</th>
<th>$B_1 B_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>01 10 11</td>
</tr>
</tbody>
</table>

- Start     | Time  | Alarm  | Stopwatch | Start  
- Menu     | Time  | Alarm  | Stopwatch | Menu   

Button = 00
Button = 01
Button = 10
Button = 11
Button = 0X, 10

Copyright © 2004-2012 by Daniel D. Gajski
4. Write 2-Process Behavioral Model

```vhdl
entity Wristwatch_B is
  Port ( clk : in STD_LOGIC;
         Button : in STD_LOGIC_VECTOR (1 downto 0);
         Mode : out STD_LOGIC_VECTOR (1 downto 0));
end Wristwatch_B;
architecture Behavioral of Wristwatch_B is

TYPE StateType IS
  (Menu_State, Time_State, Alarm_State, Stopwatch_State);

  Signal Current_State, Next_State: StateType;
begin
  CombLogic: Process (Button, Current_State)
  begin
    -- Combinational Logic Process
  end Process CombLogic;

  StateRegister: Process (clk)
  begin
    -- State Register Process
  end Process StateRegister;
end Behavioral;
```
Combilogic: Process (Button, Current_State) begin
 case Current_State IS
   When Menu_State =>
     Mode <= "00";
     if (Button = "00") then
       Next_State <= Time_State;
     elsif (Button = "01") then
       Next_State <= Alarm_State;
     elsif (Button = "10") then
       Next_State <= Stopwatch_State;
     else
       Next_State <= Menu_State;
     end if;
   When Time_State =>
     Mode <= "01";
     if (Button = "11") then
       Next_State <= Menu_State;
     else
       Next_State <= Time_State;
     end if;
   When Alarm_State =>
     Mode <= "10";
     if (Button = "11") then
       Next_State <= Menu_State;
     else
       Next_State <= Alarm_State;
     end if;
   When Stopwatch_State =>
     Mode <= "11";
     if (Button = "11") then
       Next_State <= Menu_State;
     else
       Next_State <= Stopwatch_State;
     end if;
 end case;
end Process Combilogic;

StateRegister: Process (clk) begin
  if(clk = '1' AND clk'EVENT) then
    Current_State <= Next_State;
  end if;
end Process StateRegister;
5. Write the Testbench

- Generate a new source in the project as a VHDL testbench

```vhdl
-- Stimulus process
stim_proc: process
begin
  -- insert stimulus here
  Button <= "10";
  WAIT UNTIL clk = '1' AND clk'EVENT;
  Wait for 15 NS;
  ASSERT ( Mode = "11") REPORT "Failed Menu_State --> Stopwatch_State" SEVERITY Warning;

  Button <= "11";
  WAIT UNTIL clk = '1' AND clk'EVENT;
  Wait for 15 NS;
  ASSERT ( Mode = "00") REPORT "Failed Menu_State --> Menu_State" SEVERITY Warning;

  Button <= "00";
  WAIT UNTIL clk = '1' AND clk'EVENT;
  Wait for 15 NS;
  ASSERT ( Mode = "01") REPORT "Failed Menu_State --> Time_State" SEVERITY Warning;

  wait;
end process;
```
6. Simulate and Verify in ISE Tool
7. State Encoding

<table>
<thead>
<tr>
<th>State</th>
<th>Encoding A</th>
<th>Encoding B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Menu</td>
<td>00</td>
<td>11</td>
</tr>
<tr>
<td>Time</td>
<td>01</td>
<td>00</td>
</tr>
<tr>
<td>Alarm</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Stopwatch</td>
<td>11</td>
<td>01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State Encoding A</th>
<th>Input</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>( B_1 B_0 )</td>
<td>( \text{NS}_1 ) ( \text{NS}_0 )</td>
<td>( M_1 M_0 )</td>
<td></td>
</tr>
<tr>
<td>Menu : 00</td>
<td>0 0</td>
<td>0 1</td>
<td>0 0</td>
</tr>
<tr>
<td></td>
<td>0 1</td>
<td>1 0</td>
<td>0 0</td>
</tr>
<tr>
<td></td>
<td>1 0</td>
<td>1 1</td>
<td>0 0</td>
</tr>
<tr>
<td></td>
<td>1 1</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>Time : 01</td>
<td>0 X</td>
<td>0 1</td>
<td>0 1</td>
</tr>
<tr>
<td></td>
<td>1 0</td>
<td>0 1</td>
<td>0 1</td>
</tr>
<tr>
<td></td>
<td>1 1</td>
<td>0 0</td>
<td>0 1</td>
</tr>
<tr>
<td>Alarm: 10</td>
<td>0 X</td>
<td>1 0</td>
<td>1 0</td>
</tr>
<tr>
<td></td>
<td>1 0</td>
<td>1 0</td>
<td>1 0</td>
</tr>
<tr>
<td></td>
<td>1 1</td>
<td>0 0</td>
<td>1 0</td>
</tr>
<tr>
<td>Stopwatch: 11</td>
<td>0 X</td>
<td>1 1</td>
<td>1 1</td>
</tr>
<tr>
<td></td>
<td>1 0</td>
<td>1 1</td>
<td>1 1</td>
</tr>
<tr>
<td></td>
<td>1 1</td>
<td>0 0</td>
<td>1 1</td>
</tr>
</tbody>
</table>
8. Develop Minimal Boolean Equation for the Combinatorial Logic

<table>
<thead>
<tr>
<th>State Encoding A</th>
<th>Input</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B₁ B₀</td>
<td>NS₁ NS₀</td>
<td>M₁ M₀</td>
</tr>
<tr>
<td>Menu : 00</td>
<td>0 0</td>
<td>0 1</td>
<td>0 0</td>
</tr>
<tr>
<td></td>
<td>0 1</td>
<td>1 0</td>
<td>0 0</td>
</tr>
<tr>
<td></td>
<td>1 0</td>
<td>1 1</td>
<td>0 0</td>
</tr>
<tr>
<td></td>
<td>1 1</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>Time : 01</td>
<td>0 X</td>
<td>0 1</td>
<td>0 1</td>
</tr>
<tr>
<td></td>
<td>1 0</td>
<td>0 1</td>
<td>0 1</td>
</tr>
<tr>
<td></td>
<td>1 1</td>
<td>0 0</td>
<td>0 1</td>
</tr>
<tr>
<td>Alarm : 10</td>
<td>0 X</td>
<td>1 0</td>
<td>1 0</td>
</tr>
<tr>
<td></td>
<td>1 0</td>
<td>1 0</td>
<td>1 0</td>
</tr>
<tr>
<td></td>
<td>1 1</td>
<td>0 0</td>
<td>1 0</td>
</tr>
<tr>
<td>Stopwatch : 11</td>
<td>0 X</td>
<td>1 1</td>
<td>1 1</td>
</tr>
<tr>
<td></td>
<td>1 0</td>
<td>1 1</td>
<td>1 1</td>
</tr>
<tr>
<td></td>
<td>1 1</td>
<td>0 0</td>
<td>1 1</td>
</tr>
</tbody>
</table>

NS₁ = CS₁ B₁' + CS₁ B₀' + CS₀ B₁' B₀ + CS₀ B₁ B₀'

NS₀ = CS₀ B₁' + CS₀ B₀' + CS₁ B₀'

M₁ = CS₁
M₀ = CS₀
9. Optimize Boolean Expressions

\[ NS_1 = (CS_1 B_1' + CS_1 B_0' + CS_0' B_1' B_0) + CS_0' B_1 B_0' \]
\[ (1 \text{ ns} + 2.8 \text{ ns} + 2.8 \text{ ns}) + 2.8 \text{ ns} = 9.4 \text{ ns} \]

\[ NS_0 = (CS_0' B_1' + CS_0 B_0') + CS_1' B_0' \]
\[ (1 \text{ ns} + 2.8 \text{ ns}) + 2.8 \text{ ns} = 6.6 \text{ ns} \]

\[ M_1 = CS_1 \]
\[ M_0 = CS_0 \]
10. Estimate Timing

Total Delay = 1ns + 4ns + 9.4 ns = 14.4 ns
entity Wristwatch_S is
  Port ( clk : in   STD_LOGIC;
       Button : in   STD_LOGIC_VECTOR (1 downto 0);
       Mode : out  STD_LOGIC_VECTOR (1 downto 0));
end Wristwatch_S;

architecture Behavioral of Wristwatch_S is

SUBTYPE StateType is STD_LOGIC_VECTOR (1 downto 0);
SIGNAL Current_State, Next_State : StateType := "00";

begin

  CombLogic: Process (Current_state, Button)
  begin
    -- Combination Logic Process
  end process CombLogic;

  StateRegister: Process (clk)
  begin
    -- State Register Process
  end Process StateRegister;

end Behavioral;

11. Write Structural Model

Combinational Logic

Current State

State Register

Next state

FSM Mode

Mode

FSM input

Button

Clk
11. Write Structural Model (cont...)

CombLogic: Process (Current_state, Button)
begin
   -- Combination Logic Process

   Next_State(1) <= (Current_state(1) AND (NOT Button(1))) OR
                     (Current_state(1) AND (NOT Button(0))) OR
                     ((NOT Current_state(0)) AND (NOT Button(1)) AND Button(0)) OR
                     ((NOT Current_state(0)) AND Button(1) AND (NOT Button(0))) after 9.4 ns;

   Next_State(0) <= ((NOT Current_state(0)) AND (NOT Button(1))) OR
                     (Current_state(0) AND (NOT Button(0))) OR
                     ((NOT Current_state(1)) AND (NOT Button(0))) after 6.6 ns;

   Mode(1) <= Current_state(1);
   Mode(0) <= Current_state(0);
end process CombLogic;

StateRegister: Process (clk)
begin
   if(clk = '1' AND clk'EVENT ) then
      Current_State <= Next_State after 4 ns;
   end if;
end Process StateRegister;
12. Simulate with the Testbench

- Generate a new source in the project as a vhdl Testbench

```vhdl
-- Stimulus process
stim_proc: process
begin
  -- insert stimulus here
  Button <= "10";
  WAIT UNTIL clk = '1' AND clk'EVENT;
  Wait for 15 NS ;
  ASSERT ( Mode = "11") REPORT "Failed Menu_State -> Stopwatch_State" SEVERITY Warning;

  Button <= "11";
  WAIT UNTIL clk = '1' AND clk'EVENT;
  Wait for 15 NS ;
  ASSERT ( Mode = "00") REPORT "Failed Menu_State -> Menu_State" SEVERITY Warning;

  Button <= "00";
  WAIT UNTIL clk = '1' AND clk'EVENT;
  Wait for 15 NS ;
  ASSERT ( Mode = "01") REPORT "Failed Menu_State -> Time_State" SEVERITY Warning;

  wait;
end process;
```
# 13. Cost and Performance Numbers

<table>
<thead>
<tr>
<th>State</th>
<th>Encoding A</th>
<th>Encoding B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>122</td>
<td>132</td>
</tr>
<tr>
<td>Delay</td>
<td>9.4 ns</td>
<td>6.6 ns</td>
</tr>
</tbody>
</table>
## State Encoding B

<table>
<thead>
<tr>
<th>State</th>
<th>Encoding A</th>
<th>Encoding B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Menu</td>
<td>00</td>
<td>11</td>
</tr>
<tr>
<td>Time</td>
<td>01</td>
<td>00</td>
</tr>
<tr>
<td>Alarm</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Stopwatch</td>
<td>11</td>
<td>01</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
NS_1 &= CS_1CS_0' + CS_1B_0 + B_1B_0 \\
NS_0 &= CS_1'CS_0 + CS_0B_1 + B_1B_0 \\
M_1 &= CS_1'CS_0 + CS_1CS_0' \\
M_0 &= CS_1'
\end{align*}
\]

**Max Delay = 6.6 ns**
Summary

- Lab 2 methodology

- Wristwatch example
  - Lab procedure walkthrough