Logic and Computer Design Fundamentals

Chapter 5 – Sequential Circuits

Part 3 – State Machine Design

Charles Kime

(Hyperlinks are active in View Show mode)

Overview

- **Part 1 - Storage Elements and Analysis**
- **Part 2 - Sequential Circuit Design**
- **Part 3 – State Machine Design**
  - **Issues with traditional state diagrams and table representations**
  - **The state machine diagram model**
  - **Constraint checking**
  - **State machine diagram application and design**

Finite State Machines

- A finite state machine (FSM) consists of three sets I, O, and S and two functions f and g in which:
  - I is a set of input combinations,
  - O is a set of output combinations,
  - S is a set of states
  - f is the next state function f(I, S), and
  - g is the output function f(S) [Moore model] or the output function f(I, S) [Mealy model].
- The FSM is a fundamental mathematical model used for sequential circuits.
- The details of the traditional state diagrams and state tables as we have defined them are just two of many ways of representing FSMs.

Issues with Traditional State Diagram and Table Representations

- Both of these traditional representations require:
  - Enumeration of all input combinations for each state in defining next states
  - Enumeration of all input combinations for each state in defining Mealy outputs
  - Enumeration of all applicable output combinations for each state (Moore) and for each input combination-state pair (Mealy).
- For state diagrams, all Mealy outputs must be specified on transition arcs

Issues with Traditional State Diagram and Table Representations

- These requirements may be acceptable for sequential circuits with relatively few inputs, and outputs.
- For larger numbers of inputs and outputs both representations become intractable.
- The specification of outputs only on transition arcs complicates the specification of outputs for Mealy circuits unnecessarily.

The State Machine Diagram Model

- In response to the issues listed, a broader state machine diagram (SMD) representation has been devised.
- Many other authors have used similar representations to overcome some of the issues we have listed.
- The SMD achieves the flexibility of the *algorithmic state machine* (ASM) (used in some previous editions of this text), without adopting the constraints of the ASM notation.

The State Machine Diagram

- Uses state nodes and transition arcs as in the traditional state diagram
- Adds notation for defining Mealy outputs on states as well as transitions
- Is based on input conditions, transition conditions, output conditions and output actions:
  - *Input condition*: a Boolean expression or equation which evaluates to either 0 or 1.
  - *Transition condition,* (TC): an input condition on a transition arc which evaluates to either 0 or 1.
  - *Output condition* (OC): a input condition that if equal to 1 causes an output action to occur and if 0 does not cause the output to occur.

State Machine Diagram

- Output Action Examples
  - Single Variables
    - Appearance of variable Z attached a state specifies that $Z = 1$. Z is implicitly 0 otherwise.
    - Appearance of variable Z attached to a transition condition (and possibly an output condition) from a state implies that $Z = 1$ for the condition(s) satisfied. Z is implicitly 0 otherwise unless Z is a Moore output (unconditional) attached to the state or is part of a TCI label attached to a state.
    - Separate default value statements may be used to explicitly specify by default $Z = 0$ or $Z = 1$.

State Machine Diagram

- **Output Action Examples**
  - **Vector Variables**
    - **Appearance of an equation Z = vector value attached to a state specifies the value of Z for the state.**
    - **Appearance of an equation Z = vector value attached to a transition condition (and possibly an output condition) from a state specifies the value of Z for the state, transition condition and output condition. The value of Z attached to a transition may also be specified by a Moore output (unconditional) attached to the state or as part of a TCI label attached to a state. Otherwise, Z takes on a default value if one is specified . The default value for a vector must be specified (including possibly don’t cares).**
  - **Register Transfer Outputs**
    - **Useful for describing controlled datapath operations (see Chapter 7)**

The State Machine Diagram

- **A *unconditional transition* has no transition condition on its arc or a transition condition consisting of the constant 1.**
- **A *conditional transition* has one or more transition conditions on its arc. If any one of the conditions evaluates to 1, the transition occurs.**

The State Machine Diagram

- *Moore output actions*, are unconditional, depending only on the state, and are attached by a line to the respective state.
- *Transition condition-independent* (TCI) *Mealy output actions* are preceded by their output condition and a slash and are attached by a line to the respective state. The output action occurs if the output condition evaluates to 1.
- *Transition condition-dependent* (TCD) Mealy output actions are attached by a line to their respective transition condition. The output action occurs if the output condition evaluates to 1.
- *Transition and output condition-dependent* (TCOD) Mealy output actions are preceded by an output condition and a slash and are attached by a line to their respective transition condition. The output action occurs if the transition condition and the output condition both evaluate to 1.

The State Machine Diagram

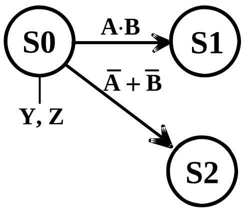
- To summarize, in a given state, an output action occurs if it is (a) unconditional (Moore), (b) TCI and its output condition OC evaluates to 1, (c) TCD and its transition condition TD evaluates to 1, or (d) TOCD and its transition condition TC and output condition OC both evaluate to 1.
- Moore and TCI output actions attached to a state, apply to *all* transitions from the state.

The State Machine Diagram

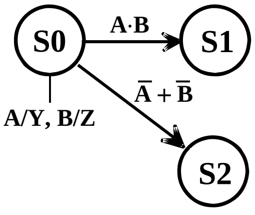
- This may seem complex, but note the following:
  - Only the unconditional output type applies to pure Moore machines
  - TDC outputs represents the traditional Mealy model and can be used exclusively at some potential cost in complexity including an increase in the number of states.
  - Mixing of Moore and Mealy types and the TCI and TCOD types provide optional opportunities to simplify the state diagram and state table and their specifications

Examples Of Transition & Output Conditions

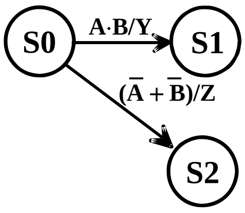
- Input Variables A, B, C
- Output Variables Y, Z
  Default: Y = 0, Z = 0



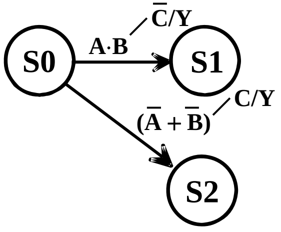
Ex. 1: Moore Outputs



Ex. 2: TCI Outputs



Ex. 3: TCD Outputs



Ex. 4: TCOD Outputs

Constraint Checking

- TC Constraints
  - Constraint 1: In state S_i, for all possible TC pairs (T_{ij}, T_{ik}) on arcs to distinct next states from S_i,

    $$T_{ij} \cdot T_{ik} = 0$$
  - Constraint 2: In state Si, for all possible TCs, T_{ij}

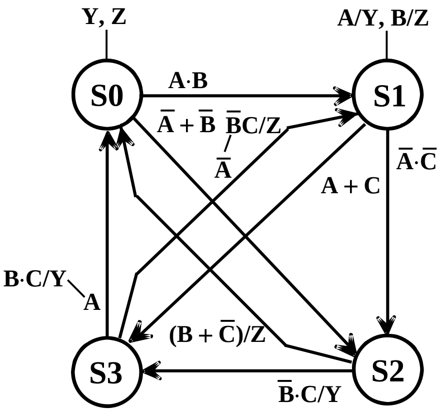
    $$\Sigma\, T_{ij} = 1$$
- OC Constraints
  - Constraint 1: For every output action in state S_i or on its transitions having coincident output variables with differing values, the corresponding pair of output condition (O_{ij}, O_{ik}) must be mutually exclusive, i. e., satisfy

    $$O_{ij} \cdot O_{ik} = 0$$
  - Constraint 2:For every output variable, the output conditions for state S_i or its transitions must cover all possible combinations of input variables that can occur, i. e.,

    $$\Sigma\, O_{ij} = 1$$
  - For both output constraints above, TCs must be used in evaluating O_{ij} for output actions of TCD and TCOD output action types
  - See text for using don't cares and defaults.

Constraint Checking Example

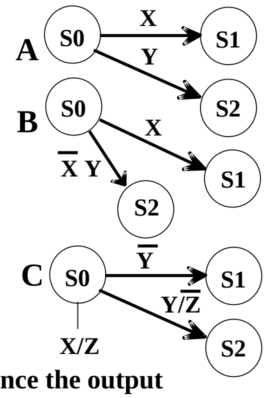
Defaults: Y = 0, Z = 0



- Transition Constraints:
  - S0: $A\cdot B\cdot(\overline{A} + \overline{B}) = 0$;
    $A\cdot B + (\overline{A} + \overline{B}) = 1$
  - S1: $\overline{A}\cdot\overline{C}\cdot(A + C) = 0$;
    $\overline{A}\cdot\overline{C} + (A + C) = 1$
  - S2: $\overline{B}\cdot C\cdot (B + \overline{C}) = 0$;
    $\overline{B}\cdot C + (B + \overline{C}) = 1$
  - S3: $A \cdot \overline{A} = 0$;
    $A + \overline{A} = 1$
- Output Constraints:
  - Satisfied for all four states by the given output conditions and values and the default constraints.

Constraint Violation Examples

- **Transition Constraints**
  - **Example A: $X \cdot Y \neq 0$ and $X + Y \neq 1$, so two constraints are violated**
  - **Example B: $X \cdot \bar{X}Y = 0$, but $X + \bar{X}Y \neq 1$. so constraint 2 is violated**
- **Outputs**
  - **Example C: For values Z = 1 and Z = 0, $X \cdot Y \neq 0$, so constraint 1 is violated**
  - **Constraint $X + Y + \bar{Y} = 1$, due to the default value of Z on $\bar{Y}$, so constraint 2 is satisfied**
  - **Example D: In general, for a given state, since the output condition for a Moore type output action is 1, no output action on a same output variable with a different value is permitted on the transitions.**



State Table Format

State	State Code	Transition Condition	Next State	Next State Code	Output Actions (and OCs)
State Name 1	State Code 1	Unused	Unconditional Next State 1	Next State Code 1	Moore or TCI Output (and OC)
		Transition Cond. 11	Next State 11	Next State Code 11	TCD or TOCD Output 11(and OC)
		Additional Transition Conditions and Entries for State Name 1			
State Name i	Entries for State Names i, i = 2, …n				

State Table Example

- State table for constraint checking example

State	State Code	Transition Condition	Next State	Next State Code	Output Actions (OCs)
S0	00				Y,Z
		A·B	S1	01	
		$\overline{A}$ + $\overline{B}$	S2	10	
S1	01				A/Y, B/Z
		$\overline{A}$·$\overline{C}$	S2	10	
		A + C	S3	11	

- Continued on next slide

State Table Example (continued)

- State table for constraint checking example

State	State Code	Transition Condition	Next State	Next State Code	Output Actions (OCs)
S2	10				
		$\overline{B}$·C	S3	11	Y*
		B + $\overline{C}$	S0	00	Z*
S3	11				
		A	S0	00	B·C/Y*
		$\overline{A}$	S1	01	$\overline{B}$·C/Y*

- * is reminder of an output action dependent on transition condition

State Machine Design Procedure

- Define the input and output variables for the circuit or system and meaning of 0 and 1 values of each variable
- Draw the state machine diagram or formulate the state machine table for the circuit or system
- If a state machine diagram is used, convert it to a state machine table
- From the state machine table, derive optimized next state equations and output equations for the circuit or system

Example State Machine Design – Elevator Control – Inputs

- Circuit: Elevator control for two-floor elevator
- Warning: Does not include safety features or all user buttons!
- C1(C2) – Call button (outside elevator) to floor 1(2)
    - 0 – no action; 1 – call for elevator
- G1(G2) – Go button (inside elevator) to floor 1(2)
    - 0 – no action; 1 – go to floor command
- F1(F2) – Senses elevator at floor 1(2)
    - 0 – elevator not at floor; 1 – elevator at floor
- S1(S2) – Senses elevator approaching floor 1(2) (Controls slowdown of elevator)
    - 0 – elevator not approaching floor; 1 – elevator approaching floor
- DO – Doors open
    - 0 – doors not fully open; 1 – doors fully open
- TO – End of time interval from button push to elevator movement starting
    - 0 – waiting for time interval to end; 1 – time interval has ended
- DC – Doors closed
    - 0 – doors not closed; 1 – doors closed

Example – Elevator Control - Outputs

- Up – elevator to go up
  - 0 – no action; 1 – commands elevator to go up
- Down – commands elevator to go down
  - 0 – no action; 1 – commands elevator to go down
- TS – timer start
  - 0 – no action; 1 – initialize and start timer
- SD – slow down
  - 0 – elevator moves as normal speed; 1 – elevator approaching target floor slows down
- OD – Open Doors
  - 0 – no action; 1 – open doors
- CD – Close Doors
  - 0 – no action; 1 – close doors

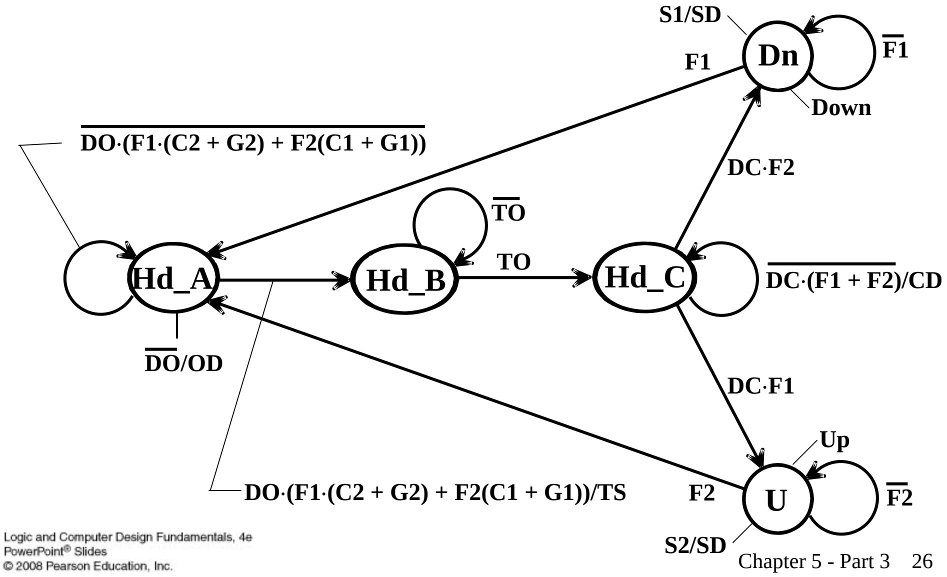
Example – Elevator Operation – Specifications

- The elevator parks at the floor to which it has last taken passengers with doors open.
- Call button C_i calls elevator to a floor.
- If the elevator is not at the floor, TS is used to initialize and start the timer;
- After TO becomes 1, the doors close, and when DC is active, the Up or Down output is activated.
- The S_i sensor detects the floor approach and activates output SD to slow elevator.
- The F_i sensor detects the elevator at the floor, forces both Up and Dn to 0, and opens the doors.
- Passenger(s) enter elevator and push the G_i button.
- After TO becomes 1, the doors close, and when DC is active, the Up or Down output is activated.
- The S_i sensor detects the approach and activates output SD to slow elevator.
- The F_i sensor detects the elevator at the floor, forces both Up and Dn to 0, and opens the doors, permitting passengers to exit.

Example – Elevator Control – States

- Initial proposed states:
  - U (Up)
  - Dn (Down)
  - Hd (Hold)
- Series of actions required in Hd state:
  - Open doors
  - Use timer to wait for passengers
  - Close doors
- Expand Hd to 3 states: Hd_A, Hd_B, Hd_C
- One-Hot State Vector:
  (U, Dn, Hd_C, Hd_B, Hd_A)

Example – Elevator Control – SMD


S1/SD
F1
Dn
$\overline{F1}$
Down
$\overline{DO \cdot (F1 \cdot (C2 + G2) + F2(C1 + G1))}$
DC·F2
$\overline{TO}$
Hd_A
Hd_B
TO
Hd_C
$\overline{DC \cdot (F1 + F2)}$/CD
$\overline{DO}$/OD
DC·F1
Up
DO·(F1·(C2 + G2) + F2(C1 + G1))/TS
F2
U
$\overline{F2}$
S2/SD

Example – Elevator Control – SMT

State	State Code	Transition Condition	Next State	Next State Code	Output Actions (OCs)
Hd_A	00001				$\overline{DO}$/OD
		$\overline{(DO\cdot(F1\cdot(C2 + G2) + F2\cdot(C1 + G1))}$	Hd_A	00001	
		$DO\cdot(F1\cdot(C2 + G2) + F2\cdot(C1 + G1)$	Hd_B	00010	TS*
Hd_B	00010	$\overline{TO}$	Hd_B	00010	
		TO	Hd_C	00100	
Hd_C	00100	$\overline{DC\cdot(F1 + F2)}$	Hd_C	00100	CD*
		DC·F2	Dn	01000	
		DC·F1	Up	10000	

Example – Elevator Control – SMT

State	State Code	Transition Condition	Next State	Next State Code	Output Actions (OCs)
Dn	01000				Down, S1/SD
		$\overline{F1}$	Dn	01000	
		F1	Hd_A	00001	
U					Up, S2/SD
		$\overline{F2}$	Up	10000	
		F2	Hd_A	00001	

Example – Elevator Control - Equations

- **Flip-Flop Input**
  - $X = DO\cdot((F1\cdot(C2 + G2) + F2\cdot(C1 + G1))$
  - $Y = DC\cdot(F1 + F2)$
  - $D_{Hd\_A} = Hd\_A\cdot\overline{X} + Dn\cdot F2 + U\cdot F1$
  - $D_{Hd\_B} = Hd\_A\cdot X + Hd\_B\cdot\overline{TO}$
  - $D_{Hd\_C} = Hd\_B\cdot TO + Hd\_C\cdot\overline{Y}$
  - $D_{Dn} = Hd\_C\cdot DC\cdot F2 + Dn\cdot\overline{F1}$
  - $D_U = Hd\_C\cdot DC\cdot F1 + U\cdot\overline{F2}$
- **Output**
  - Down = Dn
  - Up = U
  - $SD = Dn\cdot S1 + U\cdot S2$
  - $TS = Hd\_A\cdot X$
  - $OD = Hd\_A\cdot\overline{DO}$
  - $CD = Hd\_C\cdot\overline{Y}$

Terms of Use

- **All (or portions) of this material © 2008 by Pearson Education, Inc.**
- **Permission is given to incorporate this material or adaptations thereof into classroom presentations and handouts to instructors in courses adopting the latest edition of Logic and Computer Design Fundamentals as the course textbook.**
- **These materials or adaptations thereof are not to be sold or otherwise offered for consideration.**
- **This Terms of Use slide or page is to be included within the original materials or any adaptations thereof.**