CS/ECE/EEE/INSTR F215:Digital Design

Lecture 34: ASM_3

Sat, 04 Dec 2021

BITS Pilani
Hyderabad Campus

Dr. R. N. Ponnalagu, EEE

Mistakes

Easy to judge when others do it

Difficult to realize when we do it.

Never Lose Heart

Remember

The best view comes

after the hardest climb

BINARY MULTIPLIER

To Multiply two unsigned binary numbers.

Sequential Multiplier

Uses One adder and a shift register.

Less hardware but takes more clock cycles to complete the operation

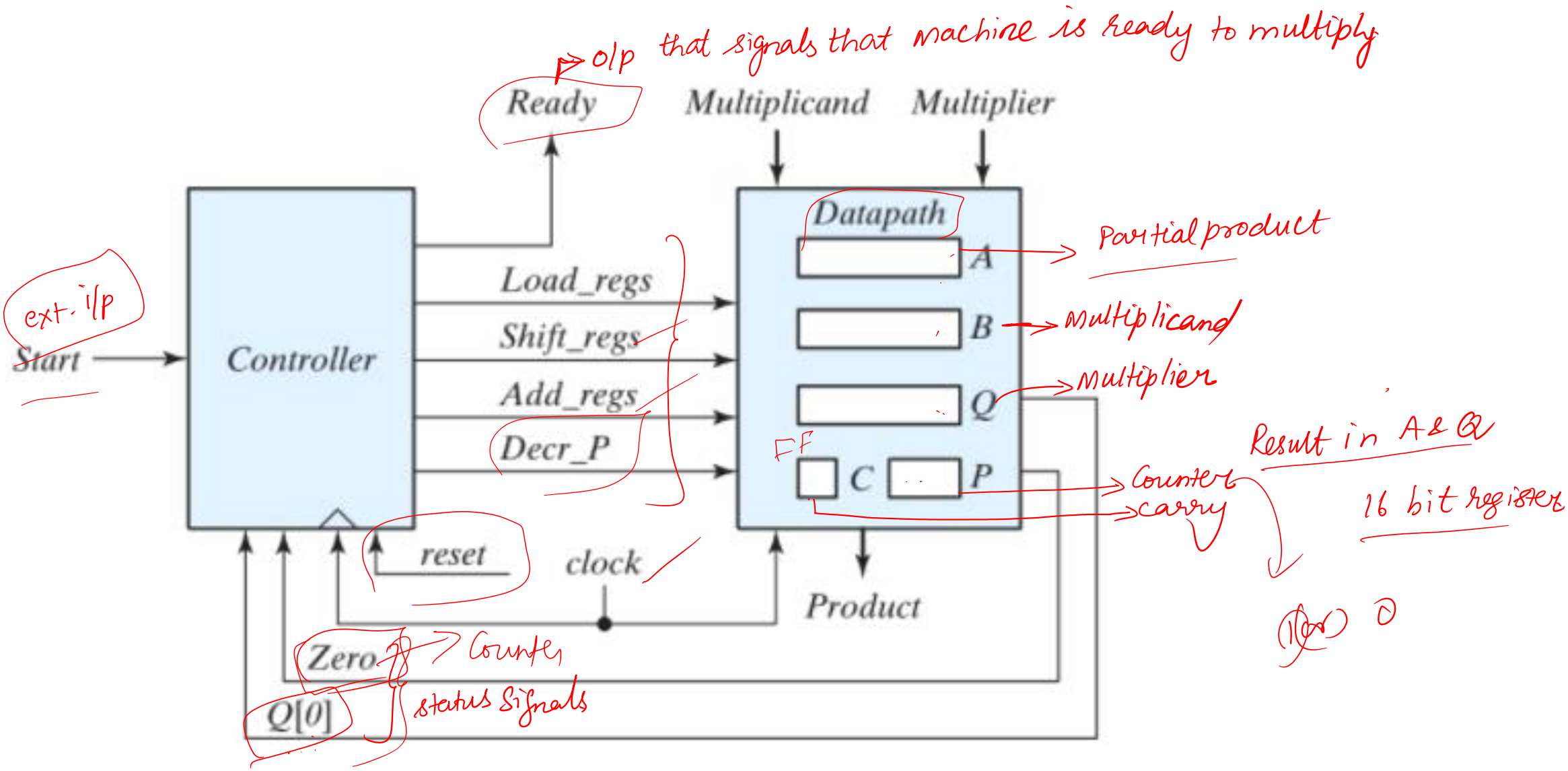
The process consists of successively adding and shifting copies of the multiplicand.

The product obtained from the multiplication of two binary numbers of n bits each can have up to 2 n bits.

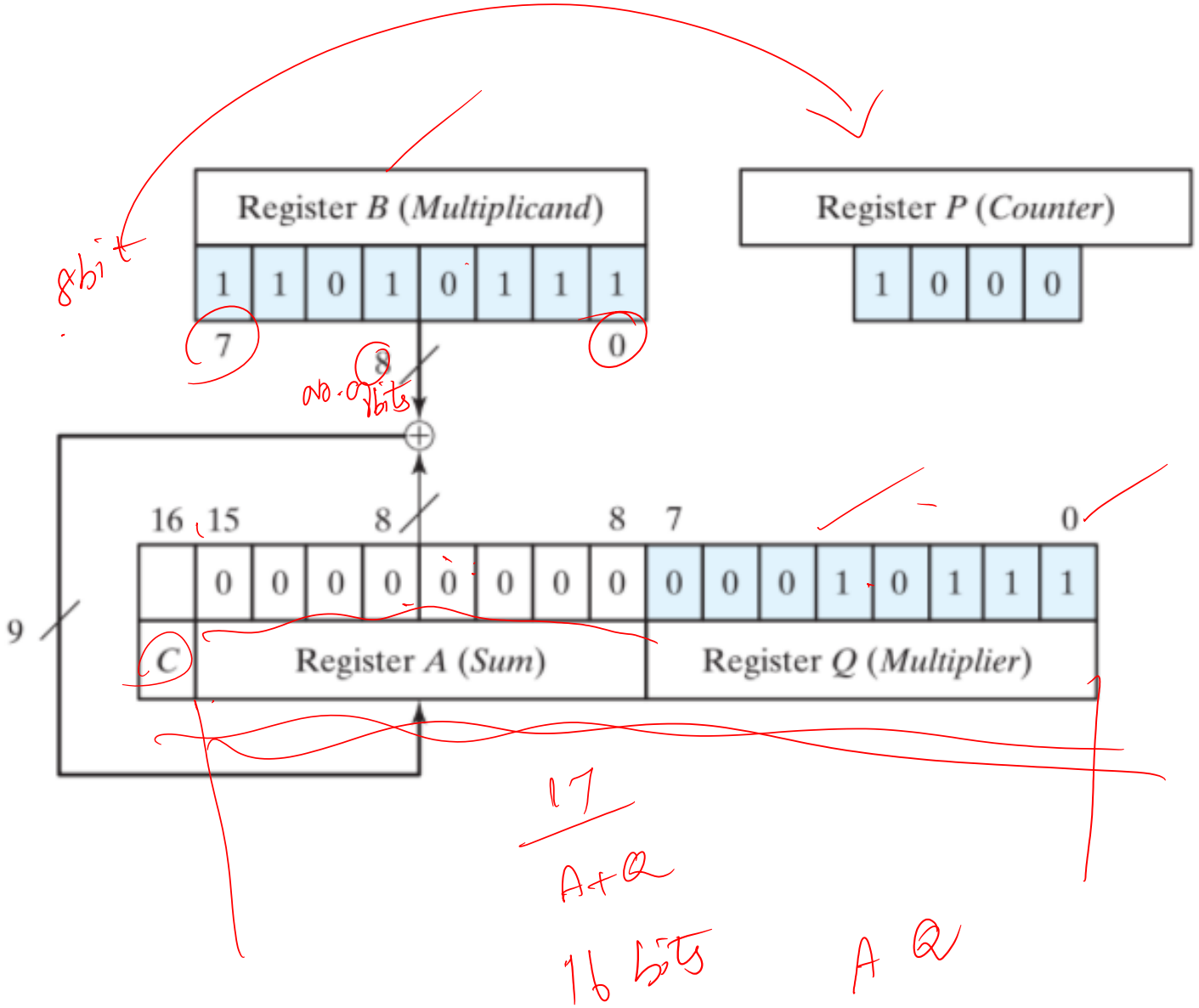
multiply the two binary numbers 10111 and 10011:

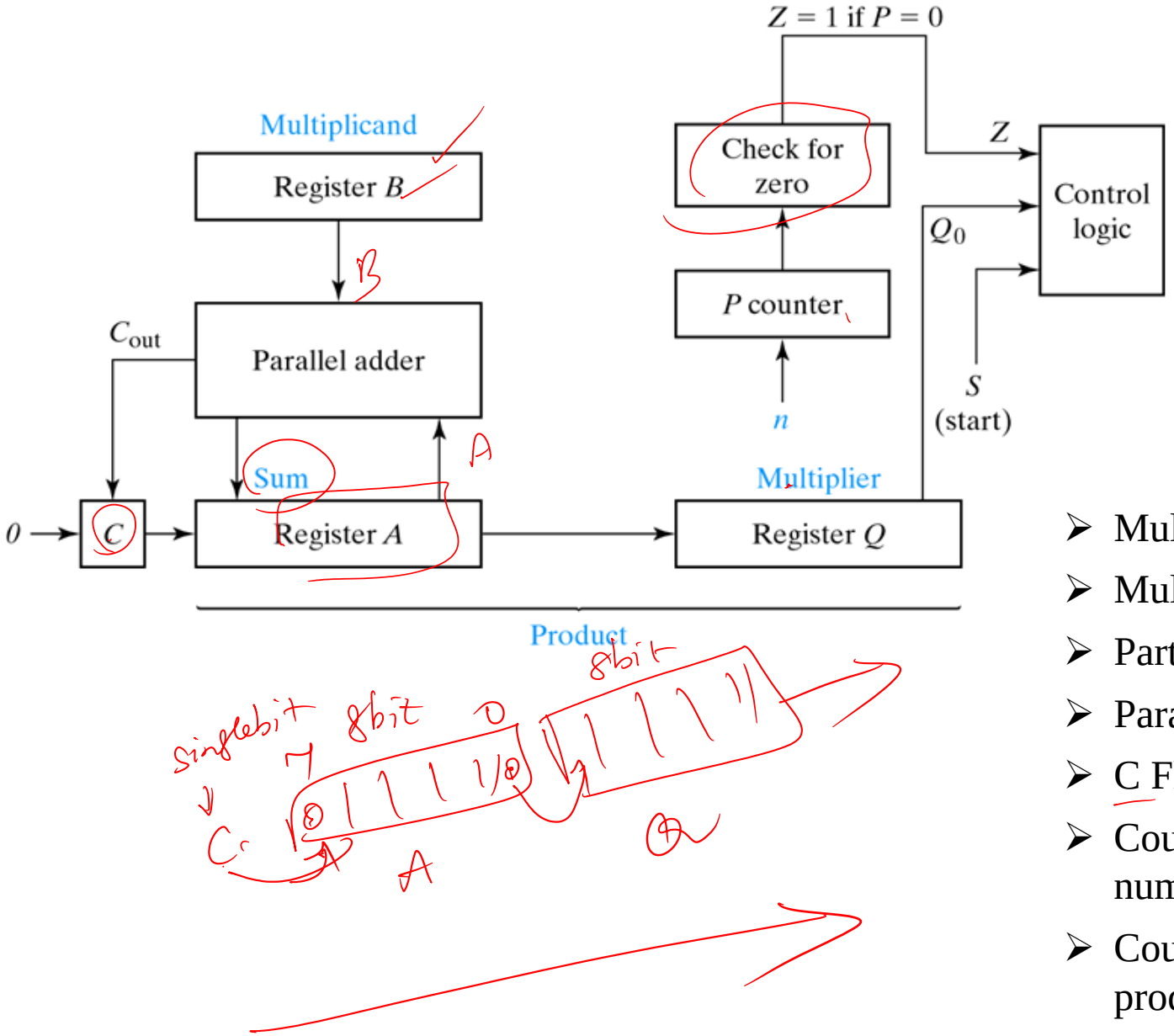
23	10111	multiplican
19	10011	multiplier
	10111	
	10111	
	00000	
	00000	
	10111	
437	110110101	product

Block diagram of Sequential Multiplier



Register configuration of the data path unit



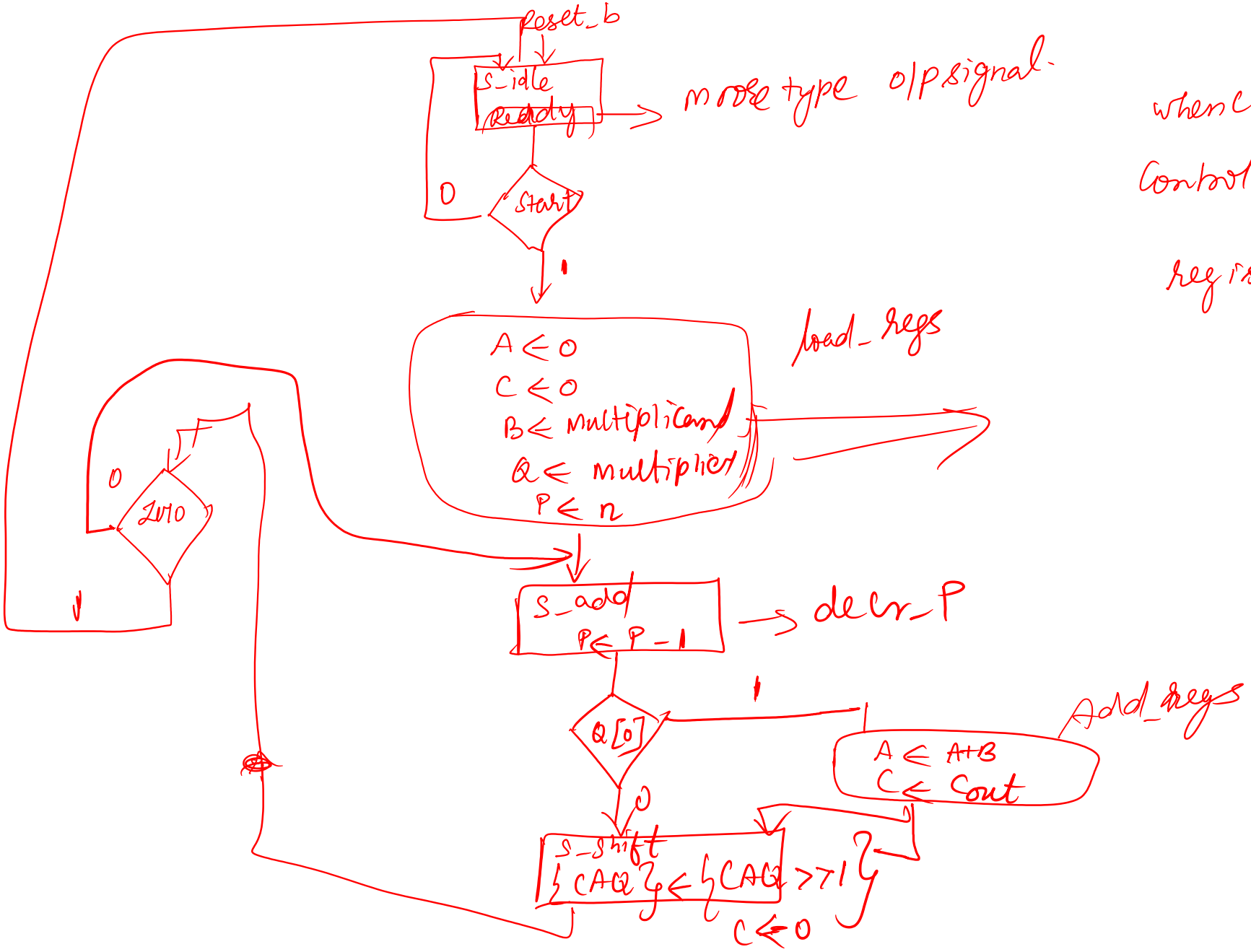


- Multiplicand in Register B
- Multiplier in register Q
- Partial product formed in register A and stored in A and Q
- Parallel adder adds the contents of register B to A
- C Flip-flop stores the carry resulting from addition
- Counter P initially set to hold a binary number equal to the number of bits in the multiplier
- Counter P is decremented after formation of each partial product
- When Counter is zero, the product is formed in the double register A and Q and the process stops

- Initial state S-idle until start signal (external input) is 1.
- After Ready signal from controller Machine starts to perform multiplication.
- Sum of A and B forms the ‘n’ most significant bits of the partial product which is transferred to A from parallel adder
- Output carry after addition (whether a 0 or 1) is transferred to C flip-flop
- The partial formed in register A and stored in A and Q are shifted to right
- LSB of A is shifted to MSB of Q, carry C is shifted into MSB of A and a 0 is shifted into C FF
- After shifting right, one bit of the partial product is transferred into Q while the multiplier bits in Q are shifted one position right
- LSB of register Q , Q[0] holds the bit of the multiplier that must be inspected next
- Control logic determines whether to add or not on the basis of Q[0]
- Q[0] from multiplier register & Zero signal from the counter P are status signals for control unit
- Based on Start, Q[0] and Zero signal controller generates five output control signals, Ready, Load_regs, Add_regs, Shift_regs, Decr_P

- Based on Start, Q[0] and Zero signal controller generates five output control signals, Ready, Load_regs, Add_regs, Shift_regs, Decr_P

ASM Chart

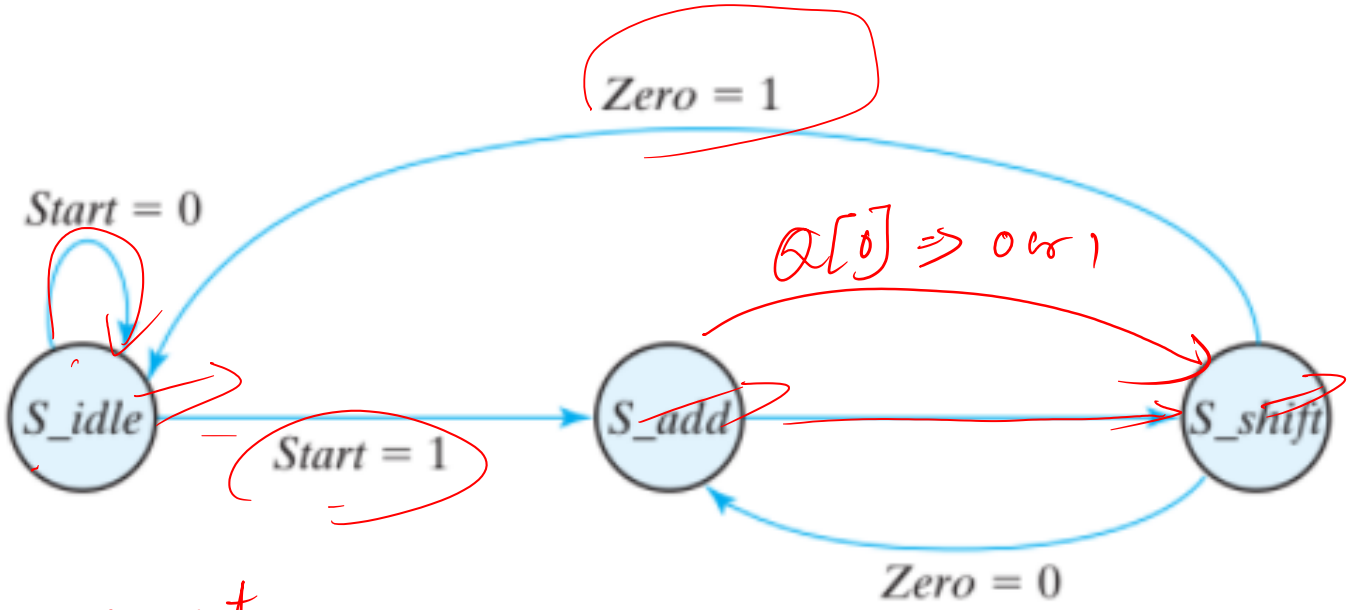


when clock is available
Control transfers to
S_add
register A & FF C
are cleared

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 $AQ = 0110110101_2 = 1b5_H$				



3 states
start
zero
Q[0]

Q[0] => 0 or 1

State assignment

Binary
Gray code
one-hot assignment

S_idle
S_add
S_shift

Binary	Gray	one_hot
00	00	001
01	01	010
10	11	100
2FFs	2 ffs	3FFs

3 bits

3 states 3 FFs

12 states 12 bits

12 FFs

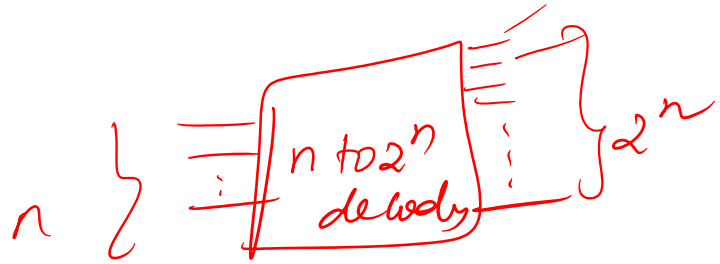
4 FFs

specialized methods for control logic design

2 methods

1) sequence register & a decoder
2) one ff per state ⇒ one hot assignment

Sequence register & decoder



ASM chart 3 states 3 i/ps

Binary state assignment 00, 01, 10

2 FFs for the register and 2 to 4 line decoder

Next state $G_1 = D_{G_1}$

when Present state is 01 → S_add

$D_{G_1} = T_1$

Next state $G_0 = S\,D_{G_0}$

$D_{G_0} = S\_idle \cdot start + S\_shift \cdot zero = 0$

$= T_0\,start + T_2\,\overline{zero}$

Moore type

D type FF

2 FFs for register

State Table for Control Circuit

Present-State Symbol	Present State G_1	Present State G_0	Inputs Start	Inputs Q[0]	Inputs Zero	Next State G_1	Next State 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

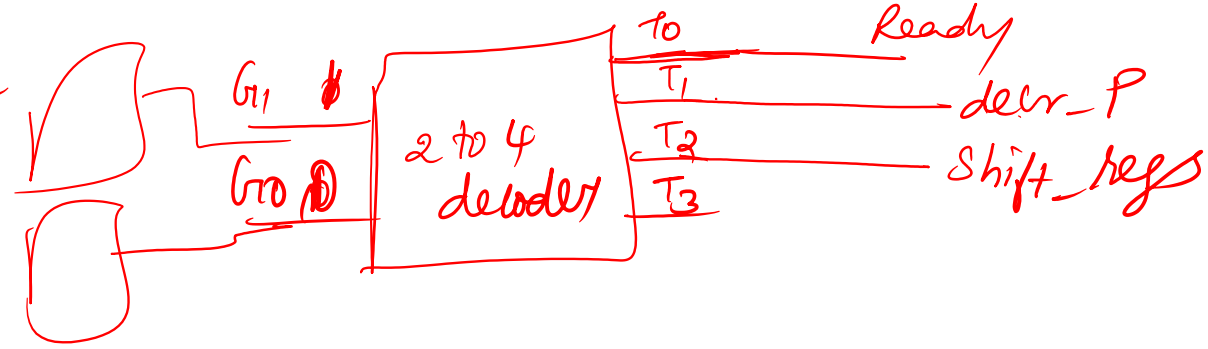
O/P

S_idle

S_add

Ready = 1 when state is S_idle

binary assignment



State Table for Control Circuit

Present-State Symbol	Present State G_1	Present State G_0	Inputs Start	Inputs Q[0]	Inputs Zero	Next State G_1	Next State 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

$D_{G0} = S\_idle \cdot Start + S\_shift \cdot Zero$

$= T_0 \cdot start + T_2 \cdot Zero$

Load_regs = state → S_idle, i/p → start

Load_regs = $T_0 \cdot start$

Add_regs = state S_add · Q[0] = 1

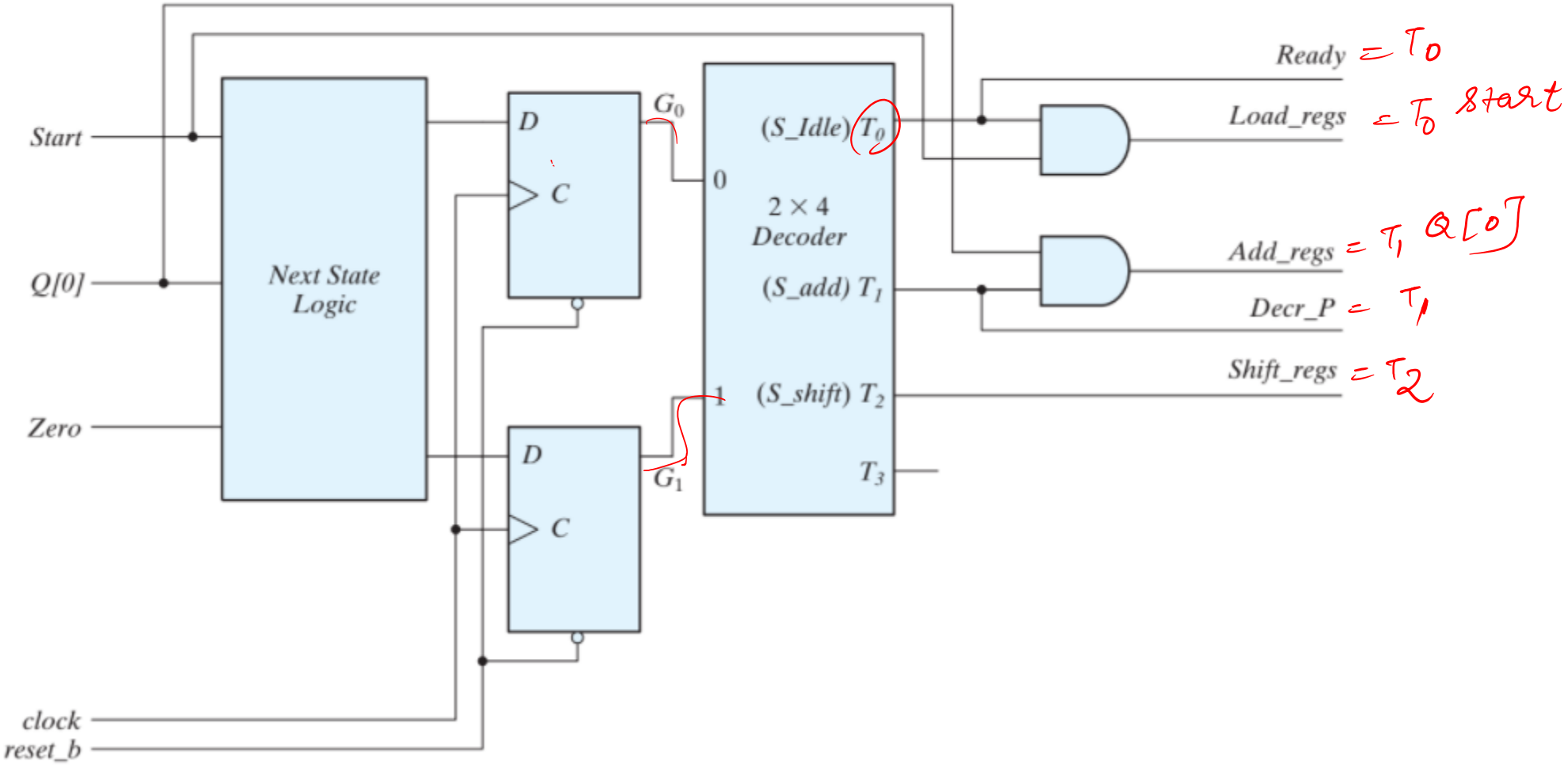
Add_regs = $T_1 Q[0]$

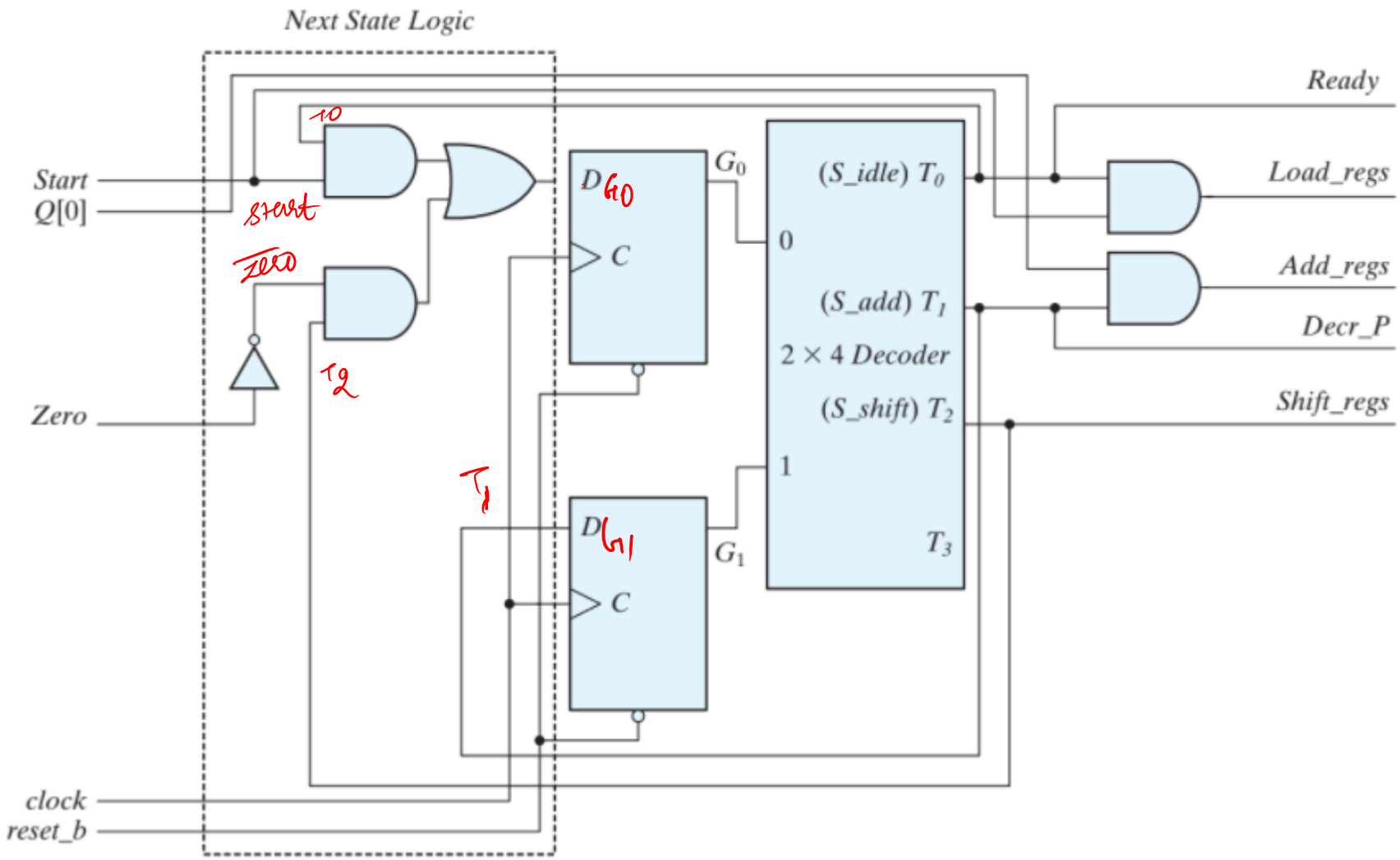
Ready = T_0

Decr_P = T_1

Shift_regs = T_2

$D_{G1} = T_1$


Start
Q[0]
Zero
clock
reset_b
Next State Logic
D
C
D
C
G_0
G_1
0
1
2 × 4 Decoder
(S_Idle) T_0
(S_add) T_1
(S_shift) T_2
T_3
Ready
Load_regs
Add_regs
Decr_P
Shift_regs
$= T_0$
$= T_0$ start
$= T_1$ Q[0]
$= T_1$
$= T_2$


Next State Logic
Start
Q[0]
Zero
clock
reset_b
D
C
G_0
G_1
0
1
2 × 4 Decoder
(S_idle) T_0
(S_add) T_1
(S_shift) T_2
T_3
Ready
Load_regs
Add_regs
Decr_P
Shift_regs

one hot design

one FF / state

3 FFs

↓

3 D FFs

S_idle
001

S_add
010

S_shift
100

~~Moore~~ o/ps G_0 G_1 & G_2.

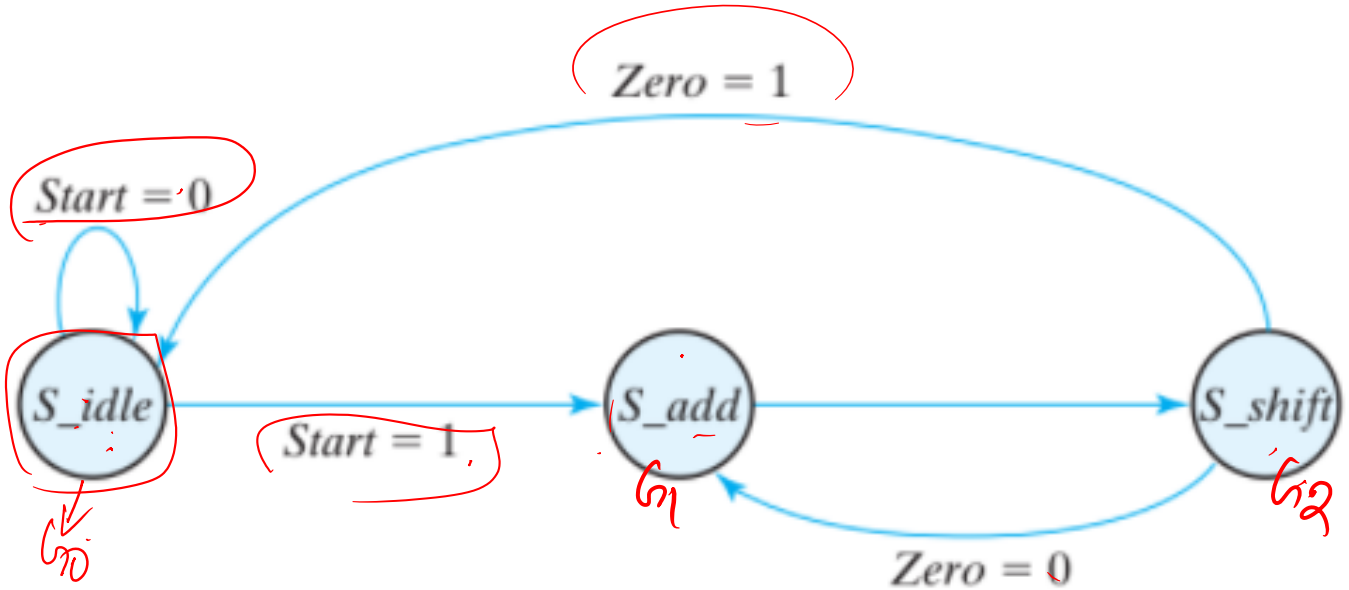
just by seeing state dgm we can design the controller in case of one hot design

001
010
100

$D_{G_0} = 1$?

$D_{G_1} = 1$

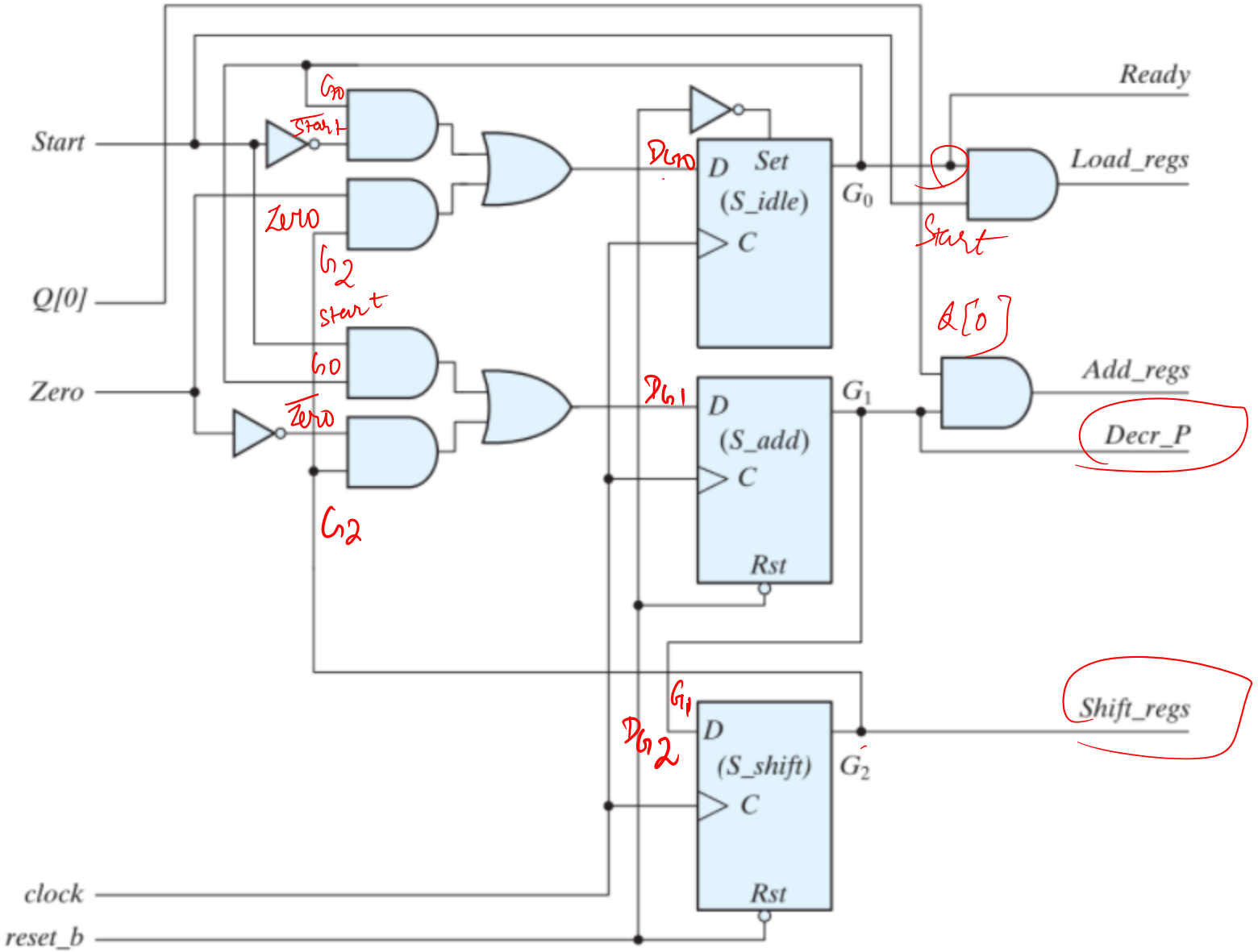
$D_{G_2} = 1$



$D_{G_0} = G_0 \overline{Start} + G_2 Zero$

$D_{G_1} = G_0 Start + G_2 \overline{Zero}$

$D_{G_2} = G_1$


Start
Q[0]
Zero
clock
reset_b
Ready
Load_regs
Add_regs
Decr_P
Shift_regs
D
Set
(S_idle)
C
Rst
(S_add)
(S_shift)
G_0
G_1
G_2
Start
Zero
Q[0]