Sequencing and Control

Instructor: Saraju P. Mohanty

PART 1

- The control units
- Algorithm State Machines
- Some Design Examples
- Hardwired Vs Microprogrammed control
- Hardwired control design

Sources

- Logic and Computer Design Fundamentals by M. M. Mano and C. R. Kime.
- Dr. Valavanis lectures

The Control Unit

- Binary information stored in a digital computer can be classified as either data or control information.
- Register timing in a synchronous digital system is controlled by a master clock generator.
- Clock pulses are applied to all ff and registers in the system, including those in the control unit.
- To prevent continuous clock pulses from changing the state of all registers on every clock cycle, some registers have a load control signal that enables and disables the loading of the register.
- The control unit that generates the signals for sequencing the operations in the datapath is a sequential circuit with states that dictate the control signals for the system.
- Using status conditions and control inputs the sequential control unit determines the next state in which additional microoperations are activated.

The Control Unit …

- Based on the overall system design, there are two distinct types of control units used in digital systems, one for a programmable system and the other for a nonprogrammable system.
- In a programmable system, a portion of the input to the processor consists of a sequence of instructions. Each instruction specifies the operation the system is to perform.
- Instructions are stored in memory (RAM or ROM).
- To execute instructions in sequence, necessary to provide the address in memory of the instruction to be executed. Address comes from a register called the program counter (PC).
- Executing an instruction means activating the necessary sequence of microoperations in the datapath required to perform the operation specified by the instruction.

Algorithm State Machines (ASM)

- Data-processing task can be defined by register transfer operations controlled by a sequencing mechanism. Such a task can be specified as a hardware algorithm that consists of a finite number of procedural steps which perform the data processing task.
- A flowchart is a convenient way to specify a sequence of procedural steps and decision paths for an algorithm.
- A special flowchart is used, called algorithmic state machine (ASM) chart.
- A state machine is another term for a sequential circuit.
- The ASM resembles a conventional flowchart, but there are differences in terms of interpretation (ASM also describes the timing relationship between the states).

Algorithm State Machines: ASM chart elements

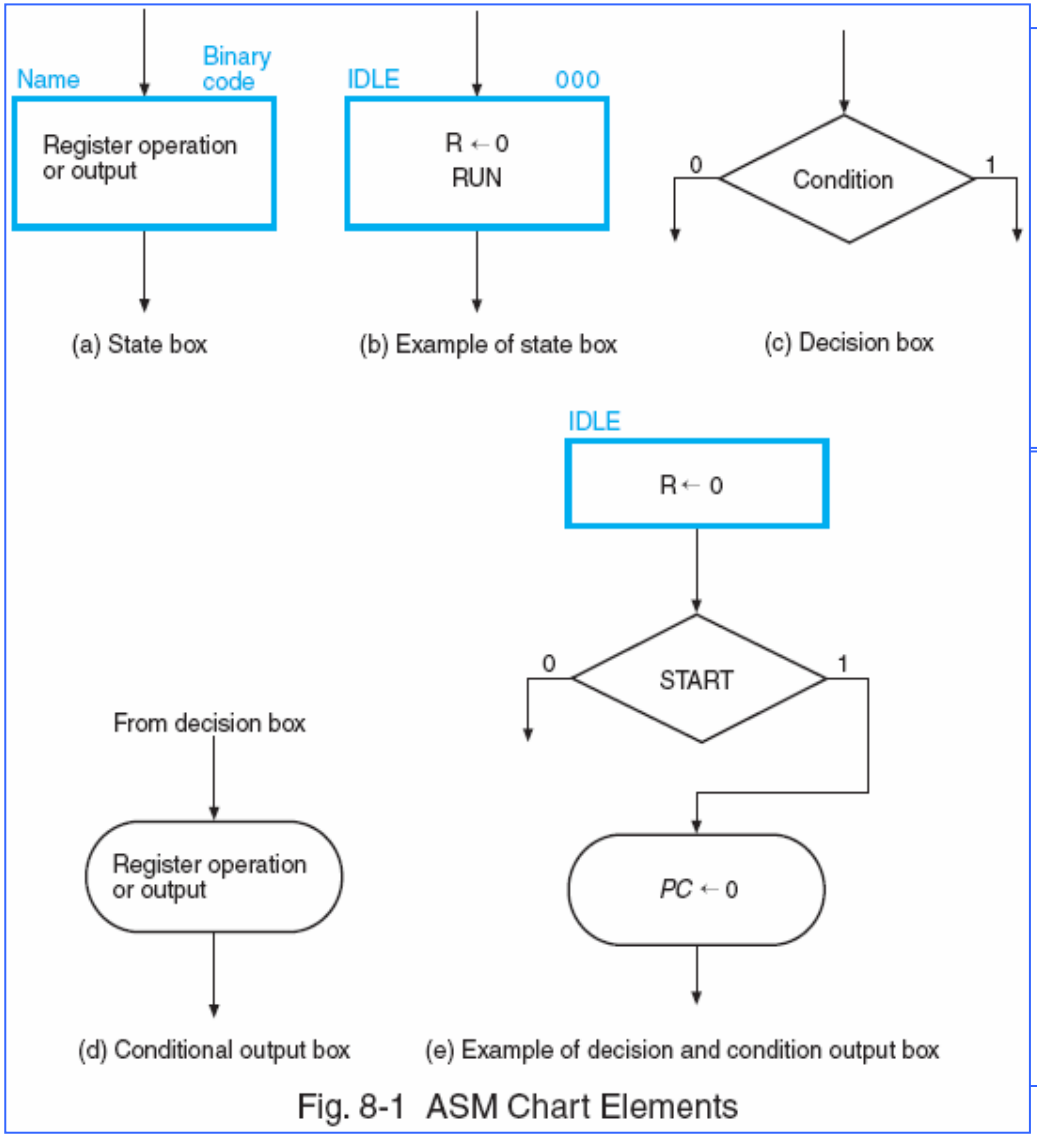


Fig. 8-1 ASM Chart Elements

Three basic elements

- State box
- decision box
- conditional output box

Register transfer indicates that the output signal RUN is to be 1 during the time that the control is in state IDLE. RUN is 1 for any state box in which it appears and it is 0 for any state box in which it does not appear.

Algorithm State Machines: ASM block

An ASM block consists of one state box and all of the decision and conditional output boxes connected between the state box exit and entry paths to the same or other state boxes.

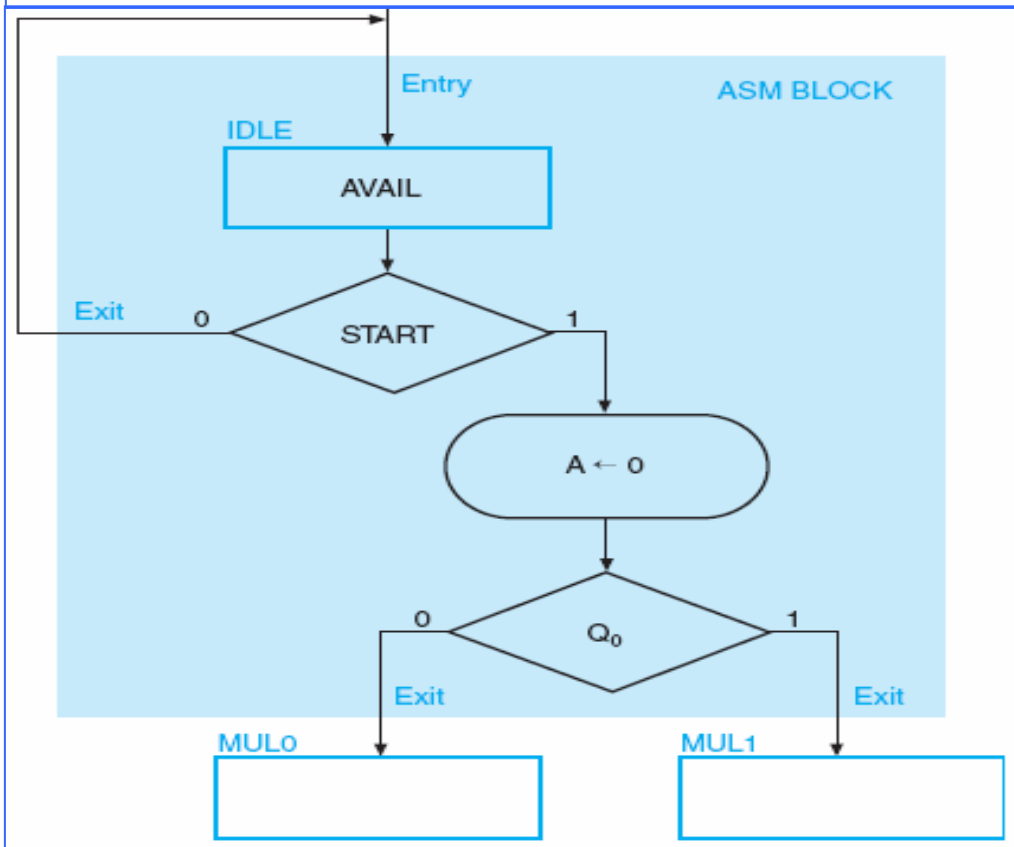


Fig. 8-2 ASM Block

- The ASM chart is a really a form of state diagram for the sequential circuit part of the control unit.
- Each state box is equivalent to a node in the state diagram.
- The decision box is equivalent to input variable.
- Outputs of the register transfer and boxes are equivalent to the output of the sequential circuit.

Algorithm State Machines: Timing Considerations

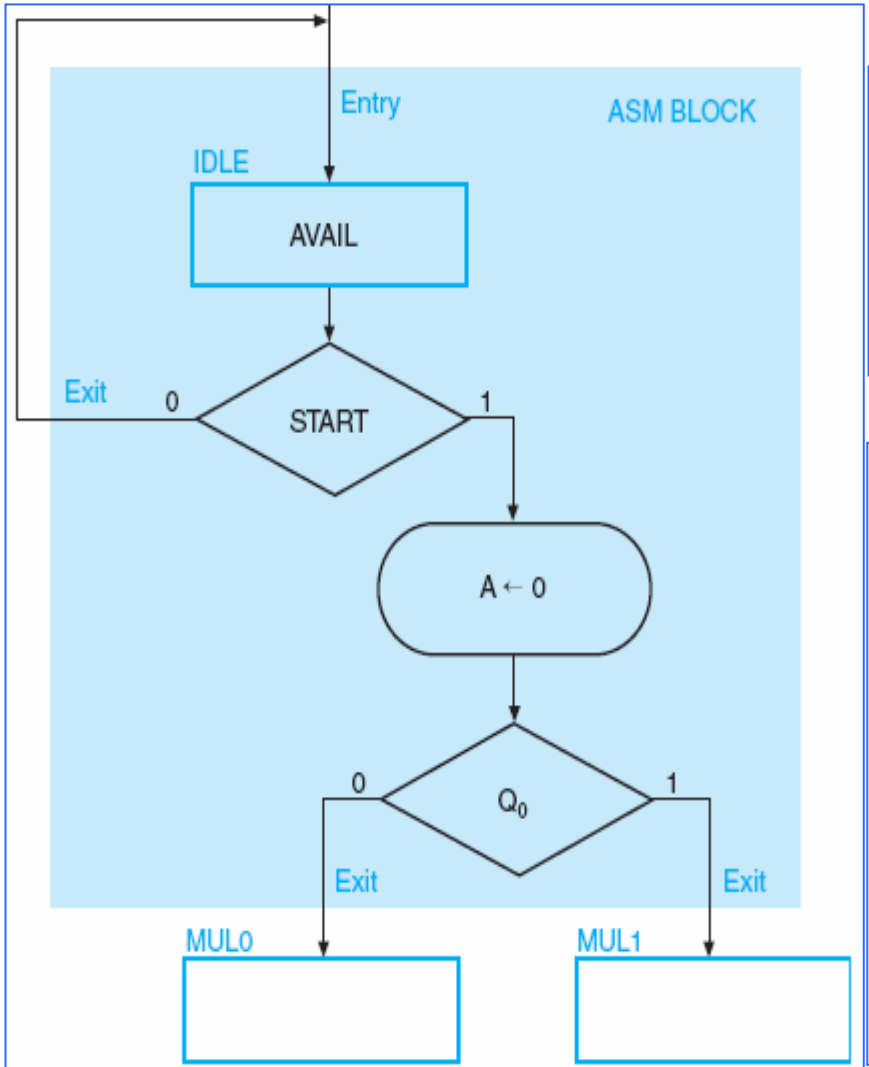


Fig. 8-2 ASM Block

- The timing of the events related to state IDLE is shown.
- All the FFs are assumed to be positive edge-triggered.

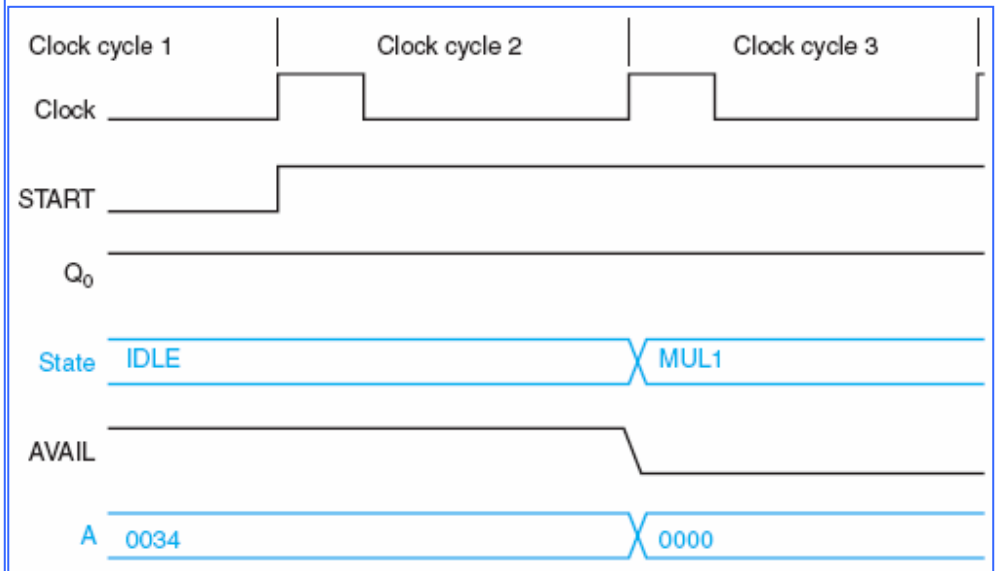


Fig. 8-3 ASM Timing Behavior

Design example: Binary multiplier

Multiply two unsigned numbers, n-bits each. The product can have up to 2n bits.

23	10111	Multiplicand
19	10011	Multiplier
	10111	
	10111	
	00000	
	00000	
	10111	
437	110110101	Product

Fig. 8-4 Hand Multiplication Example

Design example: Binary multiplier …

With digital hardware: Instead of having digital circuit that adds n binary numbers simultaneously, it is less expensive to provide a circuit that sums only two numbers. So each time a copy of the multiplicand or 0's are determined to enter into the addition, they are immediately added to a partial product, stored in a register in preparation for the shift action to follow.

Instead of shifting copies of the multiplicand to the left, partial product formed is shifted to the right – leaves partial product and the copy of multiplicand in the same relative position as the left shift. More important, adder is needed for only n bit positions instead of 2n bit positions. Addition always takes place in the same n position.

When corresponding bit in multiplier is 0, no need to add all 0's.

Design example: Binary multiplier …

23	10111	Multiplicand
19	10011	Multiplier
	00000	Initial partial product
	10111	Add multiplicand, since multiplier bit is 1
	10111	Partial product after add and before shift
	010111	Partial product after shift
	10111	Add multiplicand, since multiplier bit is 1
	1000101	Partial product after add and before shift[a]
	1000101	Partial product after shift
	01000101	Partial product after shift
	001000101	Partial product after shift
	10111	Add multiplicand, since multiplier bit is 1
	110110101	Partial product after add and before shift
437	0110110101	Product after final shift

a. Note that overflow temporarily occurred.

Fig. 8-5 Hardware Multiplication Example

Design example: Binary multiplier …

Multiplier Datapath: Counter P requires [$\log_2 n$] bits in order to count processing of n bits of multiplier. Multiplicand is loaded into register B from IN, multiplier into Q from IN , partial product is formed in A and stored in A and Q. Dual use of Q is possible because we use a right shift of the multiplier in Q to examine each successive multiplier bit. Right shift vacates space one bit at a time in Q. This space accepts the lower part of the partial product from A as it is generated. N-bit binary adder is used for adding B to A. C FF stores C_{out} from the addition and it is reset to 0 during right shift.

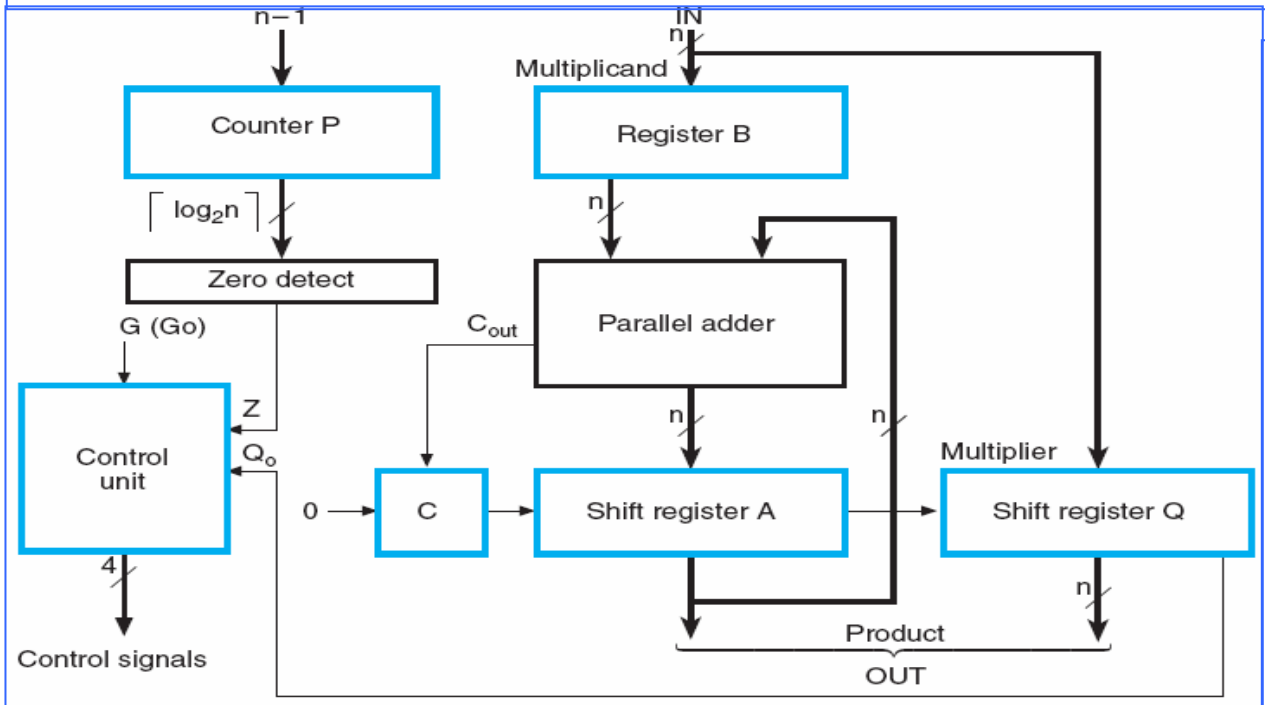


Fig. 8-6 Block Diagram for Binary Multiplier

To count # of add-shift or shift, P is provided. Initially set to n-1 and counted down after the formation of each partial product.

Unit in initial state until G becomes 1, then system performs multiplication.

Design example: Binary multiplier …
(Basic components)

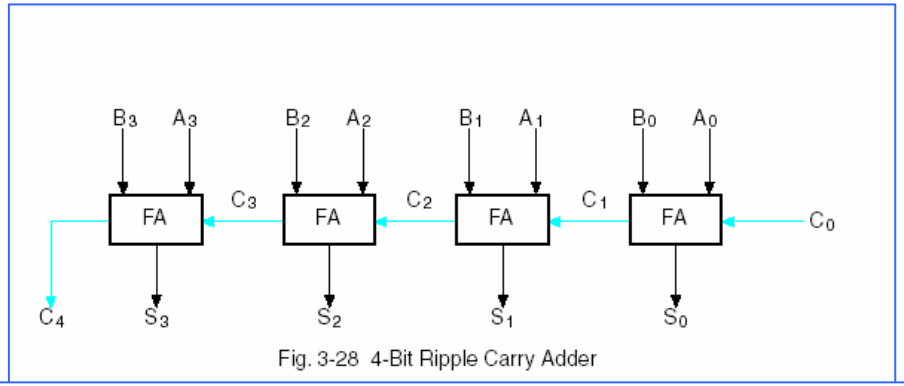


Fig. 3-28 4-Bit Ripple Carry Adder

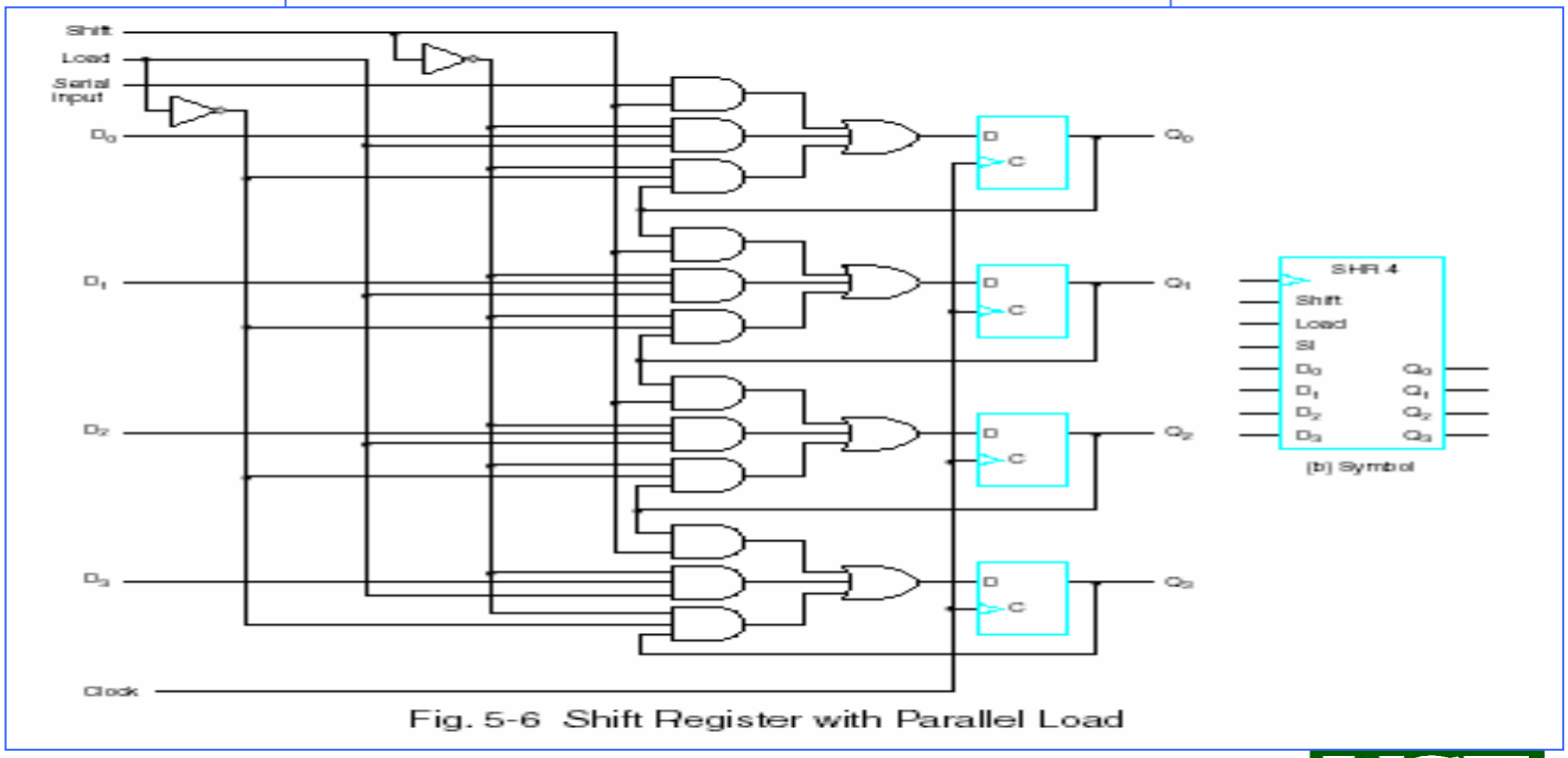


Fig. 5-6 Shift Register with Parallel Load

Design example: Binary multiplier …
(Basic components ….)

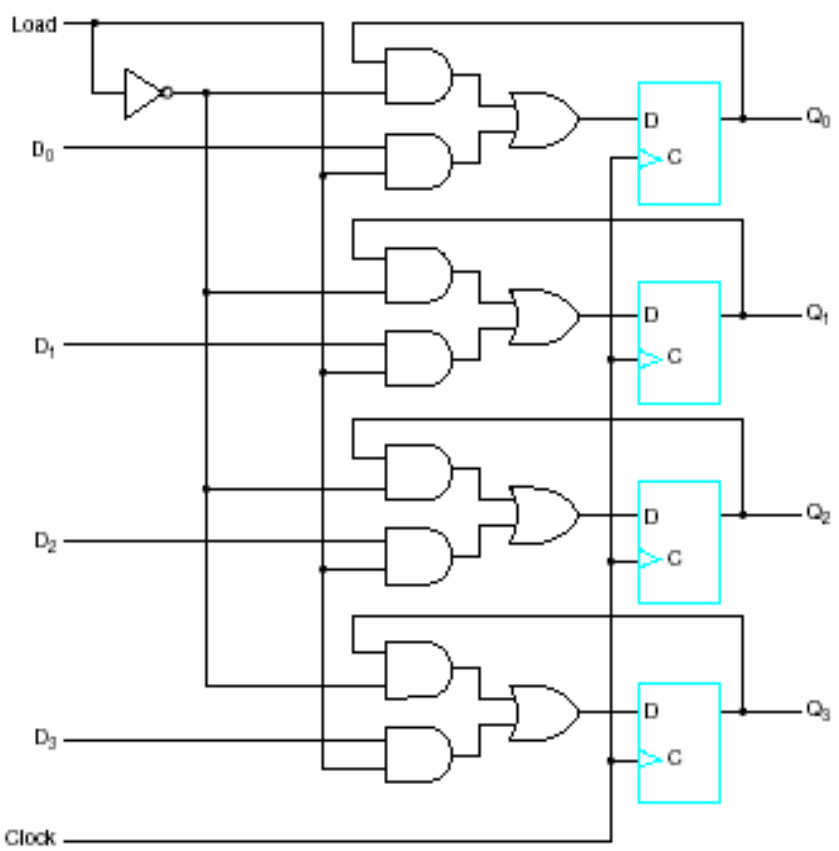
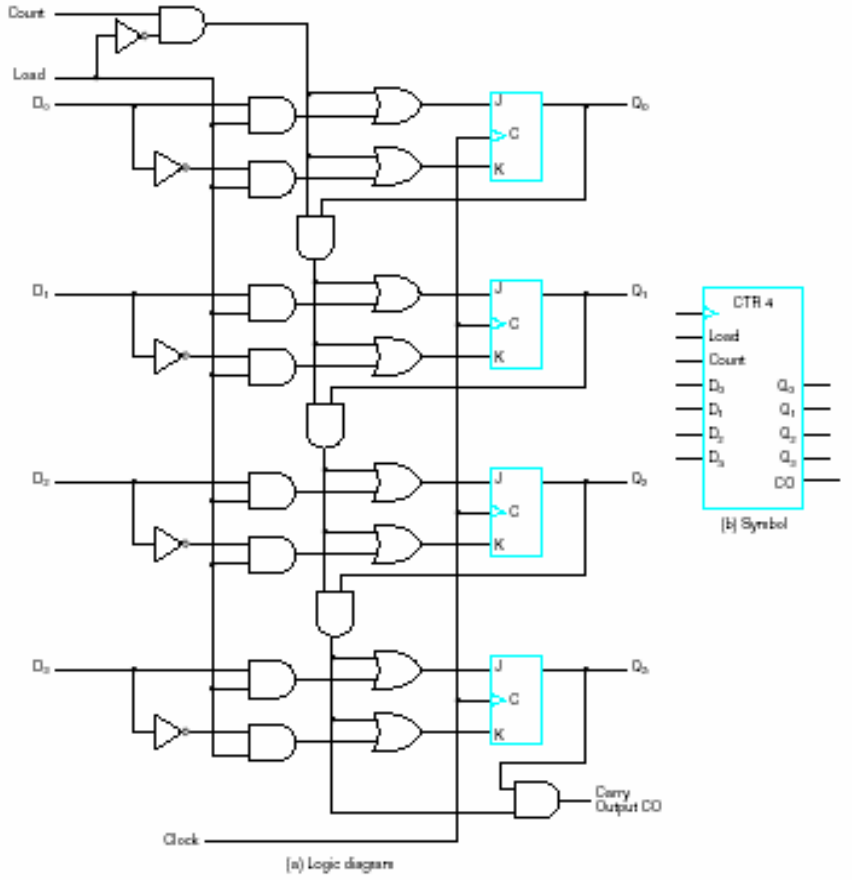


Fig. 5-2 4-Bit Register with Parallel Load



c.

Fig. 5-12 4-Bit Binary Counter with Parallel Load

Design example: Binary multiplier …
(ASM chart for multiplier)

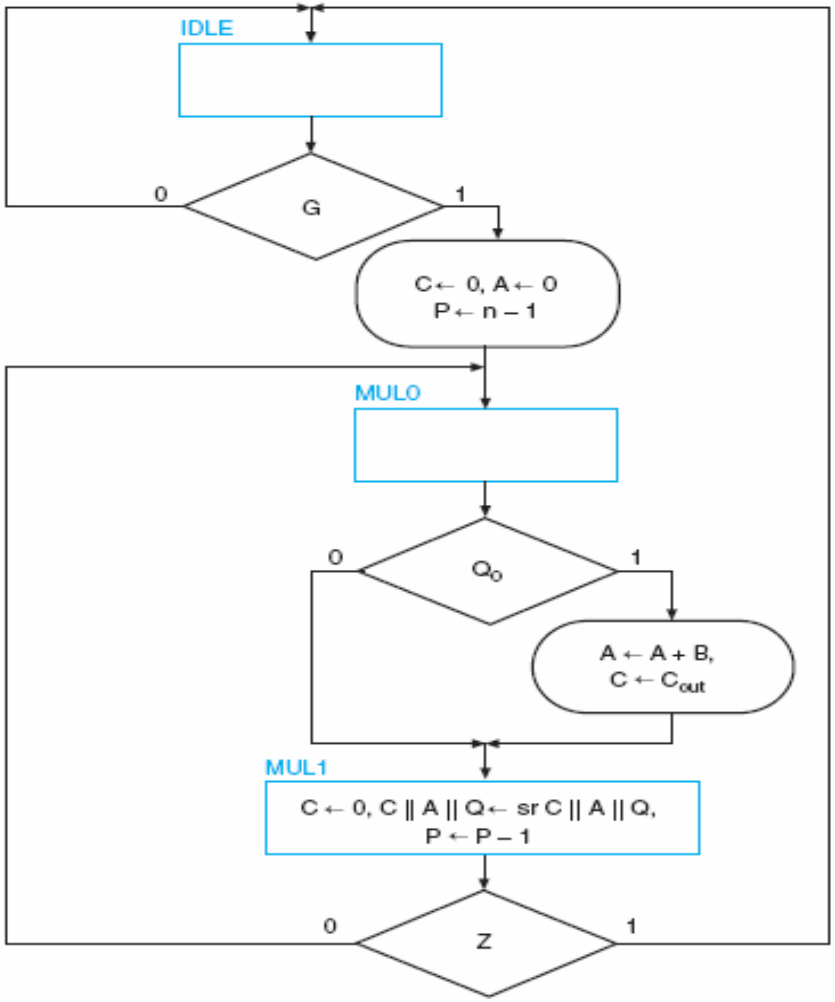


Fig. 8-7 ASM Chart for Binary Multiplier

Initially multiplicand in B, multiplier in Q. When ASM in state IDLE and G=0 no action. Process starts when G=1. ASM moves from IDLE to MUL0, C and A are cleared to 0, P loaded with constant n-1. In MUL0 decision is made based on Q_0, the LSB of Q. If 1 contents of B added to those of A with result transferred to A and carry to C. If 0, A and bit C unchanged. In both cases, next state is MUL1.

In MUL1 a right shift is performed on combined contents of C, A, Q. Shift expressed by five simultaneous transfers

C←0, A(n-1)←C, A←sr A, Q(n-1)←A(0), Q← sr Q

Define combined register C||A||Q, re-write then as C||A||Q ← sr C||A||Q

Implementing Control Units

- Two basic approaches of implementing control unit:
  - Hardwired
  - Microprogramming
- Hardwired control: Less costly for small control units, offers faster speed and useful for high-performance systems. Design involves a finite state machine, which is implemented using combinational logic, registers, etc..
- Microprogramming control: More useful for complex instructions, slower than the hardwired control. The control unit is designed as a program that implements the machine instruction in terms of simpler microinstructions.

Design example: Binary multiplier … (Hardwired Control)

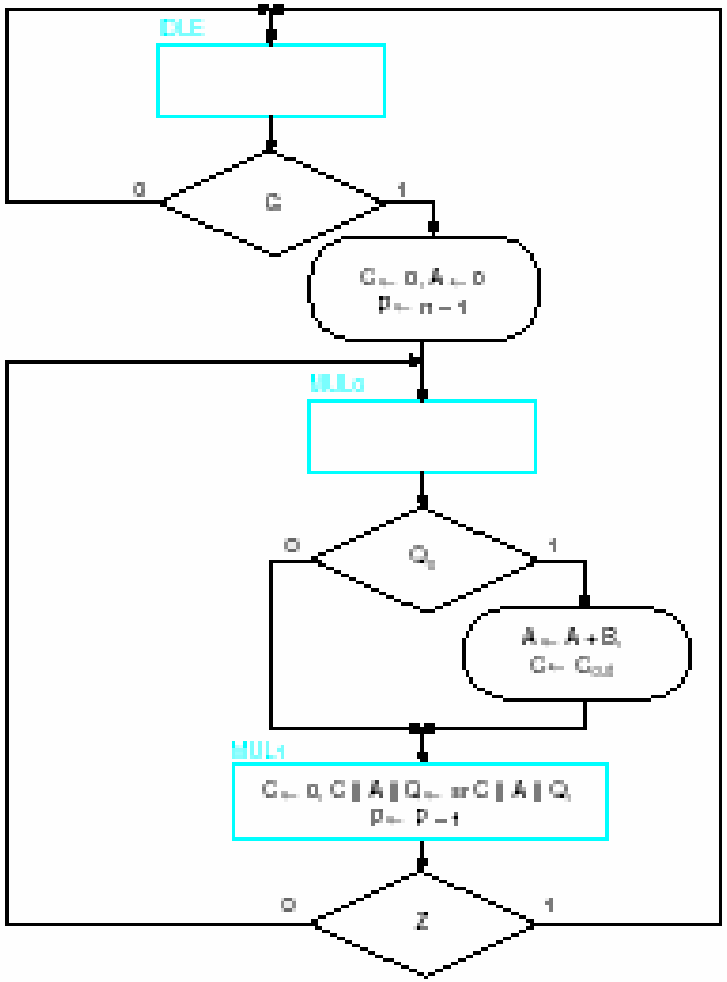


Fig. 8-7 ASM Chart for Binary Multiplier

Block Diagram Module	Microoperation	Control Signal Name	Control Expression
Register A:	$A \leftarrow 0$	Initialize	$IDLE \cdot G$
	$A \leftarrow A + B$	Load	$MUL0 \cdot Q_0$
	$C \| A \| Q \leftarrow sr\ C \| A \| Q$	Shift_dec	MUL1
Register B:	$B \leftarrow IN$	Load_B	LOADB
Flip-Flop C:	$C \leftarrow 0$	Clear_C	$IDLE \cdot G + MUL1$
	$C \leftarrow C_{out}$	Load	—
Register Q:	$Q \leftarrow IN$	Load_Q	LOADQ
	$C \| A \| Q \leftarrow sr\ C \| A \| Q$	Shift_dec	—
Counter P:	$P \leftarrow n-1$	Initialize	—
	$P \leftarrow P-1$	Shift_dec	—

Table 8-1 Control Signals for Binary Multiplier

Dividing the ASM to two parts

- A table that defines the control signals
- A simplified ASM chart

Design example: Binary multiplier … (Hardwired Control ….)

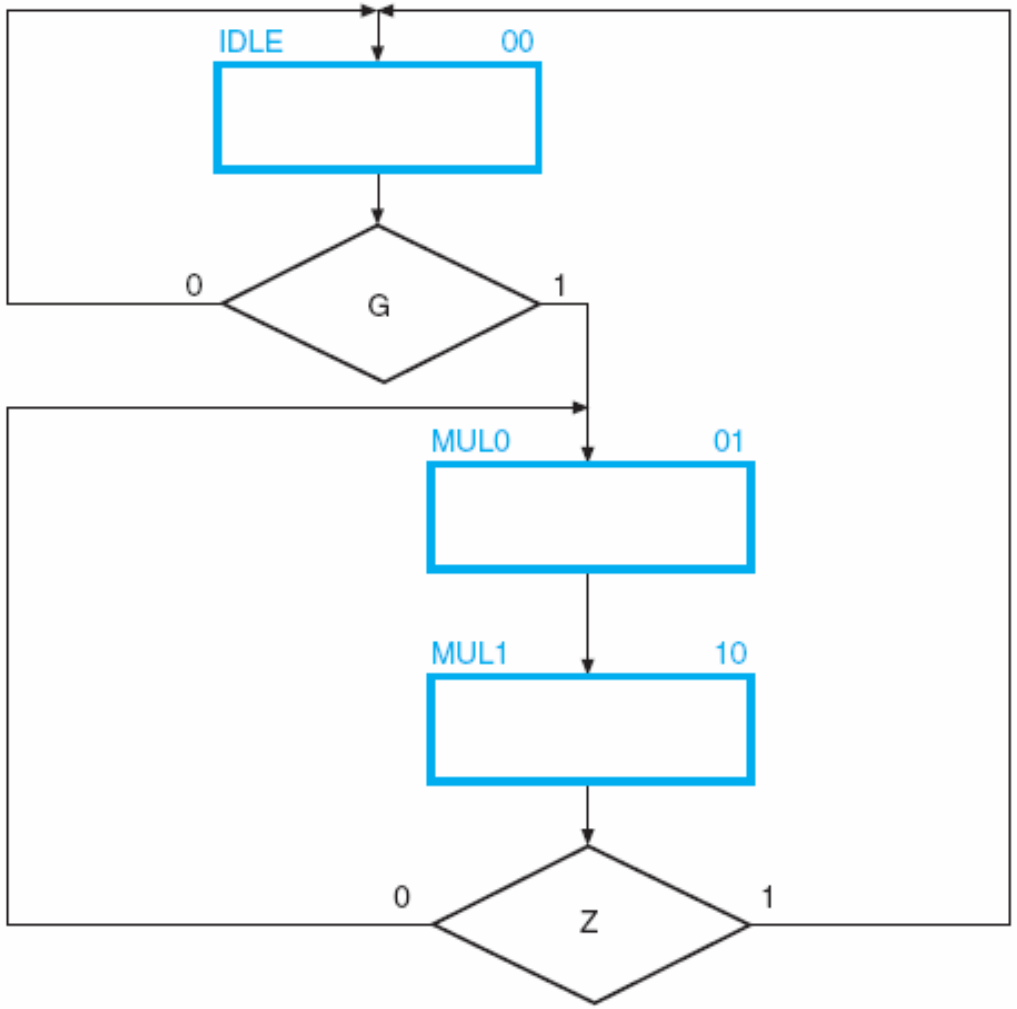


Fig. 8-8 Sequencing Part of ASM Chart for the Binary Multiplier

State diagram of a sequential circuit without outputs specified, except that representations used in diagram are different.

It may be difficult to perform such separation for large circuits. So two techniques used: (1) Sequence register and decoder method (2) One Flip-Flop per state method

Design example: Binary multiplier …
(Hardwired Control: Sequence Register and Decoder)

Present state			Inputs		Next state		Decoder Outputs		
Name	M_1	M_0	G	Z	M_1	M_0	IDLE	MUL0	MUL1
IDLE	0	0	0	×	0	0	1	0	0
	0	0	1	×	0	1	1	0	0
MUL0	0	1	×	×	1	0	0	1	0
MUL1	1	0	×	0	0	1	0	0	1
	1	0	×	1	0	0	0	0	1
—	1	1	×	×	×	×	×	×	×

Table 8-2 State Table for Sequence Register and Decoder Part of Multiplier Control Unit

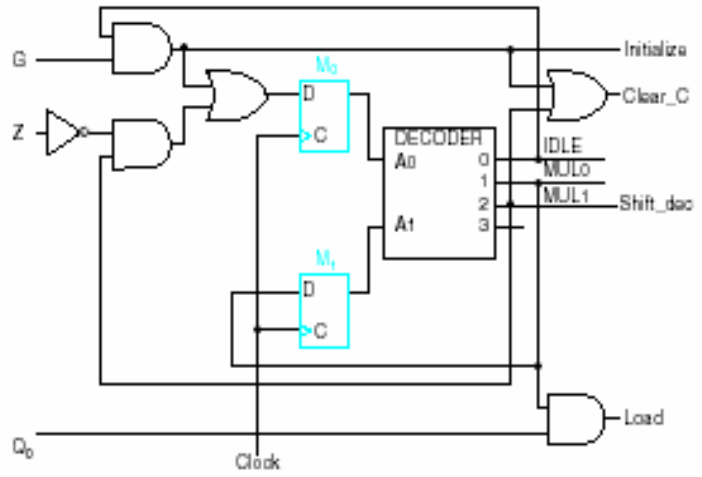


Fig. 8-9 Control Unit for Binary Multiplier Using a Sequence Register and a Decoder

- Two FFs are used for M0 and M1
- Binary states: 00-IDLE, 01-MUL0, and 10-MUL1
- Don't care (X) when the input is not used to determine the next state.
- Outputs of the sequencing par are determined by the state names.
- A 2-to-4-line decoder is used.