CS221: Digital Design

RTL Design: Serial Multiplier

A. Sahu

Dept of Comp. Sc. & Engg.

Indian Institute of Technology Guwahati

Outline

- RTL Design
- Modulo 14 Counter Example
- Serial Multiplier Example
- **Can we automate this RTL design process?**
  - **Given C code/Parallel Code**

Reference Material for Lec 33, 34, 35

- Chapter 8 of Mano Book
  - Design at Register Transfer Level
  - Classic Example: Serial Binary Multiplication
- Chapter 15 of Kumar Book
  - Section 15.5.1: Data path Subsystem for Binary Multiplier

ASM Charts: An Complete Example
Ref: Mano Book

ASM Charts: An Example

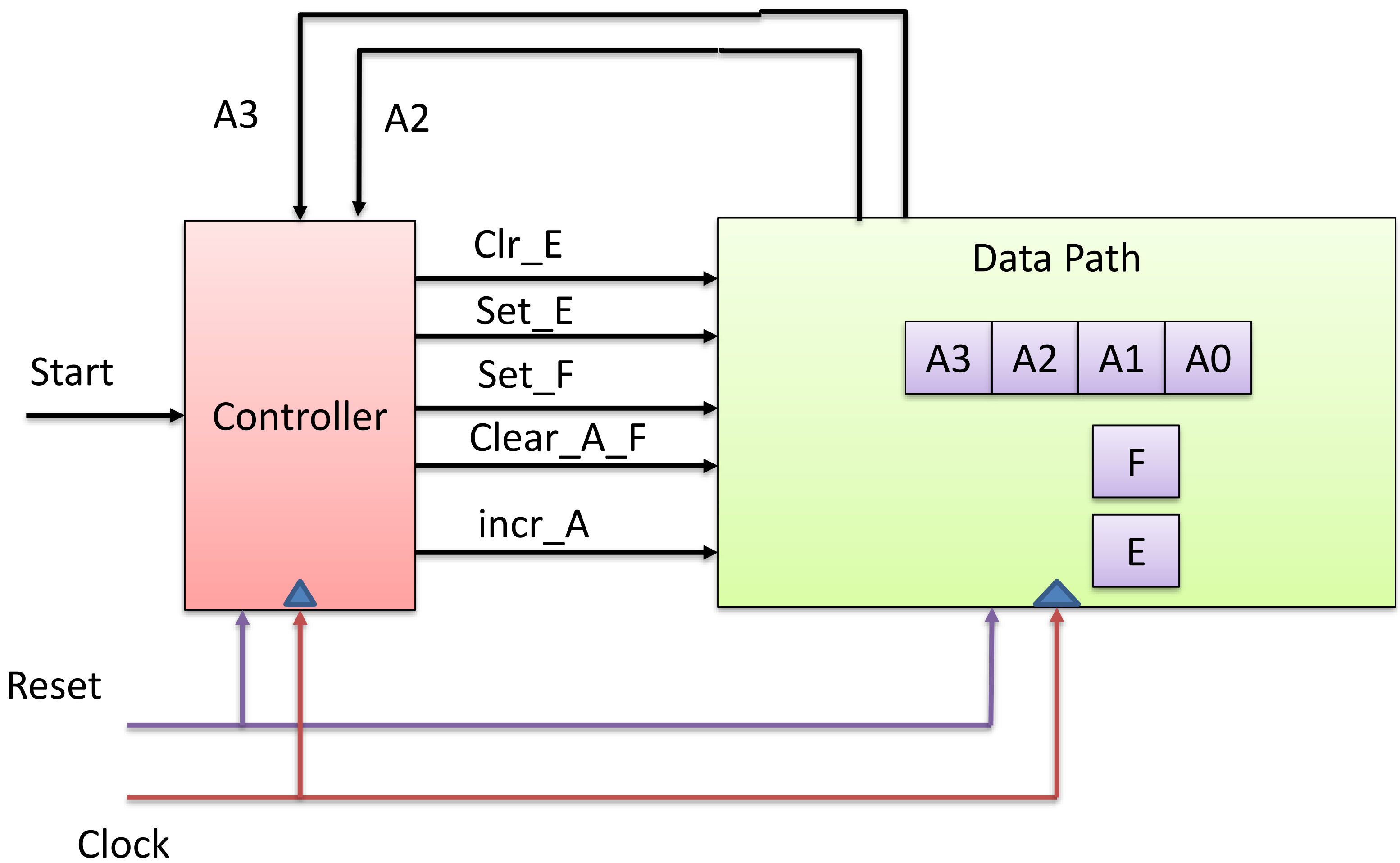
Mod 14 counter:

- There is a 4 bit counter (A)
- E specify: less than 12 or less than 4
- EF=11 specify : value =12,
  - It time to reset after next counting
- E depends of A_2, F depends on A_2, A_3

If A_2=1 ➔ E=1, else E=0

A_3A_2=1, ➔ F =1, counter reset

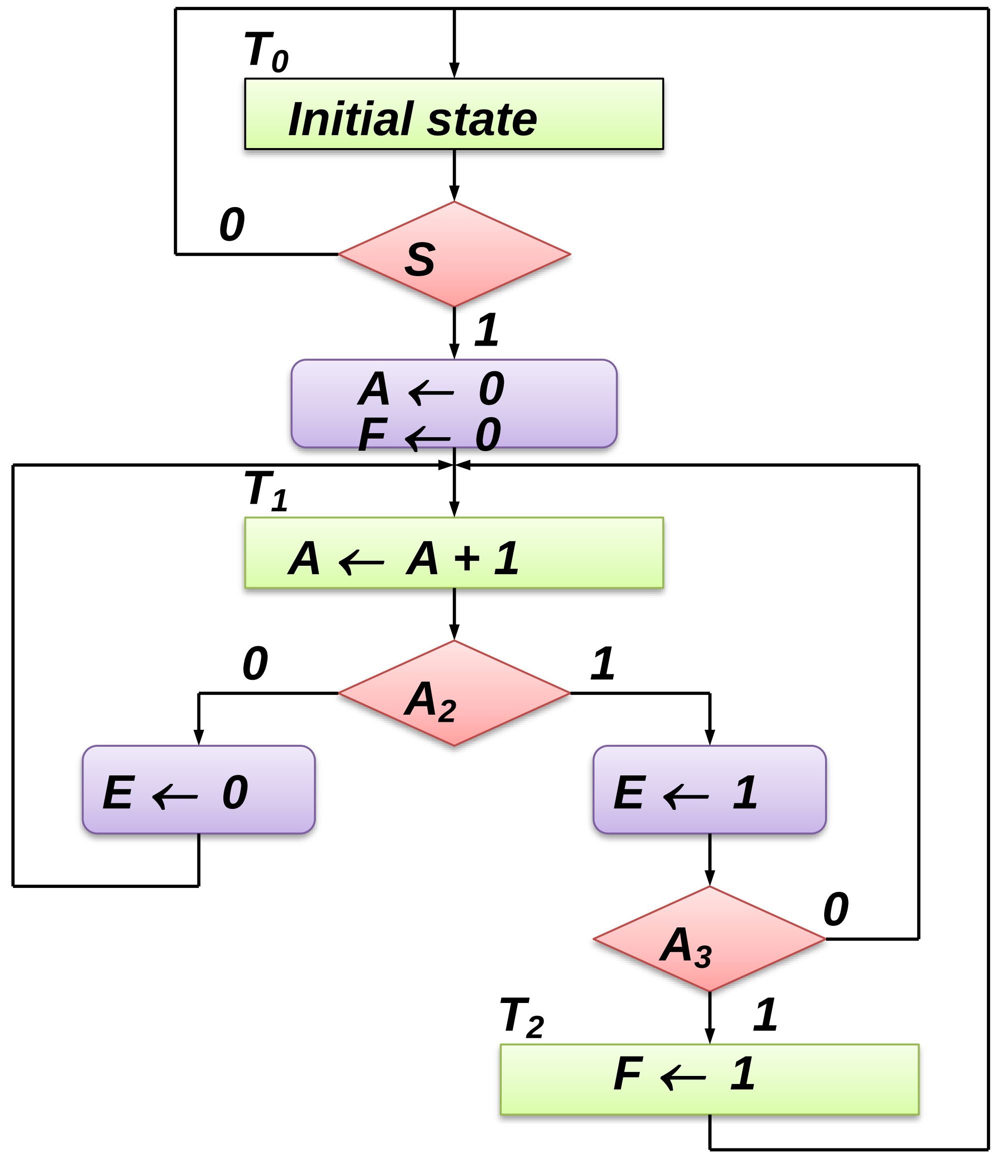
AMS DP+CP : to be High Level



ASM Charts: An Example

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

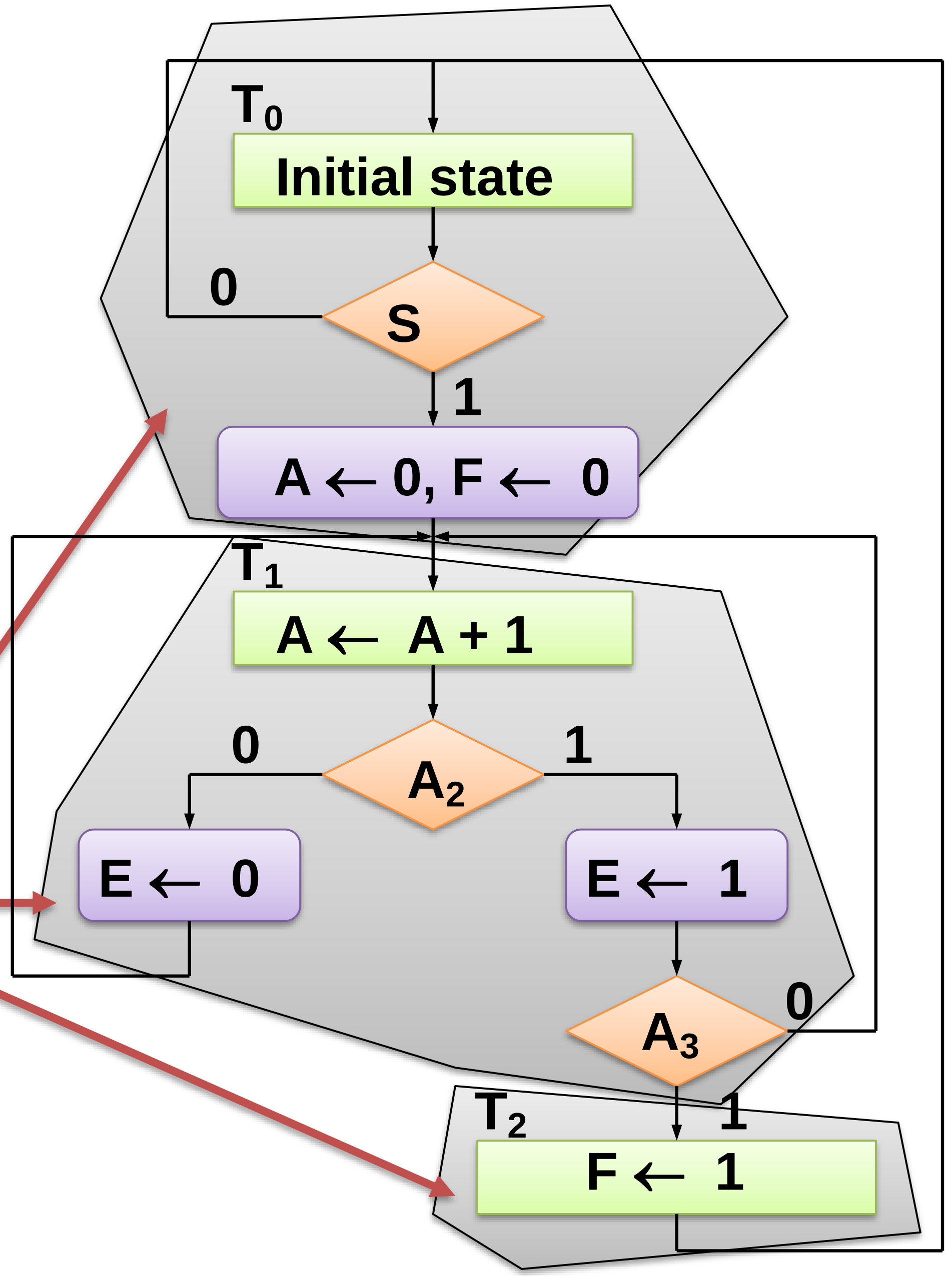
  $A = A_3A_2A_1A_0$
- E and F are single-bit flip-flops.



Timing in ASM Charts

- 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



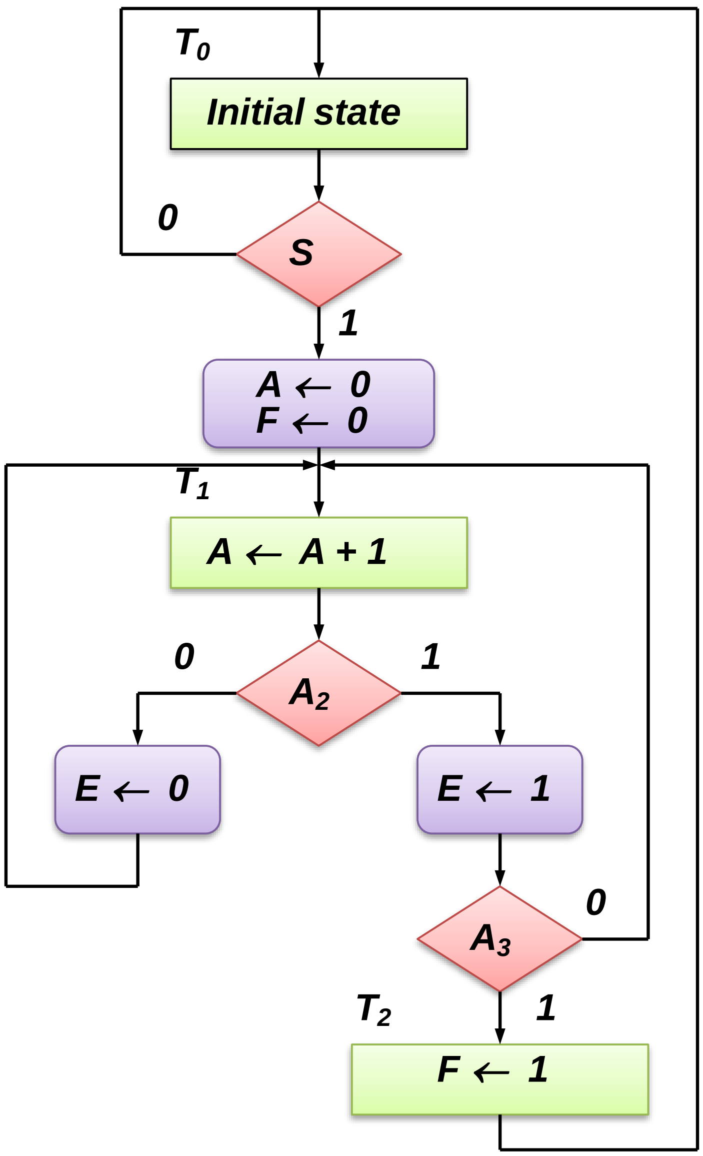
3 ASM blocks

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

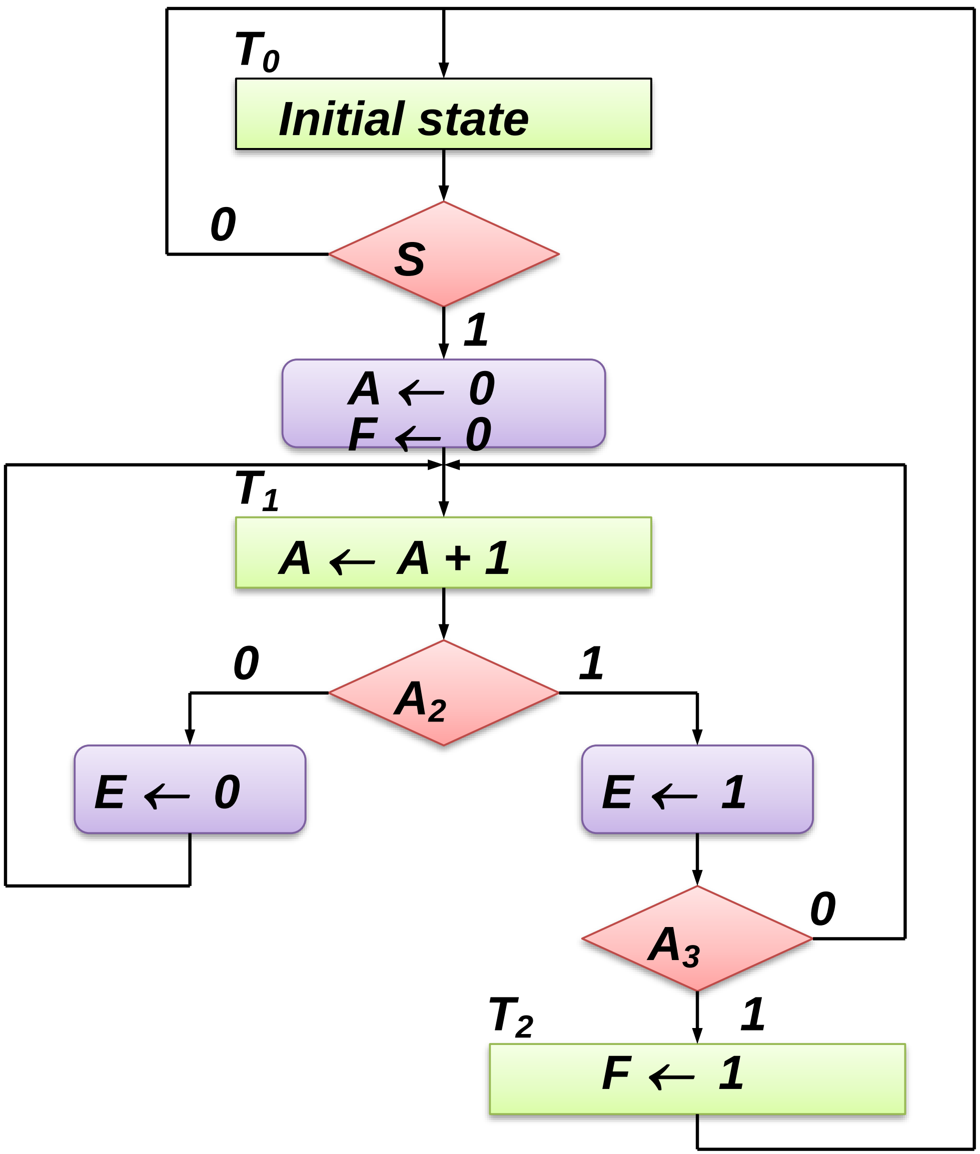
- Procedure:
  - Step 1: Identify all states and assign suitable codes.
  - Step 2: Formulate state table using

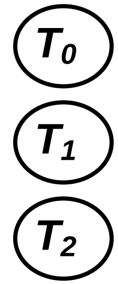
    State from state boxes

    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

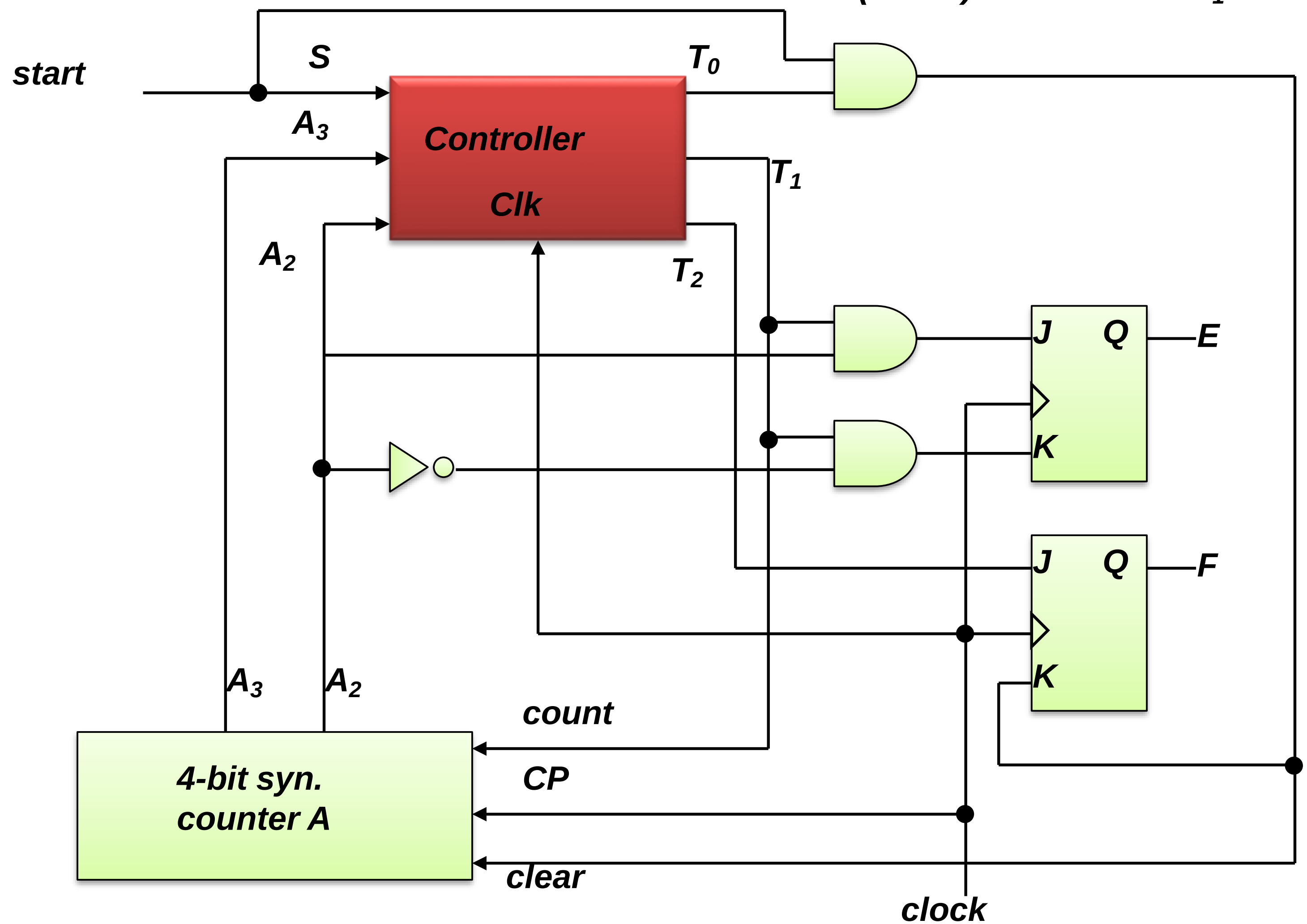
- Architecture is more difficult to design than controller.
- Nevertheless, it can be deduced from the ASM chart. In particular, the operations from the ASM chart determine:
  - What registers to use
  - How they can be connected
  - What operations to support
  - How these operations are activated.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

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



RTL/ASM example
for
8 bit Sequential Multiplier

Ref: Mano Book

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

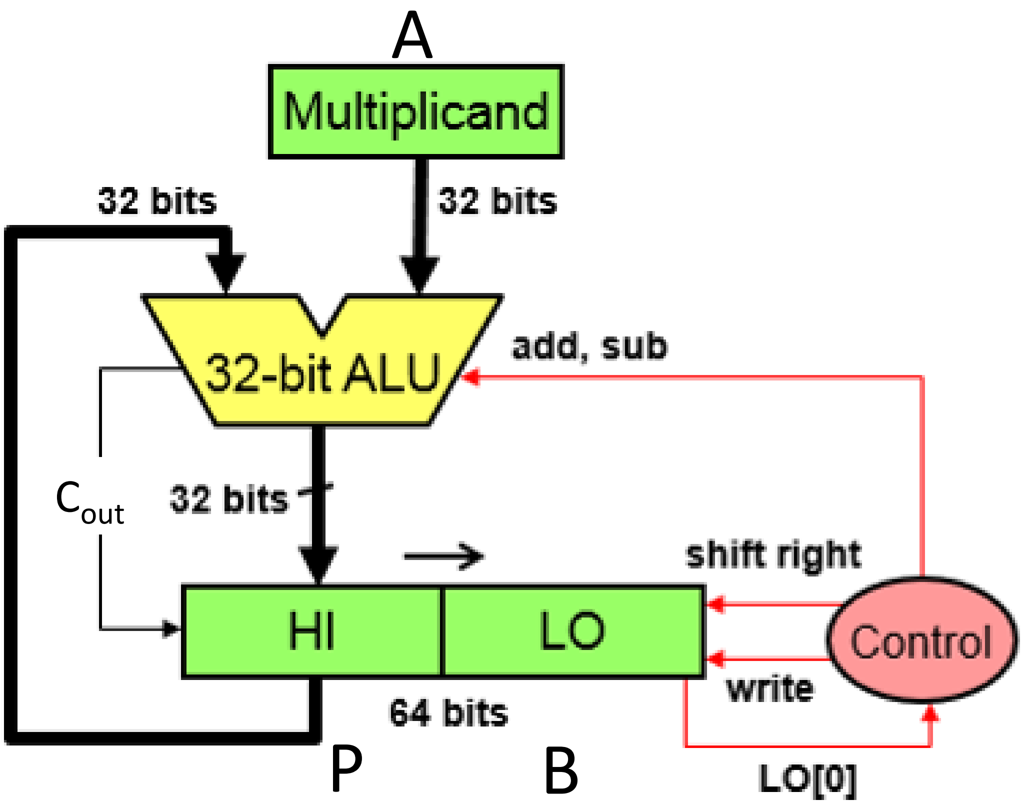
- Reorganizing example to follow hardware algorithm:

	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**

Binary Multiplication

- Polynomial Multiplication
  - You can think Binary number A, B as polynomials
  - $A(X)=A_{n-1}.2^{n-1}+....+ A_2.2^2+ A_1.2^1+ A_0.2^0$
  - Multiply polynomial to get another one
  - Time complexity: $O(n^2)$ Basic serial Algorithm, $O(n^{1.5})$ for divide conquer approach, $O(n \lg n)$ using FFT
- Booth Algorithm reduce number Partial addition
  - 99 Represent using 100-1, 95 is 100-5
  - 111 represent as 1000-1 in binary: possibly reduce

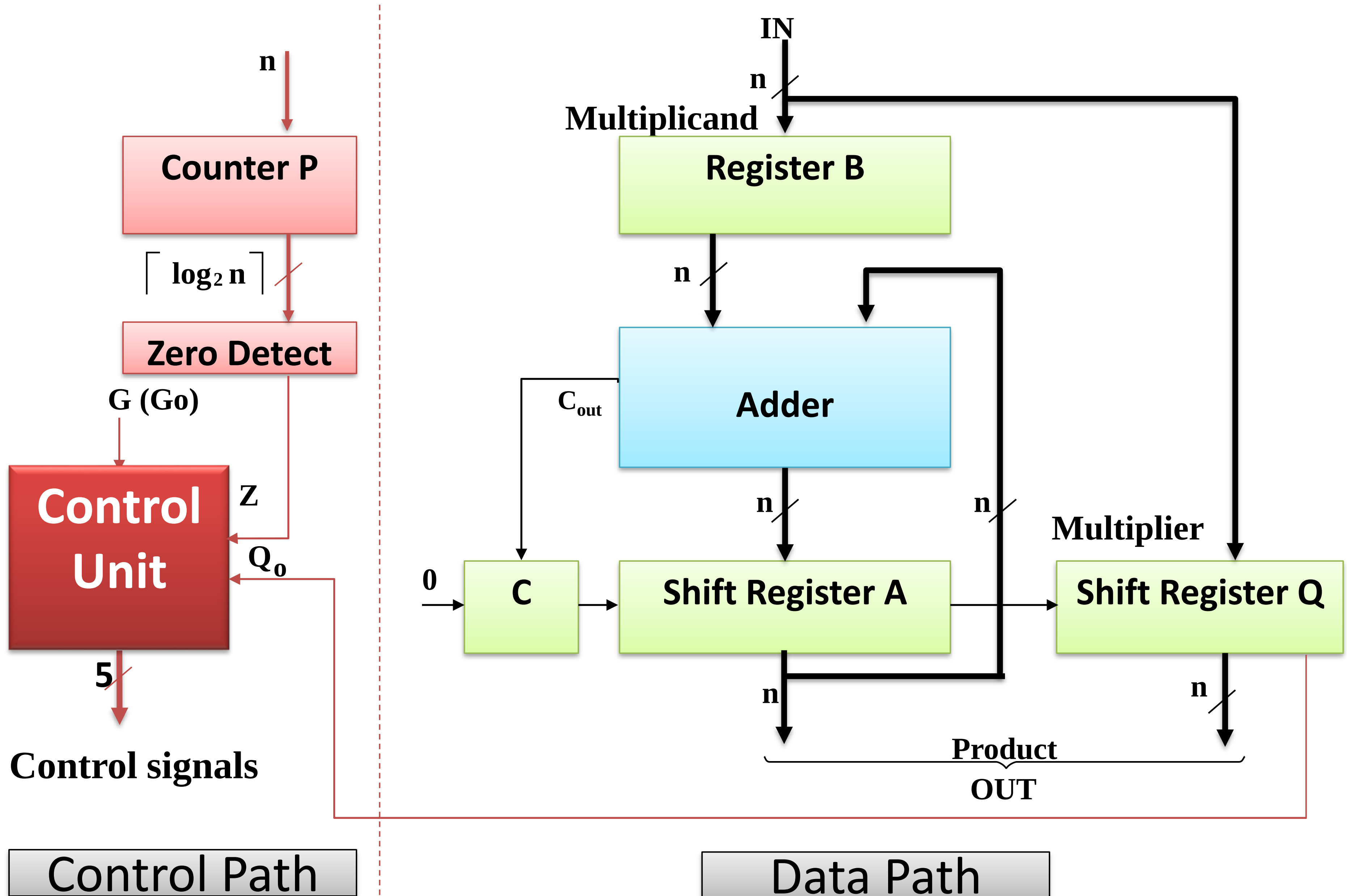
Sequential Multiplier



Control Algorithm:

1. P ← 0, A ← multiplicand, B ← multiplier **//Initialization**
2. If LSB of B==1 then add A to P else add 0
3. Shift [P][B] right 1
4. Repeat steps 2 and 3 n-1 times.
5. [P][B] has product.

Multiplier Example: Block Diagram



Multiplier Example: Operation

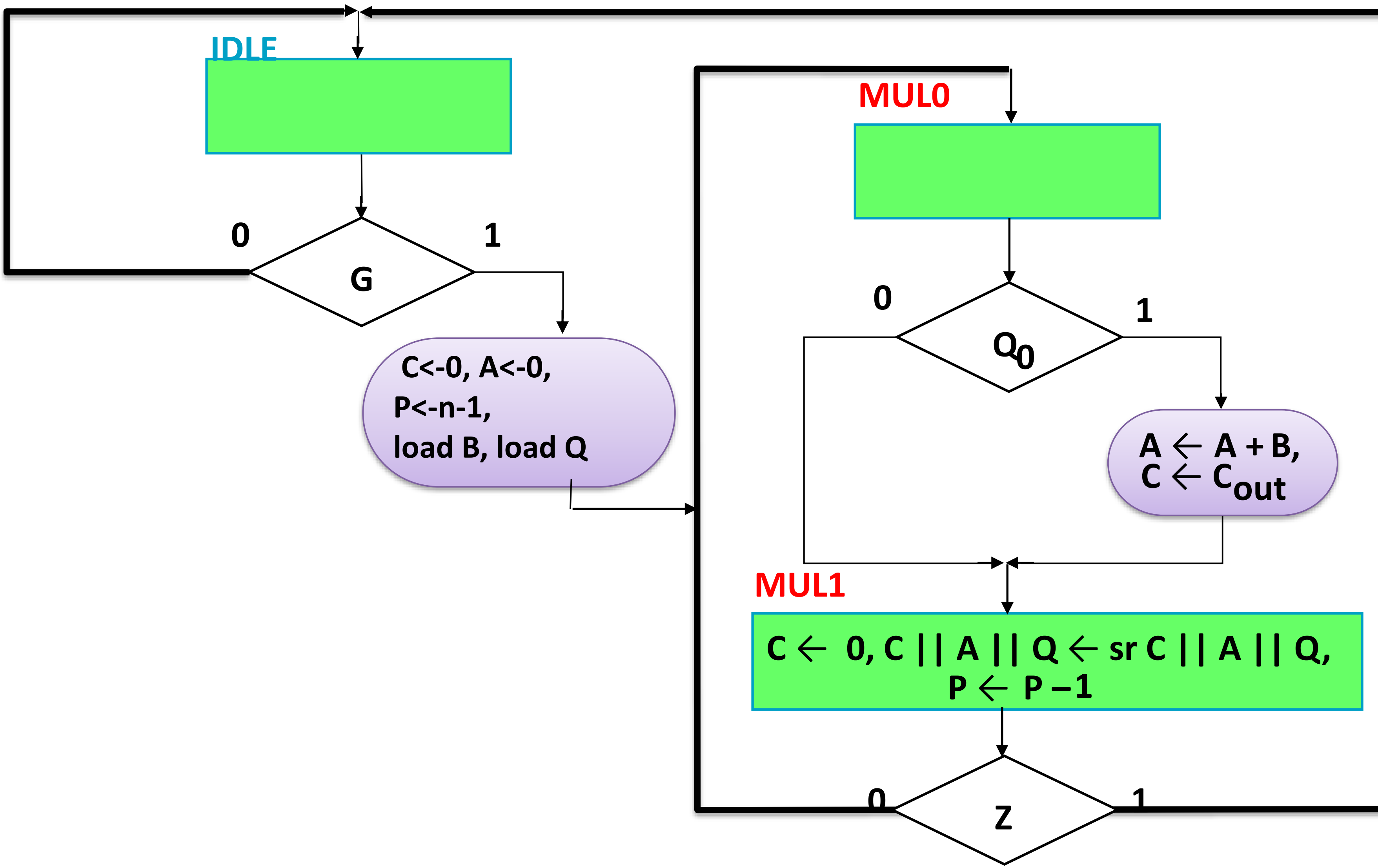
- Step1: The multiplicand (top operand) is loaded into register B.
- Step2: The multiplier (bottom operand) is loaded into register Q.
- Step3: Register C||A is initialized to 0 when G becomes 1.
- Step4: The partial products are summed iteratively in register C||A||Q.

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

<table>
<tr><th>Module</th><th>Micro Operations</th><th>Control Signal Name</th><th>Control Expression</th></tr>
<tr><td>Reg A</td><td>A <- 0
A <- A +B
C ||A||Q <- sr (C ||A||Q)</td><td>Load_regs
Add_regs
Shift CAQ</td><td>IDLE.G
MUL0.Q_0
MUL1</td></tr>
<tr><td>Reg B</td><td>B <- IN</td><td>Load_regs</td><td>LOAD_B</td></tr>
<tr><td>FF C

Reg Q</td><td>C <- 0
C <- Cout
Q <- IN
C ||A||Q <- sr (C ||A||Q)</td><td>Clear C
Load
Load_regs
ShiftCAQ</td><td>IDLE.G+MUL1
--
LOAD_Q</td></tr>
<tr><td>Ctr P</td><td>P <- N-1
P <- P-1</td><td>Load_regs
Decr_P</td><td>
MUL1</td></tr>
</table>

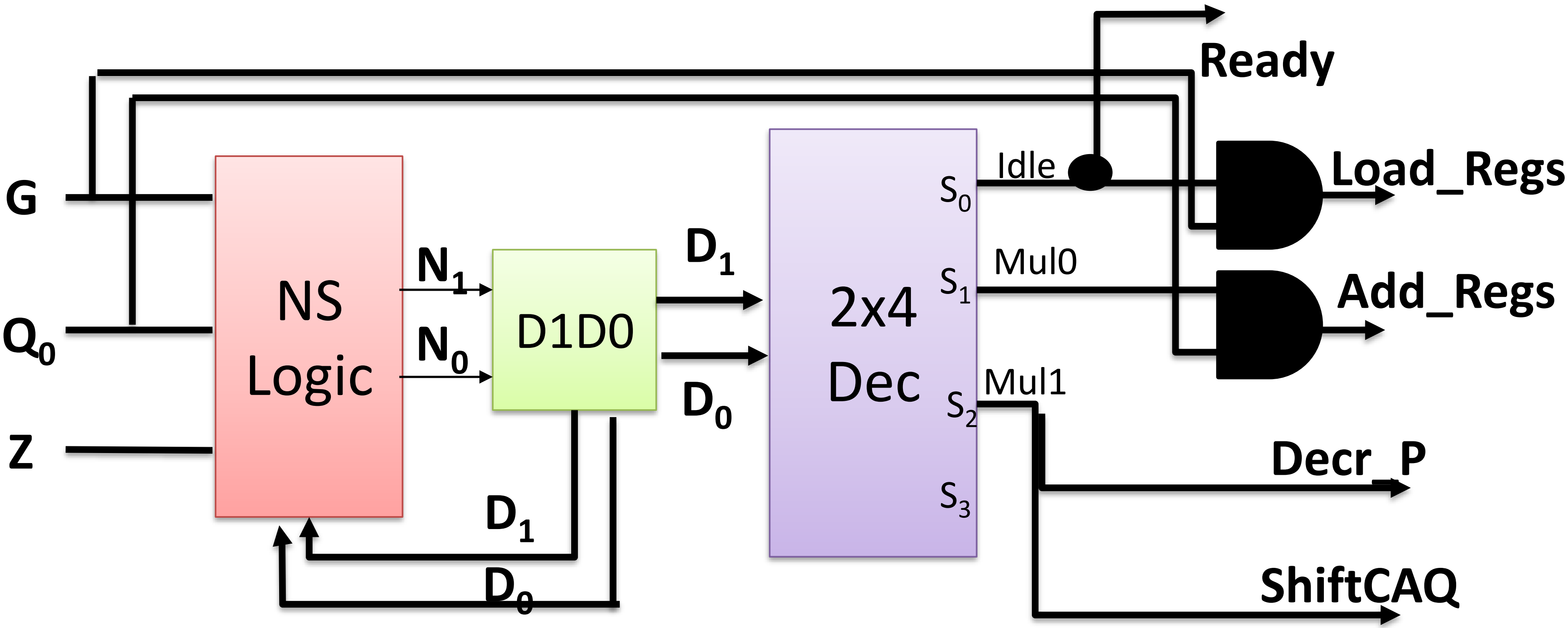
Multiplier Example: Control Circuit

PS	PS	Inputs			NS	Ready	Load Regs	Decr_P	Add_regs	Shift_CAQ
		G	Q0	Z						
Idle	00	0	x	X	00	1				
Idle	00	1	X	x	01	1	1			
Mul0	01	x	0	X	10			1		
Mul0	01	x	1	X	10				1	
Mul1	10	x	x	0	01			1		1
Mul1	10	x	x	1	00			1		1

Ready=Idle   Load_Regs=Idle.G   Decr_P=Mul0

Add_regs=Mul0.Q_0 ShiftCAQ=Mul1

Multiplier Example: Control Circuit



$N1 = D_1'D_0$

$N0 = D_1'D_0'G + D_1D_0'Z'$

Ready=Idle   Load_Regs=Idle.G   Decr_P=Mul1

$\text{Add\_regs} = \text{Mul0}.Q_0$ ShiftCAQ=Mul1