CS221: Digital Design

http://jatinga.iitg.ernet.in/~asahu/cs221

RTL Examples with ASMD and FSMD

A. Sahu

Dept of Comp. Sc. & Engg.

Indian Institute of Technology Guwahati

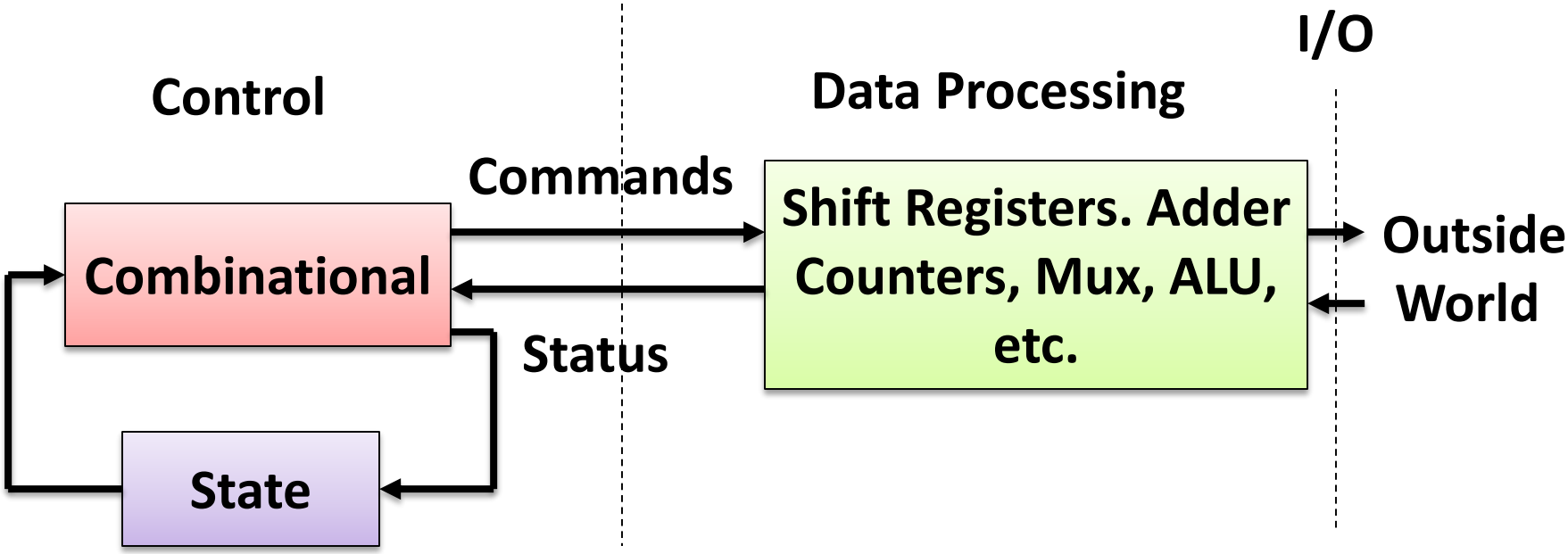
RTL Design

- We need to separate controller & data processor
  - Controller – What actions need to be taken? What is fundamental operating mode?
  - Processor – Undertake the action. Manipulate the data

The ultimate Goal of this course : Design using Control Path + Data Approach : RTL Design

RTL Design

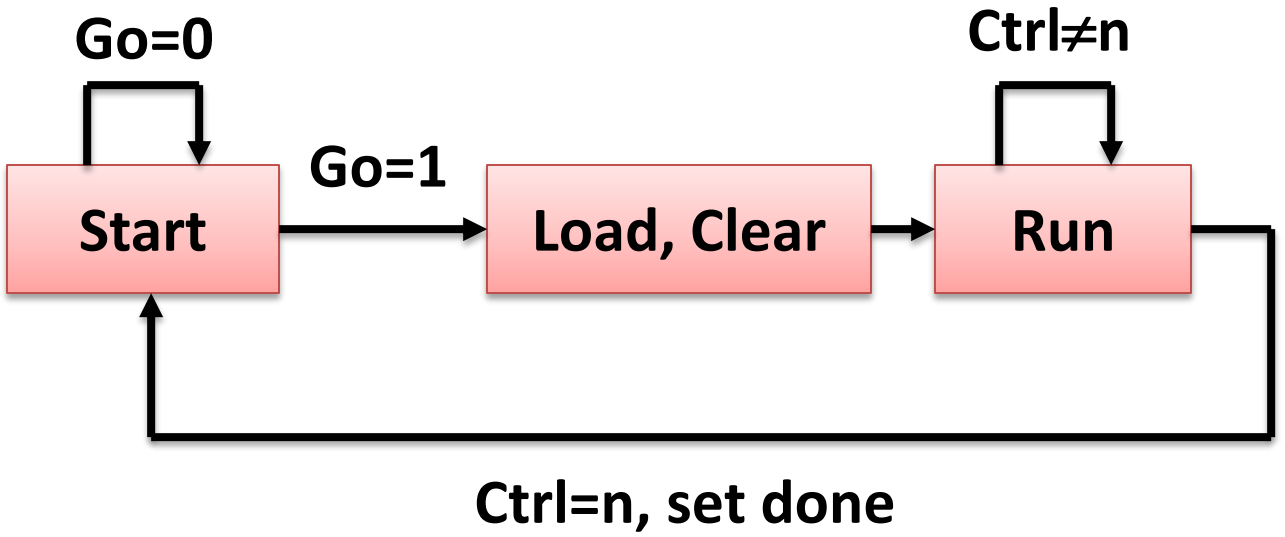
- Control and data path interaction



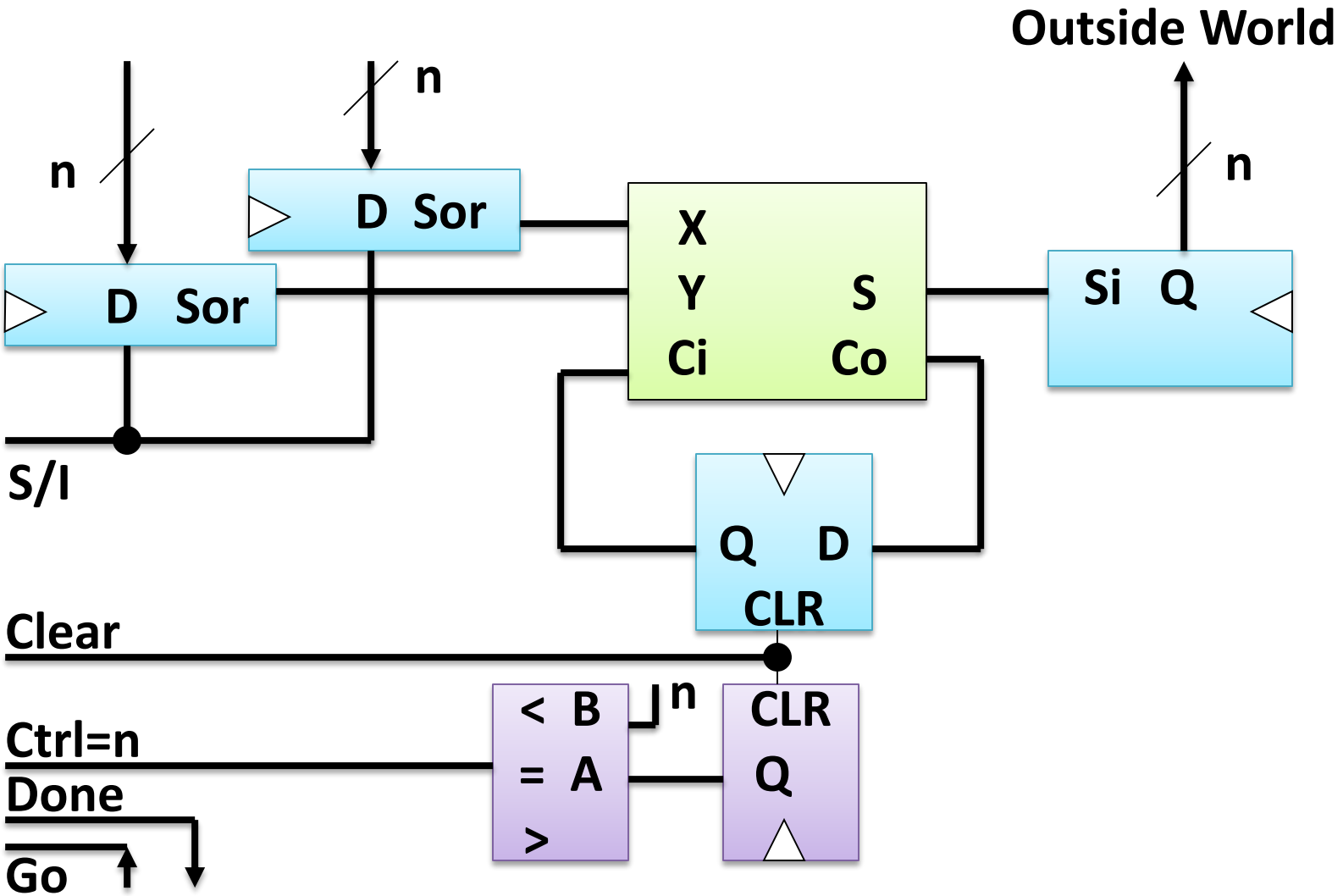
- Our circuit is now explicitly separated

ASM/FSM Overview : High Level

- Ex. Serial Addition
- Control Part/Path



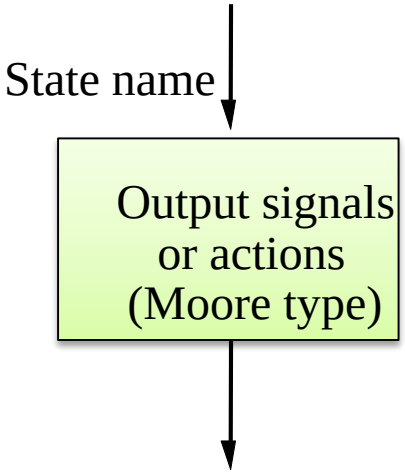
Serial Addition Data Path



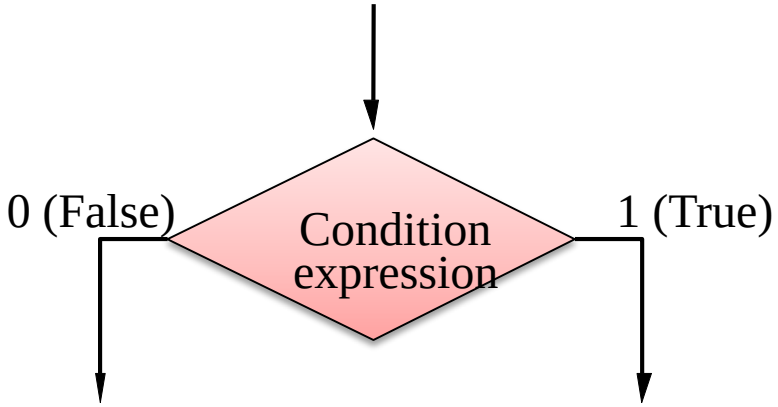
ASM Design

- ASM charts are like flowcharts, with a few crucial differences.
- Be careful, especially with timing.
- Three type components/Box
  - **State Box**
  - **Decision Box**
  - Combinational Box/Transition Box/**Conditional Box**

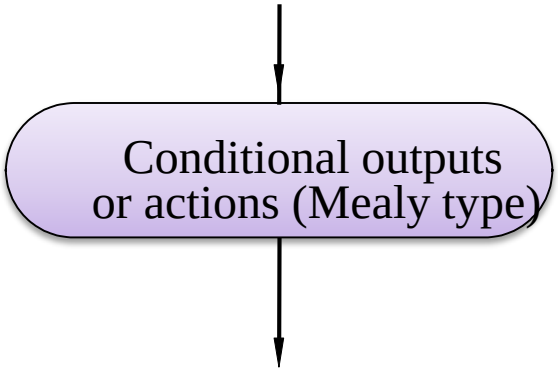
Elements used in ASM charts



(a) State box



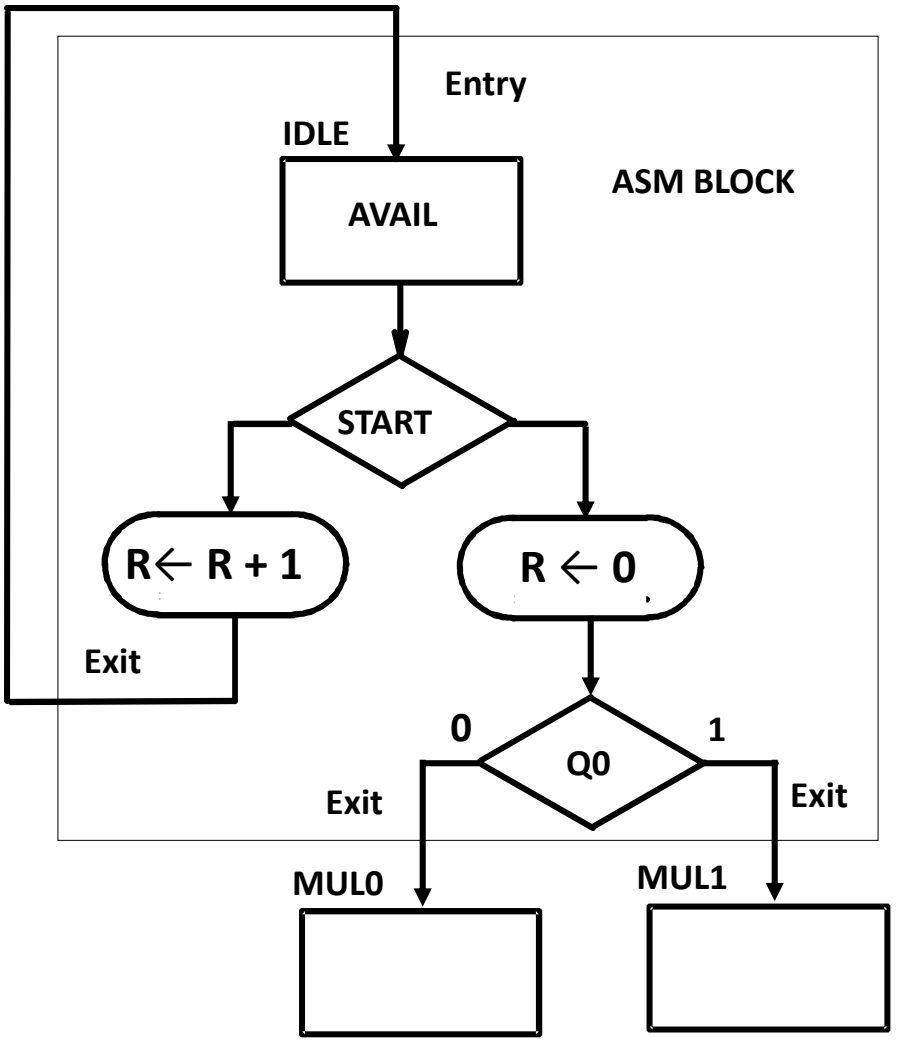
(b) Decision box



(c) Conditional output box

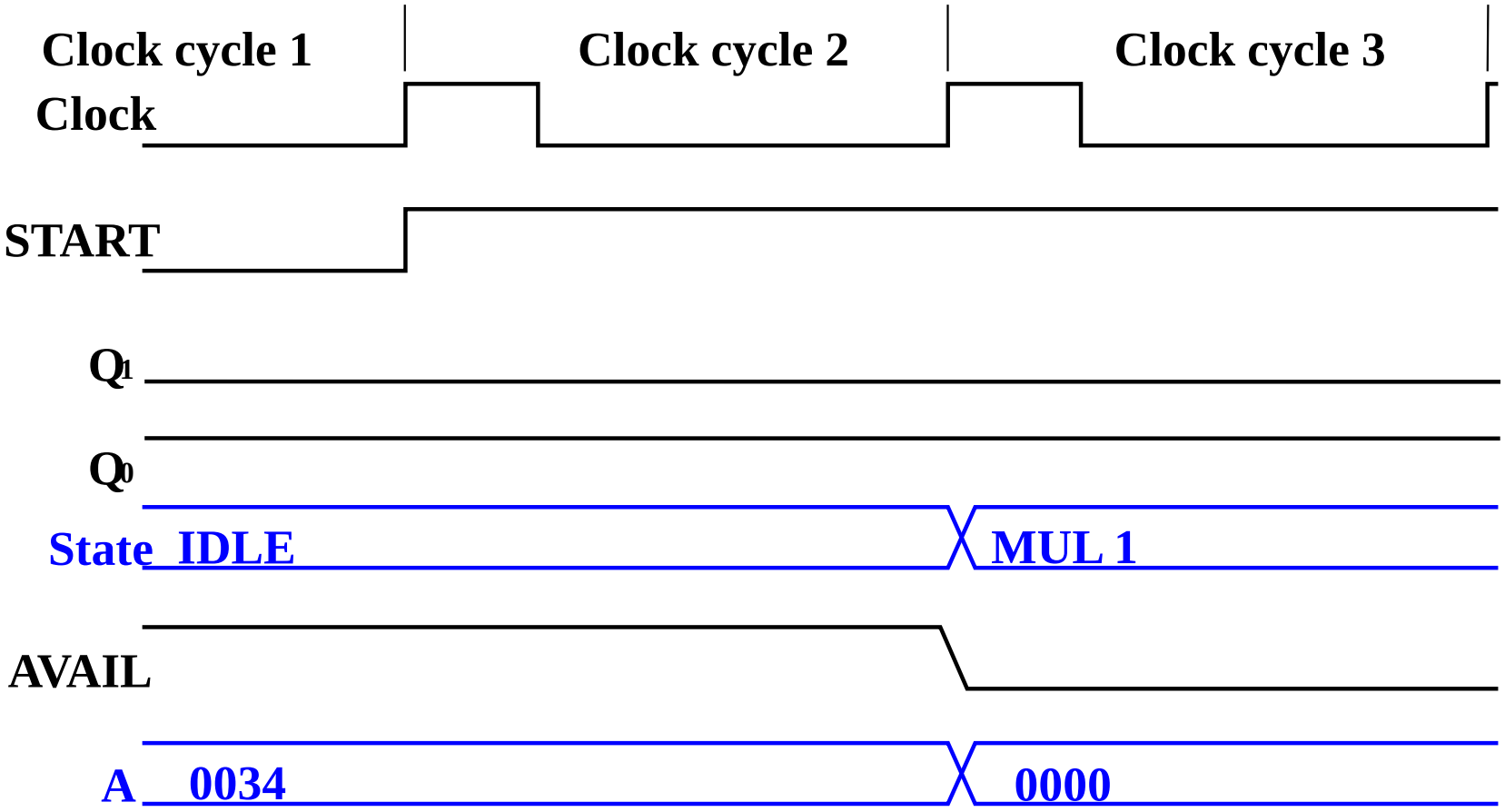
ASM Blocks

One ASM block
execute in one cycle



ASM Timing

- Outputs appear while in the state
- Register transfers occur at the clock while exiting the state - New value occur in the next state!



ASM : DP+CP Example

- **Find the Data Path and ASM for the following problem: Addition of two numbers**
  - We **first** need to load two registers (R1, R2) with some value.
  - We will **then** need to add the two Registers (R1, R2) and save the result in Register R3.
  - All these operations **should occur if** a "**start**" Signal is activated.

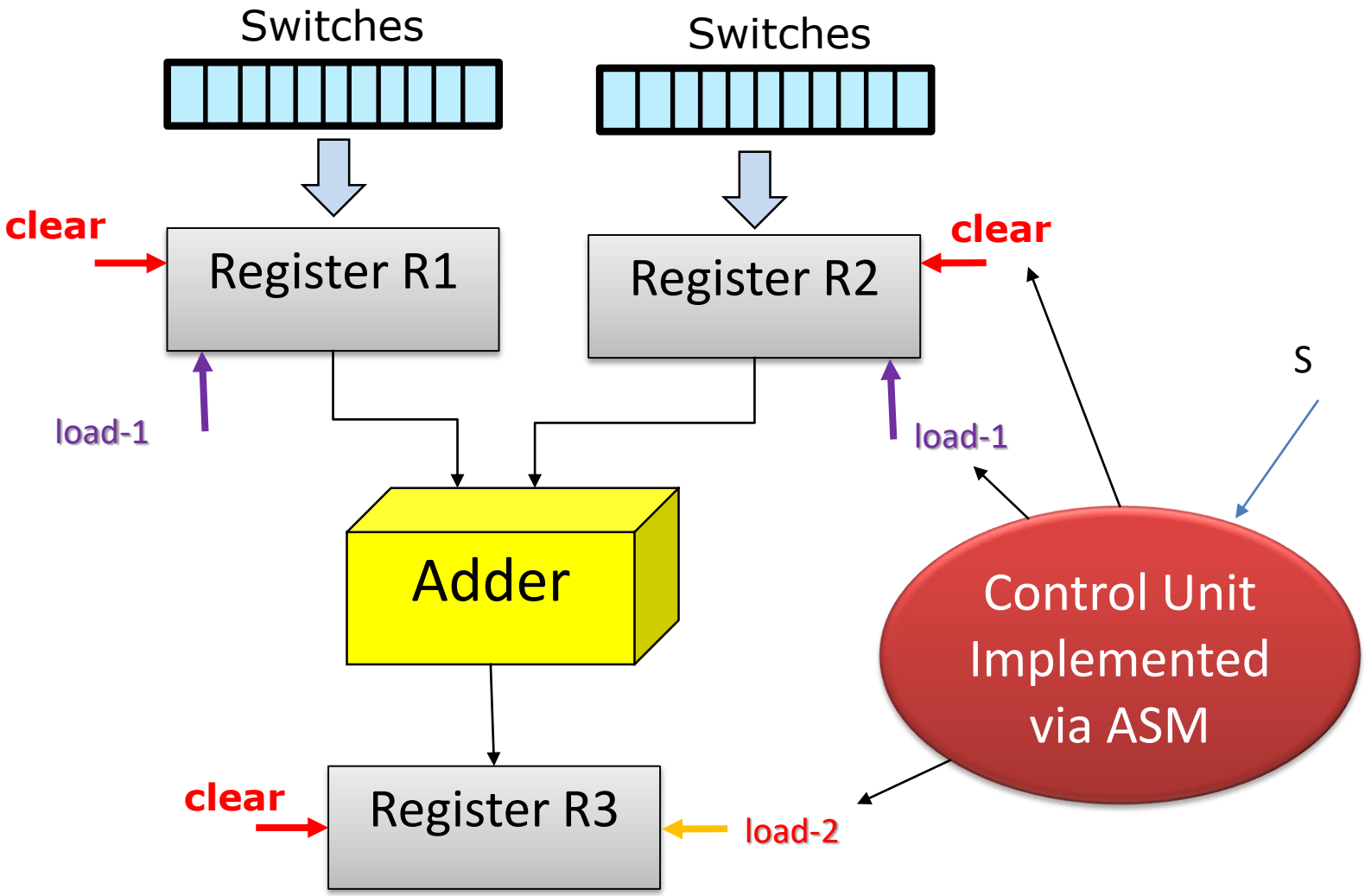
ASM : DP+CP Example

- Translation to Hardware:
  - We need to clear the registers first.
  - If the "**start**" signal is set to 0, I do nothing
  - Else If the "**start**" signal is set to 1, I will **load** R1, R2 with values
  - Next, enable R3 to be loaded (load-2)by the results of R1+R2

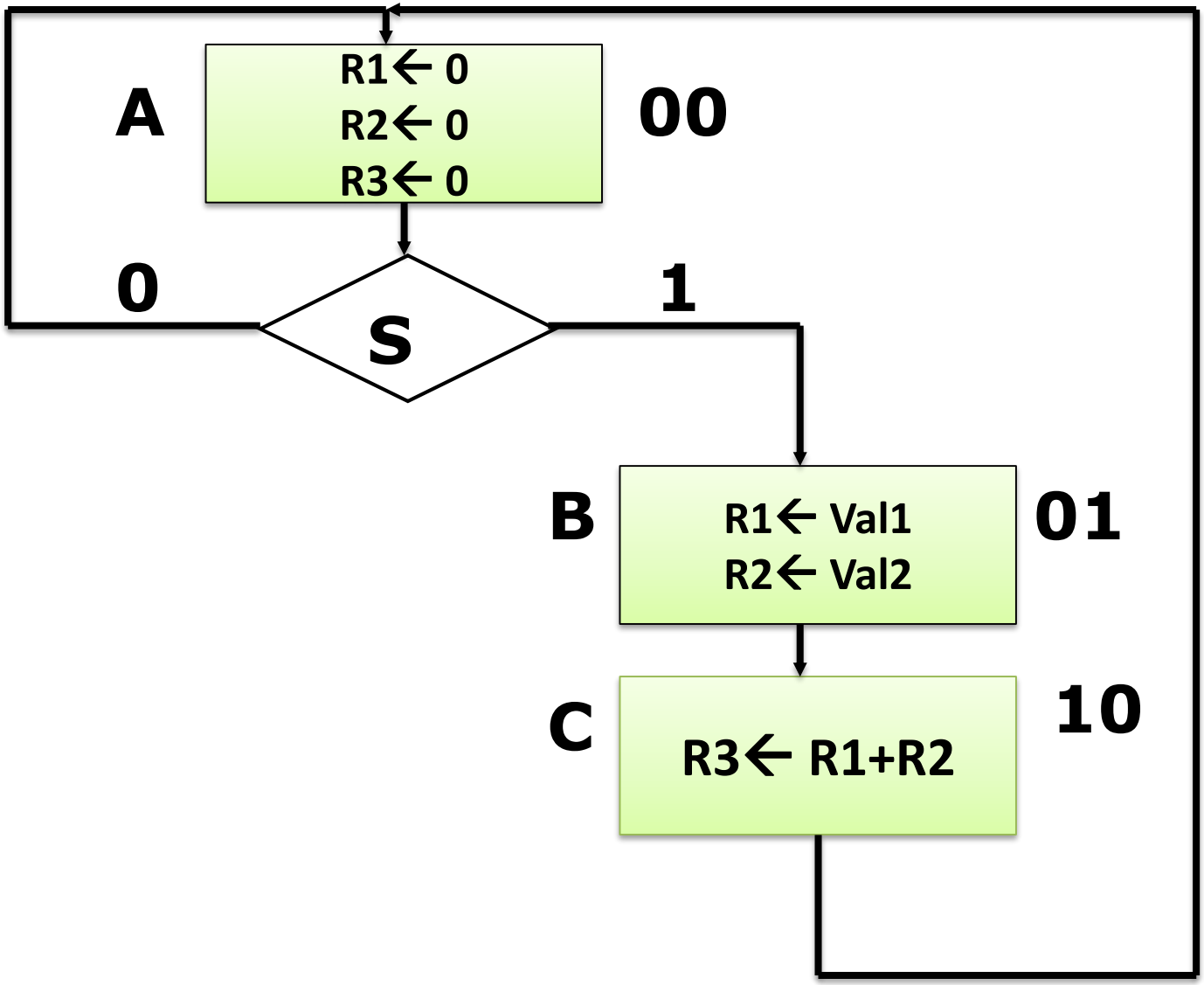
ASM : DP+CP Example

- Inputs/Outputs:
  - **start** is an input signal (Use a switch)
  - **clear** is an output signal generated and connected to R1, R2, R3
  - **load-1** is an output signal generated and connected to R1, R2
  - **load-2** is an output signal generated and connected to R3

Data Path


Switches
Switches
clear
Register R1
Register R2
clear
S
load-1
load-1
Adder
Control Unit
Implemented
via ASM
clear
Register R3
load-2

ASM : DP+CP Example



ASM : DP+CP Example Controller

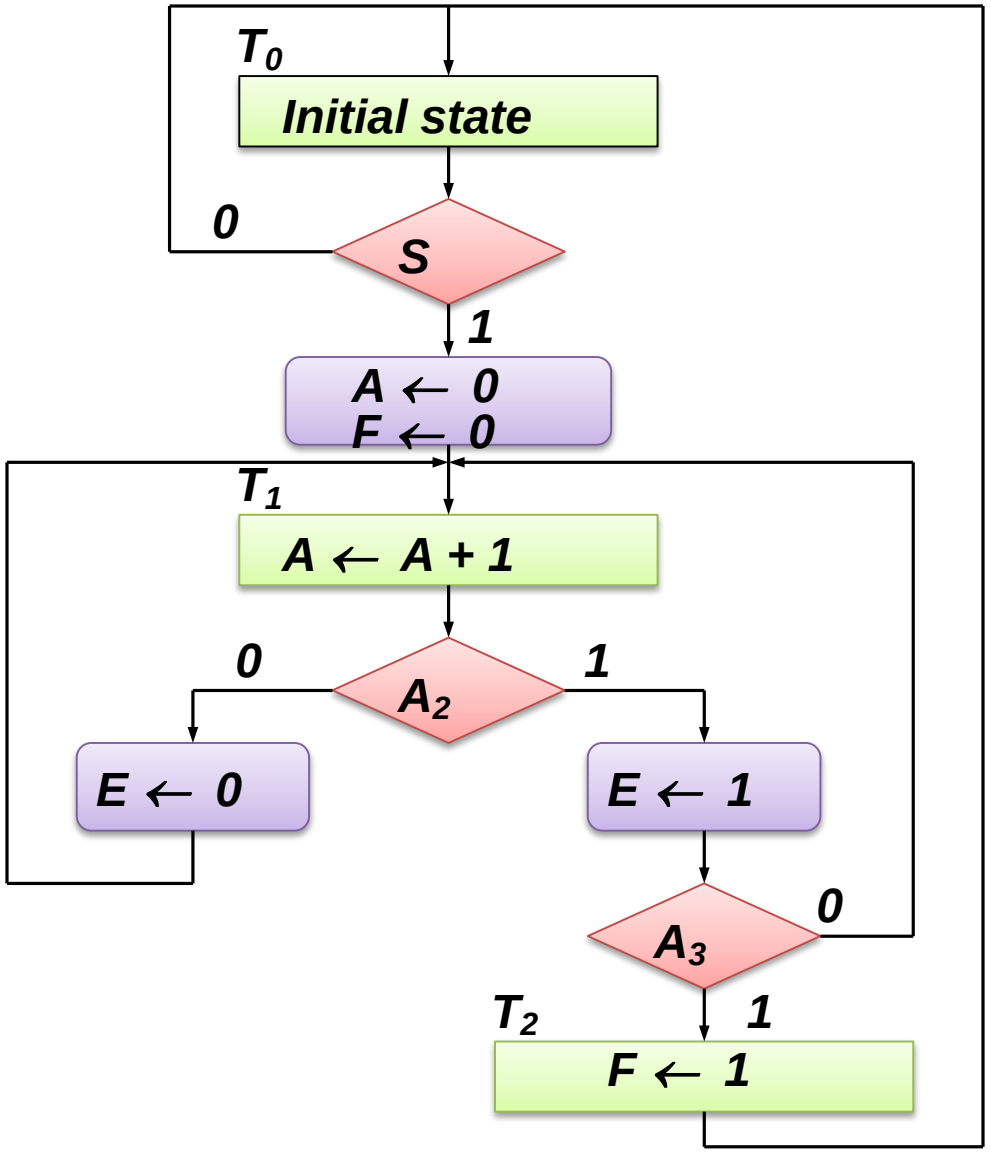
PS		S	NS	
0	0	0	0	0
0	0	1	0	1
0	1	X	1	0
1	0	X	0	0
1	1	X	0	0

PS		CLR	L1	L2
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1
1	1	X	X	X

ASM Charts: An Example

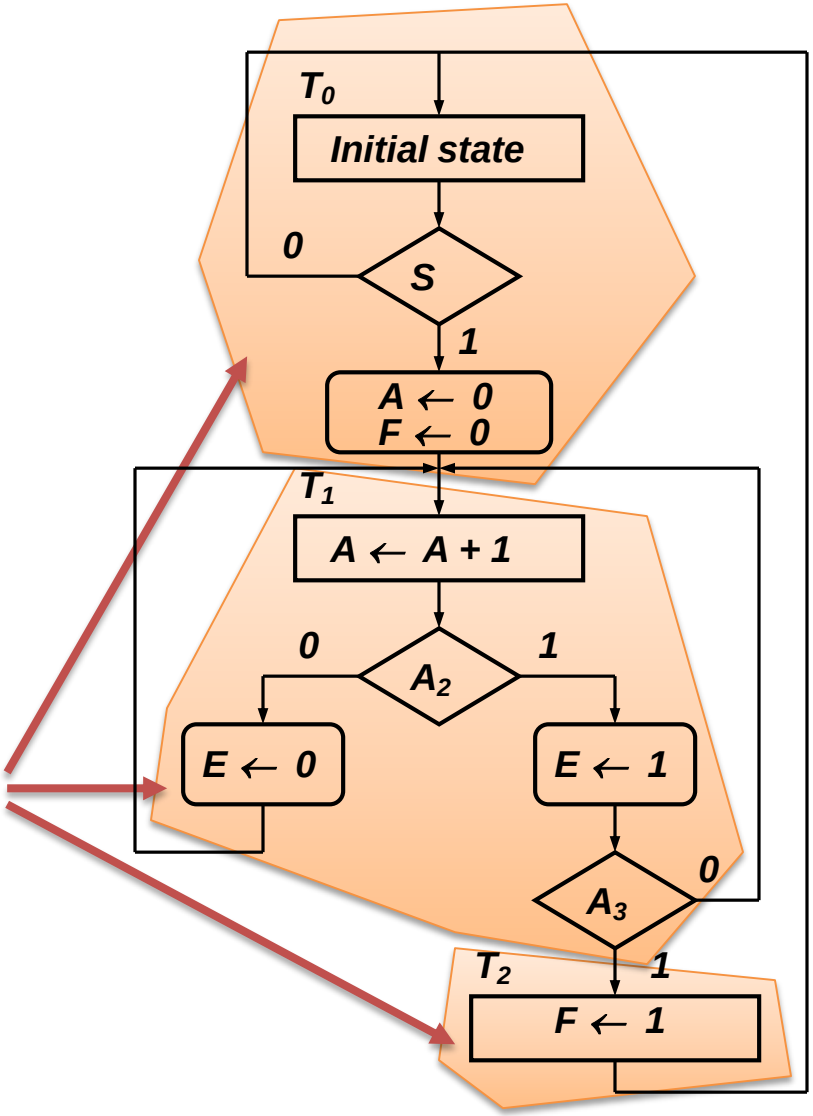
- A is a register;
- A_i stands for i^{th} bit of the A register.

  $A = A_3A_2A_1A_0$
- E and F are single-bit flip-flops.



Timing in ASM Charts

3 ASM blocks

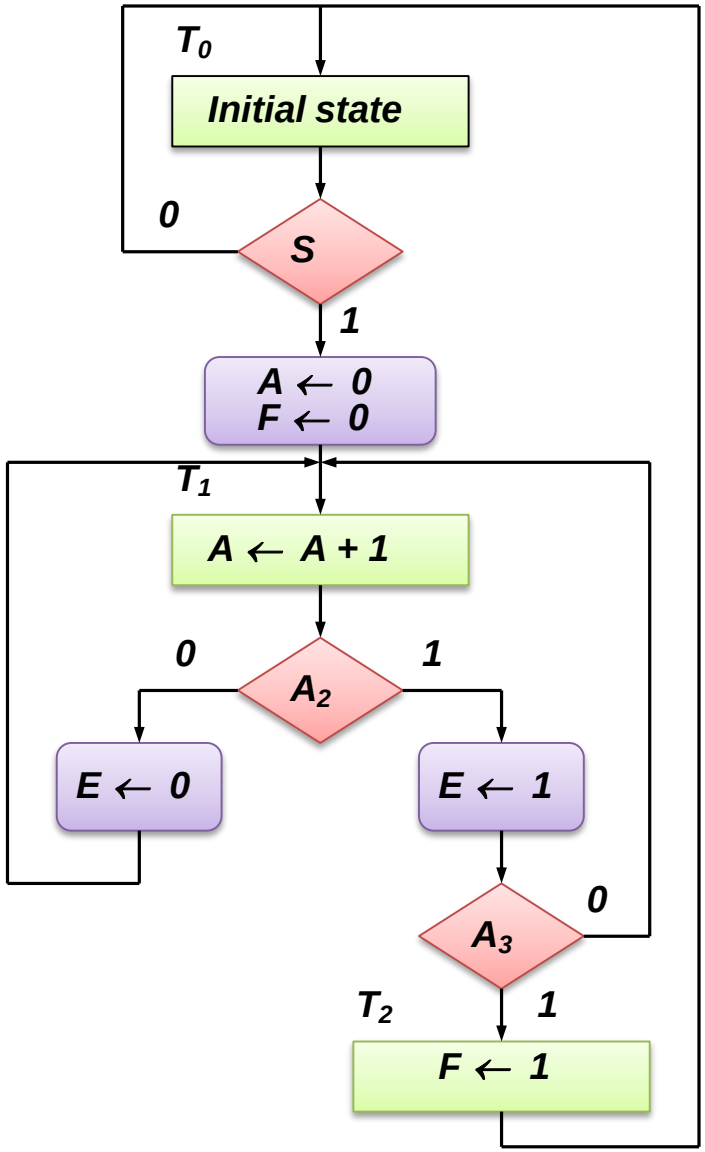


Timing in ASM Charts

- Operations of ASM can be illustrated through a timing diagram.
- Two factors which must be considered are
  - Operations in an ASM block occur at the same time in ***one clock cycle***
  - Decision boxes are dependent on the status of the ***previous clock cycle*** (that is, they do not depend on operations of current block)

Timing in ASM Charts

CTR	E, F	Conditions	State
0000	1,0	A_2=0,A_3=0	T1
0001	0,0	--	--
0010	0,0	---	--
0011	0,0	---	--
0100	0,0	A_2=1,A_3=0	--
0101	1,0	--	--
0110	1,0	--	--
0111	1,0	--	--
1000	1,0	A_2=0,A_3=1	--
1001	0,0	--	--
1010	0,0	--	--
1011	0,0	--	--
1100	0,0	A_2=1,A_3=1	--
1101	1,0		T2
1101	1,1		T0



ASM Chart => Digital System

- ASM chart describes a digital system. From ASM chart, we may obtain:
  - Controller logic (via State Table/Diagram)
  - Architecture/Data Processor
- Design of controller is determined from the decision boxes and the required state transitions.
- Design requirements of data processor can be obtained from the operations specified with the state and conditional boxes.

ASM Chart => Controller

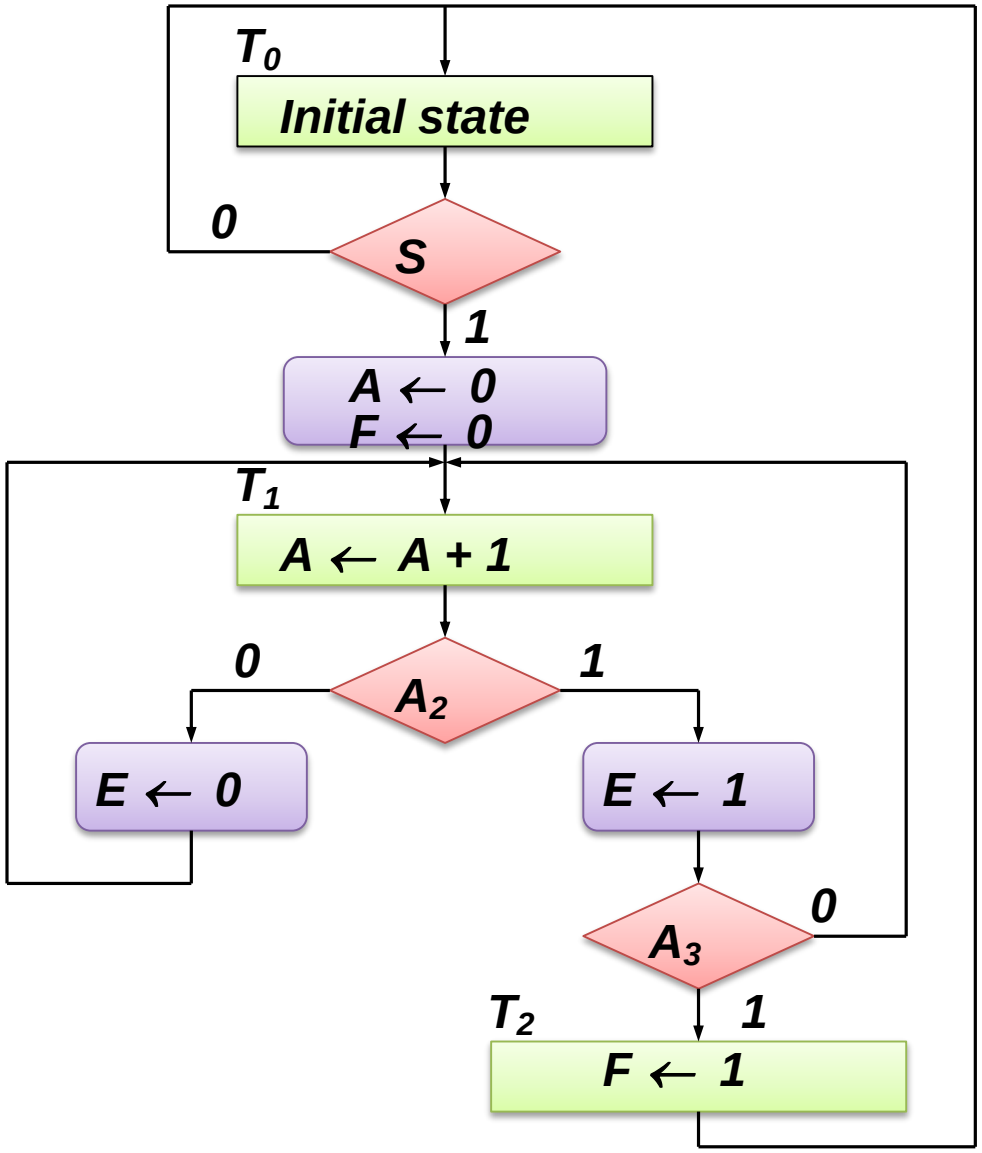
- Procedure:
  - Step 1: Identify all states and assign suitable codes.
  - Step 2: Formulate state table using

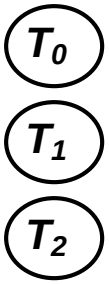
    State from state boxes

    Inputs from decision boxes

    Outputs from operations of state/conditional boxes.
  - Step 3: Obtain state/output equations and draw circuit.

ASM Chart => Controller





Assign codes to states:

$T_0 = 00$

$T_1 = 01$

$T_2 = 11$

Present state		inputs			Next state		outputs		
G_1	G_0	S	A_2	A_3	G_1^+	G_0^+	T_0	T_1	T_2
0	0	0	X	X	0	0	1	0	0
0	0	1	X	X	0	1	1	0	0
0	1	X	0	X	0	1	0	1	0
0	1	X	1	0	0	1	0	1	0
0	1	X	1	1	1	1	0	1	0
1	1	X	X	X	0	0	0	0	1

Inputs from conditions in decision boxes.
Outputs = present state of controller.

ASM Chart => Architecture/Data Processor

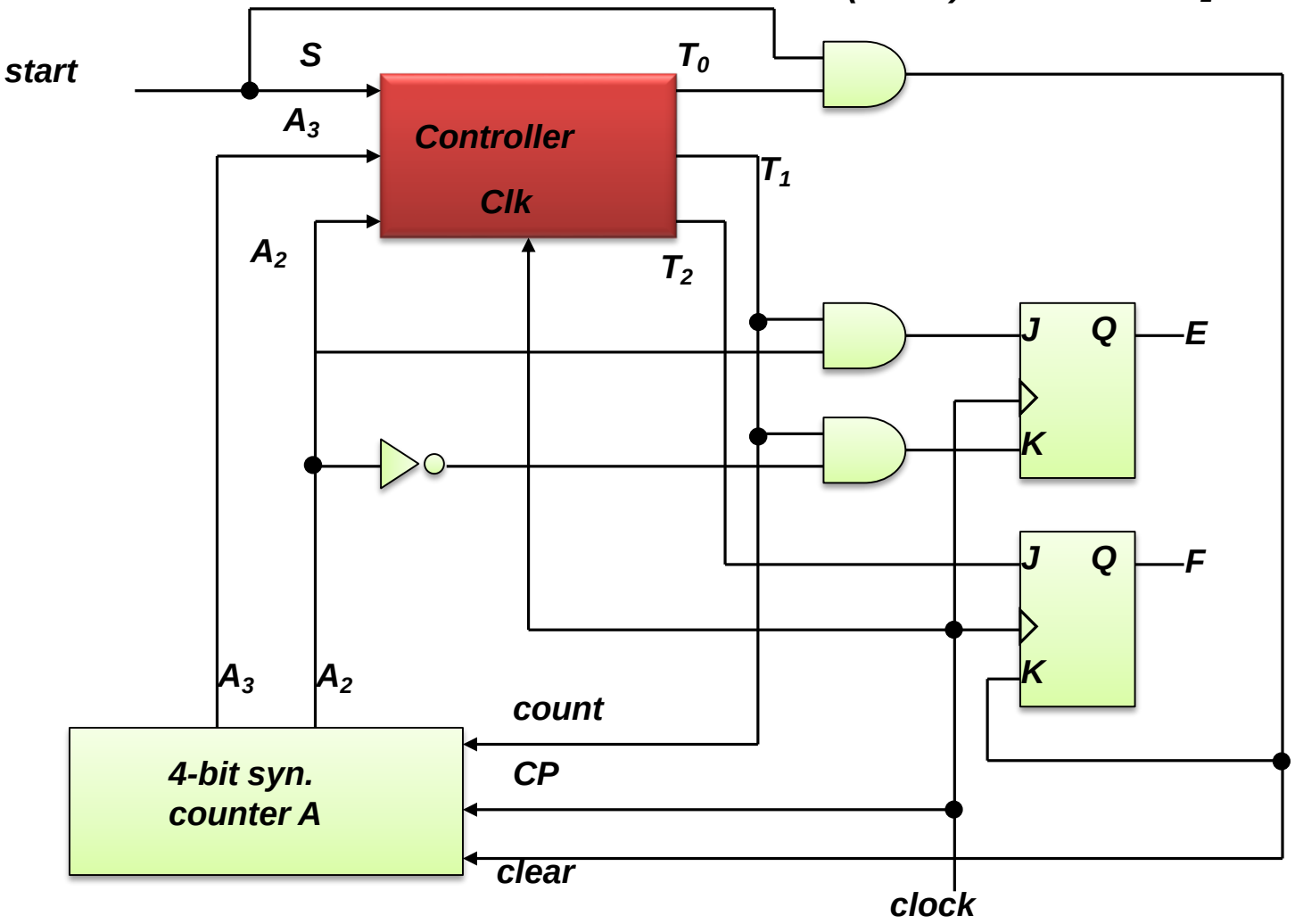
- Architecture is more difficult to design than controller.
- Nevertheless, it can be deduced from the ASM chart. In particular, the operations from the ASM chart determine:
  - What registers to use
  - How they can be connected
  - What operations to support
  - How these operations are activated.
- Guidelines:
  - always use high-level units
  - simplest architecture possible.

ASM Chart => Architecture/Data Processor

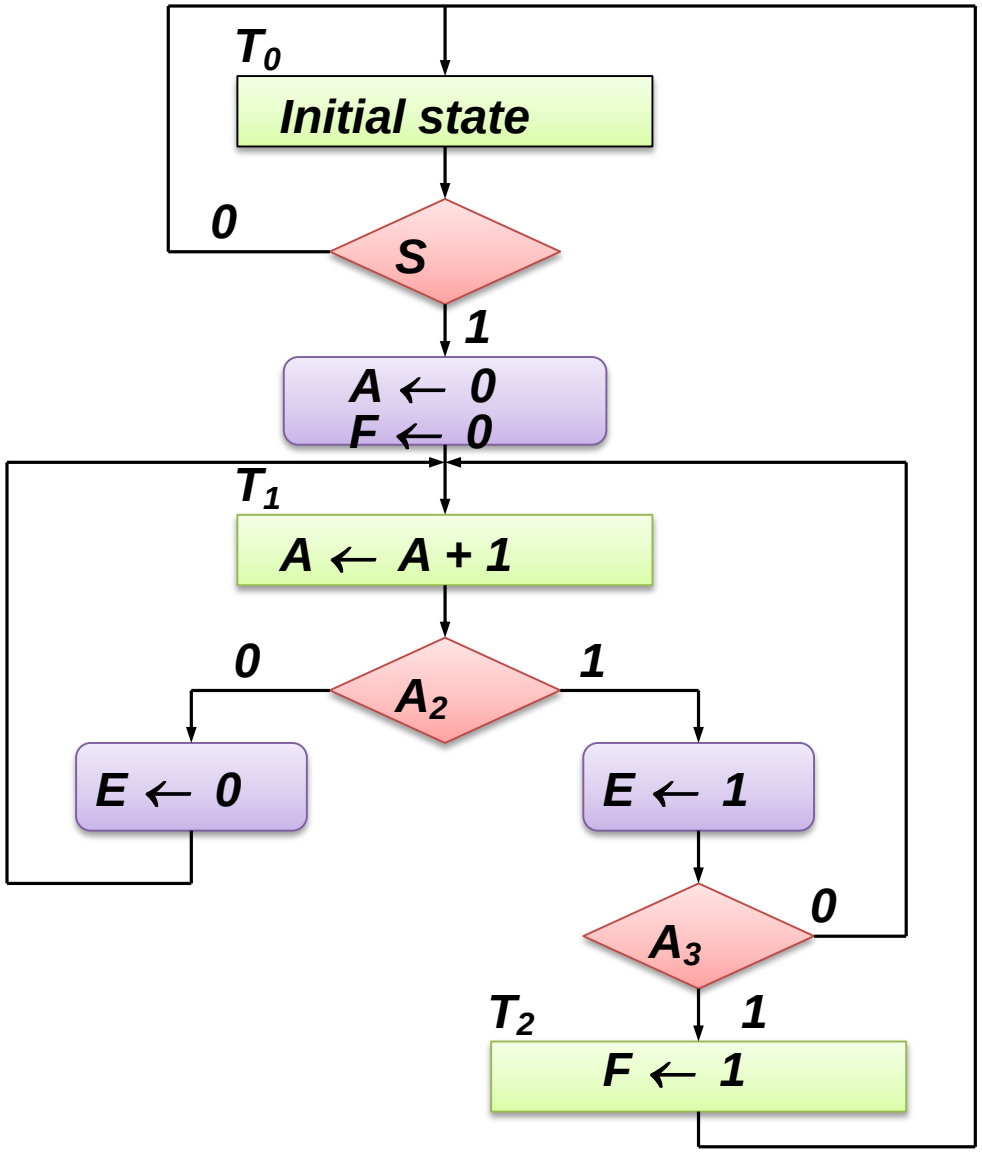
- Various operations are:
  - Counter incremented ($A \leftarrow A + 1$) when state = T_1.
  - Counter cleared ($A \leftarrow 0$) when state = T_0 and S = 1.
  - E is set ($E \leftarrow 1$) when state = T_1 and $A_2 = 1$.
  - E is cleared ($E \leftarrow 0$) when state = T_1 and $A_2 = 0$.
  - F is set ($F \leftarrow 0$) when state = T_2.
  - F is cleared ($F \leftarrow 0$) when state = T_0 and S = 1.
- Deduce:
  - One 4-bit register A (e.g.: 4-bit synchronous counter with clear/increment).
  - Two flip-flops needed for E and F (e.g.: JK/D flip-flops).

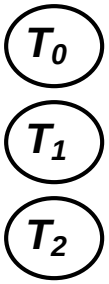
ASM Chart => Architecture/Data Processor

(A ← A + 1) when state = T_1.
(A ← 0) when state = T_0 and S = 1.
(E ← 1) when state = T_1 and A_2 = 1.



ASM Chart => Controller





Assign codes to states:

$T_0 = 00$

$T_1 = 01$

$T_2 = 11$

Present state		inputs			Next state		outputs		
G_1	G_0	S	A_2	A_3	G_1^+	G_0^+	T_0	T_1	T_2
0	0	0	X	X	0	0	1	0	0
0	0	1	X	X	0	1	1	0	0
0	1	X	0	X	0	1	0	1	0
0	1	X	1	0	0	1	0	1	0
0	1	X	1	1	1	1	0	1	0
1	1	X	X	X	0	0	0	0	1

Inputs from conditions in decision boxes.
Outputs = present state of controller.

RTL: Multiplier Example

- Example: (101 X 011) Base 2
- Note that the partial product summation for *n* digits, base 2 numbers requires adding up to *n* digits (with carries) in a column.
- Note also *n* X *m* digit multiply generates up to an *m + n* digit result (same as decimal).

- **Partial products are:**
  101 X 0, 101 X 1, and 101 x 1

			1	0	1
		X	0	1	1
			1	0	1
		1	0	1	
	0	0	0		
0	0	1	1	1	1

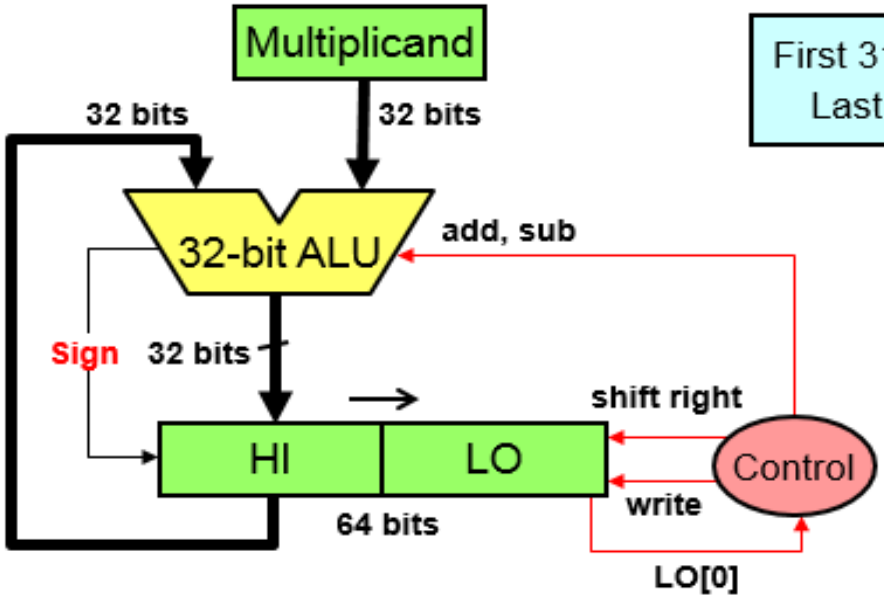
Example (1 0 1) x (0 1 1) Again

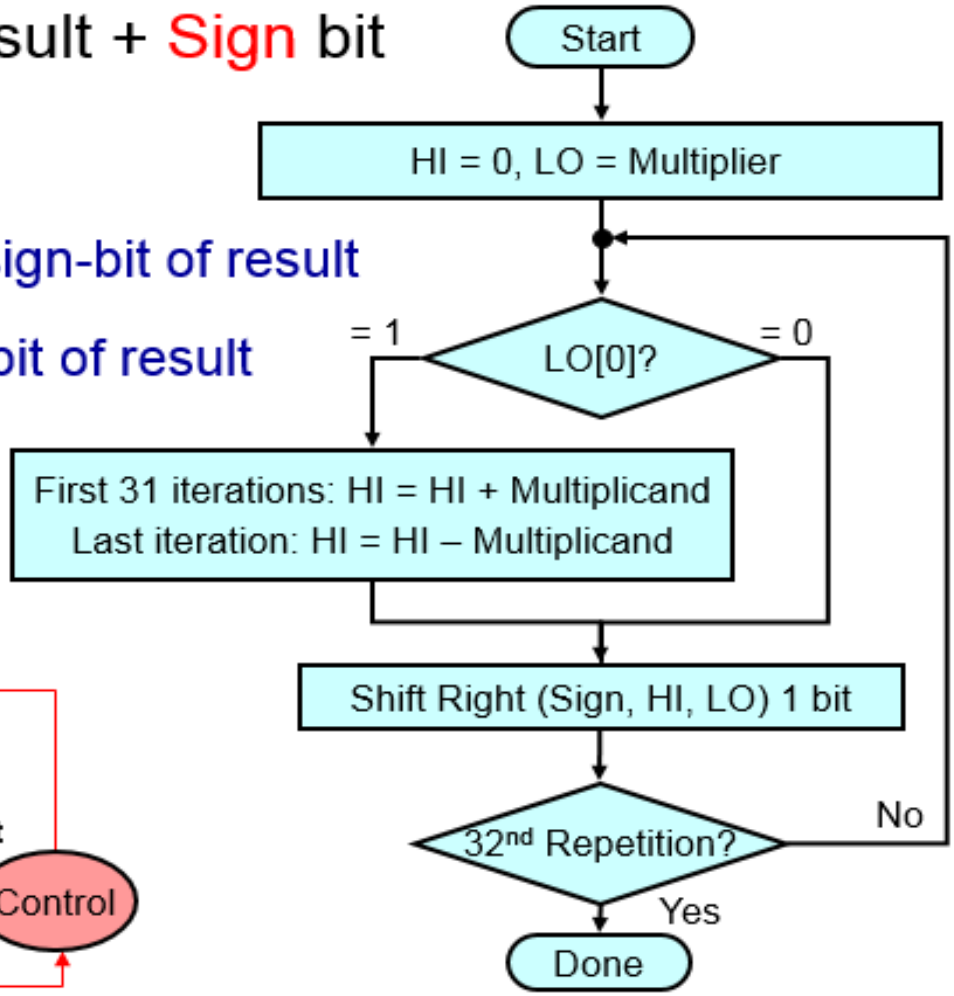
- Reorganizing example to follow hardware algorithm:

		1	0	1				
	x	0	1	1				
→	0	0	0	0				Clear C || A
→	+	1	0	1				Multipler0 = 1 => Add B
→	0	1	0	1				Addition
→	0	0	1	0	1			Shift Right (Zero-fill C)
→	+	1	0	1				Multipler1 = 1 => Add B
→	0	1	1	1	1			Addition
→	0	0	1	1	1	1		Shift Right
→	0	0	0	1	1	1	1	Multipler2 = 0 => No Add, Shift Right

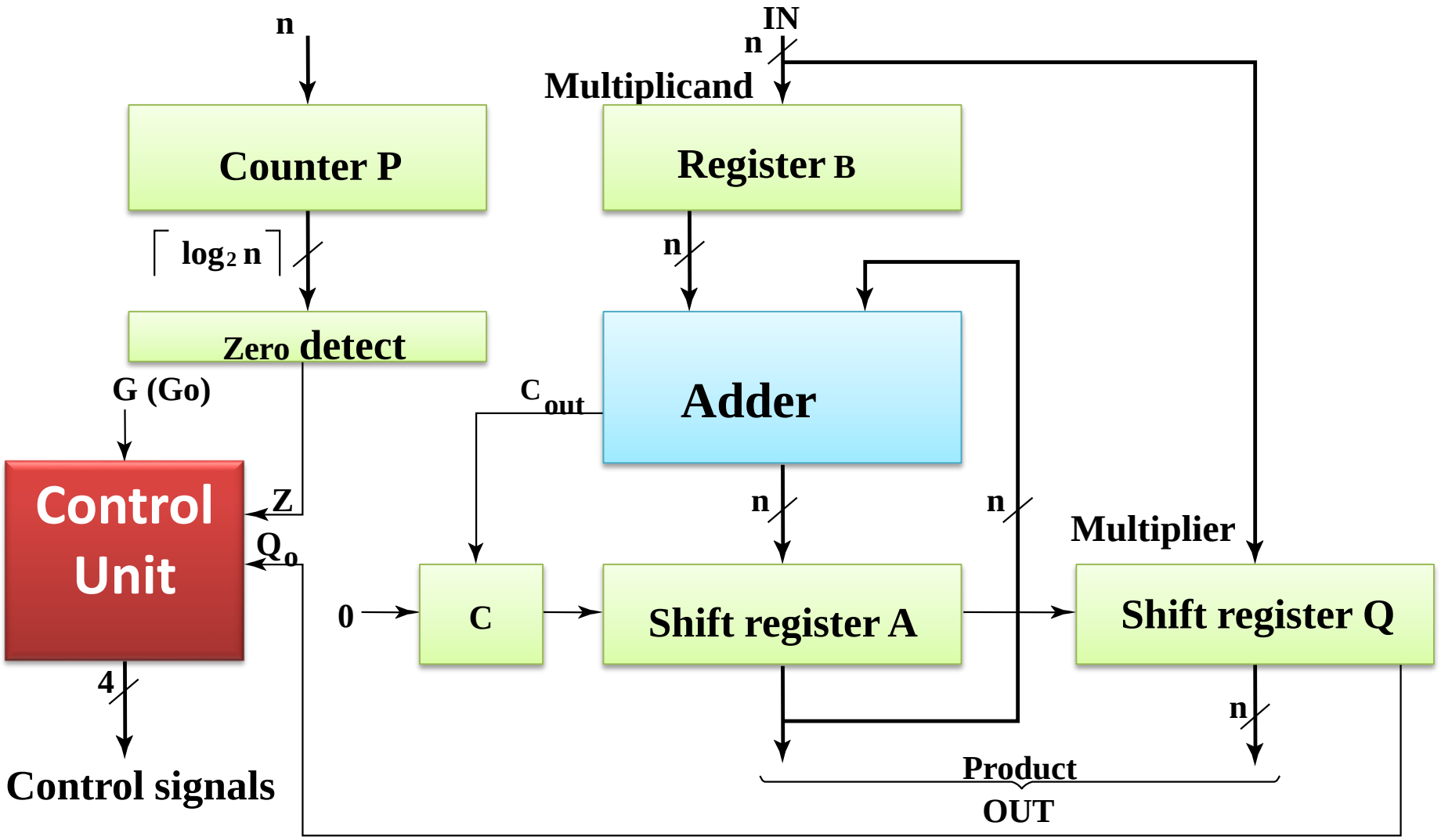
Sequential Signed Multiplier

- ALU produces 32-bit result + Sign bit
- Sign bit set as follows:
  - No overflow ➔ Extend sign-bit of result
  - Overflow ➔ Invert sign-bit of result





Multiplier Example: Block Diagram



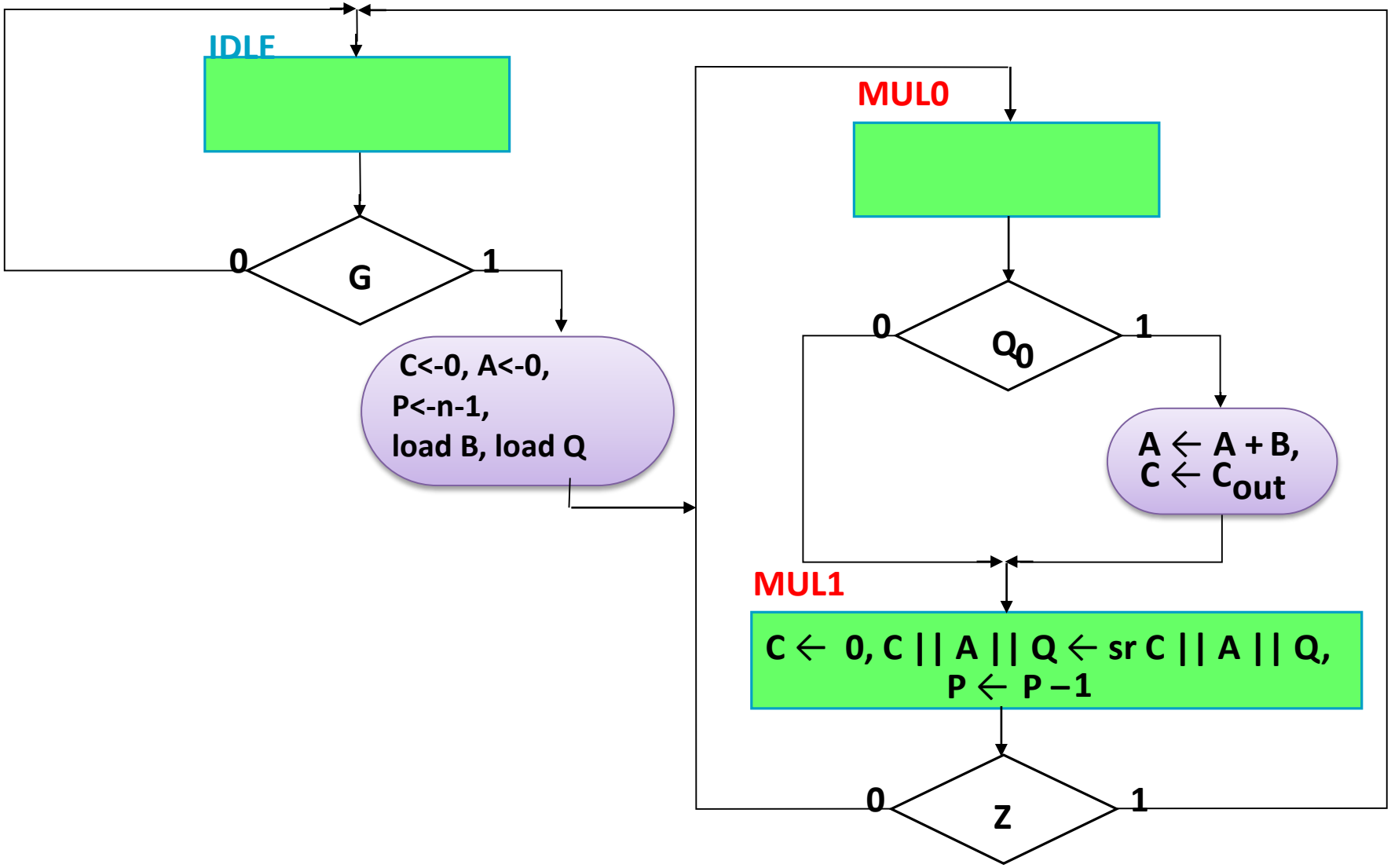
Multiplier Example: Operation

- Step1: The multiplicand (top operand) is loaded into register B.
- Step2: The multiplier (bottom operand) is loaded into register Q.
- Step3: Register C||A is initialized to 0 when G becomes 1.
- Step4: The partial products are summed iteratively in register C||A||Q.

Multiplier Example: Operation

- Step5: Each multiplier bit, beginning with the LSB, is processed (if bit is 1, use adder to add B to partial product; if bit is 0, do nothing)
- Step6: C||A||Q is shifted right using the shift register
  - Partial product bits fill vacant locations in Q as multiplier is shifted out
  - If overflow during addition, the outgoing carry is recovered from C during the right shift
- Step 7: Steps 5 and 6 are repeated until Counter P = 0 as detected by Zero detect.
  - Counter P is initialized in step 4 to n – 1, n = number of bits in multiplier

Multiplier Example: ASM Chart



Multiplier Example: ASM Chart (continued)

- Combined Mealy - Moore output model
- **IDLE - state**
  - Input G is used as the condition for starting the multiplication, and
  - *C, A, and P are initialized and Load B, Load Q*
- **MUL0 - state**
  - Conditional addition is performed based on the value of Q_0.
- **MUL1 - state**
  - Right shift is performed to capture the partial product and position the next bit of the multiplier in Q_0
  - the terminal count of 0 for down counter P is used to sense completion or continuation of the multiply.

Multiplier Example: Control Signal Table

Control Signals 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	$MUL0 \cdot Q$
	$C \mid\mid A \mid\mid Q \leftarrow sr\ C \mid\mid A \mid\mid Q$	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	$LOADQ$
	$C \mid\mid A \mid\mid Q \leftarrow sr\ C \mid\mid A \mid\mid Q$	Shift_dec	—
Counter P:	$P \leftarrow n - 1$	Initialize	—
	$P \leftarrow P - 1$	Shift_dec	—