Controller Design

ECE 152A – Fall 2006

Coke® Machine

- This example illustrates the design a controller for a Coke® machine
  - The machine accepts only nickels and dimes, and a Coke® costs 15 cents
    - …this is a very old Coke® machine

Coke® Machine

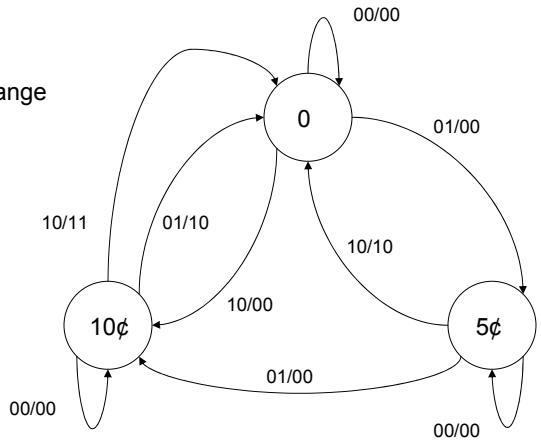
- There are three inputs to the controller
  - clk
  - dime
    - when = 1, indicates that a dime has been inserted
  - nickel
    - when = 1, indicates that a nickel has been inserted
  - Assume that dime and nickel will never be 1 at the same time

Coke® Machine

- The controller has 2 outputs
  - dispense
    - which when = 1, dispenses the Coke®
  - change
    - which when = 1, returns a nickel in change
- Design the controller as a Mealy machine
  - Implement the design in Verilog

Coke® Machine

- State diagram
  - Input = Dime Nickel
  - Output = Coke® Change
  - Dime = Nickel = 1
    → invalid input
- Three Nickels
- Dime Dime
- Nickel Dime



PLD Implementation

- Logic

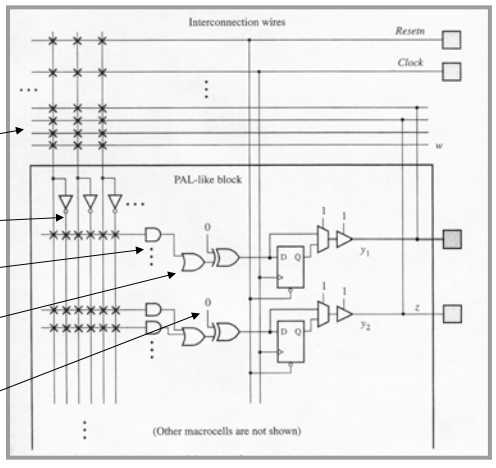
All variables

Complements

AND Level

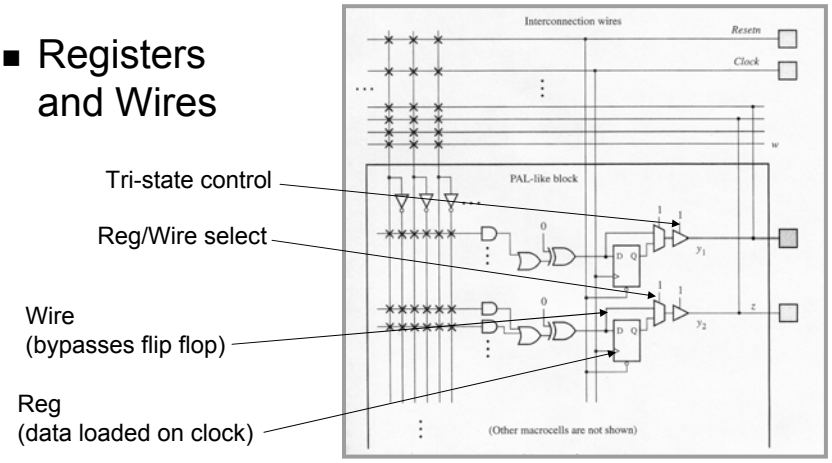
OR Level

Selectable inversion



PLD Implementation

- Registers and Wires



Continuous Assignment of Output

- Recall Mealy and Moore implementations of 101 sequence detector output
  - Moore Machine
    - assign z = state [1] & state [0]
  - Mealy Machine
    - assign z = x & state [1]
  - Both outputs are "wires"
    - Output assigned to "continuously"
    - Output changes when any input changes
  - What if output assigned to procedurally?

Coke® Machine

Wire inputs
Reg outputs

Outputs assigned to procedurally
(and synchronously)

```
module coke (clk, dime, nickel, dispense, change);

input clk, dime, nickel;
output dispense, change;

reg dispense, change;
reg [1:0] state;

parameter [1:0] idle = 2'b00, five_cents = 2'b01, ten_cents = 2'b10;

always @ (posedge clk)

    case (state)

        idle :  begin
                dispense <= 0;
                change <= 0;
                if (dime == 0 && nickel == 1)
                state <= five_cents;
                else if (dime ==1 && nickel == 0)
                state <= ten_cents;
                else
                state <= idle;
                end

        five_cents : if (dime == 0 && nickel == 1)
                     state <= ten_cents;
                     else if (dime ==1 && nickel == 0)
                     begin
                     state <= idle;
                     dispense <= 1;
                     end
                     else
                     state <= five_cents;

        ten_cents : if (dime == 0 && nickel == 1)
                    begin
                    state <= idle;
                    dispense <= 1;
                    end
                    else if (dime ==1 && nickel == 0)
                    begin
                    state <= idle;
                    dispense <= 1;
                    change <= 1;
                    end
                    else
                    state <= ten_cents;
        default : state <= idle;
    endcase

endmodule
```

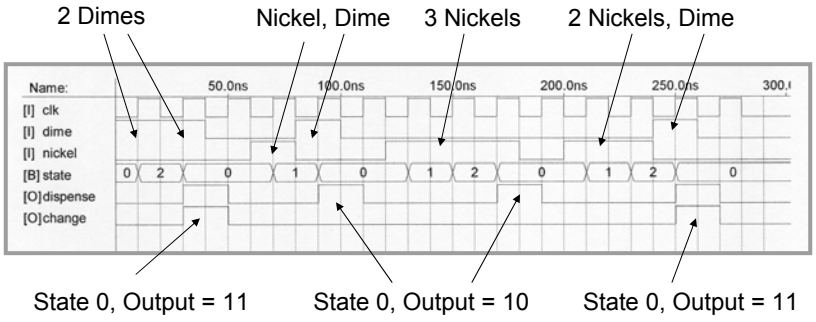
- Verilog code
  - Procedural vs. Continuous Assignment

Coke® Machine

- Timing diagram verifying correct operation of the controller
  - The timing diagram verifies the following input sequences
    - 2 dimes → Coke® and change
    - 1 nickel, 1 dime → Coke®
    - 3 nickels → Coke®
    - 2 nickels, 1 dime → Coke® and change

Coke® Machine

- Timing Diagram (Functional Simulation)
  - Note “pseudo Mealy Machine” timing



CD Player Controller

- This example illustrates the design a controller for a portable CD player
  - The CD player has only two control buttons
    - Play and Stop
  - When a CD is inserted, the controller automatically goes to a reset state and the laser is positioned at the beginning of the first track.
- The CD player then operates as follows:

CD Player Controller

- Play Button
  - Press Play once (while stopped)
    - Begin playing current track at current position
  - Press Play again
    - Advance to beginning of next track and play
  - Press Play again
    - As above, advance to beginning of next track and play

CD Player Controller

- Stop Button
  - Press Stop (while track is playing)
    - Stop at current position
  - Press Stop again
    - Return to beginning of first track

CD Player Controller

- Stop and Play Buttons Together
  - Press Stop and Play at the same time
    - Return to beginning of previous track and play
  - Press Stop and Play at the same time again
    - As above, return to beginning of previous track and play

CD Player Controller

- There are four outputs from the controller
  - Play
  - Forward (find the next track)
  - Reverse (find previous track)
  - Zero (return to the beginning of track 1)
  - These outputs should be given the variable names P F R and Z, respectively.

CD Player Controller

- Design the controller as a Moore machine
  - You can assume that the play and forward (as well as the play and reverse) outputs can be active at the same time
  - Another piece of circuitry (of no concern to you) insures that playing doesn't begin until the laser is in the correct position
- Convert the Moore machine to the equivalent Mealy machine

CD Player Controller

- The construction of the state diagram is critical in this problem
  - Clearly define (through comments) your states and the transitions between states
  - Include the following
    - A state diagram (both Moore and Mealy)
    - A state table (both Moore and Mealy)
    - Next state maps for Mealy machine
    - Implementation with D flip-flops
      - Flip-flop input equations only (no schematic)

CD Player Controller

- P F R Z
- 0 0 0 0
  - Stopped
- 0 0 0 1
  - Stopped, reposition laser
- 0 0 1 0
  - X → reverse causes play
- 0 0 1 1
  - X → reverse and zero
- 0 1 0 0
  - X → forward causes play
- P F R Z
- 0 1 0 1
  - X → forward and zero
- 0 1 1 0
  - X → forward and reverse
- 0 1 1 1
  - X → forward, reverse and zero
- 1 0 0 0
  - Play
- 1 0 0 1
  - X → play and zero

CD Player Controller

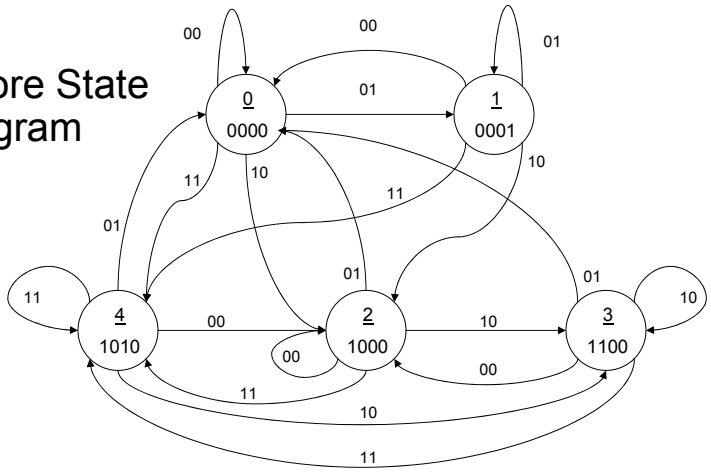
- P F R Z
- 1 0 1 0
  - Reverse and play
- 1 0 1 1
  - X → play, reverse and zero
- 1 1 0 0
  - Forward and play
- 1 1 0 1
  - X → play, forward and zero
- P F R Z
- 1 1 1 0
  - X → play, forward and reverse
- 1 1 1 1
  - X → play, forward, reverse and zero

CD Player Controller

- Five valid outputs
  - Construct Moore machine state diagram with five valid outputs (states)
    - Stopped = 0 (PFRZ = 0000)
    - Zero = 1 (PFRZ = 0001)
    - Play = 2 (PFRZ = 1000)
    - Forward/Play = 3 (PFRZ = 1100)
    - Reverse/Play = 4 (PFRZ = 1010)

CD Player Controller

- Moore State Diagram



CD Player Controller

- State Table (output state encoding)

		NS	(input = play stop)		
PS	00	01	10	11	PFRZ
0000	0000	0001	1000	1010	0000
0001	0000	0001	1000	1010	0001
1000	1000	0000	1100	1010	1000
1100	1000	0000	1100	1010	1100
1010	1000	0000	1100	1010	1010

CD Player Controller

- State Table (no secondary state assignment)

	NS				
PS	00	01	10	11	PFRZ
0	0	1	2	4	0000
1	0	1	2	4	0001
2	2	0	3	4	1000
3	2	0	3	4	1100
4	2	0	3	4	1010

CD Player Controller

- Mealy Conversion (direct)

	NS,PFRZ			
PS	00	01	10	11
0	0,0000	1,0001	2,1000	4,1010
1	0,0000	1,0001	2,1000	4,1010
2	2,1000	0,0000	3,1100	4,1010
3	2,1000	0,0000	3,1100	4,1010
4	2,1000	0,0000	3,1100	4,1010

CD Player Controller

- Note that rows 0 and 1 are identical and rows 2, 3 and 4 are identical
  - Identical rows can be combined into a single state
    - "Row matching" for state machine reduction

	NS, PFRZ			
PS	00	01	10	11
0	0,0000	1,0001	2,1000	4,1010
1	0,0000	1,0001	2,1000	4,1010
2	2,1000	0,0000	3,1100	4,1010
3	2,1000	0,0000	3,1100	4,1010
4	2,1000	0,0000	3,1100	4,1010

Identical rows indicate redundant (or same) state

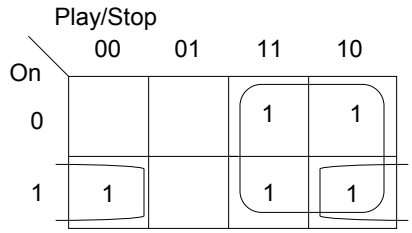
CD Player Controller

- Reduced Mealy Machine

	NS,PFRZ			
PS	00	01	10	11
0	0,0000	0,0001	1,1000	1,1010
1	1,1000	0,0000	1,1100	1,1010

CD Player Controller

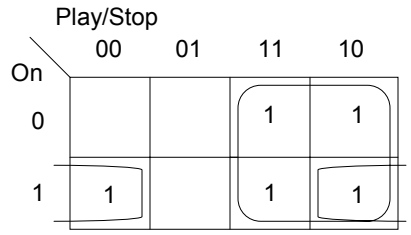
- Next State Map
  - Single state variable is labeled “On”



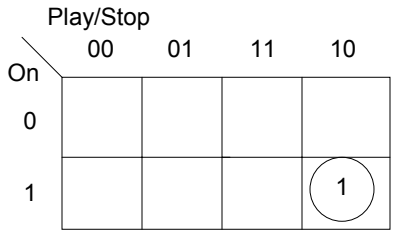
$On^{+} = Play + (On \cdot Stop')$

CD Player Controller

- Output Maps (Play & Forward)



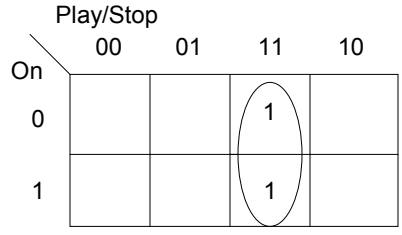
Play = Play + (On · Stop')



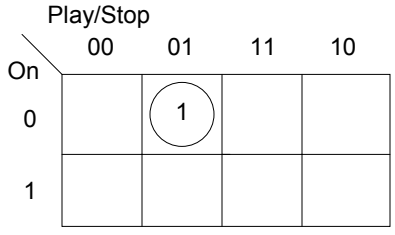
Forward = On · Play · Stop'

CD Player Controller

- Output Maps (Reverse & Zero)



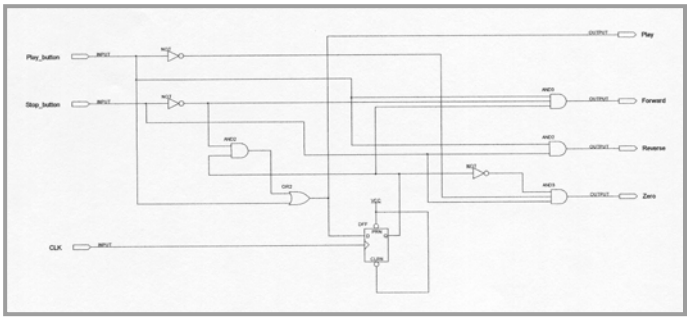
Reverse = Play · Stop



Zero = On' · Play' · Stop

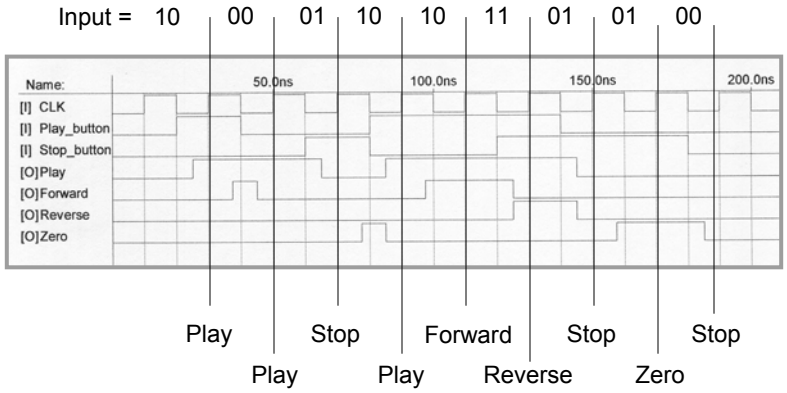
CD Player Controller

- Schematic



CD Player Controller

- Timing Simulation



Coin Operated Car Wash

- This problem illustrates the design of a controller for a coin operated car wash
  - Based on *Danny's Deli/Car Wash/Bait and Tackle*, Carpinteria, CA
- The controller has a two-bit input
  - COIN1 and COIN0
    - The number of quarters (0, 1, 2 or 3) currently deposited in the coin box

Coin Operated Car Wash

- The controller has five bits of output
  - TIME1 and TIME0
    - The number of minutes of operation remaining
  - ALARM
    - Only one minute of operation remains
  - WATER
    - Turns on the water
  - ACTIVE
    - The car wash is in the active mode

Coin Operated Car Wash

- Design Specification
  - The coin box accepts only quarters and holds a maximum of three quarters
    - The COIN1 and COIN0 inputs to the controller
    - Any quarters added to a full coin box will fall into the coin return slot

Coin Operated Car Wash

- Design Specification (cont)
  - Each quarter gives one minute of car wash operation
    - The controller will allow a maximum operating time of 3 minutes
    - Additional quarters can be added while the car wash is ACTIVE and time will be added
    - Controller outputs TIME1 and TIME0 indicate the remaining time

Coin Operated Car Wash

- Design Specification (cont)
  - The car wash doesn't begin operation until it finds 3 quarters in the coin box
    - When COIN1, COIN0 = 11 is detected, the controller becomes ACTIVE and the WATER is turned on
  - Once operation begins, the controller checks the contents of the coin box once per minute
    - Based on the contents of the coin box, the controller determines the remaining minutes of operation
      - TIME1 and TIME0 are adjusted accordingly

Coin Operated Car Wash

- Design Specification (cont)
  - If the contents of the coin box would cause the number of remaining minutes to exceed the maximum of 3, the remaining time should be set to 3 minutes
    - The excess quarters will be held in the coin box
      - The number of coins in the coin box will automatically be reduced by the number necessary to set the remaining time to 3 minutes
        - The COIN1, COIN0 inputs will be correct by construction

Coin Operated Car Wash

- Design Specification (cont)
  - When there is one minute left on the timer, the ALARM signal is asserted, advising the user that more quarters must be added for additional time
    - If the timer times out (TIME1, TIME0 = 00) and no quarters have been added, the ALARM will cease and the WATER will be turned off but the controller will remain ACTIVE for one more minute

Coin Operated Car Wash

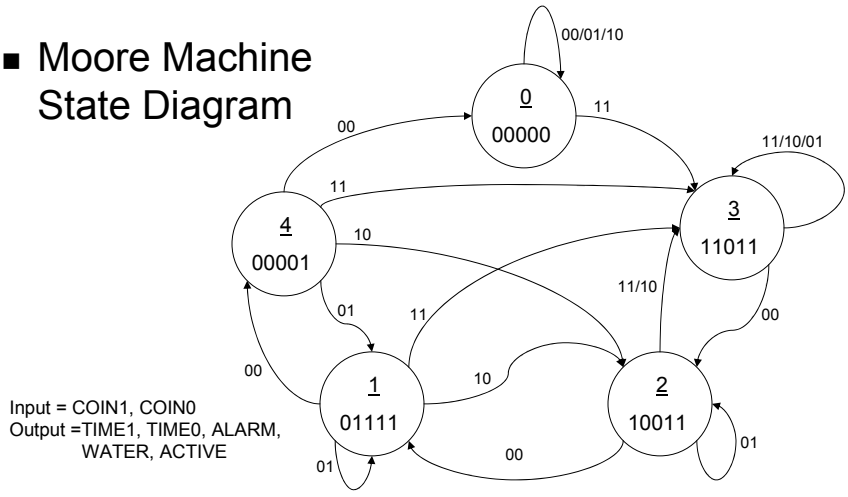
- Design Specification (cont)
  - If any quarters are added during that minute, additional time will be added.
    - Unlike the initial state which requires three quarters to commence operation, any number of quarters added during this one minute period will add the appropriate number of minutes.
      - This gives the user a chance to add time if he was unable to get to the controller during the last minute of operation.
    - If no quarters are added during this minute, the controller becomes inACTIVE (ACTIVE = 0)

Coin Operated Car Wash

- Design the controller as a Moore machine.
  - You only need to construct a state diagram, but make sure that the operation of the controller is clear
- Design Approach
  - Unlike the CD player controller, this controller (once activated) operates on a real time clock
    - The state diagram should follow the passage of time, one minute at a time

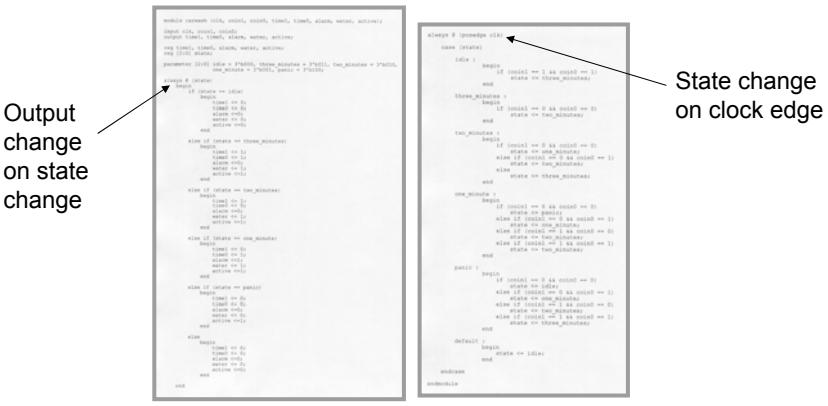
Coin Operated Car Wash

- Moore Machine State Diagram



Coin Operated Car Wash

- Verilog Implementation



Coin Operated Car Wash

- Timing Diagram (timing simulation)
  - Note Moore machine timing