

Lecture 24: 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

Final Exam

- This Sat (03/14), from 3pm to 6pm.
Location will be announced on Piazza
- Cumulative, however emphasis will be on material after Midterm 1

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

{

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 B₁₅ = 0 goto S4 || i $\leftarrow$ i+1;

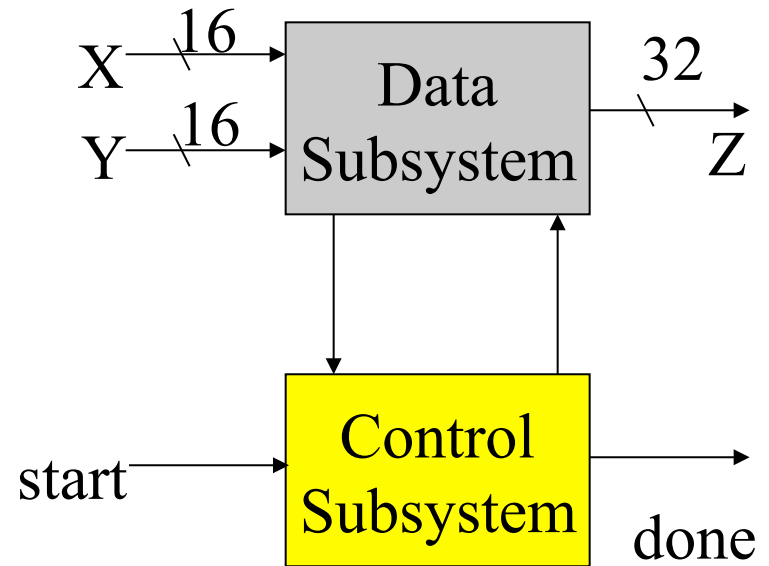
S3: M $\leftarrow$ M+A;

S4: if i >= 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

}



RTL Design (contd)

1. Design datapath
2. Identify signals to datapath from controller
3. Identify signals from datapath to controller
4. Design controller

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

{

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 B₁₅ = 0 goto S4 || i $\leftarrow$ i+1;

S3: M $\leftarrow$ M+A;

S4: if i >= 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

}

operation

A $\leftarrow$ Load (X)

B $\leftarrow$ Load (Y)

B $\leftarrow$ SHL(B)

M $\leftarrow$ Clear(M)

M $\leftarrow$ Add(M,A)

M $\leftarrow$ SHL(M)

i $\leftarrow$ Clear(i)

i $\leftarrow$ 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 $C_3=1$

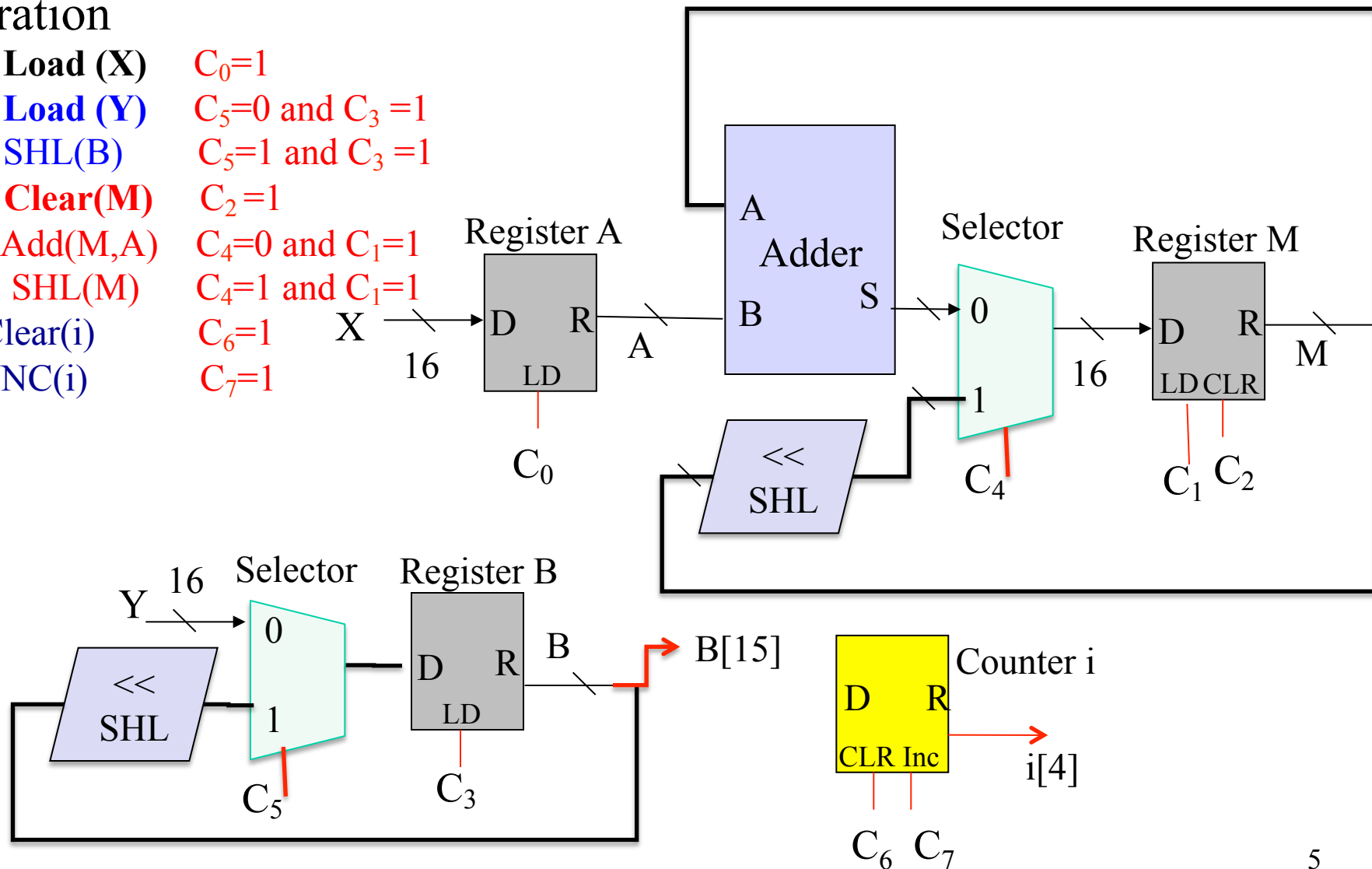
M $\leftarrow$ Clear(M) $C_2=1$

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) $C_7=1$



Control Subsystem

```
Multiply(X, Y, Z, start, done)
{
S0: If start' goto S0 || done ← 1;
S1:  $C_0=1$  ||  $C_5=0$  and  $C_3=1$  ||  $C_6=1$  ||  $C_2=1$  || done ← 0;
S2: If  $B_{15}=0$  goto S4 ||  $C_7=1$ ;
S3:  $C_4=0$  and  $C_1=1$ ;
S4: if i[4], goto S6
S5:  $C_4=1$  and  $C_1=1$  ||  $C_5=1$  and  $C_3=1$  || goto S2;
S6:  $Z \leftarrow M$  || done ← 1 || goto S0
}
```

Control Subsystem: One-Hot State Machine Design

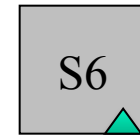
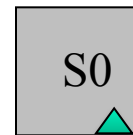
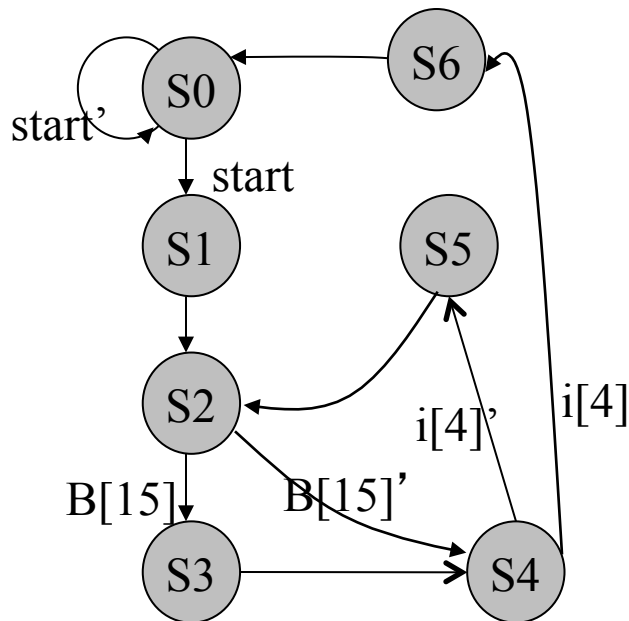
Input: State Diagram

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.

Control Subsystem: One-Hot State Machine Design

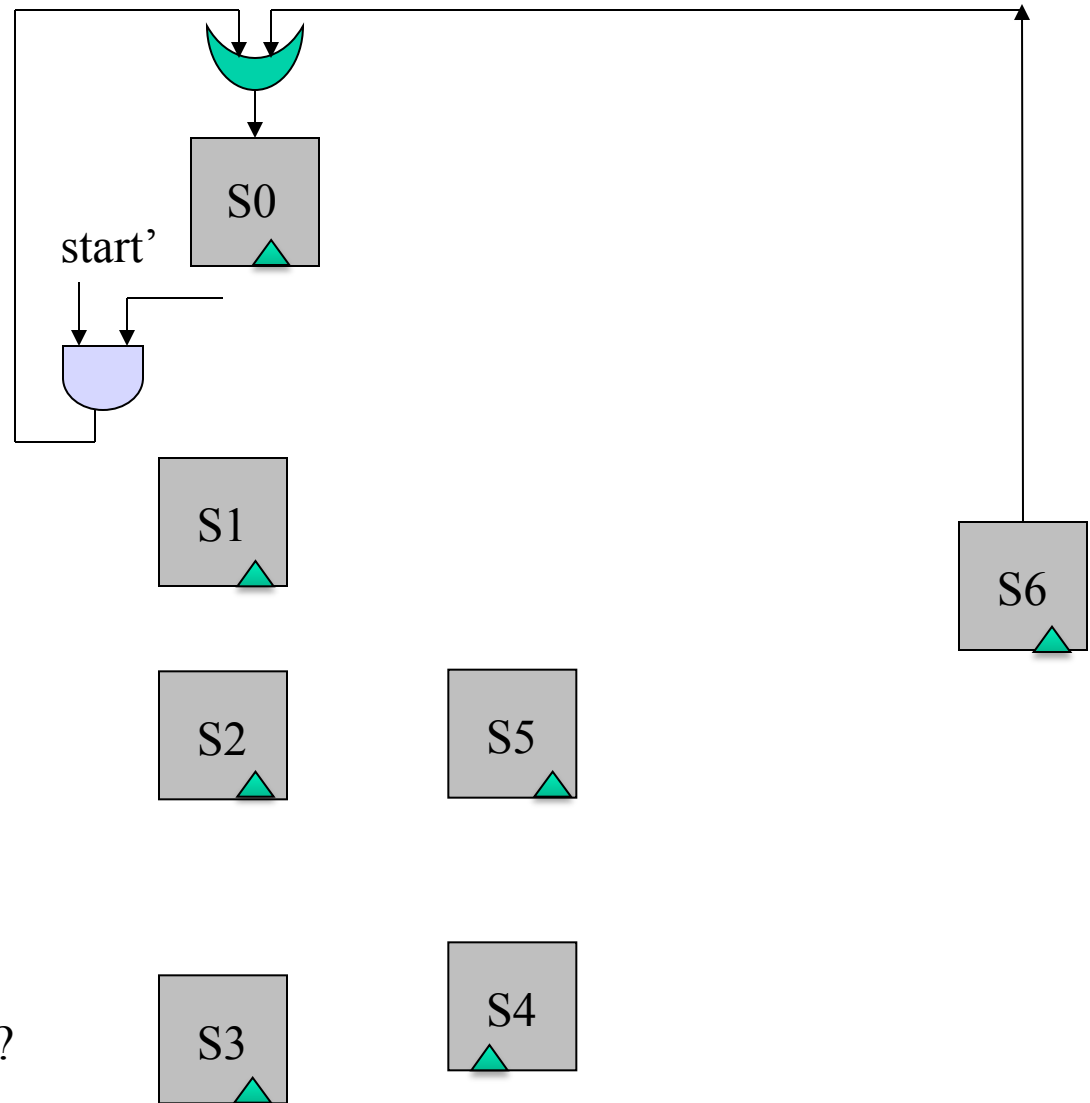
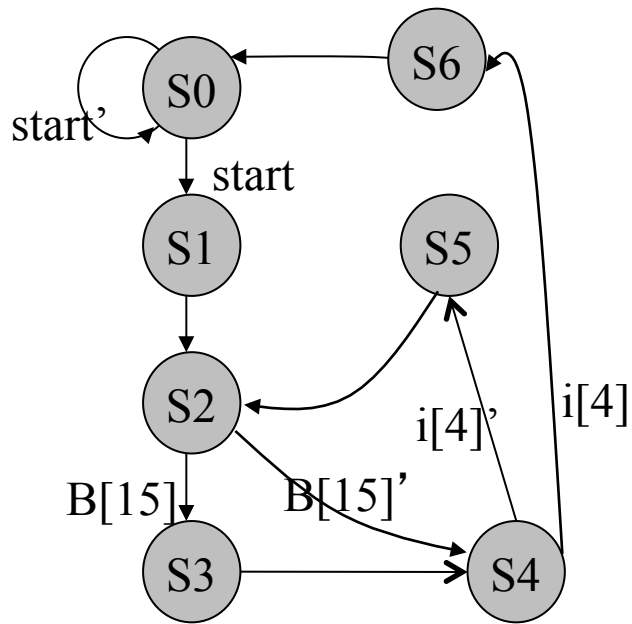
Input: State Diagram

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 (or a network of AND gates) to distribute the outward edges.



zic  *output logic*

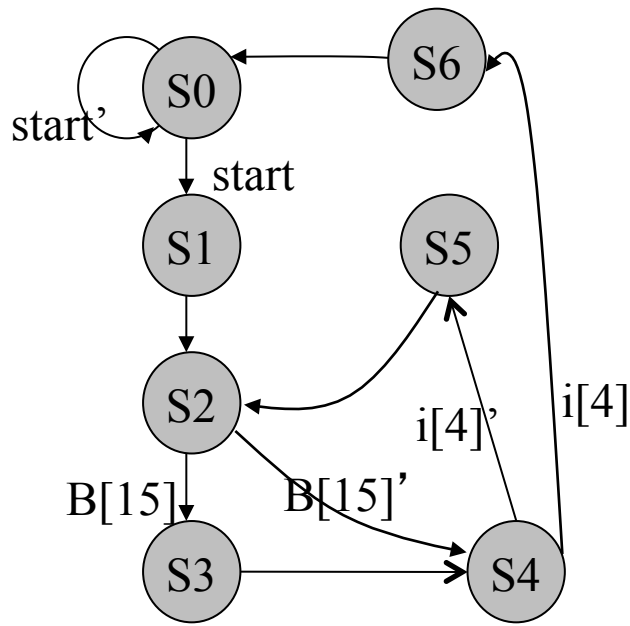
One-Hot State Machine



Is the next state logic for S1 correct?

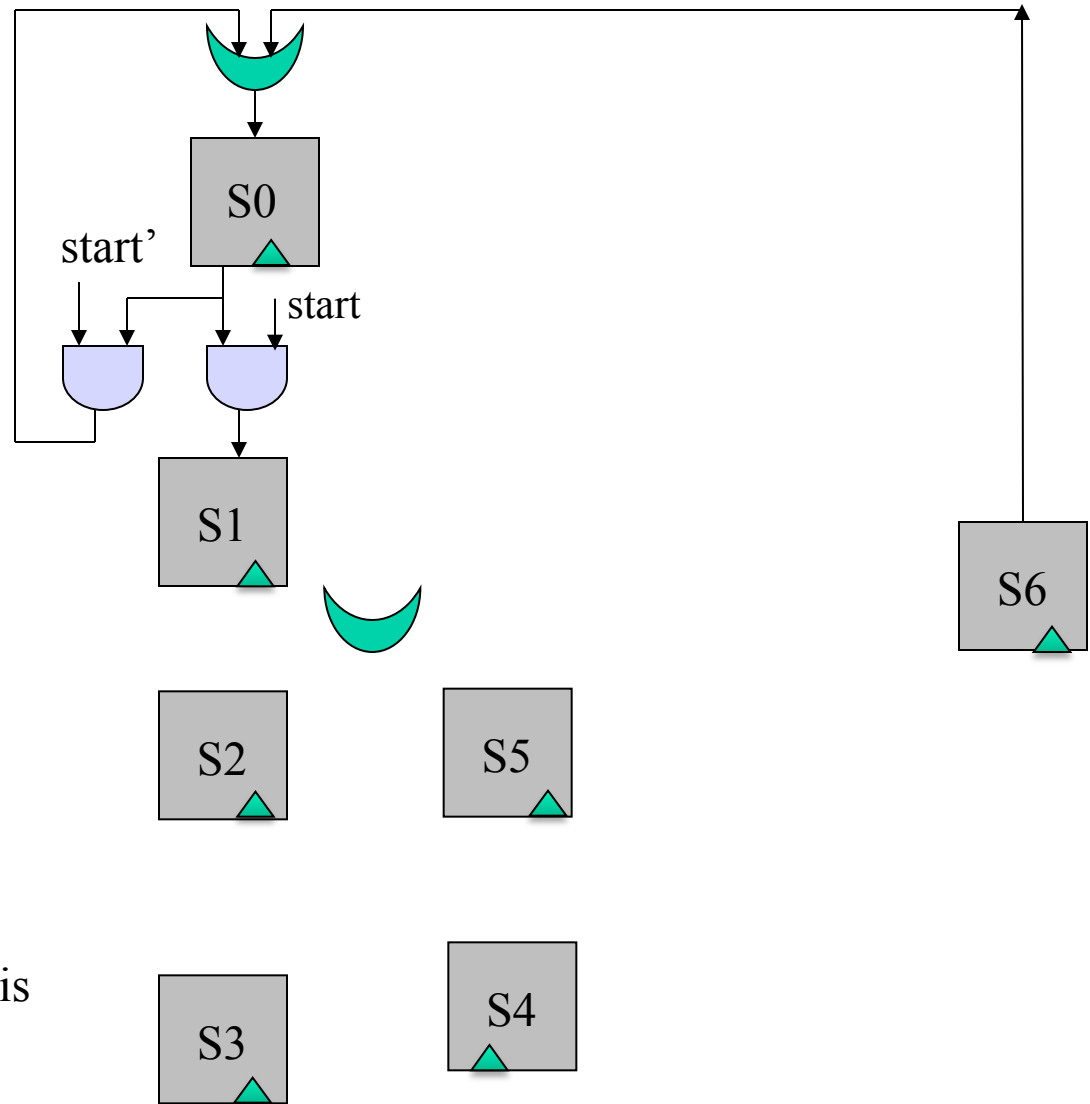
- A. Yes
- B. No

One-Hot State Machine

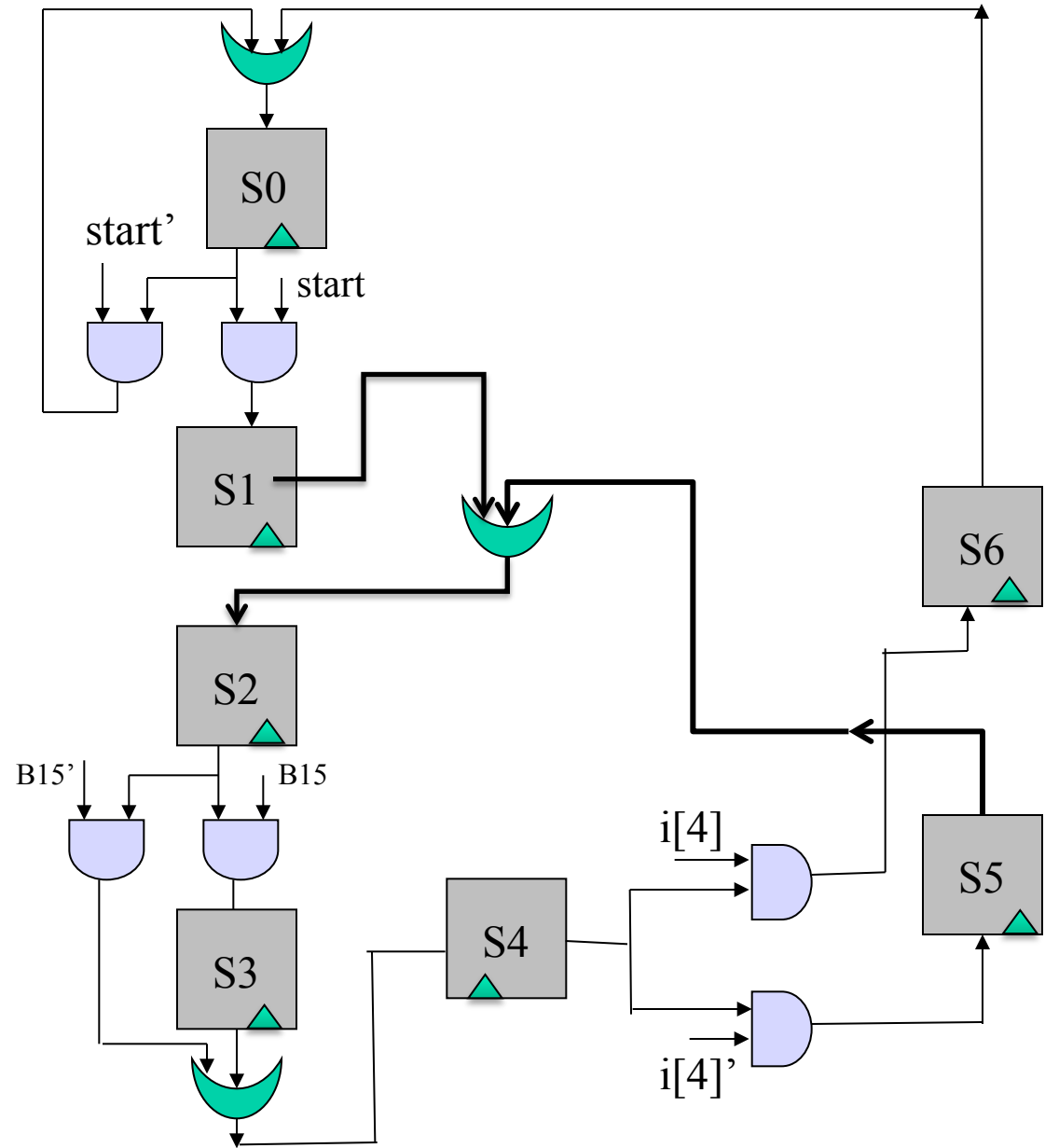
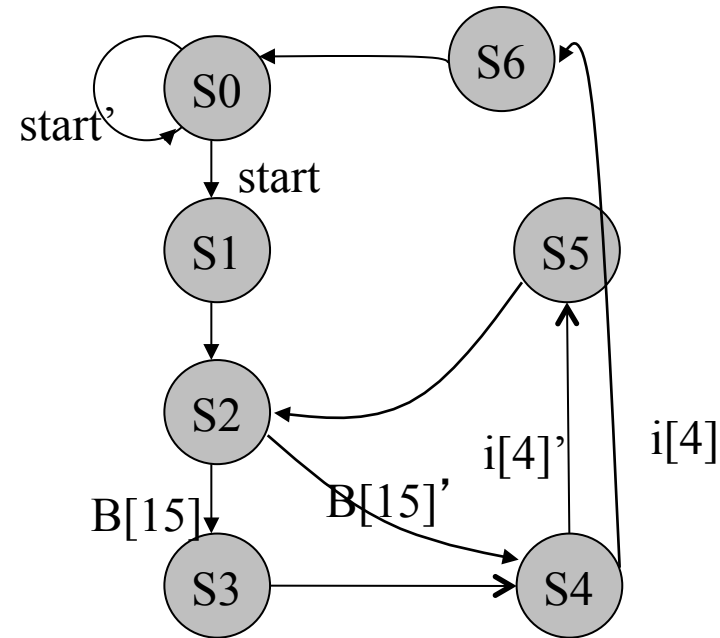


If S0 is currently 1 and start is 1, what is the next state value of (S0, S1)

- A. (0, 0)
- B. (1, 1)
- C. (1, 0)
- D. (0, 1)



One-Hot State Machine



```
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									
S1									
S2									
S3									
S4									
S5									
S6									

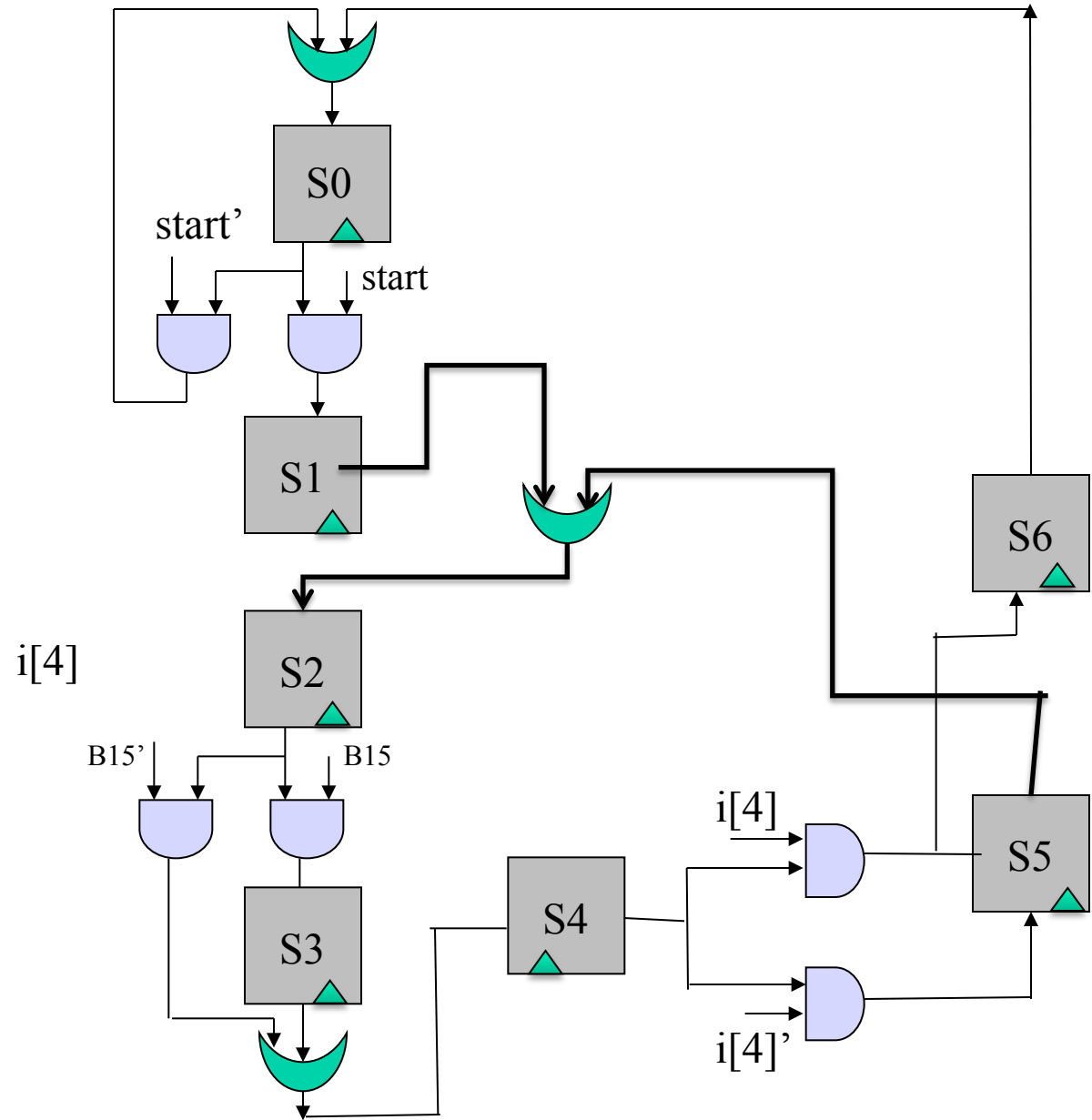
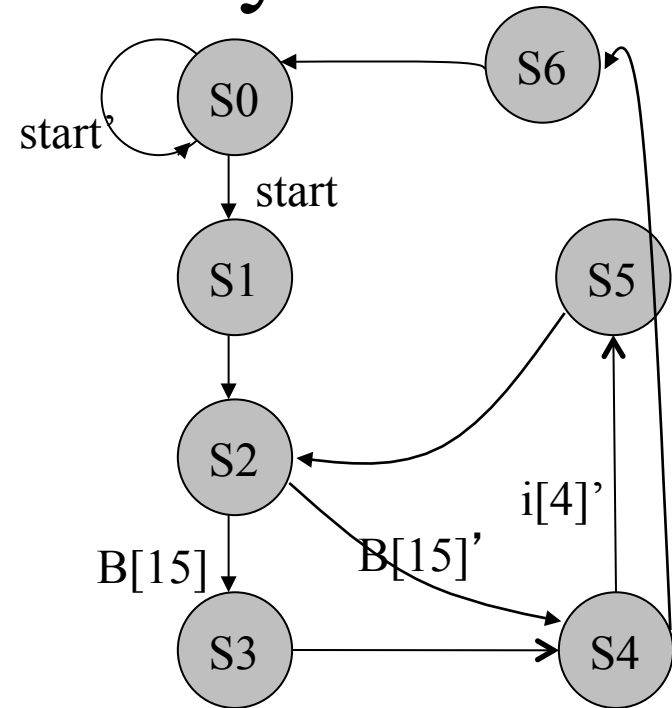
Multiply(X, Y, Z, start, done)

```
{
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
}
```

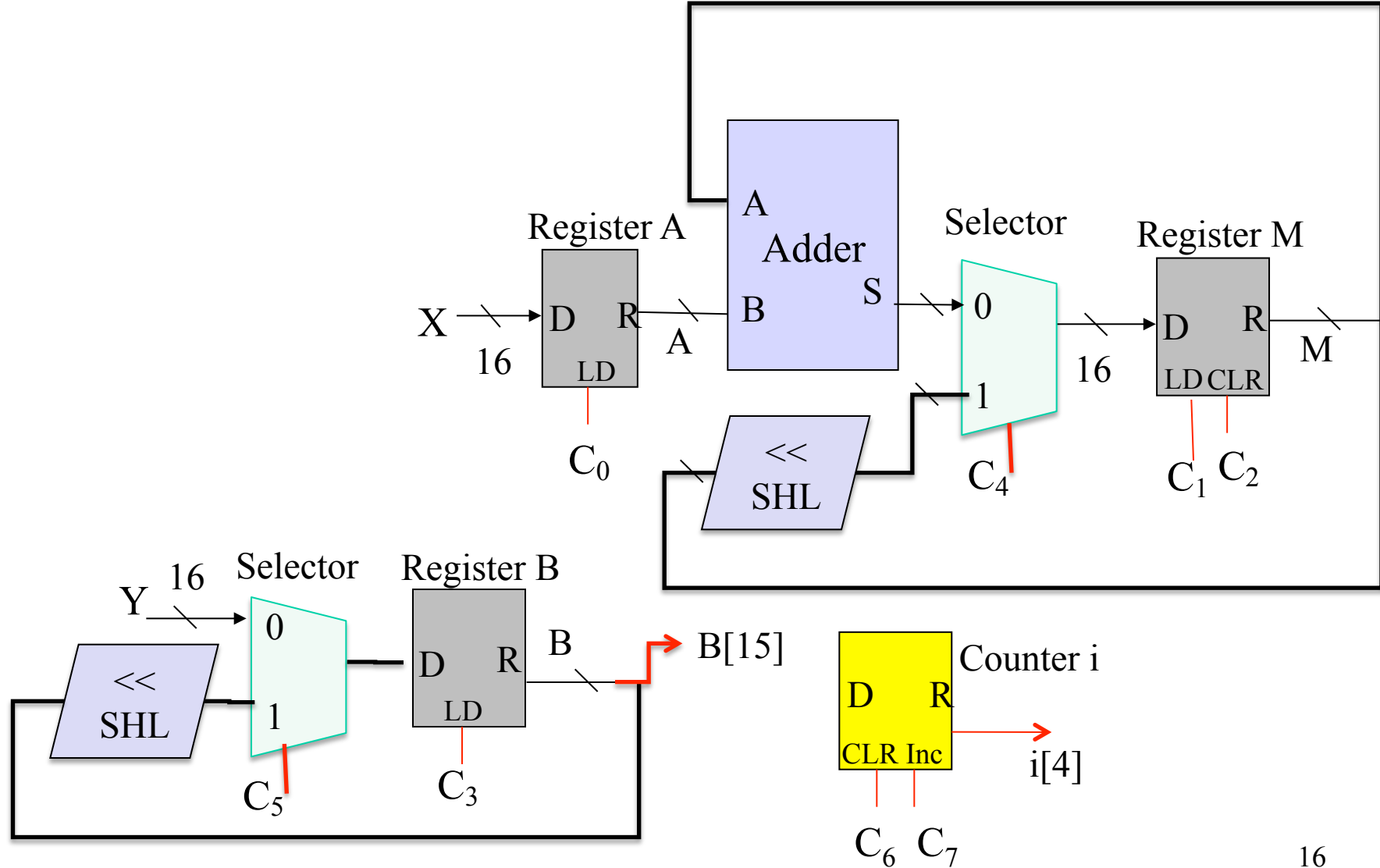
	C0	C1	C2	C3	C4 (mux)	C5 (mux)	C6	C7	done
S0	0	0	0	0	X	X	0	0	1
S1	1	0	1	1	X	0	1	0	0
S2	0	0	0	0	X	X	0	1	0
S3	0	1	0	0	0	X	0	0	0
S4	0	0	0	0	X	X	0	0	0
S5	0	1	0	1	1	1	0	0	0
S6	0	0	0	0	X	X	0	0	1

	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	1	X	0	1	0	0
S2	0	0	0	0	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

Control subsystem



Data Subsystem



Implementation: Example

Given a hardware program, implement data path and control subsystems

```
{ Input X[7:0], Y[7:0] type bit-vector,  
    start type boolean;  
  Local-Object A[7:0], B[7:0] type bit-vector;  
  Output Z[7:0] type bit-vector,  
    done type boolean;  
Wait: If start' goto Wait;  
  S1: A  $\leftarrow$  X || B  $\leftarrow$  Y || done  $\leftarrow$  0;  
  S2: If B  $\geq$  0 goto S4;  
  S3: B  $\leftarrow$  -B;  
  S4: If A  $\geq$  B goto S6;  
  S5: A  $\leftarrow$  A + 1 || B  $\leftarrow$  B-1 || goto S4;  
  S6: Z  $\leftarrow$  4 * A || done  $\leftarrow$  1 || goto Wait;  
}
```

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

Some_function

{ Input X[7:0], Y[7:0] type bit-vector,

start type boolean;

Local-Object A[7:0], B[7:0] type bit-vector;

Output Z[7:0] type bit-vector,

done type boolean;

Wait: If start' goto Wait;

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

S2: If $B \geq 0$ goto S4;

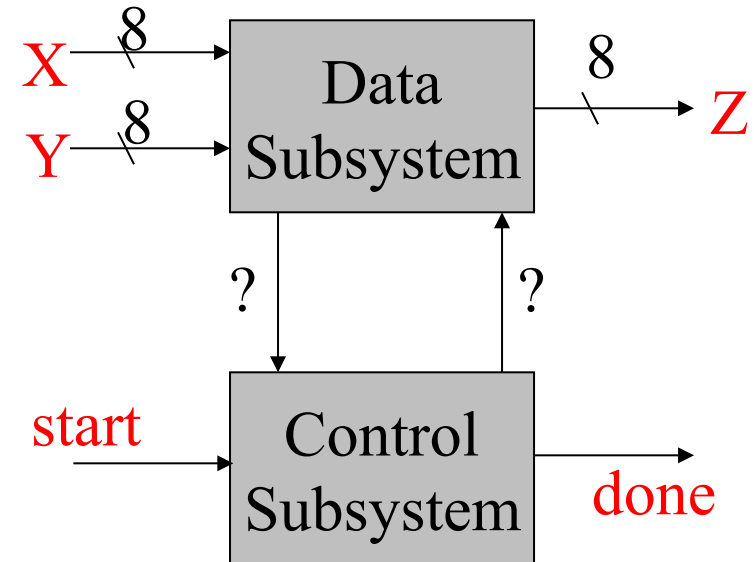
S3: $B \leftarrow -B$;

S4: If $A \geq B$ goto S6;

S5: $A \leftarrow A + 1 \parallel B \leftarrow B - 1 \parallel \text{goto S4}$;

S6: $Z \leftarrow 4 * A \parallel \text{done} \leftarrow 1 \parallel \text{goto Wait}$;

}



Step 2: Identify Data Subsystem Operations

Some_function

```
{ Input X[7:0], Y[7:0] type bit-vector,  
    start type boolean;  
  Local-Object A[7:0], B[7:0] type bit-vector;  
  Output Z[7:0] type bit-vector,  
    done type boolean;
```

Wait: If start' goto Wait;

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

S2: If $B \geq 0$ goto S4;

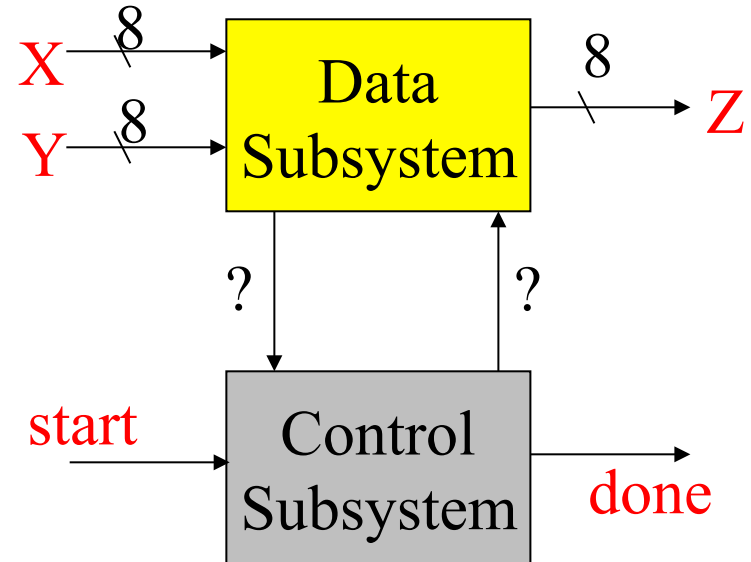
S3: $B \leftarrow -B$;

S4: If $A \geq B$ goto S6;

S5: $A \leftarrow A + 1 \parallel B \leftarrow B - 1 \parallel \text{goto S4}$;

S6: $Z \leftarrow 4 * A \parallel \text{done} \leftarrow 1 \parallel \text{goto Wait}$;

}



Step 2: Identify Data Subsystem Operations

Some_function

```
{ Input X[7:0], Y[7:0] type bit-vector,  
    start type boolean;  
  Local-Object A[7:0], B[7:0] type bit-vector;  
  Output Z[7:0] type bit-vector,  
    done type boolean;
```

Wait: If start' goto Wait;

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

S2: If $B \geq 0$ goto S4;

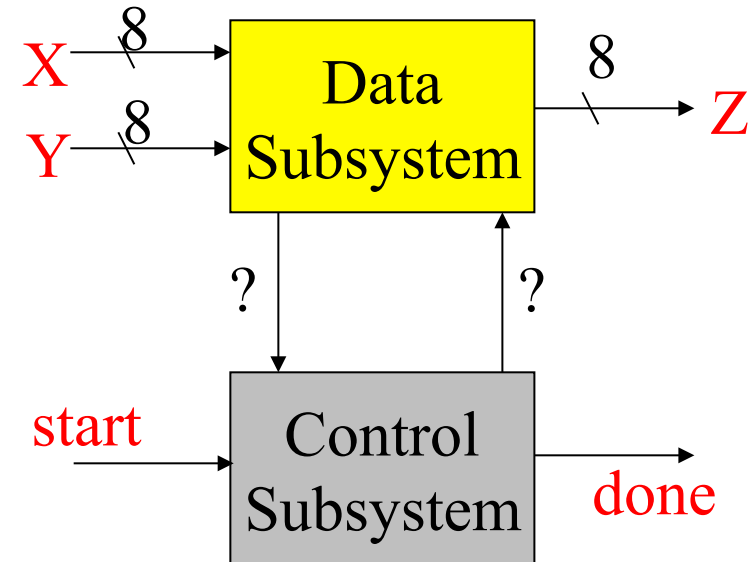
S3: $B \leftarrow -B$;

S4: If $A \geq B$ goto S6;

S5: $A \leftarrow A + 1 \parallel B \leftarrow B - 1 \parallel \text{goto S4}$;

S6: $Z \leftarrow 4 * A \parallel \text{done} \leftarrow 1 \parallel \text{goto Wait}$;

}



Step 2: Map Data Operations to Implementable functions

{Input X[7:0], Y[7:0] type bit-vector,
 start type boolean;
 Local-Object A[7:0], B[7:0] type bit-vector;
 Output Z[7:0] type bit-vector,
 done type boolean;

Wait: If start' goto Wait;

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

S2: If $B \geq 0$ goto S4;

S3: $B \leftarrow -B$;

S4: If $A \geq B$ goto S6;

S5: $A \leftarrow A + 1 \parallel B \leftarrow B - 1 \parallel \text{goto S4}$;

S6: $Z \leftarrow 4 * A \parallel \text{done} \leftarrow 1 \parallel \text{goto Wait}$;

}

	operation
$A \leftarrow X$	$A \leftarrow \text{Load}(X)$
$B \leftarrow Y$	$B \leftarrow \text{Load}(Y)$
$B \leftarrow -B$	$B \leftarrow \text{SUB}(0, B)$
$A \geq B$	$\text{Comp}(A, B)$
$A \leftarrow A + 1$	$A \leftarrow \text{INC}(A)$
$B \leftarrow B - 1$	$B \leftarrow \text{DEC}(B)$
$Z \leftarrow 4A$	$Z \leftarrow \text{SHL}(A, 2)$
If $B \geq 0$	$B[7]$

Step 3: Implement the Data Subsystem from Standard Modules

operation

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

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

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

$B \leftarrow \text{SUB}(0, B)$

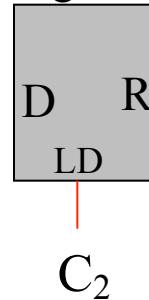
$B \leftarrow \text{DEC}(B)$

$\text{Comp}(A, B)$

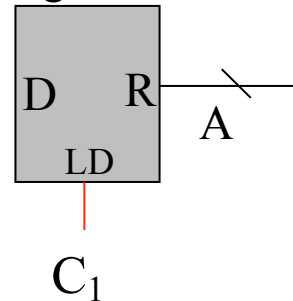
$Z \leftarrow \text{SHL}(A, 2)$

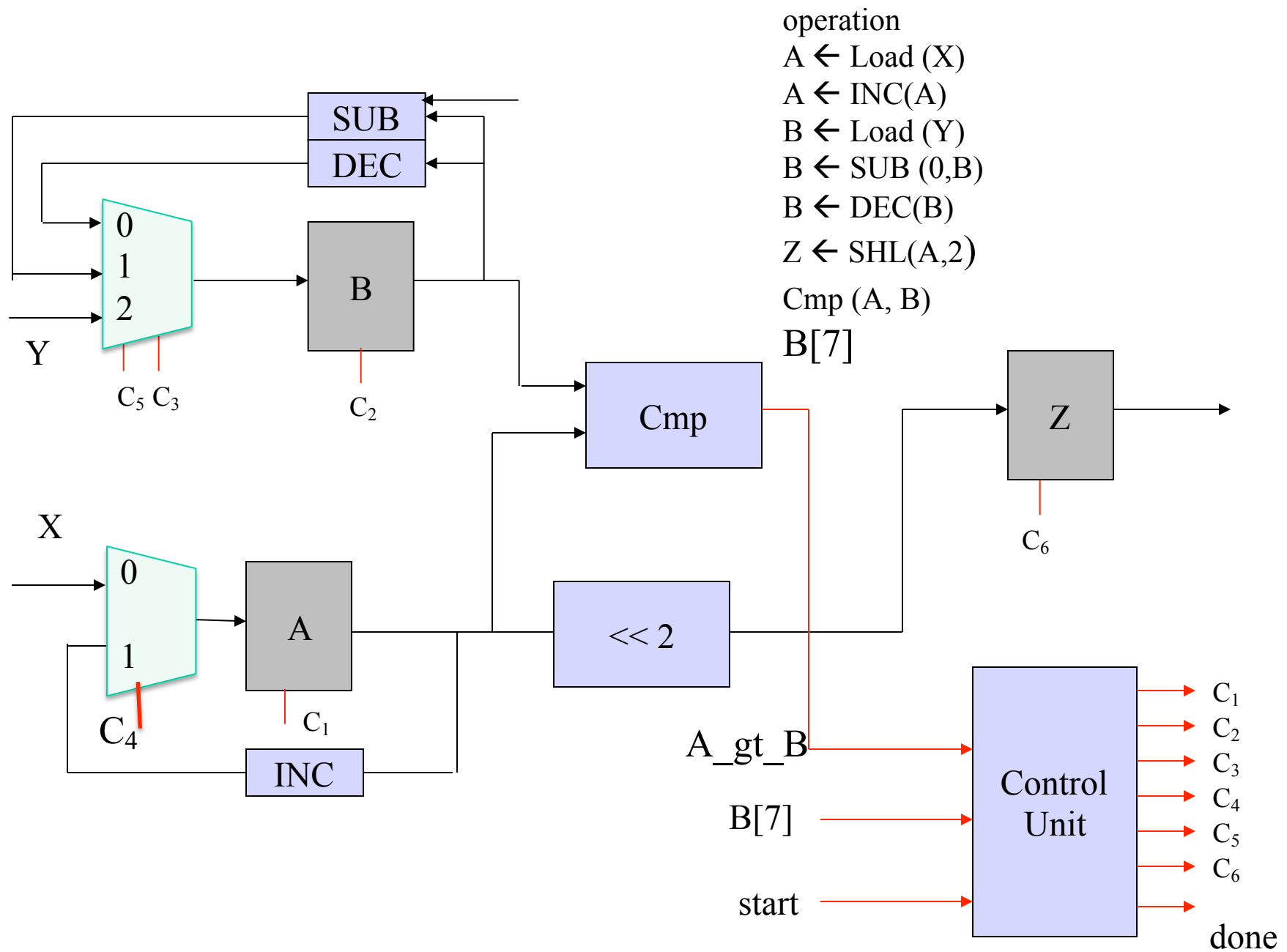
$B[7]$

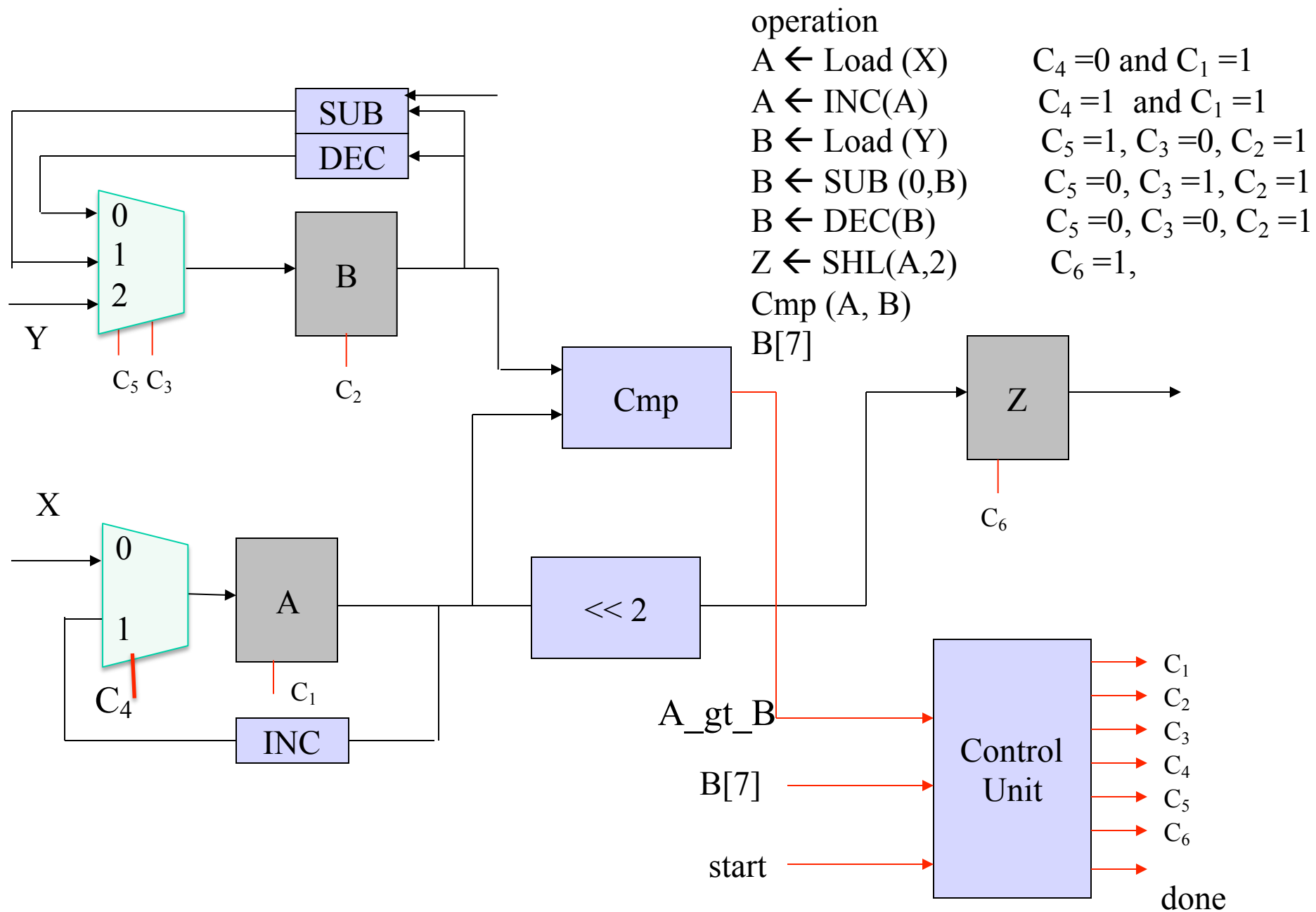
Register B



Register A



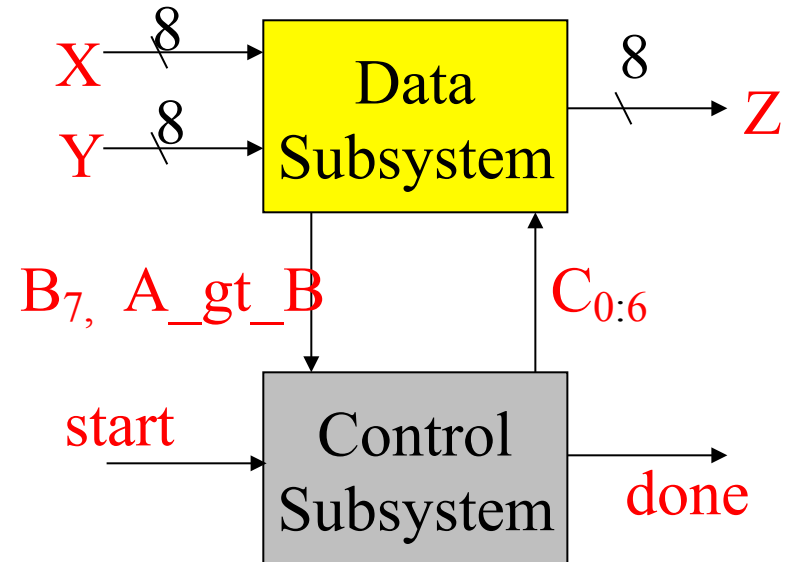




Step 4: Identify Condition bits to Control Subsystem

{Input X[7:0], Y[7:0] type bit-vector,
start type boolean;
Local-Object A[7:0], B[7:0] type bit-vector;
Output Z[7:0] type bit-vector,
done type boolean;

Wait: If start' goto Wait;
S1: $A \leftarrow X \parallel B \leftarrow Y \parallel \text{done} \leftarrow 0$;
S2: **If** B $\geq$ 0 goto S4;
S3: $B \leftarrow -B$;
S4: **If** A $\geq$ B goto S6;
S5: $A \leftarrow A + 1 \parallel B \leftarrow B - 1 \parallel \text{goto S4}$;
S6: $Z \leftarrow 4 * A \parallel \text{done} \leftarrow 1 \parallel \text{goto Wait}$;
}



Designing the control unit

Some_function

```
{
  S0: If start' goto Wait;
  S1: A ← X || B ← Y || done ← 0;
  S2: If B ≥ 0 goto S4;
  S3: B ← -B;
  S4: If A ≥ B goto S6;
  S5: A ← A + 1 || B ← B - 1 || goto S4;
  S6: Z ← 4 * A || done ← 1 || goto S0;
}
```

operation

A ← Load (X)	C ₄ = 0 and C ₁ = 1
A ← INC(A)	C ₄ = 1 and C ₁ = 1
B ← Load (Y)	C ₅ = 1, C ₃ = 0, C ₂ = 1
B ← SUB (0,B)	C ₅ = 0, C ₃ = 1, C ₂ = 1
B ← DEC(B)	C ₅ = 0, C ₃ = 0, C ₂ = 1
Z ← SHL(A,2)	C ₆ = 1,
Cmp (A, B)	A_gt_B
B ≥ 0	B[7]

Some_function

```
{
  S0: If start' goto Wait;
  S1: C4 = 0, C1 = 1 || C5 = 1, C3 = 0, C2 = 1 || done ← 0;
  S2: If      goto S4;
  S3:
  S4: If      goto S6;
  S5: C4 = 1 and C1 = 1 || C5 = 0, C3 = 0, C2 = 1 || goto S4;
  S6: C6 = 1 || done ← 1 || goto S0;
}
```

Designing the control unit

Some_function

```
{
  S0: If start' goto Wait;
  S1: A ← X || B ← Y || done ← 0;
  S2: If B ≥ 0 goto S4;
  S3: B ← -B;
  S4: If A ≥ B goto S6;
  S5: A ← A + 1 || B ← B - 1 || goto S4;
  S6: Z ← 4 * A || done ← 1 || goto S0;
}
```

operation

A ← Load (X)	C ₄ = 0 and C ₁ = 1
A ← INC(A)	C ₄ = 1 and C ₁ = 1
B ← Load (Y)	C ₅ = 1, C ₃ = 0, C ₂ = 1
B ← SUB (0,B)	C ₅ = 0, C ₃ = 1, C ₂ = 1
B ← DEC(B)	C ₅ = 0, C ₃ = 0, C ₂ = 1
Z ← SHL(A,2)	C ₆ = 1,
Cmp (A, B)	A_gt_B
B ≥ 0	B[7]

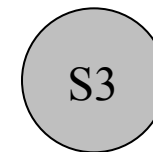
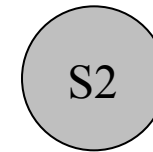
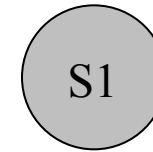
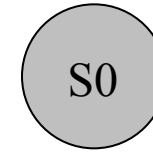
Some_function

```
{
  S0: If start' goto Wait;
  S1: C4 = 0, C1 = 1 || C5 = 1, C3 = 0, C2 = 1 || done ← 0;
  S2: If B[7]' goto S4;
  S3: C5 = 0, C3 = 1, C2 = 1
  S4: If A_gt_B goto S6;
  S5: C4 = 1 and C1 = 1 || C5 = 0, C3 = 0, C2 = 1 || goto S4;
  S6: C6 = 1 || done ← 1 || goto S0;
}
```

State Machine

Some_function

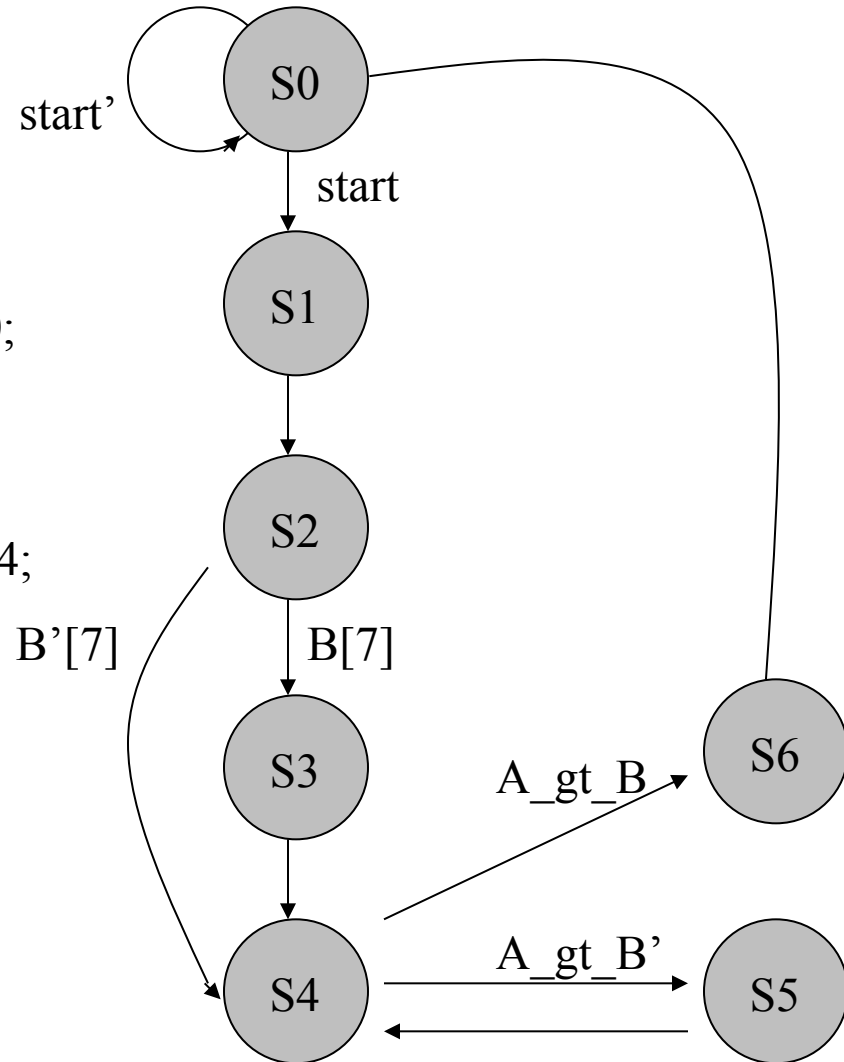
```
{  
  S0: If start' goto Wait;  
  S1:  $C_4 = 0, C_1 = 1 \parallel C_5 = 1, C_3 = 0, C_2 = 1 \parallel \text{done} \leftarrow 0$ ;  
  S2: If B[7]' goto S4;  
  S3:  $C_5 = 0, C_3 = 1, C_2 = 1$   
  S4: If A_gt_B goto S6;  
  S5:  $C_4 = 1 \text{ and } C_1 = 1 \parallel C_5 = 0, C_3 = 0, C_2 = 1 \parallel \text{goto S4}$ ;  
  S6:  $C_6 = 1 \parallel \text{done} \leftarrow 1 \parallel \text{goto S0}$ ;  
}
```



State Machine

Some_function

```
{  
  S0: If start' goto Wait;  
  S1:  $C_4 = 0, C_1 = 1 \parallel C_5 = 1, C_3 = 0, C_2 = 1 \parallel \text{done} \leftarrow 0$ ;  
  S2: If  $B[7]'$  goto S4;  
  S3:  $C_5 = 0, C_3 = 1, C_2 = 1$   
  S4: If  $A\_gt\_B$  goto S6;  
  S5:  $C_4 = 1$  and  $C_1 = 1 \parallel C_5 = 0, C_3 = 0, C_2 = 1 \parallel \text{goto S4}$ ;  
  S6:  $C_6 = 1 \parallel \text{done} \leftarrow 1 \parallel \text{goto S0}$ ;  
}
```



One-Hot State Machine

