Chapter 8
Design at the Register Transfer Level

Outline

- Register Transfer Level (RTL) Notation
- Algorithmic State Machines (ASMs)
- Design Example
- Sequential Binary Multiplier
- Control Logic
- Design with Multiplexers
- HDL Description of Design Examples

Introduction

- A digital system is a sequential logic system constructed with flip-flops and gates.
  - To specify a large digital system with a state table is very difficult .
  - Modular subsystems
  - Registers, decoders, multiplexers, arithmetic elements and control logic.
  - They are interconnected with datapaths and control signals.

Register Transfer Level

- A digital system is represented in register transfer level when it is specified by the three components.
  - 1. The set of registers in the system
  - 2. The operations that are performed on the data in the registers
  - 3. The control that supervises the sequence of operations.
- Register : a group of flip-flops that stores binary information and performs elementary operations.
- Operation: execution on the information stored in register in parallel during one clock cycle. The results may replace the original register or transfer to another register
- Control: initiation of a sequence of operations consisting of timing signals that sequence the operations in a prescribed manner.

Operations

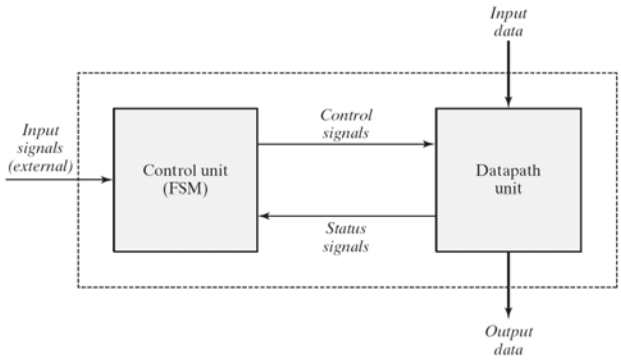
- Information Transfer
  - (R2 ← R1)
  - From output of R1 to input of R2
  - Parallel load capability
- Conditioned Transfer
  - (If (T1 = 1) then (R2 ← R1))
  - Clock transfer occurs during clock edges.
- Two or more Transfer
  - (If (T3 = 1) then (R2 ← R1, R1 ← R2))
  - Exchange of the content of R1 and R2 during clock edge, provided that T3 = 1.

Operation Types

- Four categories of operational types in digital systems.
  - Transfer operations that transfer data from one register to another
  - Arithmetic operations that perform arithmetic on data in register
  - Logic operations that perform bit manipulation of non-numeric data in register
  - Shift operations that shift data in register

Algorithmic State Machine (ASM)

- Control logic and data processing in a digital system
- Control signal
  - Initiates a sequence of commands to the data path.
  - Uses the status conditions from datapath as decision variable for determining the sequence of control signals.



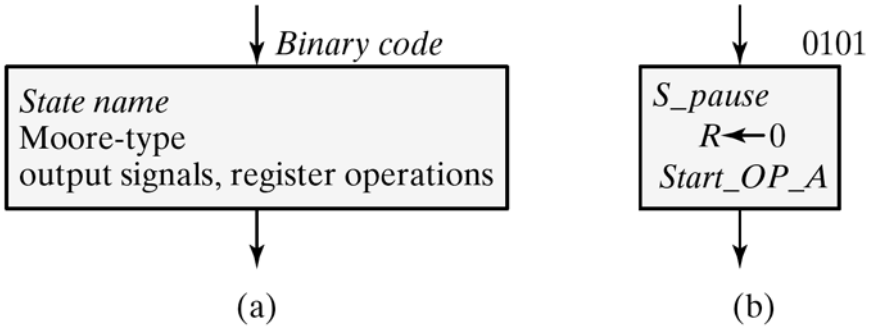
- A hardware algorithm is used to describe the objective of the digital design.
- An algorithmic state machine (ASM) is a flow-chart method which can be used to defined the digital hardware algorithm.

ASM Chart

- The ASM chart is a special type of flow-chart for describing the sequential operations of a digital system.
- Three basic element
  - The state box
  - The decision box
  - The condition box
- They are connected by directed edges indicating the sequential precedence and evolution of the states as the machine operates.

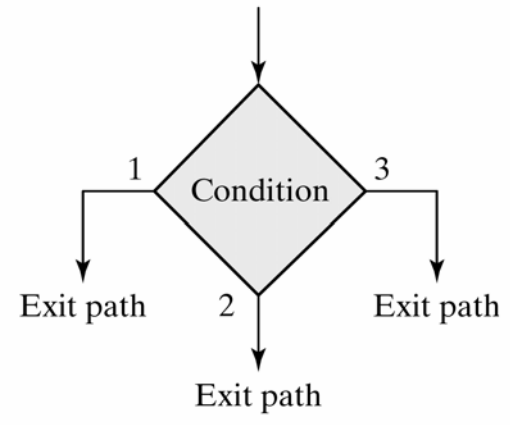
State Box

- State box is rectangular
  - Register Operation or Output
  - State Specification



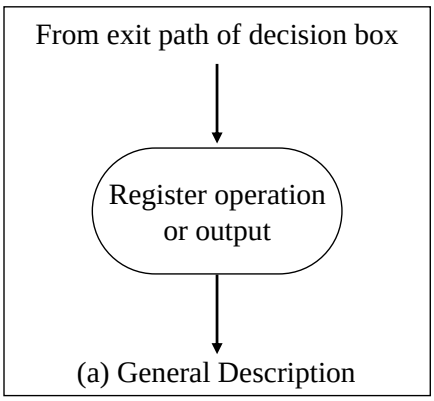
Decision Box

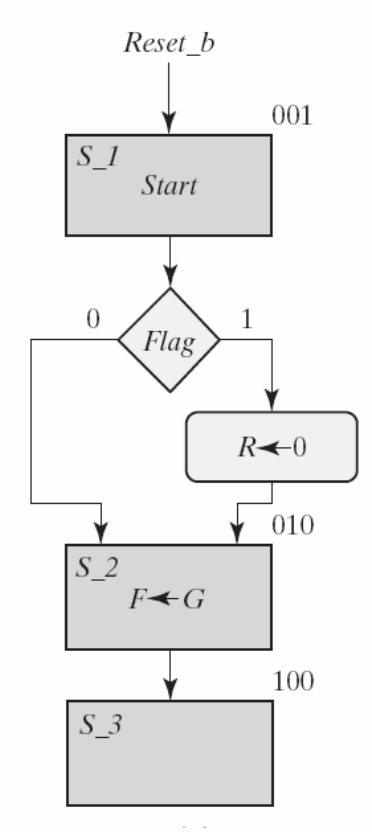
- The decision box is diamond-shaped box
  - The effect of an input on the control system
  - Indication of decision condition



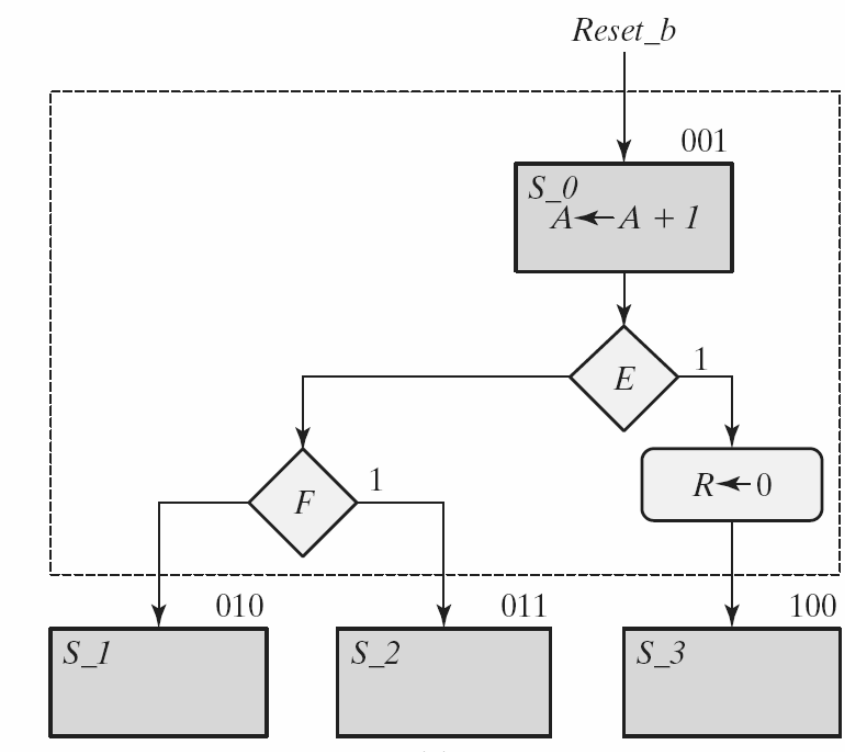
Condition Box

- The condition box is oval-shaped.
  - Unique to ASM chart
  - The input path must come from one of the exit paths of decision box
  - The operation or output is generated if input condition is satisfied.



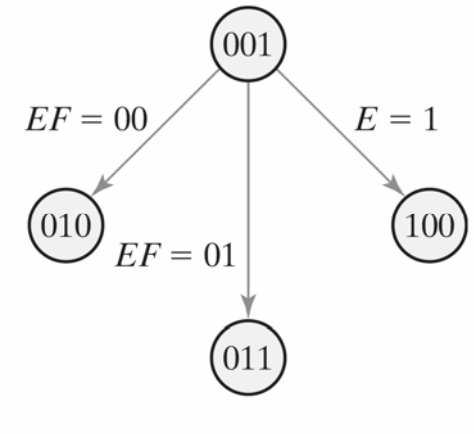


ASM Block Example



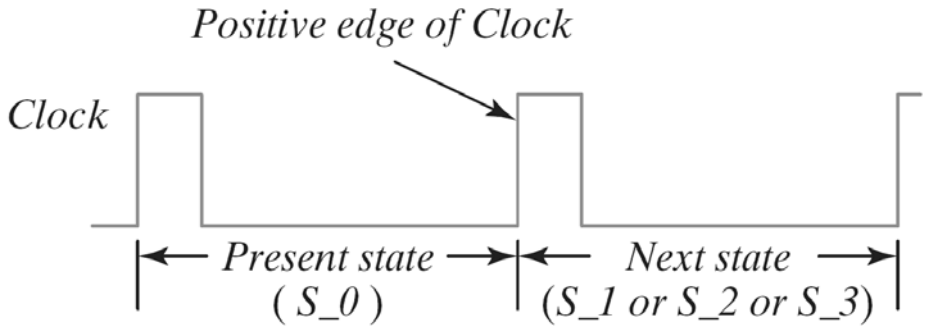
Equivalent State Diagram

- Each block in ASM chart describes the state of the system during one clock-pulse interval.
- Each state block is equivalent to a state in a sequential circuit, thus, is easy to convert to a state diagram.



Timing Consideration

- The timing for all registers and flip-flops in a digital system is controlled by a master-clock.
- In ASM chart, all operations specified in a block must occur during the edge transition of the clock pulse.
  - Example of state transition



ASMD Chart

- Contrasted between Algorithmic State Machine and Datapath (ASMD) charts & ASM charts.
  - An ASMD chart are does not list register operations within a state box.
  - The edges of an ASMD charts are annotated with register operations that are concurrent with the state transition indicated by the edge.
  - An ASMD chart includes conditional boxes identifying the singles which control the register operation s that annotate the edges of the chart.
  - An ASMD chart associates register operations with state transitions rather than with state.

ASMD Chart

- Designed an ASMD chart have three-step:
  - Form an ASM chart displaying only how the inputs to the controller determine its state transitions.
  - Convert the ASM chart to an ASMD chart by annotating the edges of ASM chart to indicate to the concur rent register operations of the datapath unit.
  - Modify the ASMD chart to identify the control singles that are generated by the controller and the ca use the indicated register operations in the datapath unit.

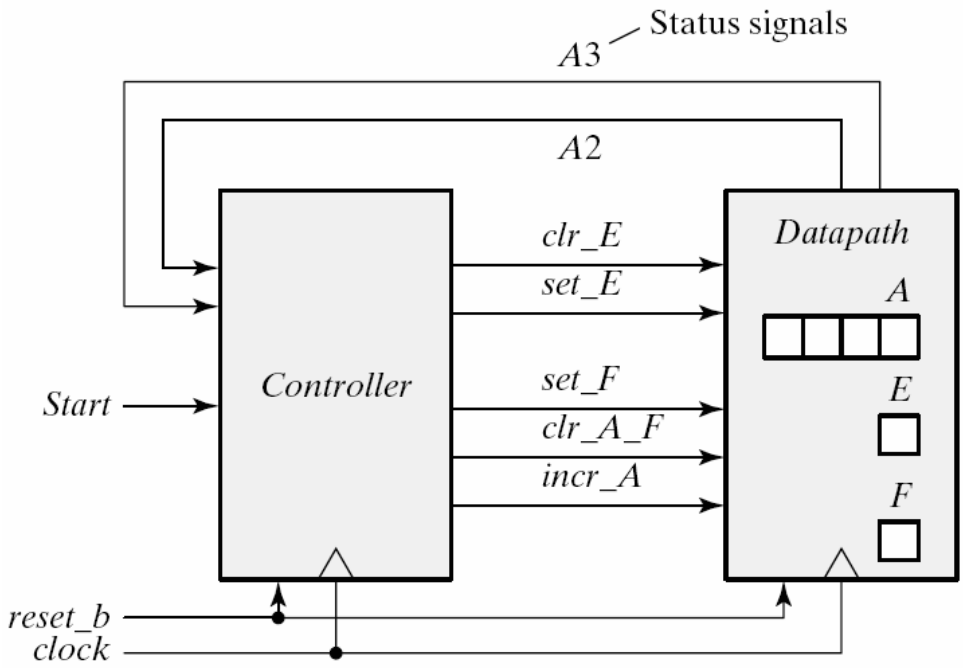
Design Example I - Counter

- Specification
  - Two flip-flops, E and F.
  - One 4-bit binary counter A ($A_3A_2A_1A_0$)
  - Start signal S
  - Counter bits A_4 A_3 determine the sequence of operations
    - If A_2 = 0, E is 0, and count continues.
    - If A_2 = 1, E is 1, then if A_3 = 0, the count continues, but if A_3 = 1, F is 1 on next cycle, system stops counting.
    - If *Start* = 0, the system remains in initial state, but if *Start* = 1, the operation cycle repeats.

Design Example

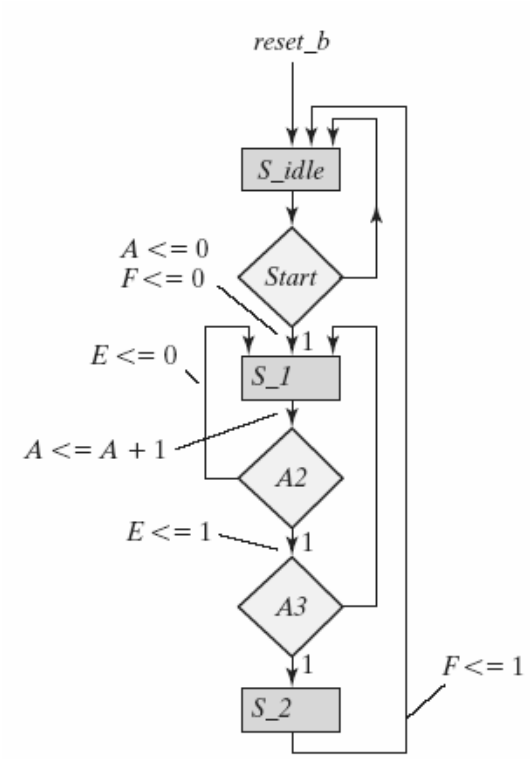
Block Diagram

Note: A3 denotes A[3],
A2 denotes A[2],
<= denotes nonblocking assignment
reset_b denotes active-low reset condition

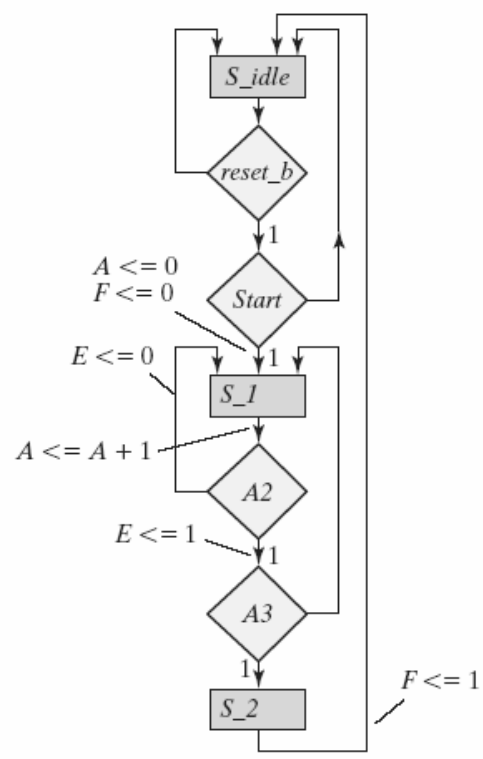


ASMD Charts

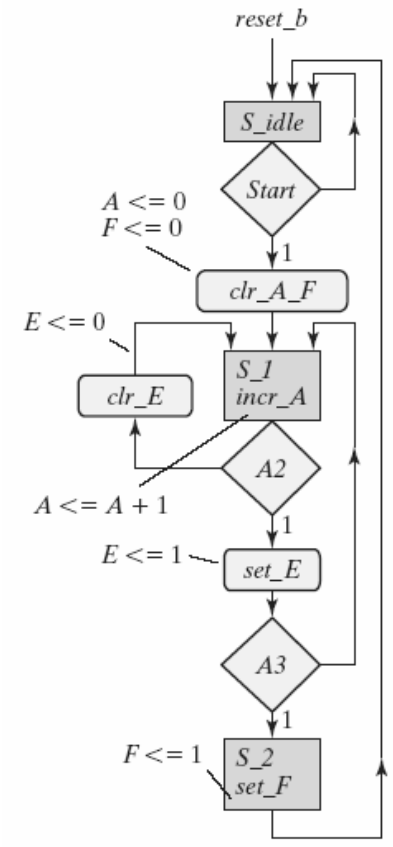
Asynchronous Reset



Synchronous Reset



ASMD Charts

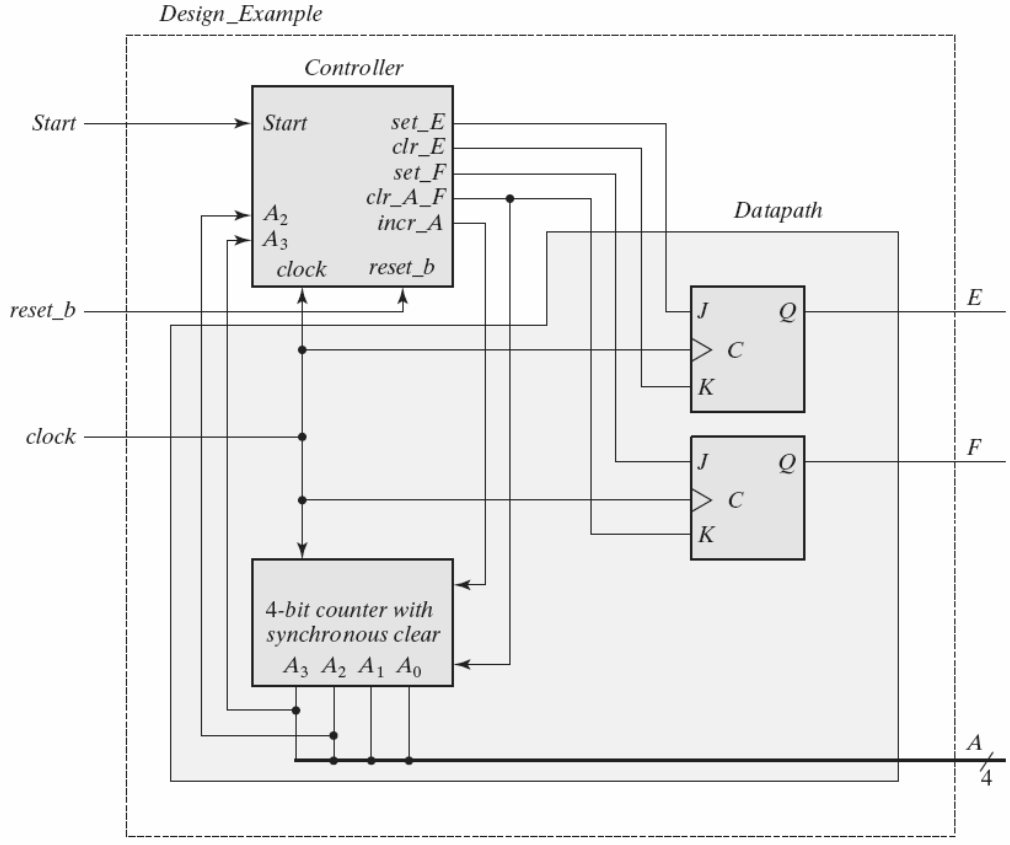


Timing Sequence

Sequence of Operations for Design Example

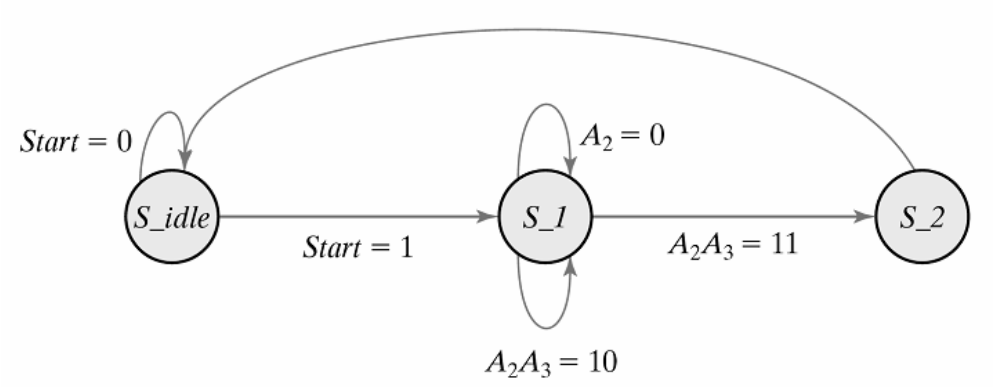
Counter				Flip-Flops			
A_3	A_2	A_1	A_0	E	F	Conditions	State
0	0	0	0	1	0	$A_2 = 0, A_3 = 0$	*S_1*
0	0	0	1	0	0		
0	0	1	0	0	0		
0	0	1	1	0	0		
0	1	0	0	0	0	$A_2 = 1, A_3 = 0$	
0	1	0	1	1	0		
0	1	1	0	1	0		
0	1	1	1	1	0		
1	0	0	0	1	0	$A_2 = 0, A_3 = 1$	
1	0	0	1	0	0		
1	0	1	0	0	0		
1	0	1	1	0	0		
1	1	0	0	0	0	$A_2 = 1, A_3 = 1$	
1	1	0	1	1	0		*S_2*
1	1	0	1	1	1		*S_idle*

Controller and Datapath Design



Register Transfer Level

- Register transfer level description



S_idle ⟶ S_1, clr_A_F: $A \leftarrow 0$, $F \leftarrow 0$

S_1 ⟶ S_1, incr_A: $A \leftarrow A + 1$

if ($A_2 = 1$) then set_E: $E \leftarrow 1$

if ($A_2 = 0$) then clr_E: $E \leftarrow 0$

S_2 ⟶ S_idle, set_F: $F \leftarrow 1$

State Table

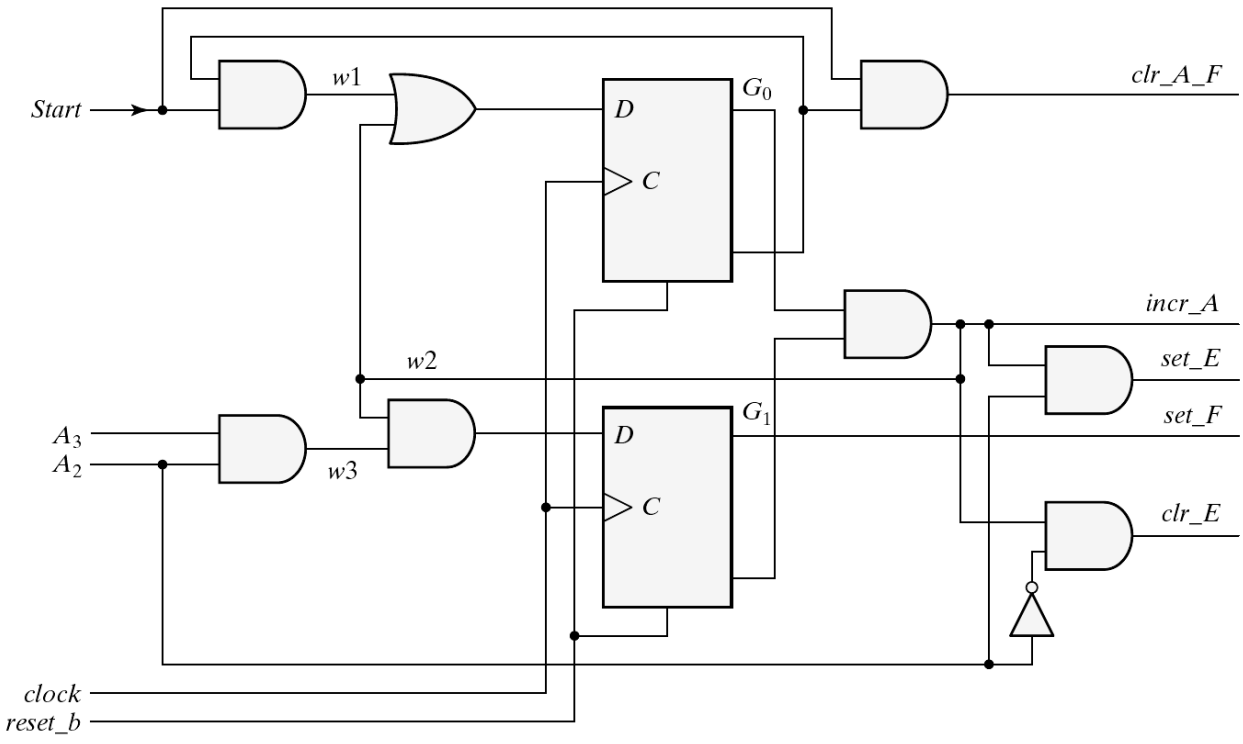
State Table for the Controller of Fig. 8.10

	Present State		Inputs			Next State		Outputs				
Present-State Symbol	G_1	G_0	*Start*	A_2	A_3	G_1	G_0	*set_E*	*clr_E*	*set_F*	*clr_A_F*	*incr_A*
S_idle	0	0	0	X	X	0	0	0	0	0	0	0
S_idle	0	0	1	X	X	0	1	0	0	0	1	0
S_1	0	1	X	0	X	0	1	0	1	0	0	1
S_1	0	1	X	1	0	0	1	1	0	0	0	1
S_1	0	1	X	1	1	1	1	1	0	0	0	1
S_2	1	1	X	X	X	0	0	0	0	1	0	0

Control

- Method 1
  - Five-variable map using $G_1\ G_0\ S\ A_3\ A_2$
- Method 2
  - Simply the circuit by inspection and set each flip-flop to 1 for next-state.
    - $D_{G1} = S\_1A_3A_2$
    - $D_{G0} = Start\ S\_idle + S\_1$
  - Output
    - $set\_E = S\_1\ A_2$
    - $clr\_E = S\_1\ A_2'$
    - $set\_F = S\_2$
    - $clr\_A\_F = Start\ S\_idle$
    - $incr\_A = S\_1$

Logic Diagram



Sequential Binary Multiplier

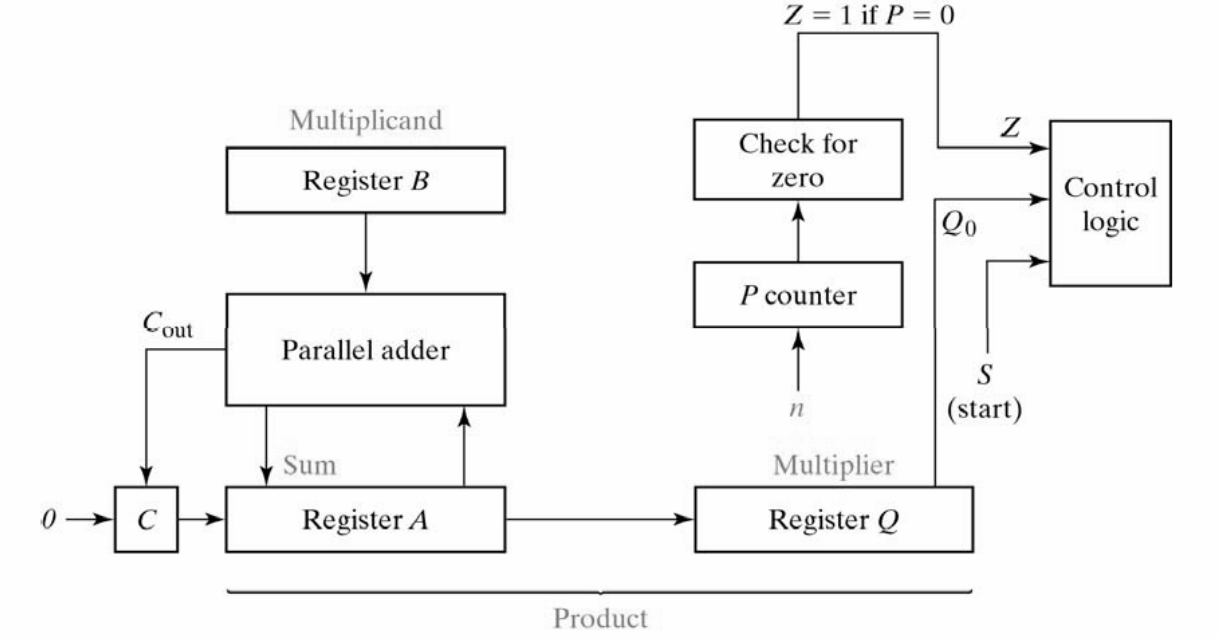
- Specification: Binary Multiplication
- Method: Shift left and accumulate (direct way)
- Example: multiplying two binary numbers 10111 $(23)_{10}$ and 10011 $(19)_{10}$:

```
23        10111  multiplicand
19        10011  multiplier
          10111
         10111
        00000
       00000
      10111
437   110110101 product
```

The product obtained from the multiplication of two n-bit binary numbers can be up to $2n$ bits.

Sequential Binary Multiplier

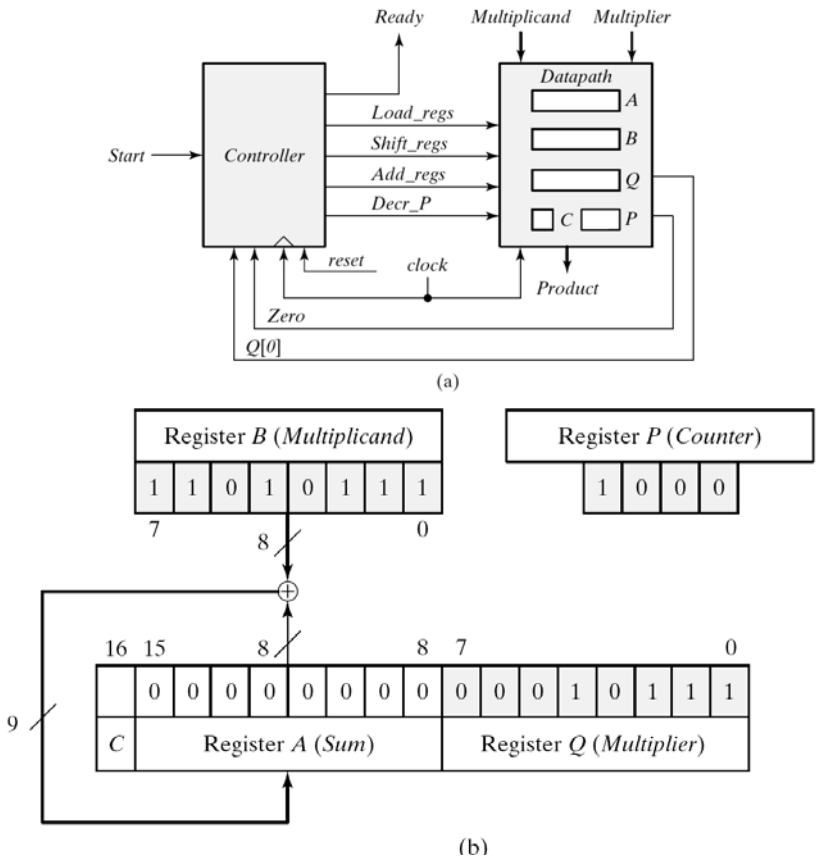
- An alternative method: Shift Right and accumulate
- Block Diagram of Binary Multiplier



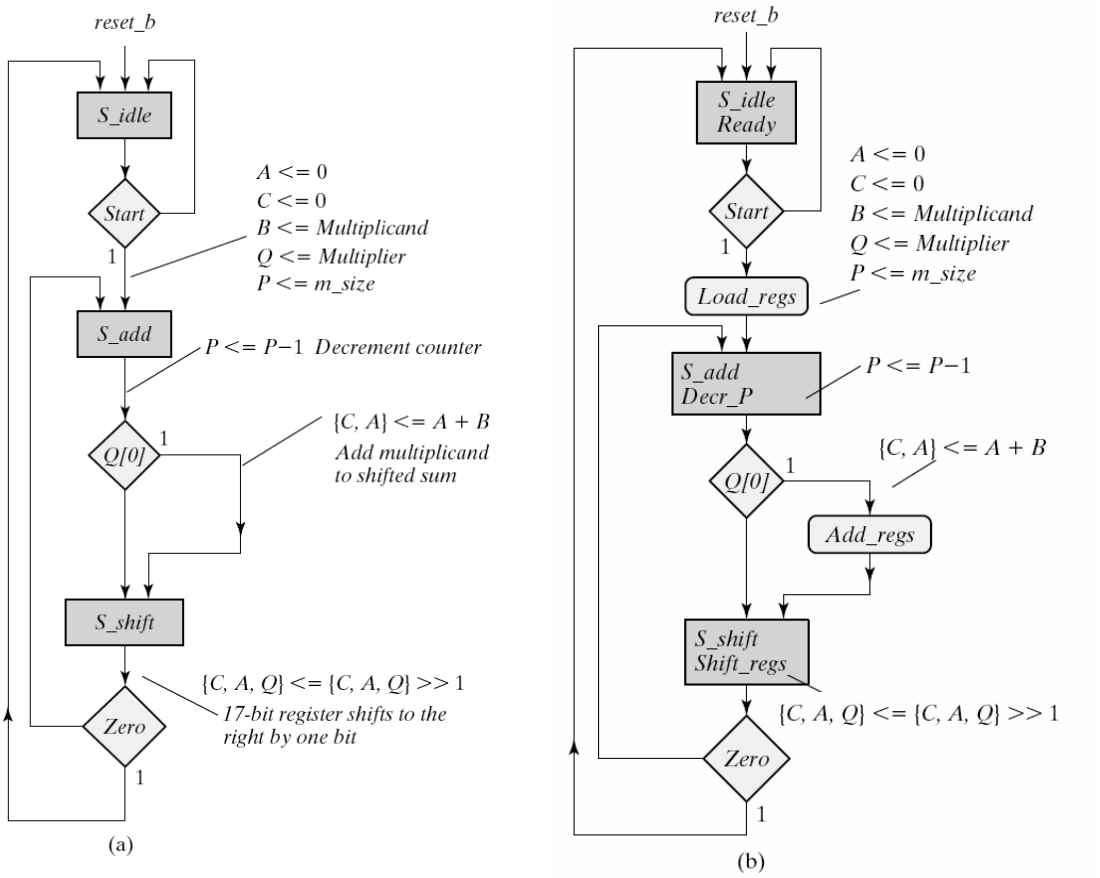
ASM Chart

- Register A: A shift register with parallel load to accept the sum of the adder, and clear to reset to 0.
- Register B and Q: A shift registers with parallel load to received multiplicand and multiplier.
- Counter P: Binary counter with a parallel load to accept a binary constant.
  - Z status register: Z = 0 when P = 0; Z = 1 when P = 0
- C flip-flop: flip-flop to accept input carry and a synchronous reset.
- Shift right CAQ in the chart
  - A ← shr A, A_{n-1} ← C
  - Q ← shr Q, Q_{n-1} ← A0
  - C ← 0

Register Configuration



ASMD for Binary Multiplier



Numerical Example

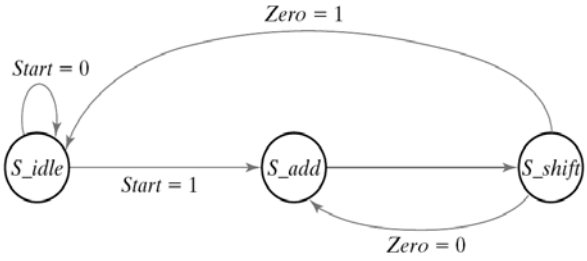
Numerical Example For Binary Multiplier

Multiplicand $B = 10111_2 = 17_H = 23_{10}$ **Multiplier $Q = 10011_2 = 13_H = 19_{10}$**

	C	A	Q	P
Multiplier in Q	0	00000	10011	101
$Q_0 = 1$; add B		10111		
First partial product	0	10111		100
Shift right CAQ	0	01011	11001	
$Q_0 = 1$; add B		10111		
Second partial product	1	00010		011
Shift right CAQ	0	10001	01100	
$Q_0 = 0$; shift right CAQ	0	01000	10110	010
$Q_0 = 0$; shift right CAQ	0	00100	01011	001
$Q_0 = 1$; add B		10111		
Fifth partial product	0	11011		
Shift right CAQ	0	01101	10101	000
Final product in $AQ = 0110110101_2 = 1b5_H$				

Control Logic

- The design of a digital system has two parts
  - The design of register transfers in the datapath
  - The design of control logic
- The control logic design is a sequential circuit design problem.
- Converting from ASM Chart to state diagram.



(a)

State Transition		Register Operations
From	To	
S_idle		Initial state
S_idle	*S_add*	$A <= 0, C <= 0, P <= dp\_width$
S_add	*S_shift*	$P <= P - 1$ if $(Q[0])$ then $(A <= A + B, C <= C_{out})$
S_shift		shift right $\{CAQ\}, C <= 0$

State Assignment for Control

State	Binary	Gray Code	One-Hot
S_idle	00	00	001
S_add	01	01	010
S_shift	10	11	100

Control Logic and State Assignment

- The are two distinct aspects
  - Establish the required sequence of states. (State Diagram)
  - Provide signals to control the register operations. (register transfer)
- State assignment
  - Binary number
  - Gray code
  - One-hot assignment
- One method
  - Sequential circuit may use binary number for control logic design
  - Use sequence register and decoder
- The other method
  - Use one flip-flop per state (one-hot assignment)

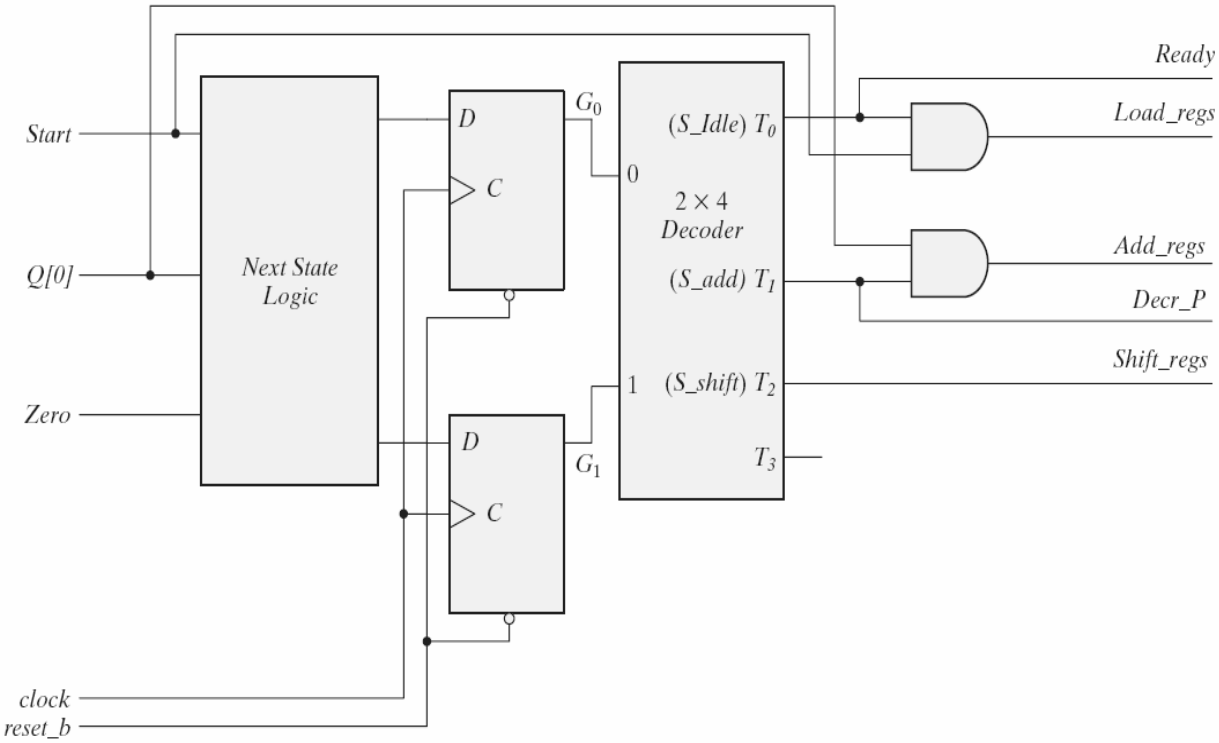
Sequence Register and Decoder

- Control logic using Sequence Register and Decoder
  - User binary-valued register for control states
  - Decoder provides corresponding output to each states

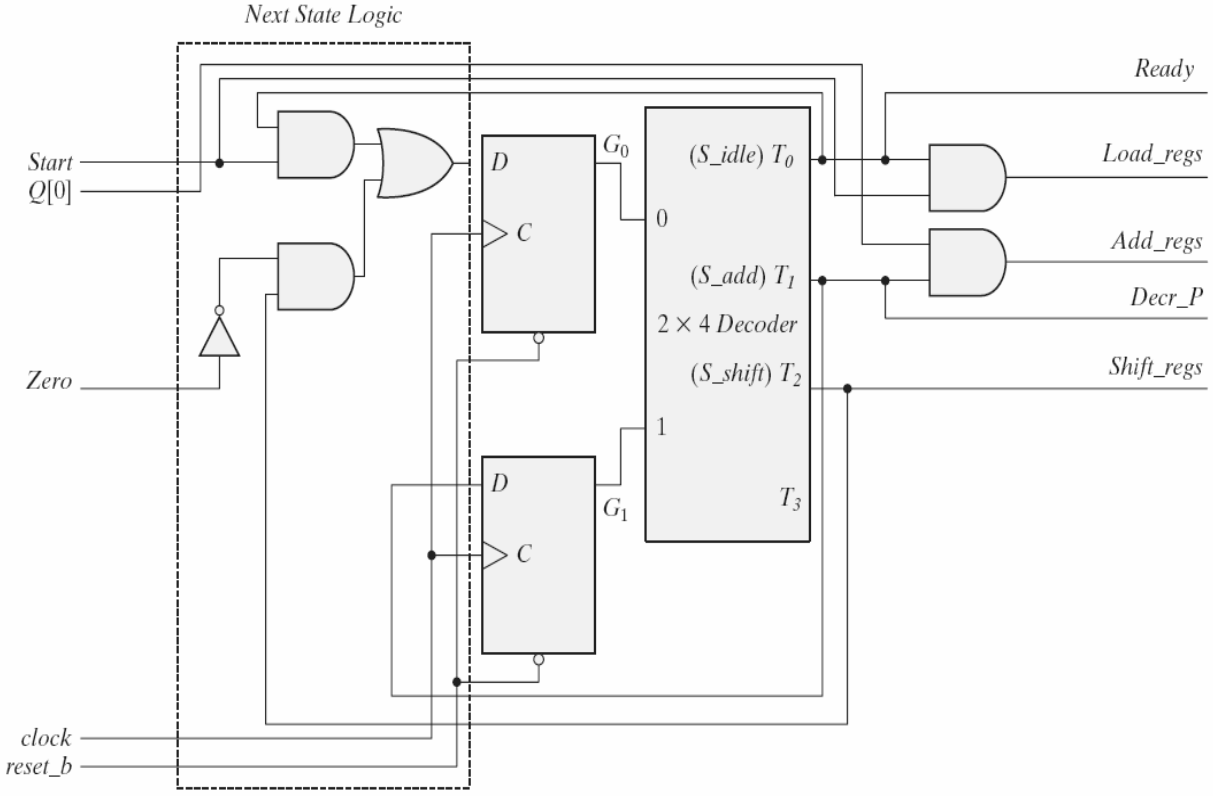
State Table for Control Circuit

	Present State		Inputs			Next State						
Present-State Symbol	G_1	G_0	*Start*	*Q[0]*	*Zero*	G_1	G_0	*Ready*	*Load_regs*	*Decr_P*	*Add_regs*	*Shift_regs*
S_idle	0	0	0	X	X	0	0	1	0	0	0	0
S_idle	0	0	1	X	X	0	1	1	1	0	0	0
S_add	0	1	X	0	X	1	0	0	0	1	0	0
S_add	0	1	X	1	X	1	0	0	0	1	1	0
S_shift	1	0	X	X	0	0	1	0	0	0	0	1
S_shift	1	0	X	X	1	0	0	0	0	0	0	1

Logic Diagram of Control for Binary Multiplier



Logic Diagram of Control for Binary Multiplier



Example

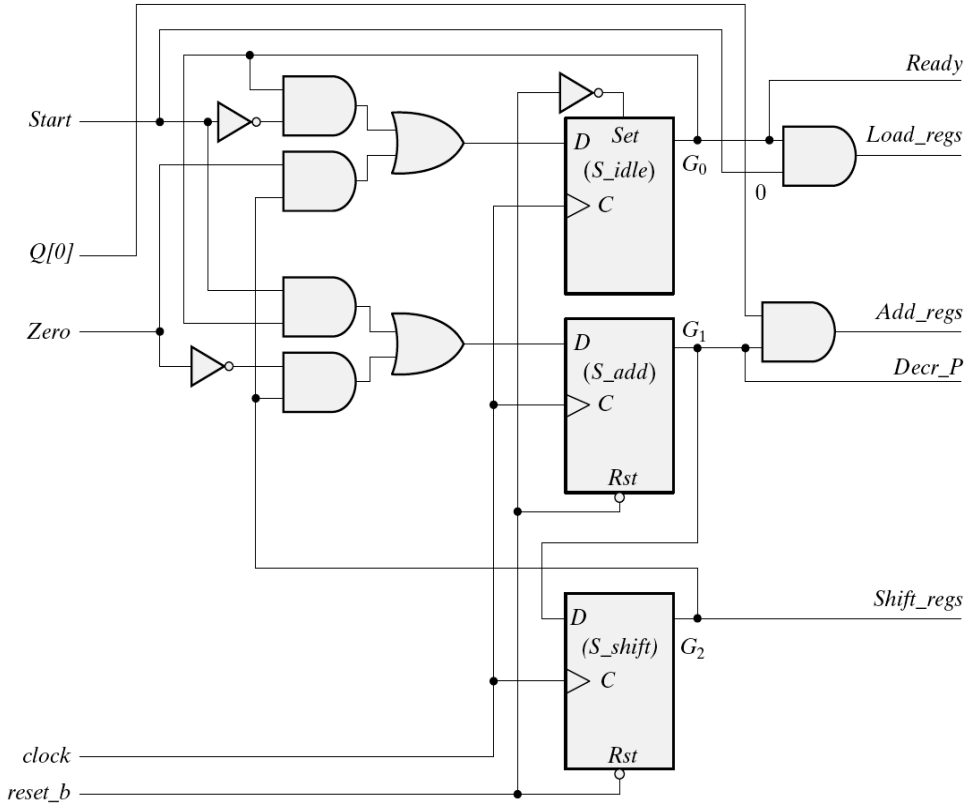
- From the next-state conditions in the state table, we find the next state G_1 is equal to 1 when the present state is *S_add* and is equal to 0 when the present state is *S_idle* or *S_shift*. These conditions can be specified by the equation

$$D_{G1} = T_1$$

where D_{G1} is D input of flip-flop G_1. Similarly, the D input of G_0 is

$$D_{G0} = T_0\,Start' + T_2\,Zero'$$

Logic Diagram for One-Hot State Controller



Example

- D_{G0} the input to flip-flop G_0, is set to 1 if the machine is in state G_0 and *Start* is not asserted, or if the machine is in state G_2 and *Zero* is asserted. The conditions are specified by the input equation:

$$D_{G0} = G_0\ Start' + G_2\ Zero$$

Using this procedure for the other three flip-flops, we obtain the remaining input equations:

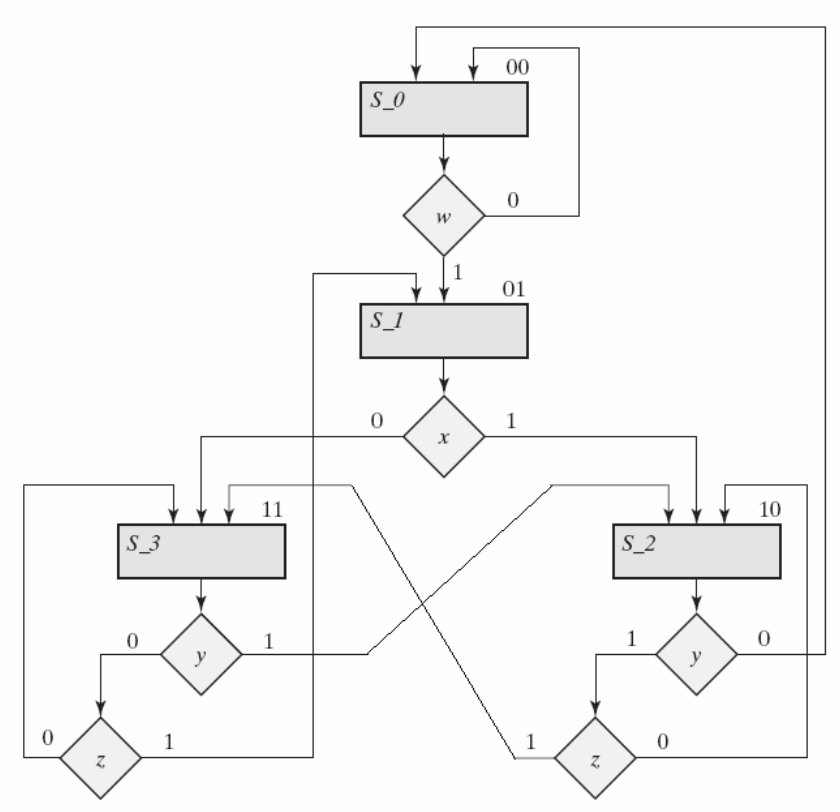
$$D_{G1} = G_0\ Start + G_2\ Zero'$$

$$D_{G2} = G_1$$

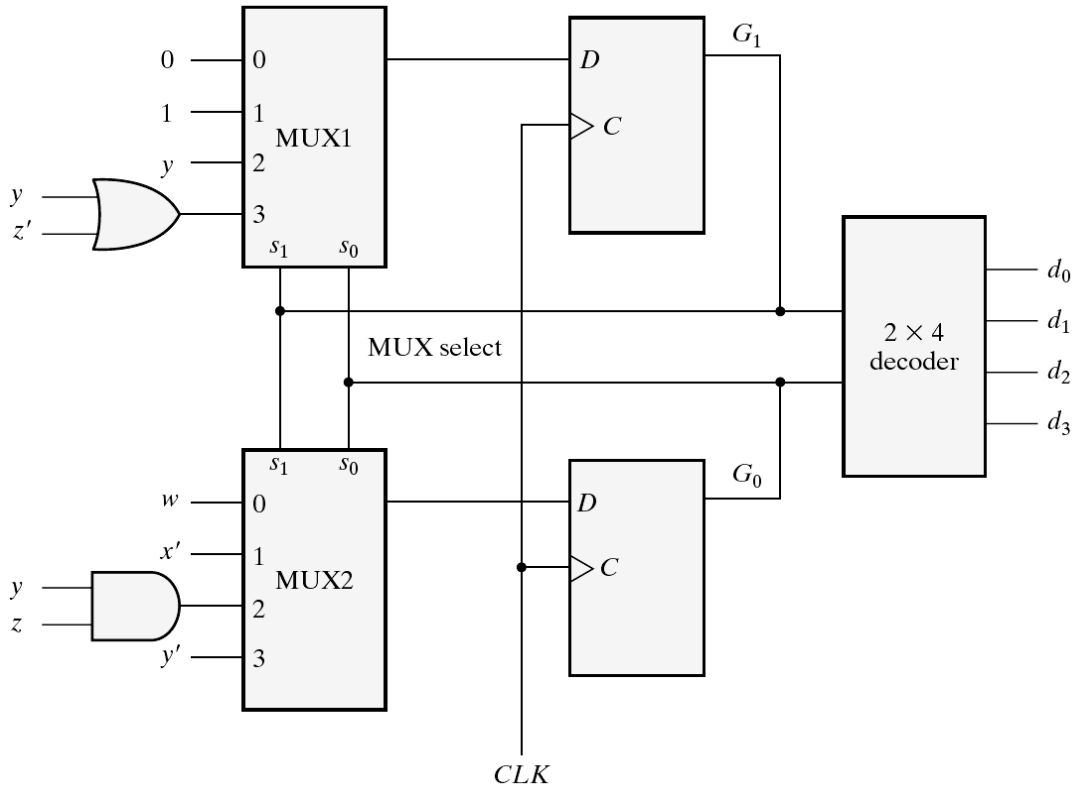
Design with Multiplexers

- Replacing the gates with multiplexers results in a regular pattern of three levels of components.
  - The first level consist of multiplexers that determine the next state of the register.
  - The second level contains a register that holds the present binary state.
  - The three level has a decoder that asserts a unique output line for each control state.

Example of ASM Chart with Four Control Inputs



Control Implementation with Multiplexers



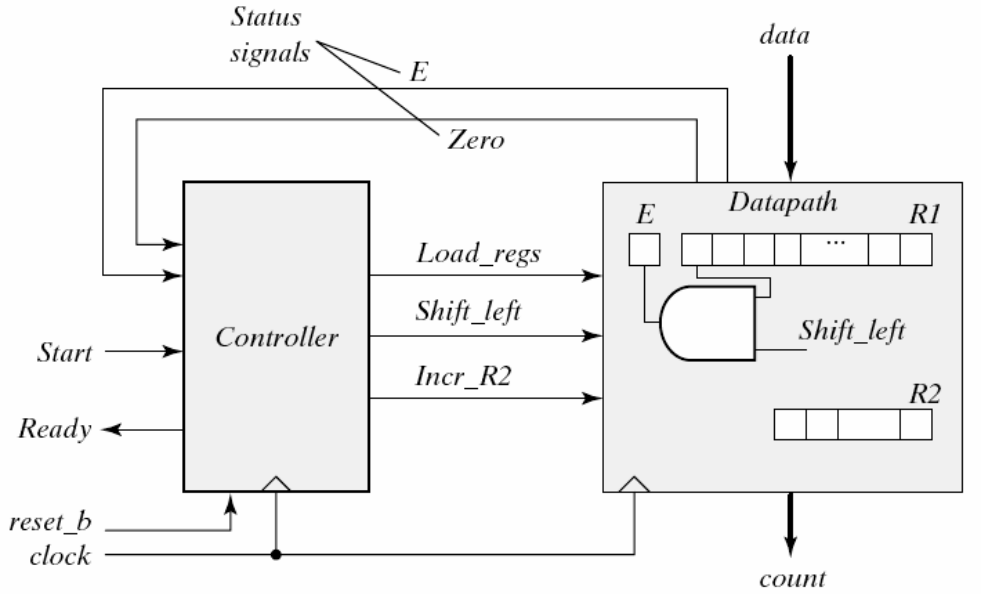
State Table

Multiplexer Input Conditions

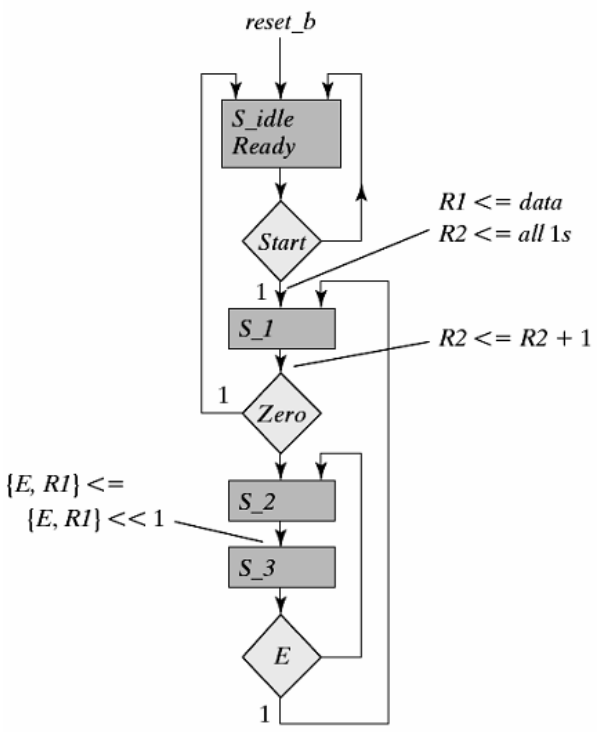
Present State		Next State		Input Condition	Inputs	
G_1	G_0	G_1	G_0	s	MUX1	MUX2
0	0	0	0	w'		
0	0	0	1	w	0	w
0	1	1	0	x		
0	1	1	1	x'	1	x'
1	0	0	0	y'		
1	0	1	0	yz'		
1	0	1	1	yz	$yz' + yz = y$	yz
1	1	0	1	$y'z$		
1	1	1	0	y		
1	1	1	1	$y'z'$	$y + y'z' = y + z'$	$y'z + y'z' = y'$

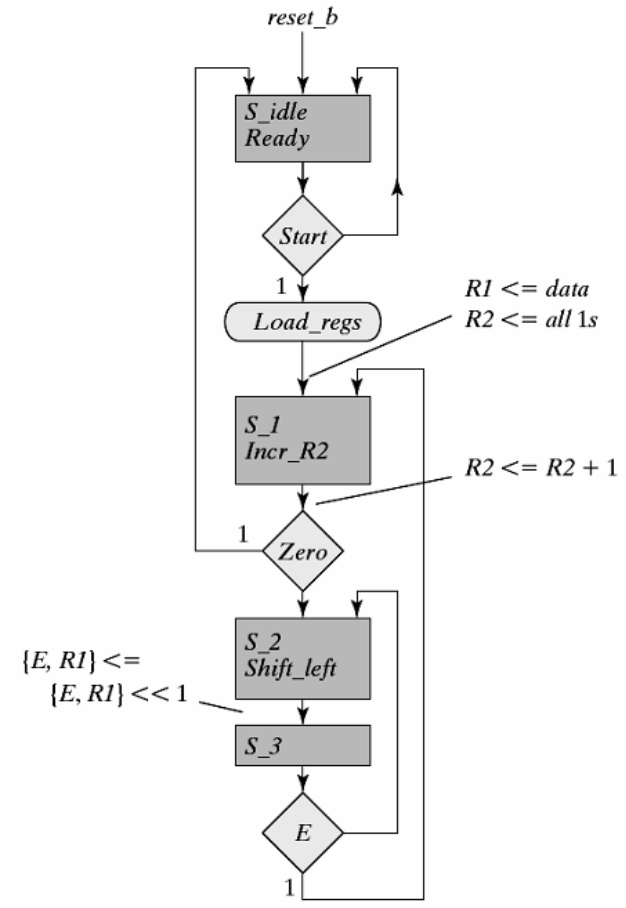
Design Example: Count the Number of Ones in a Register

- A system that is to count the number of 1's in a word of data



ASMD Chart





State Table

Multiplexer Input Conditions for Design Example

Present State		Next State		Input Conditions	Multiplexer Inputs	
G_1	G_0	G_1	G_0		MUX1	MUX2
0	0	0	0	*Start'*		
0	0	0	1	*Start*	0	*Start*
0	1	0	0	*Zero*		
0	1	1	0	*Zero'*	*Zero'*	0
1	0	1	1	None	1	1
1	1	1	0	*E'*		
1	1	0	1	*E*	*E'*	*E*

Control Implementation