

Lecture 22:

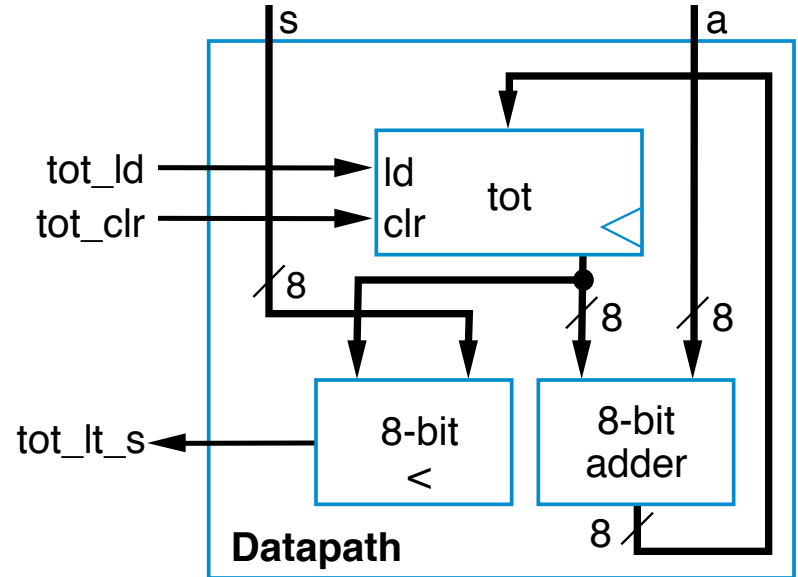
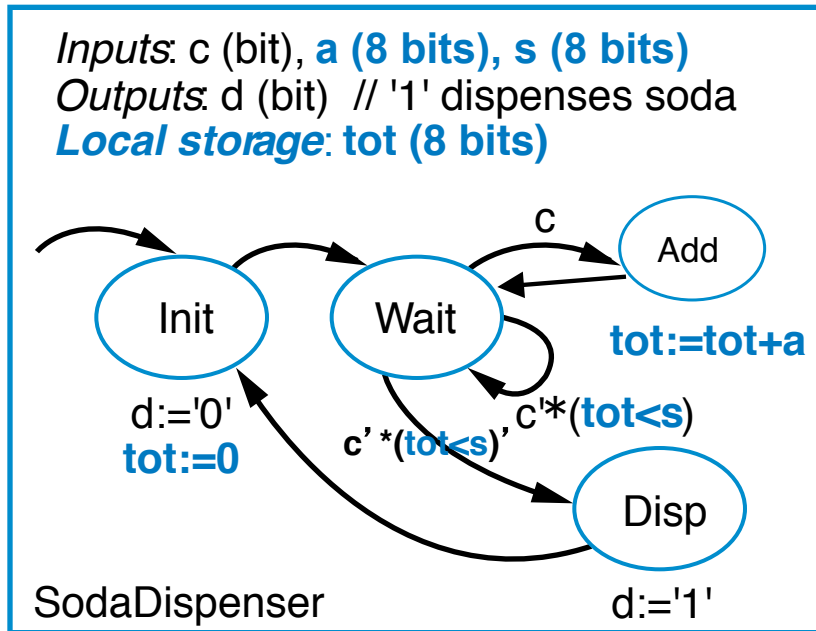
System (RTL) Design

CSE 140: Components and Design Techniques for Digital Systems

Diba Mirza

Dept. of Computer Science and Engineering
University of California, San Diego

Design the datapath (2A)

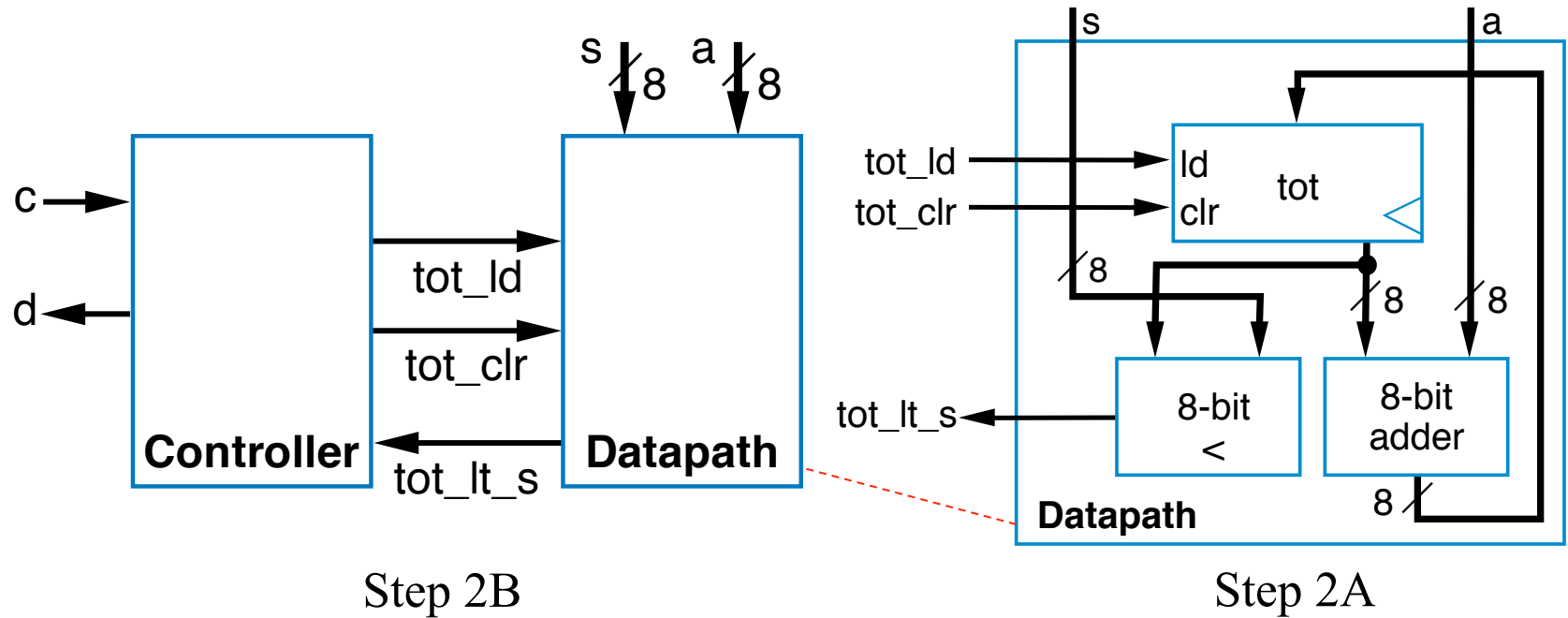


Step 2A

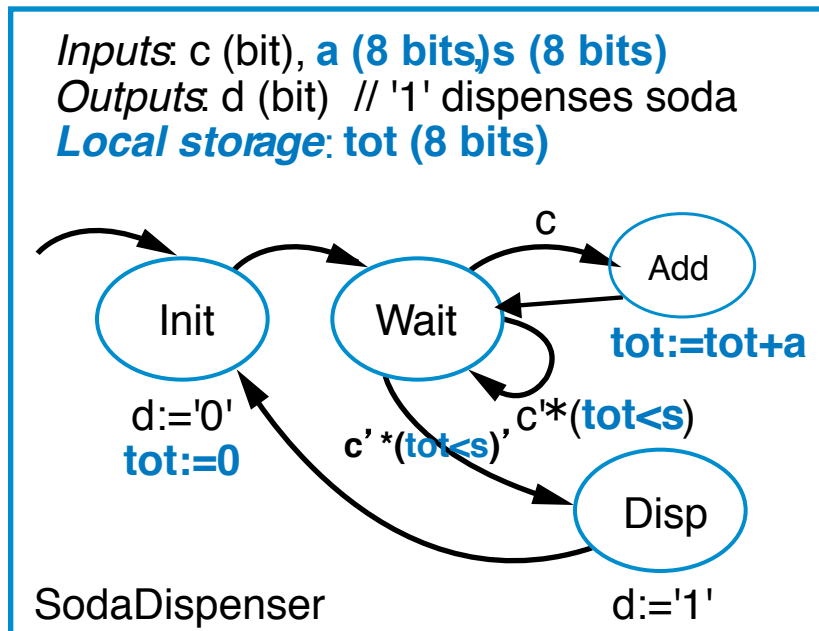
Q: According to the current design, under which of the following conditions does the register output 'tot' change?

- A. Whenever the value of the coin inserted ('a') changes
- B. Whenever the cost of the soda ('s') changes
- C. When the signal tot_ld becomes high
- D. When the signal tot_clr becomes high

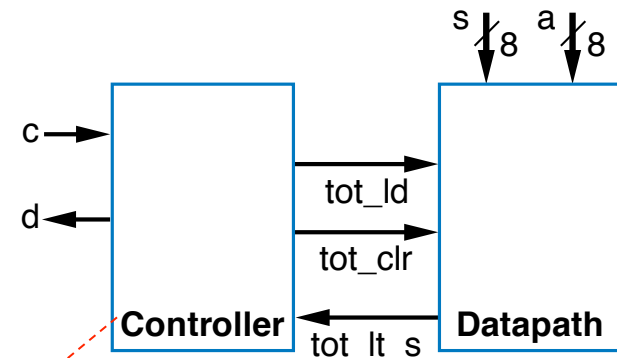
Connect data path to controller(2B)



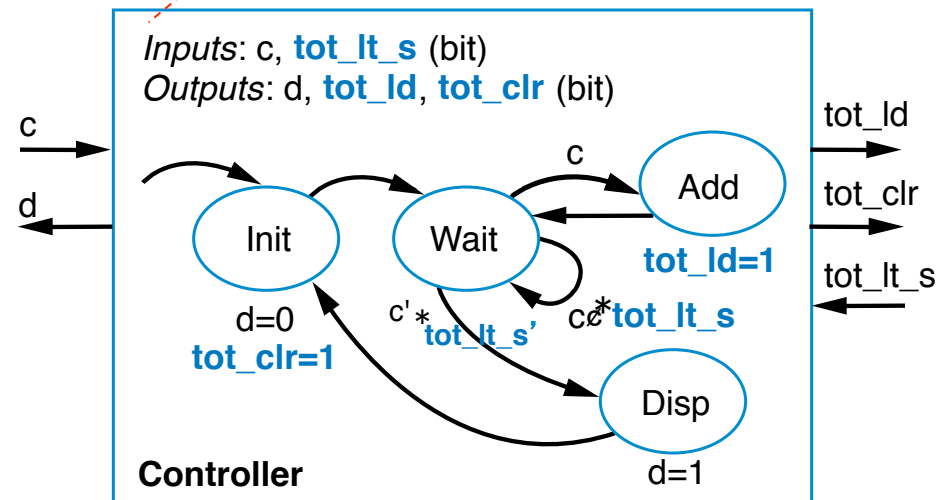
Design control path FSM



Step 1



Step 2B

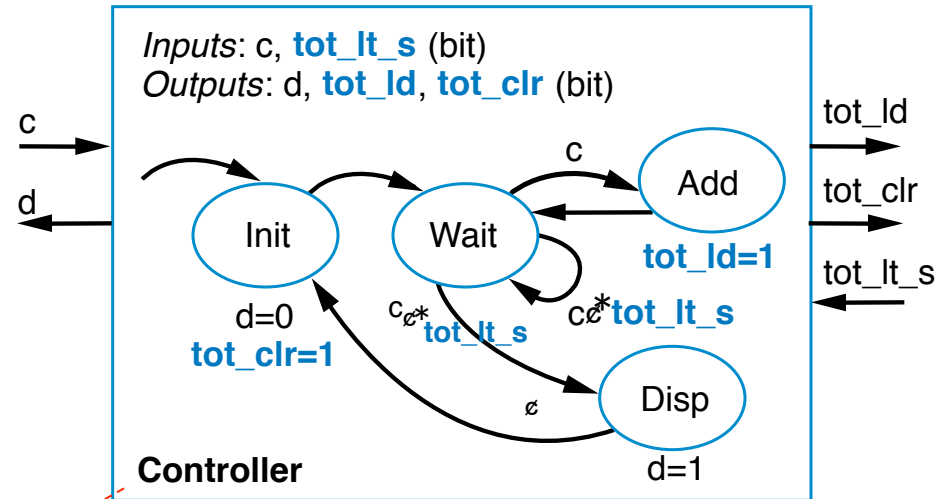


Step 2C

Source: Vahid

Implement controller FSM

	s1	s0	c	tot_lt_s	n1	n0	d	tot_ld	tot_clr
Init	0	0	0	0	0	1	0	0	1
	0	0	0	1	0	1	0	0	1
	0	0	1	0	0	1	0	0	1
	0	0	1	1	0	1	0	0	1
Wait	0	1	0	0	1	1	0	0	0
	0	1	0	1	0	1	0	0	0
	0	1	1	0	1	0	0	0	0
	0	1	1	1	1	0	0	0	0
Add	1	0	0	0	0	1	0	1	0
		...				...			
Disp	1	1	0	0	0	0	1	0	0
		...				...			

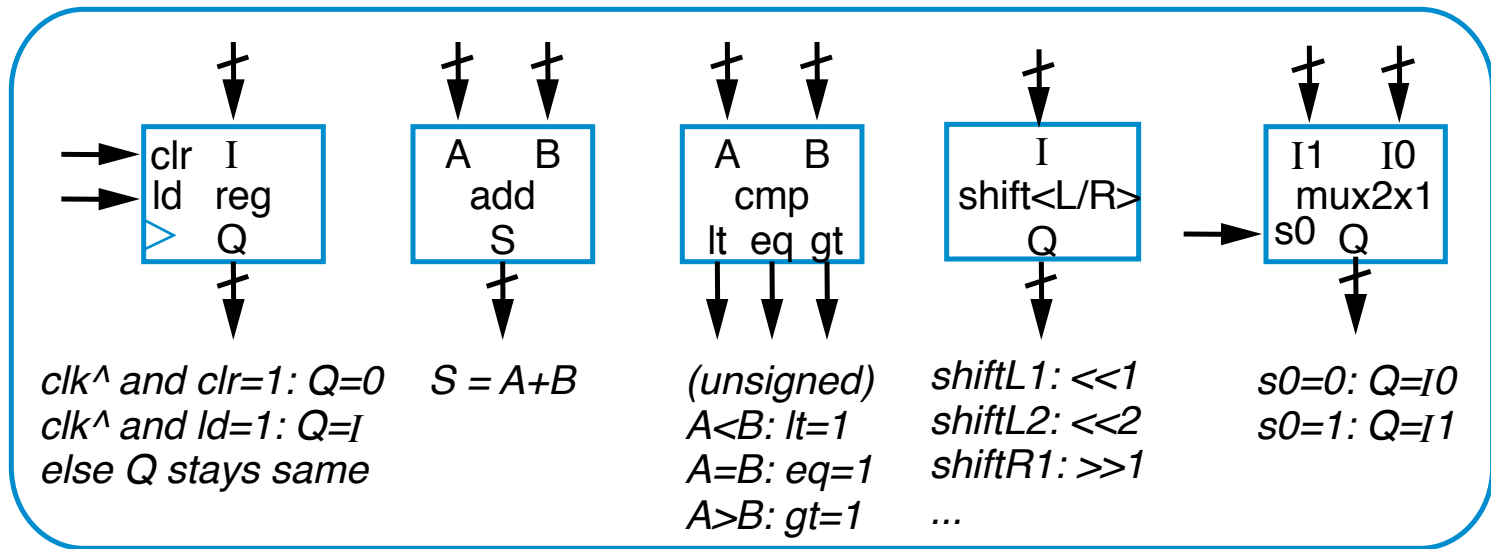


Step 2C

Use controller (FSM) design process from previous lectures to complete the design

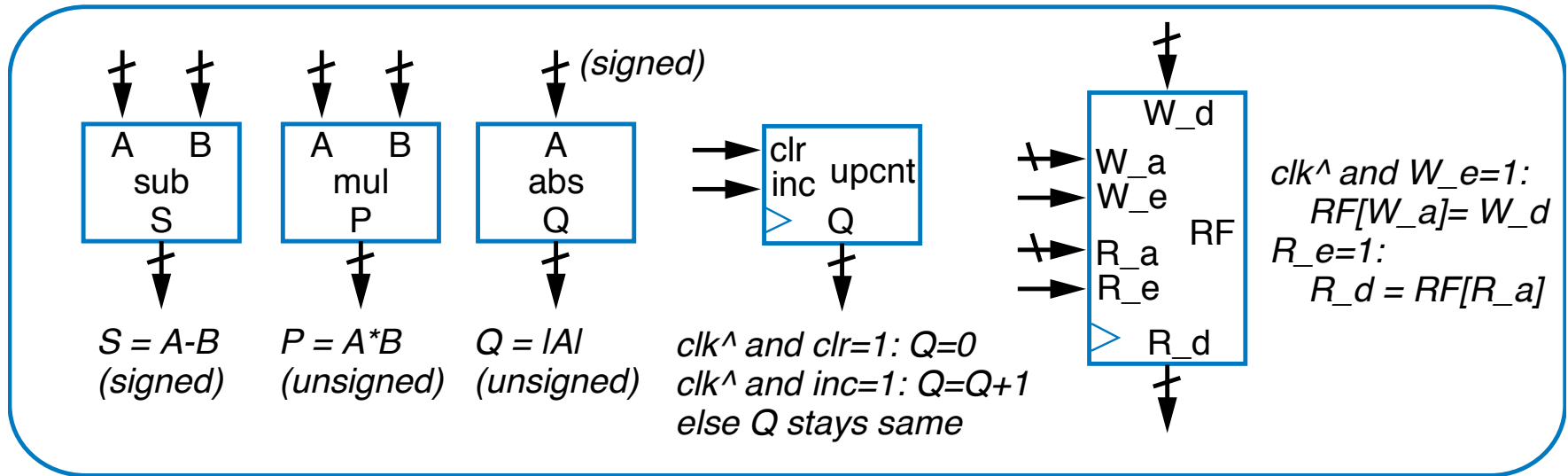
Datapath components

- Need component library from which to choose

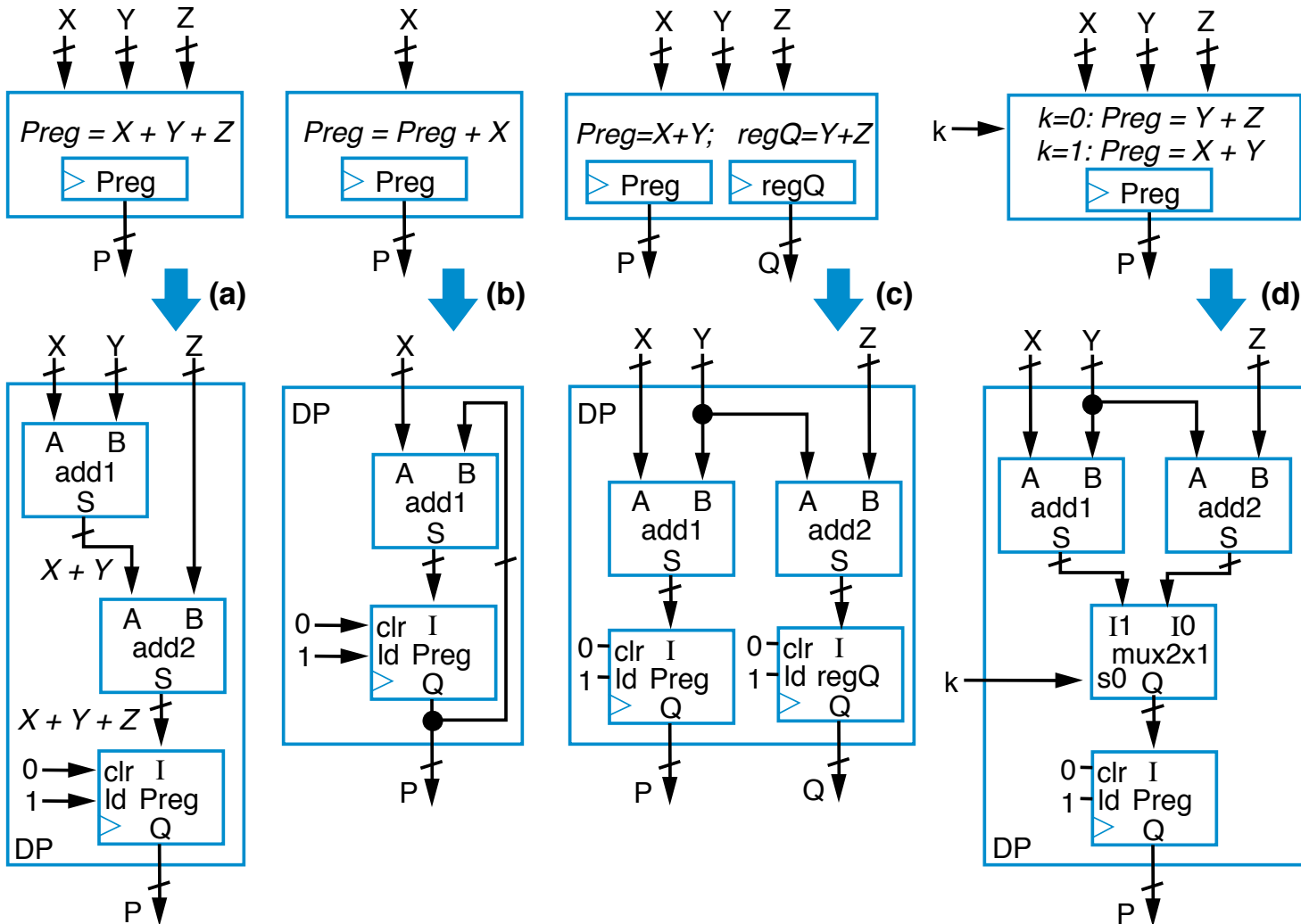


More RTL Design

- Additional datapath components



Step 2A: Datapath—Simple Examples



Design Process

- Describe system in programs
- Data subsystem
 - List data operations
 - Map operations to functional blocks
 - Add interconnect for data transport
 - Input control signals and output conditions
- Control Subsystem
 - Derive the sequence according to the hardware program
 - Create the sequential machine
 - Input conditions and output control signals

Example: Multiplication

Arithmetic

$$Z = X \times Y$$

- $M = 0$
- For $i = n-1$ to 0
 - If $Y_i = 1$, $M = M + X * 2^i$
- $Z = M$

Input X, Y

Output Z

Variable M, i

- $M = 0$
- For $i = n-1$ to 0
 - If $Y_{n-1} = 1$, $M = M + X$
 - Shift Y left by one bit
 - If $i \neq 0$, shift M left by one bit
- $Z = M$

Implementation: Example

Multiply(X, Y, Z, start, done)

{ Input X[15:0], Y[15:0] type bit-vector,

start type boolean;

Local-Object A[15:0], B[15:0], M[31:0], i[4:0] type bit-vector;

Output Z[31:0] type bit-vector,

done type boolean;

S0: If start' goto S0 || done $\leftarrow$ 1;

S1: A $\leftarrow$ X || B $\leftarrow$ Y || i $\leftarrow$ 0 || M $\leftarrow$ 0 || done $\leftarrow$ 0;

S2: If $B_{15} = 0$ goto S4 || i $\leftarrow$ i+1;

S3: M $\leftarrow$ M+A;

S4: if i $\geq$ 16, goto S6

S5: M $\leftarrow$ Shift(M,L,1) || B $\leftarrow$ Shift(B,L,1) || goto S2;

S6: Z: $\leftarrow$ M || done $\leftarrow$ 1 || goto S0;

}

Step 0: Syntax

S1: $A \leftarrow X \parallel B \leftarrow Y \parallel i \leftarrow 0 \parallel M \leftarrow 0 \parallel \text{done} \leftarrow 0;$

S2: If $B_{15} = 0$ goto S4 $\parallel i \leftarrow i+1;$

S3: $M \leftarrow M+A;$

S5: $M \leftarrow \text{Shift}(M, L, 1) \parallel B \leftarrow \text{Shift}(B, L, 1) \parallel \text{goto S2};$

S6: $Z \leftarrow M \parallel \text{done} \leftarrow 1 \parallel \text{goto S0};$

Step 1: Identify Input and Output of data and control subsystems

Multiply(X, Y, Z, start, done)

{ Input: $X[15:0]$, $Y[15:0]$ type bit-vector,

$start$ type boolean;

Local-Object : $A[15:0]$, $B[15:0]$, $M[31:0]$,
 $i[4:0]$ type bit-vector;

Output $Z[31:0]$ type bit-vector,

$done$ type boolean;

S0: If $start$ goto S0 || $done \leftarrow 1$;

S1: $A \leftarrow X$ || $B \leftarrow Y$ || $i \leftarrow 0$ || $M \leftarrow 0$ || $done \leftarrow 0$;

S2: If $B_{15} = 0$ goto S4 || $i \leftarrow i+1$;

S3: $M \leftarrow M+A$;

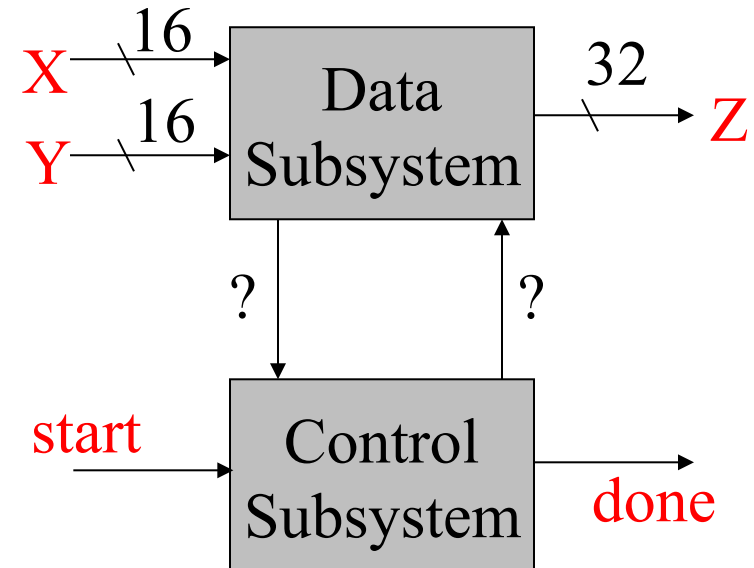
S4: if $i \geq 16$, goto S6

S5: $M \leftarrow \text{Shift}(M, L, 1)$ || $B \leftarrow \text{Shift}(B, L, 1)$ || goto S2;

S6: $Z \leftarrow M$ || $done \leftarrow 1$ || goto S0;

}

$$Z=XY$$



Step 2a: Identify Data Subsystem Operations

Multiply(X, Y, Z, start, done)

{ Input: X[15:0], Y[15:0] type bit-vector,
start type boolean;

Local-Object : A[15:0], B[15:0], M[31:0],
i[4:0] type bit-vector;

Output Z[31:0] type bit-vector,
done type boolean;

S0: If start goto S0 || done $\leftarrow$ 1;

S1: $A \leftarrow X \parallel B \leftarrow Y \parallel i \leftarrow 0 \parallel M \leftarrow 0$ || done $\leftarrow$ 0;

S2: If $B_{15} = 0$ goto S4 || $i \leftarrow i+1$;

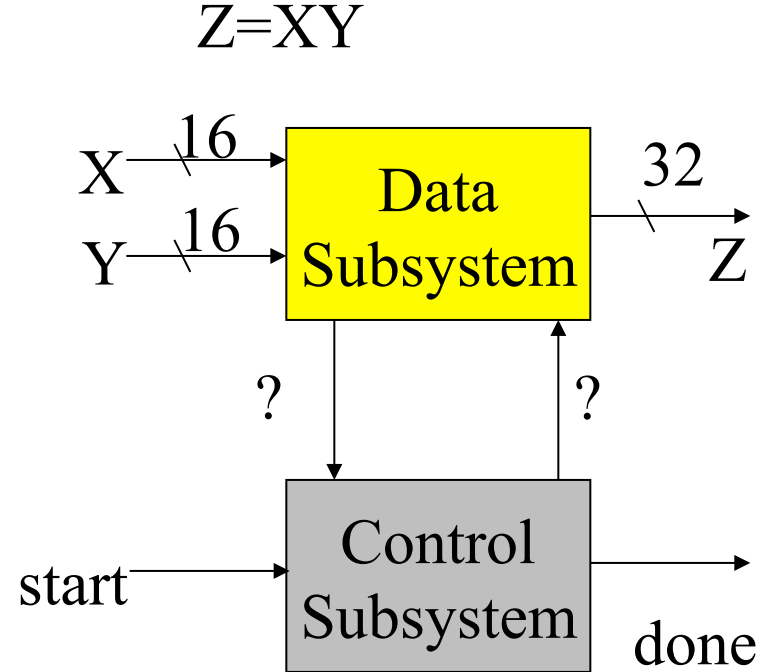
S3: $M \leftarrow M+A$;

S4: if $i \geq 16$, goto S6

S5: $M \leftarrow \text{Shift}(M, L, 1) \parallel B \leftarrow \text{Shift}(B, L, 1)$ || goto S2;

S6: $Z \leftarrow M$ || done $\leftarrow$ 1 || goto S0;

}



Step 2b: Map Data Operations to functional modules in data library

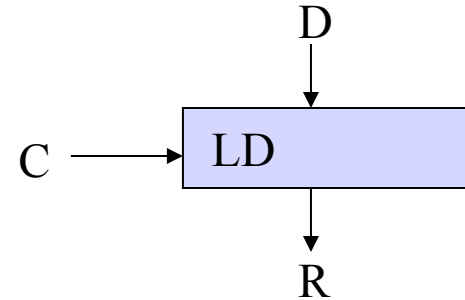
Multiply(X, Y, Z, start, done)

```
{ Input:  X[15:0], Y[15:0] type bit-vector,
        start type boolean;
  Local-Object : A[15:0], B[15:0], M[31:0],
                i[4:0] type bit-vector;
  Output Z[31:0] type bit-vector,
        done type boolean;
S0: If start' goto S0 || done ← 1;
S1: A ← X || B ← Y || i ← 0 || M ← 0 || done ← 0;
S2: If B15 = 0 goto S4 || i ← i + 1;
S3: M ← M + A;
S4: if i ≥ 16, goto S6
S5: M ← Shift(M, L, 1) || B ← Shift(B, L, 1) || goto S2;
S6: Z ← M || done ← 1 || goto S0;
}
```

	operation
A ← X	A ← Load (X)
B ← Y	B ← Load (Y)
M ← 0	M ← Clear(M)
i ← 0	i ← Clear(i)
i ← i + 1	i ← INC(i)
M ← M + A	M ← Add(M, A)
M ← Shift(M, L, 1)	M ← SHL(M)
B ← Shift(B, L, 1)	B ← SHL(B)
Z ← M	Wires

Step 2c: Implement the data subsystem

Registers: If C then $R \leftarrow D$



operation

$A \leftarrow \text{Load}(X)$

$B \leftarrow \text{Load}(Y)$

$B \leftarrow \text{SHL}(B)$

$M \leftarrow \text{Clear}(M)$

$M \leftarrow \text{Add}(M, A)$

$M \leftarrow \text{SHL}(M)$

$i \leftarrow \text{Clear}(i)$

$i \leftarrow \text{INC}(i)$

Function Modules: Registers to store data (A, B, M)

operation

$A \leftarrow \text{Load}(X)$

$B \leftarrow \text{Load}(Y)$

$B \leftarrow \text{SHL}(B)$

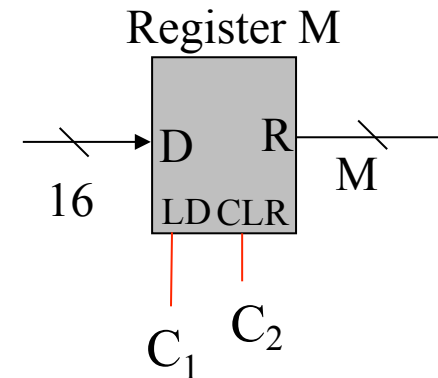
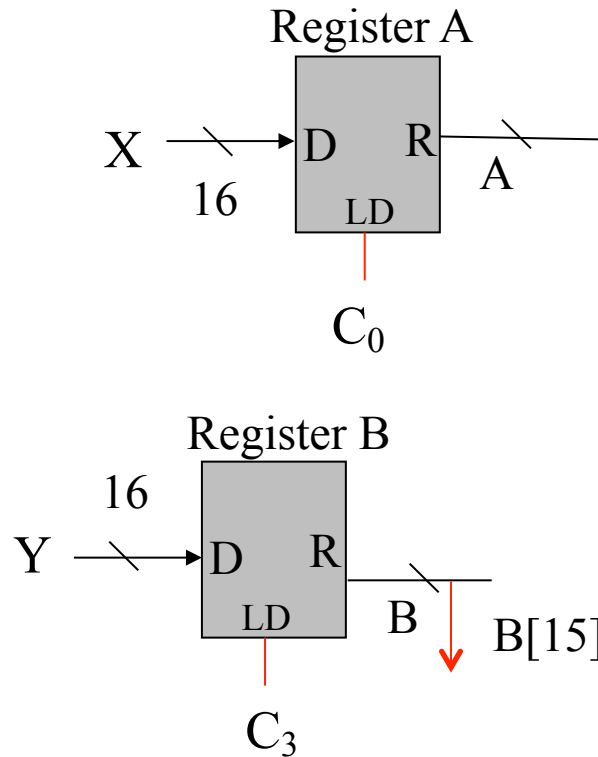
$M \leftarrow \text{Clear}(M)$

$M \leftarrow \text{Add}(M, A)$

$M \leftarrow \text{SHL}(M)$

$i \leftarrow \text{Clear}(i)$

$i \leftarrow \text{INC}(i)$



Function Modules: Adder, Shifter (updates to M)

operation

$A \leftarrow \text{Load}(X)$

$B \leftarrow \text{Load}(Y)$

$B \leftarrow \text{SHL}(B)$

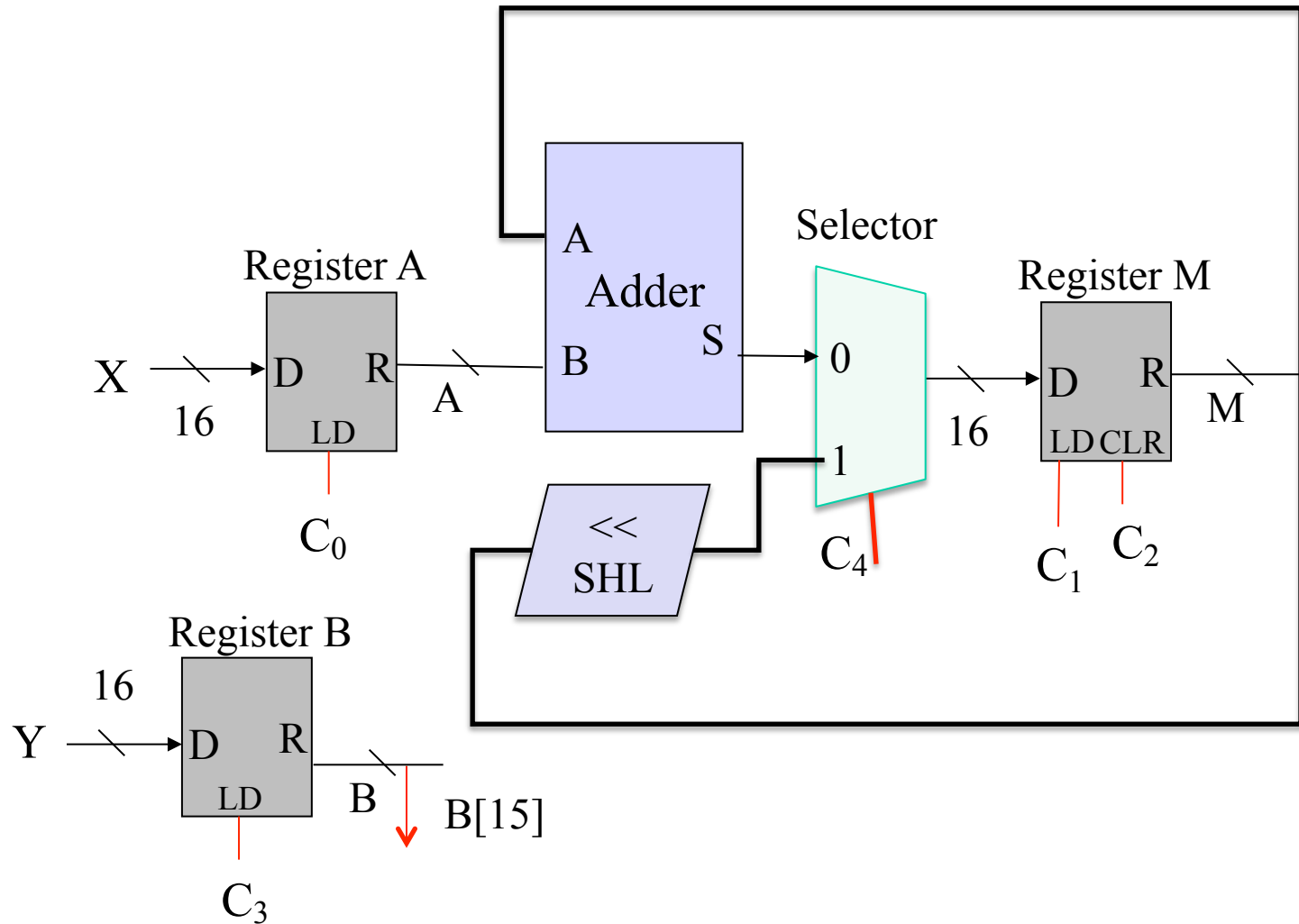
$M \leftarrow \text{Clear}(M)$

$M \leftarrow \text{Add}(M, A)$

$M \leftarrow \text{SHL}(M)$

$i \leftarrow \text{Clear}(i)$

$i \leftarrow \text{INC}(i)$



Function Modules: Adder, Shifter (updates to B)

operation

$A \leftarrow \text{Load}(X)$

$B \leftarrow \text{Load}(Y)$

$B \leftarrow \text{SHL}(B)$

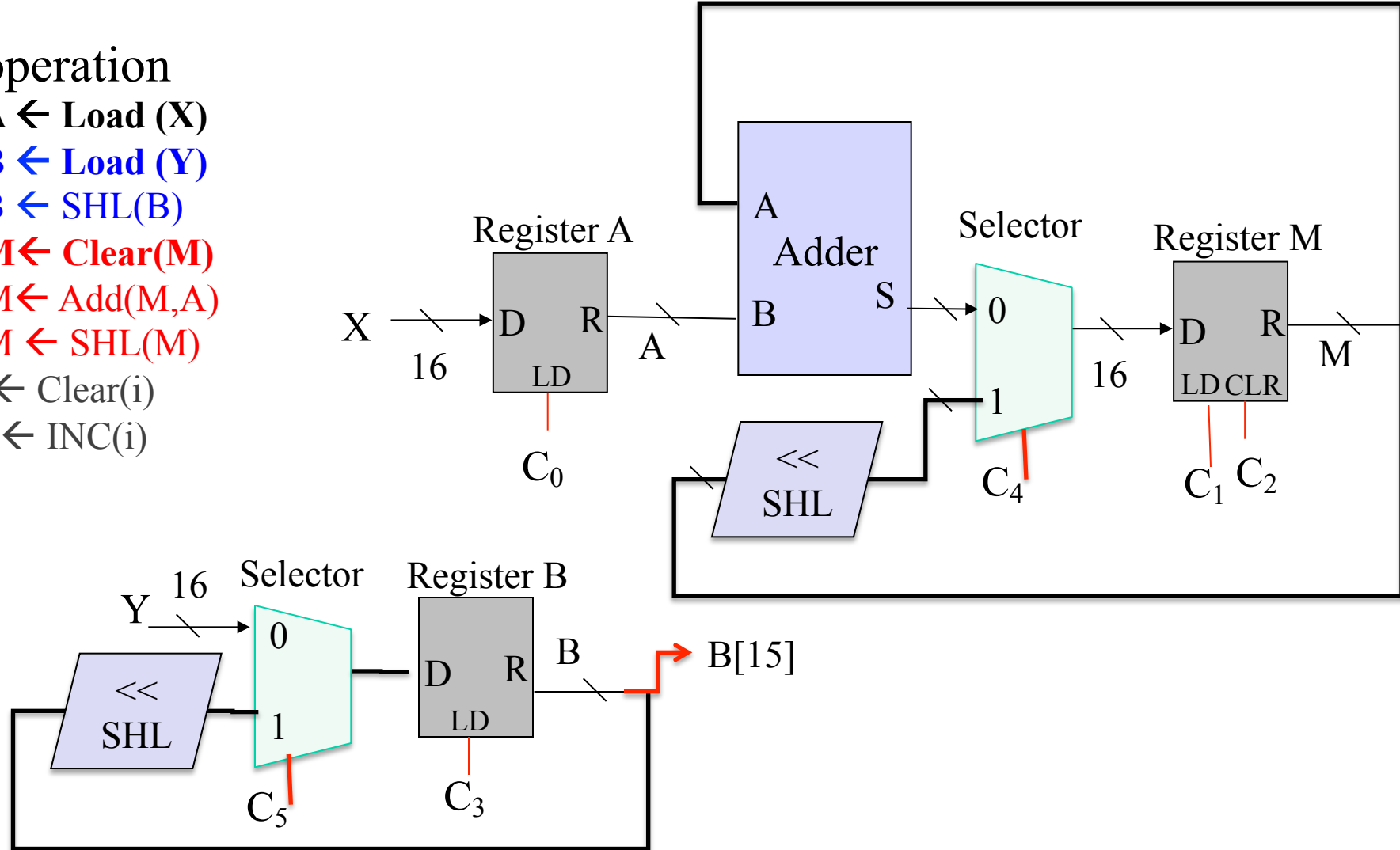
$M \leftarrow \text{Clear}(M)$

$M \leftarrow \text{Add}(M, A)$

$M \leftarrow \text{SHL}(M)$

$i \leftarrow \text{Clear}(i)$

$i \leftarrow \text{INC}(i)$



Function Modules: Counter (updates to i)

operation

$A \leftarrow \text{Load}(X)$

$B \leftarrow \text{Load}(Y)$

$B \leftarrow \text{SHL}(B)$

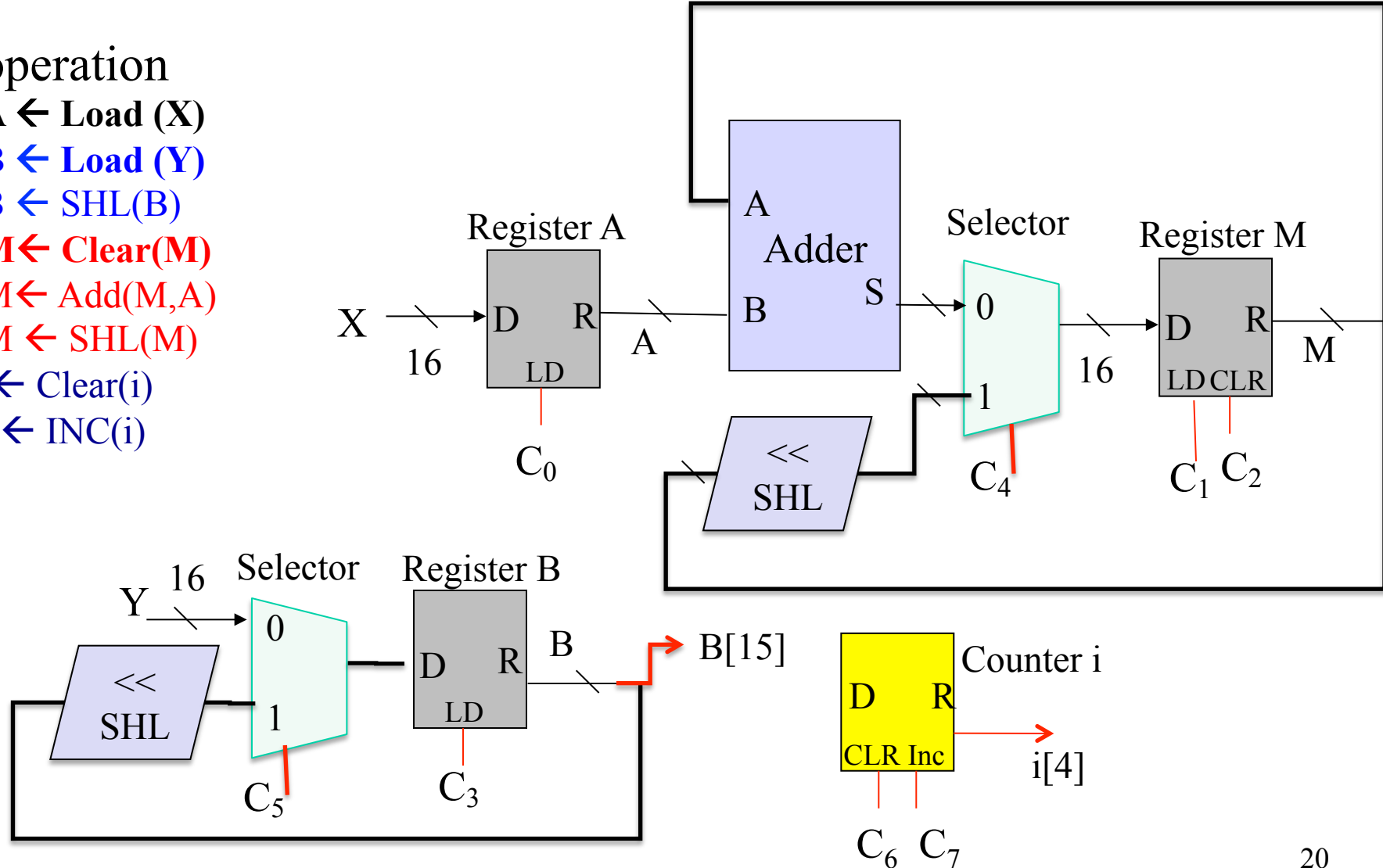
$M \leftarrow \text{Clear}(M)$

$M \leftarrow \text{Add}(M, A)$

$M \leftarrow \text{SHL}(M)$

$i \leftarrow \text{Clear}(i)$

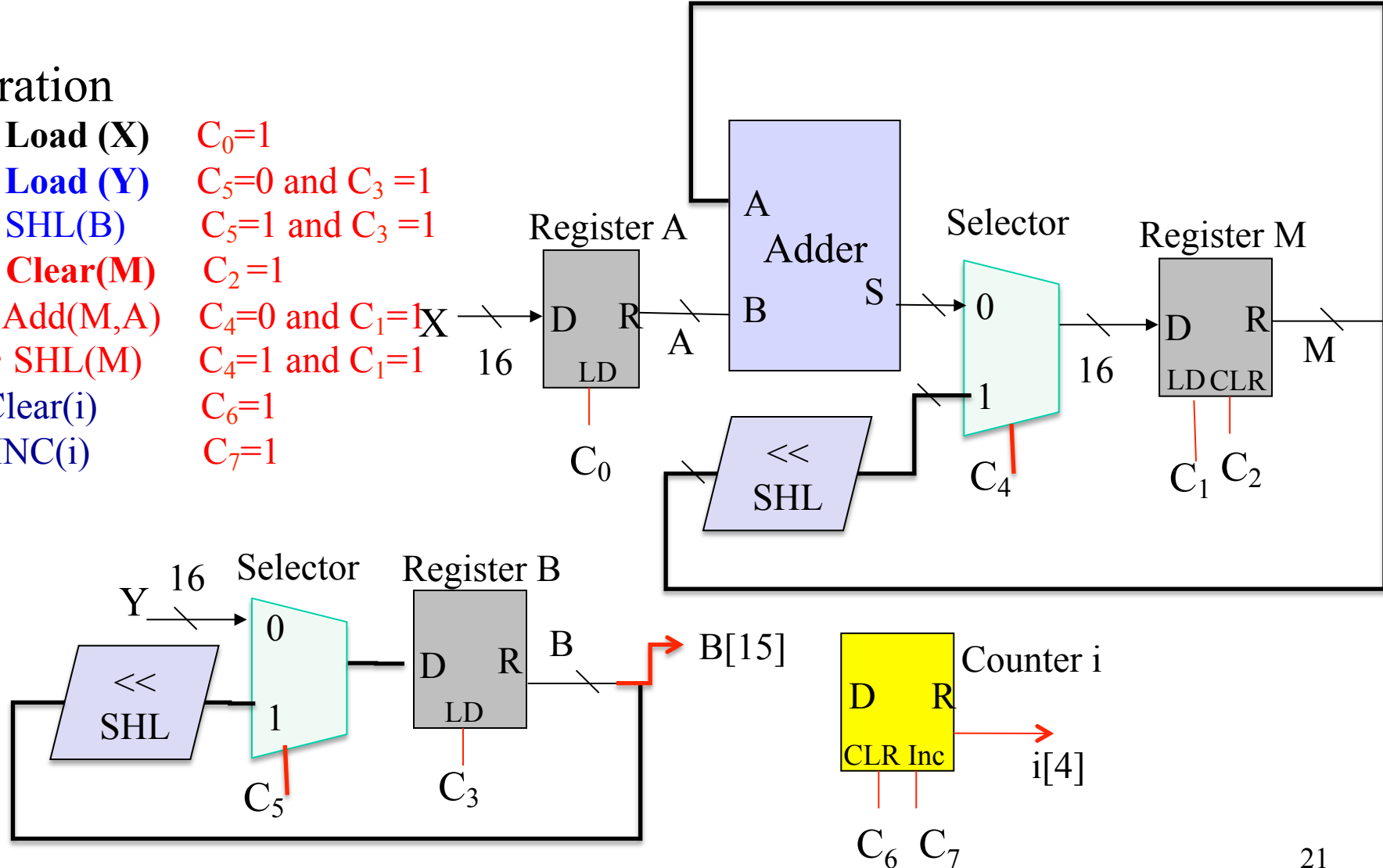
$i \leftarrow \text{INC}(i)$

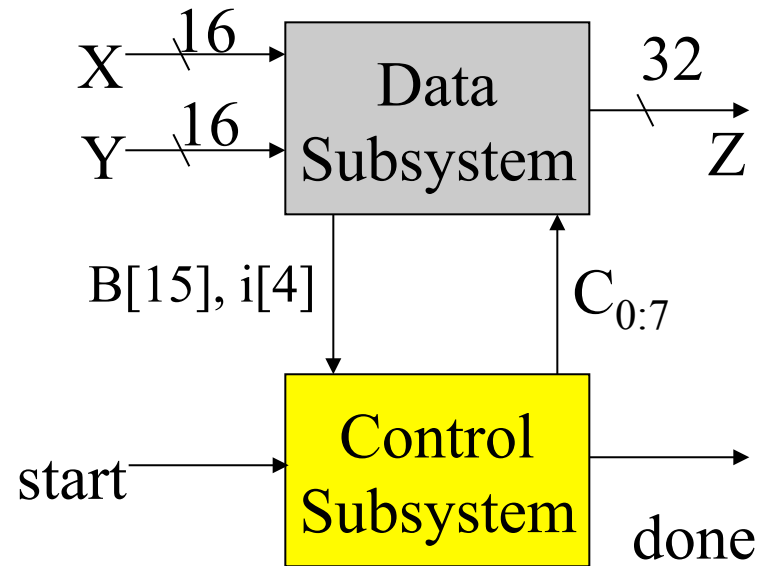


Step 2d: Map Control Signals to Operations

operation

A $\leftarrow$ Load (X) $C_0=1$
B $\leftarrow$ Load (Y) $C_5=0$ and $C_3=1$
B $\leftarrow$ SHL(B) $C_5=1$ and $C_3=1$
M $\leftarrow$ Clear(M) $C_2=1$
M $\leftarrow$ Add(M,A) $C_4=0$ and $C_1=1$
M $\leftarrow$ SHL(M) $C_4=1$ and $C_1=1$
i $\leftarrow$ Clear(i) $C_6=1$
i $\leftarrow$ INC(i) $C_7=1$





Design the Control Subsystem

Multiply(X, Y, Z, start, done)

```
{
S0: If start' goto S0 || done ← 1;
S1: A ← X || B ← Y || i ← 0 || M ← 0 || done ← 0;
S2: If B15 = 0 goto S4 || i ← i + 1;
S3: M ← M + A;
S4: if i ≥ 16, goto S6
S5: M ← Shift(M, L, 1) || B ← Shift(B, L, 1) || goto S2;
S6: Z ← M || done ← 1 || goto S0
}
```

Multiply(X, Y, Z, start, done)

```
{
S0: If start' goto S0 || done ← 1;
S1: C0 = 1 || C5 = 0 and C3 = 1 || C6 = 1 || C2 = 1 || done ← 0;
S2: If B15 = 0 goto S4 || C7 = 1;
S3: C4 = 0 and C1 = 1;
S4: if i[4], goto S6
S5: C4 = 1 and C1 = 1 || C5 = 1 and C3 = 1 || goto S2;
S6: Z ← M || done ← 1 || goto S0
}
```

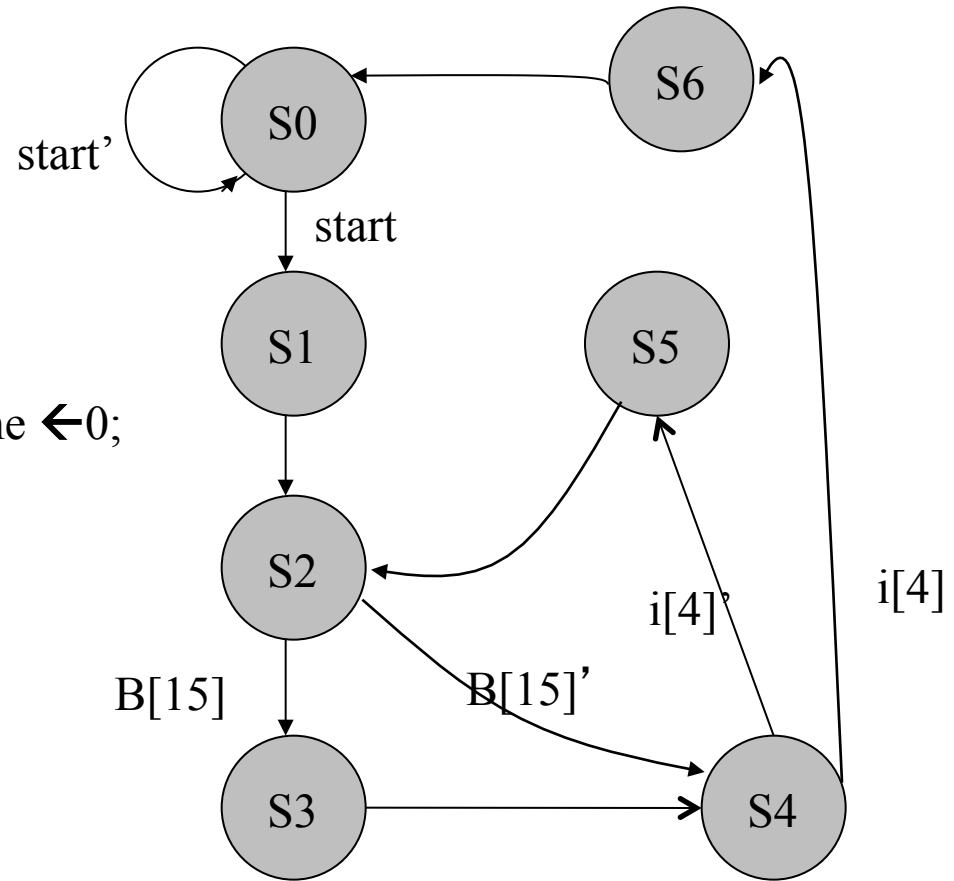
operation

A ← Load (X)	C ₀ = 1
B ← Load (Y)	C ₅ = 0 and C ₃ = 1
B ← SHL(B)	C ₅ = 1 and C ₃ = 1
M ← Clear(M)	C ₂ = 1
M ← Add(M, A)	C ₄ = 0 and C ₁ = 1
M ← SHL(M)	C ₄ = 1 and C ₁ = 1
i ← Clear(i)	C ₆ = 1
i ← INC(i)	C ₇ = 1

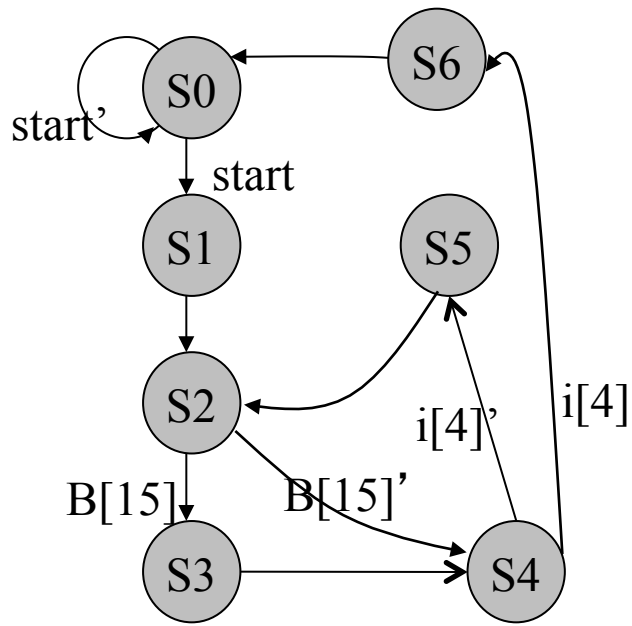
Control Subsystem

```

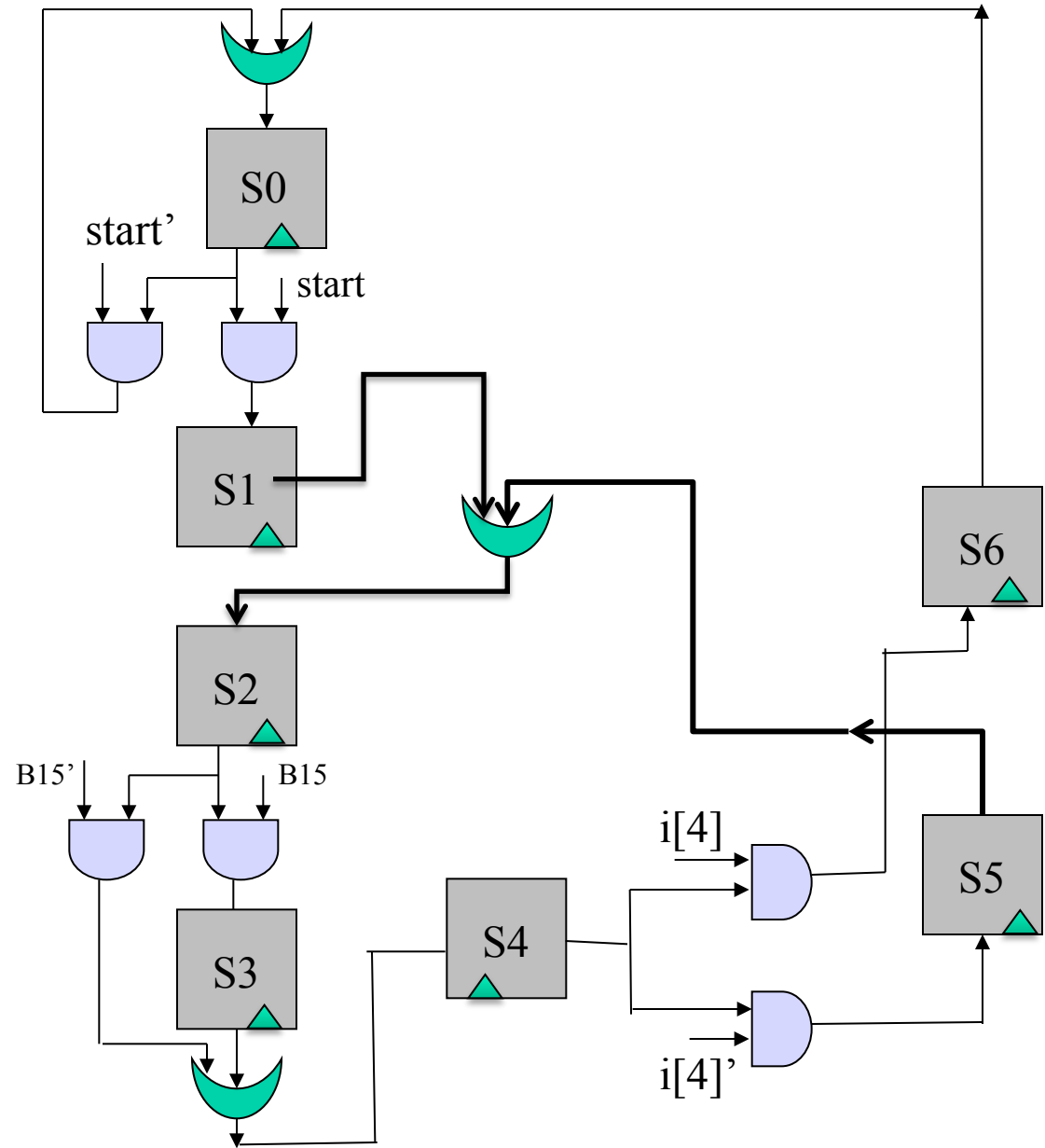
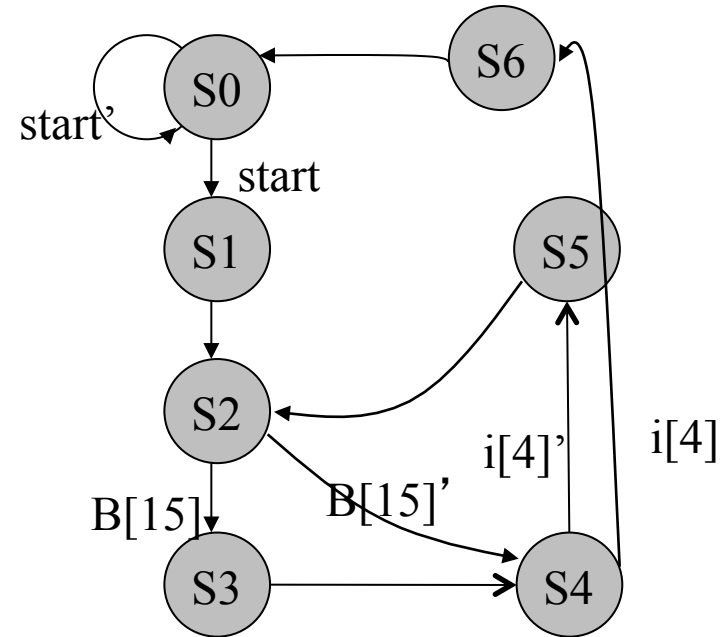
Multiply(X, Y, Z, start, done)
{
  S0: If start' goto S0 || done ← 1;
  S1: C0=1 || C5=0 and C3=1 || C6=1 || C2=1 || done ← 0;
  S2: If B15 = 0 goto S4 || C7=1;
  S3: C4=0 and C1=1;
  S4: if i[4], goto S6
  S5: C4=1 and C1=1 || C5=1 and C3=1 || goto S2;
  S6: Z ← M || done ← 1 || goto S0
}
  
```



One-Hot State Machine



One-Hot State Machine

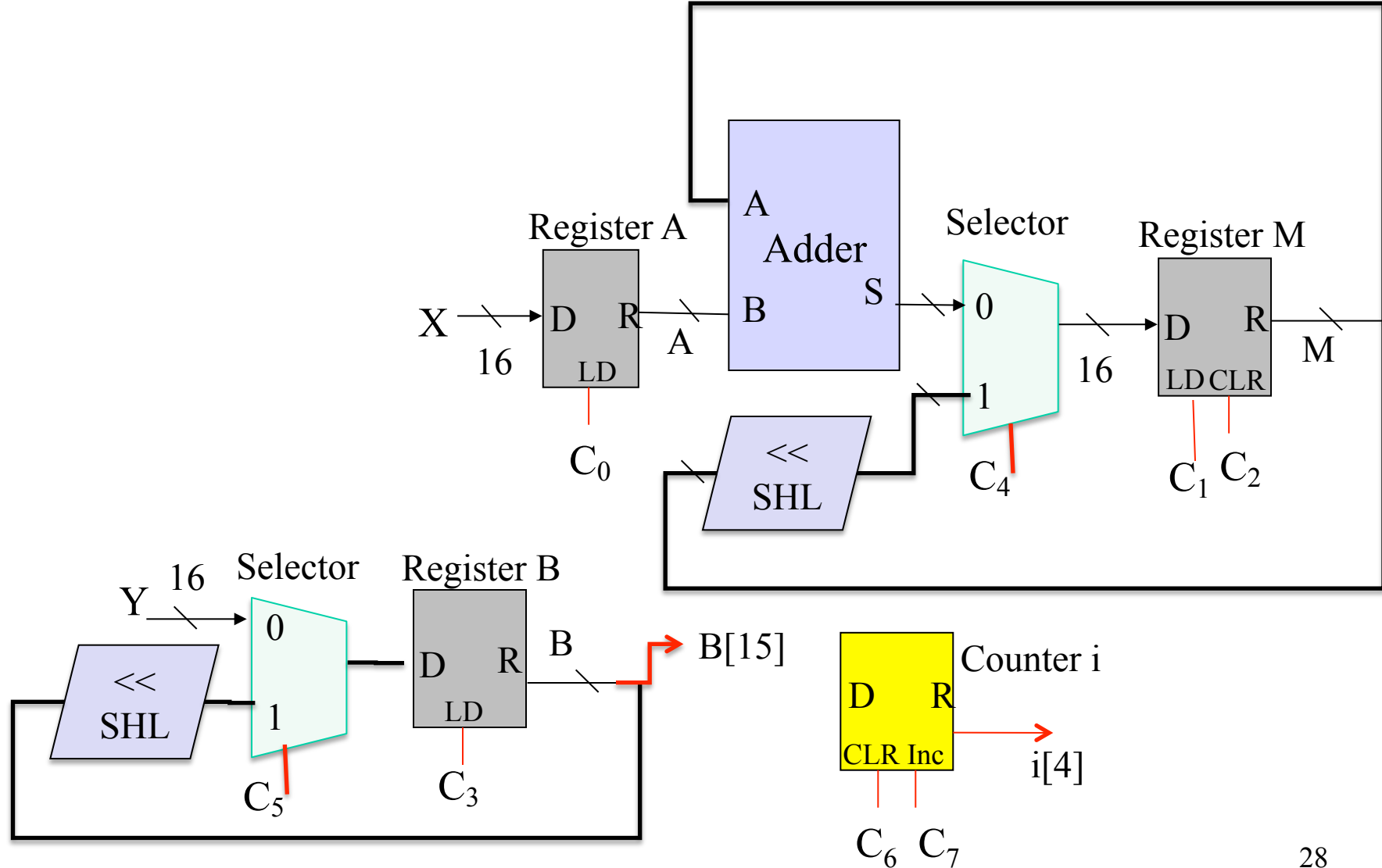


Control Subsystem: One-Hot State Machine Design

Input: State Diagram

1. Use a flip flop to replace each state.
2. Set the flip flop which corresponds to the initial state and reset the rest flip flops.
3. Use an OR gate to collect all inward edges.
4. Use a Demux to distribute the outward edges.

Data Subsystem



Multiply(X, Y, Z, start, done)

```
{
S0: If start' goto S0 || done ← 1;
S1: C0=1 || C5=0 and C3 =1 || C6=1 || C2=1 || done ← 0;
S2: If B15 = 0 goto S4 || C7=1;
S3: C4=0 and C1=1;
S4: if i[4], goto S6
S5: C4=1 and C1=1 || C5=1 and C3 =1 || goto S2;
S6: Z: ← M || done ← 1 || goto S0
}
```

	C0	C1	C2	C3	C4 (mux)	C5 (mux)	C6	C7	done
S0	0	0	0	0	X	X	0	0	1
S1	1	0	1	0	X	0	1	0	0
S2	0	0	0	1	X	X	0	1	0
S3	0	1	0	0	0	X	0	0	0
S4	0	0	0	0	X	X	0	0	0
S5	0	1	0	1	1	1	0	0	0
S6	0	0	0	0	X	X	0	0	1

	C0	C1	C2	C3	C4 (mux)	C5 (mux)	C6	C7	done
S0	0	0	0	0	X	X	0	0	1
S1	1	0	1	0	X	0	1	0	0
S2	0	0	0	1	X	X	0	1	0
S3	0	1	0	0	0	X	0	0	0
S4	0	0	0	0	X	X	0	0	0
S5	0	1	0	1	1	1	0	0	0
S6	0	0	0	0	X	X	0	0	1

One-Hot State Machine

