

CSE 140 Lecture 17

System Design II

CK Cheng
CSE Dept.
UC San Diego

1

High Level Synthesis

- Hierarchical Description
 - 4-Phase Handshaking
- Graph Representation
 - Control/Data Flow Graphs
- Data Subsystem
 - Allocation *Components (cover all operations)*
 - Binding *spatial vs time sharing.*
- Scheduling
 - Finite State Machine

Reference:

Bluespec System

FPGAs for Software Programmers, D. Koch, F. Hannig, D. Ziener, Springer 2016

2

High Level Synthesis

- 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

Allocation.

Binding.

3

Example: Multiplication

Arithmetic

$$Z = X \times Y$$

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

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
- $Z = M$

$$\begin{array}{r}
 110 \times \\
 101 \times Y \\
 \hline
 y_2=1, M=M+X \\
 =110 \\
 1100 \\
 \hline
 y_2=0, 11000 \\
 \hline
 y_2=1, 11000 \\
 1100 \\
 \hline
 11100 \\
 \hline
 \rightarrow 30
 \end{array}$$

4

$$\begin{array}{r}
 \begin{array}{l}
 i=0 \quad M=0. \\
 y_2=1 \\
 M = M + X = 110
 \end{array}
 \quad
 \begin{array}{r}
 \begin{array}{r}
 110 \times 6 \\
 \times 101 \quad Y \quad 5 \\
 \hline
 110 \\
 000 \\
 110 \\
 \hline
 11110 \\
 168421 \Rightarrow 30
 \end{array}
 \end{array}
 \end{array}$$

$$\begin{array}{r}
 \begin{array}{l}
 i=1 \\
 y_2=0 \\
 M = M_{(\text{shift left})} = 11000
 \end{array}
 \end{array}$$

$$\begin{array}{r}
 \begin{array}{l}
 i=2. \\
 y_2=1 \\
 M = M + X = 11000 + 110 \\
 = \cancel{11000} + 110 \\
 168421
 \end{array}
 \quad
 \begin{array}{r}
 1100 \\
 110 \\
 5
 \end{array}
 \end{array}$$

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 B₁₅ = 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

```

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 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;
}

```

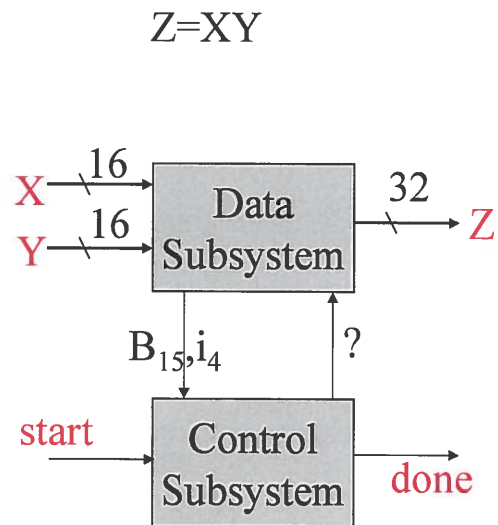
7

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 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;
}

```



8

Step 2: Identify Condition Bits to Control Subsystem

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;

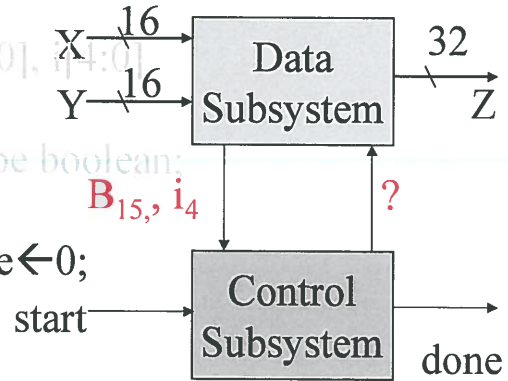
S3: M $\leftarrow$ M+A;

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;

}



9

Step 3: 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 || B $\leftarrow$ Y || i $\leftarrow$ 0 || M $\leftarrow$ 0 || done $\leftarrow$ 0;

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

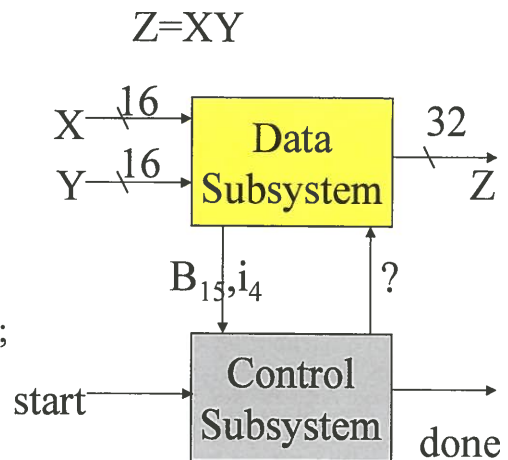
S3: M $\leftarrow$ M+A;

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;

}



10

Step 4: Map Data Operations to Implementable functions

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: $A \leftarrow X \parallel B \leftarrow Y \parallel i \leftarrow 0 \parallel M \leftarrow 0 \parallel$ 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: $M \leftarrow \text{Shift}(M, L, 1) \parallel B \leftarrow \text{Shift}(B, L, 1) \parallel$ goto S2;

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

}

Operation

$A \leftarrow X$

$B \leftarrow Y$

$M \leftarrow 0$

$i \leftarrow 0$

$i \leftarrow i+1$

$M \leftarrow M+A$

$M \leftarrow \text{Shift}(M, L, 1)$

$B \leftarrow \text{Shift}(B, L, 1)$

$Z \leftarrow M$

Operators

operation

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

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

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

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

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

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

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

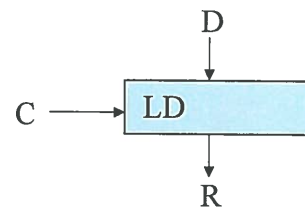
$B \leftarrow \text{SHL}(B)$

Wires

11

Step 5: Implement the Data Subsystem from Standard Modules

Registers: If C then $R \leftarrow D$



operation

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

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

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

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

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

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

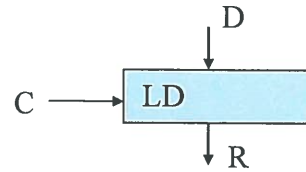
$M \leftarrow \text{SHL}(M)$

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

12

Storage Component: Registers with control signals

Registers: If C then $R \leftarrow D$



operation

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

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

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

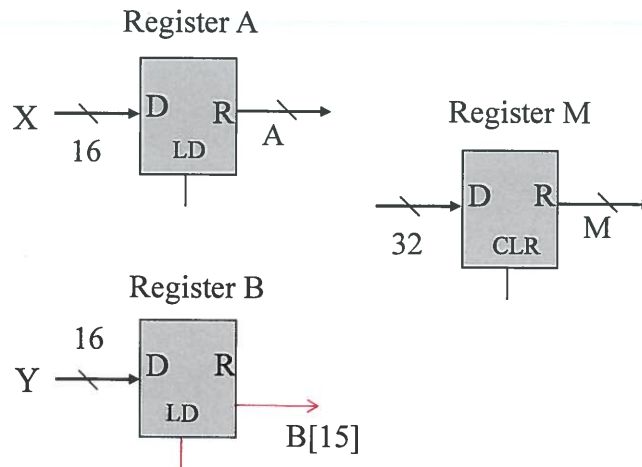
$i \leftarrow \text{Clear}(i)$

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

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

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

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



13

Data Subsystem

operation

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

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

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

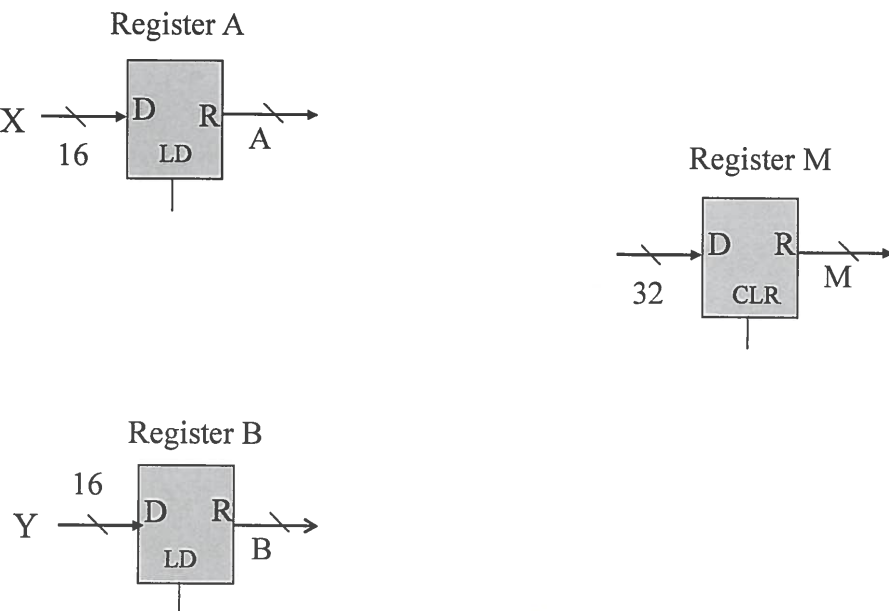
$i \leftarrow \text{Clear}(i)$

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

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

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

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



14

Function Modules: Adder, Shifter

operation

A $\leftarrow$ Load (X)

B $\leftarrow$ Load (Y)

M $\leftarrow$ Clear(M)

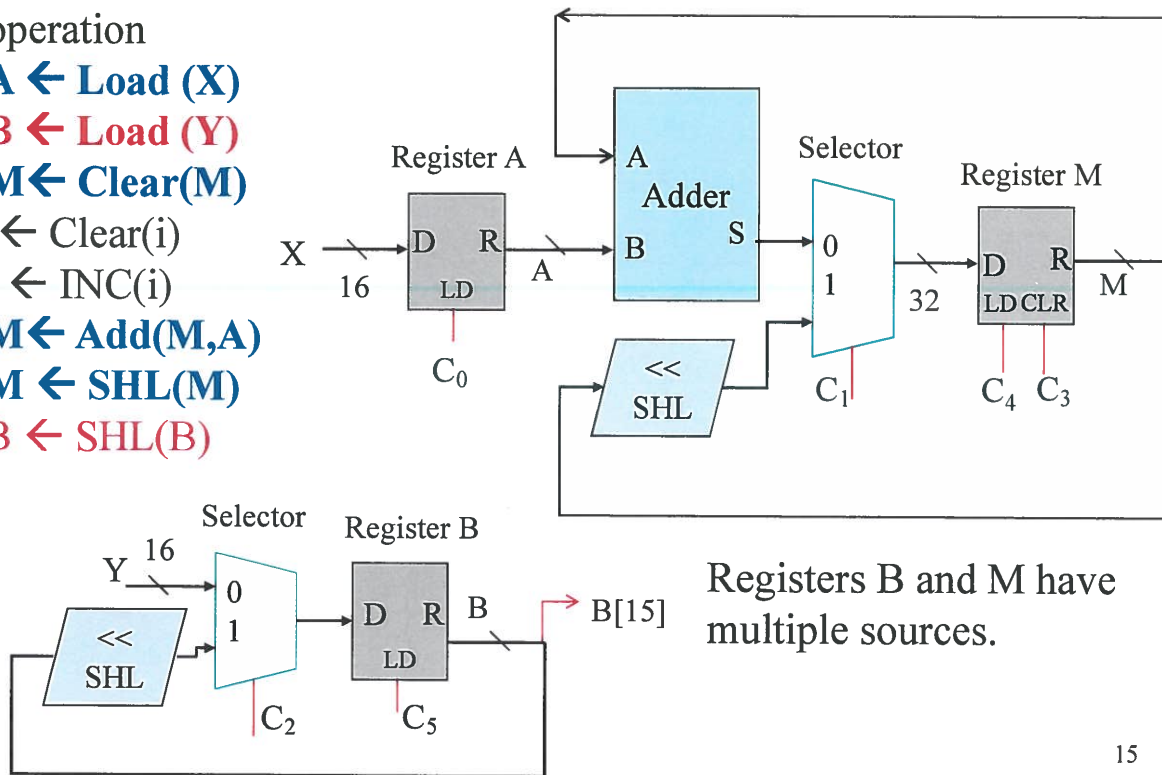
i $\leftarrow$ Clear(i)

i $\leftarrow$ INC(i)

M $\leftarrow$ Add(M,A)

M $\leftarrow$ SHL(M)

B $\leftarrow$ SHL(B)



15

Function Modules: Adder, Shifter, Counter

operation

A $\leftarrow$ Load (X)

B $\leftarrow$ Load (Y)

M $\leftarrow$ Clear(M)

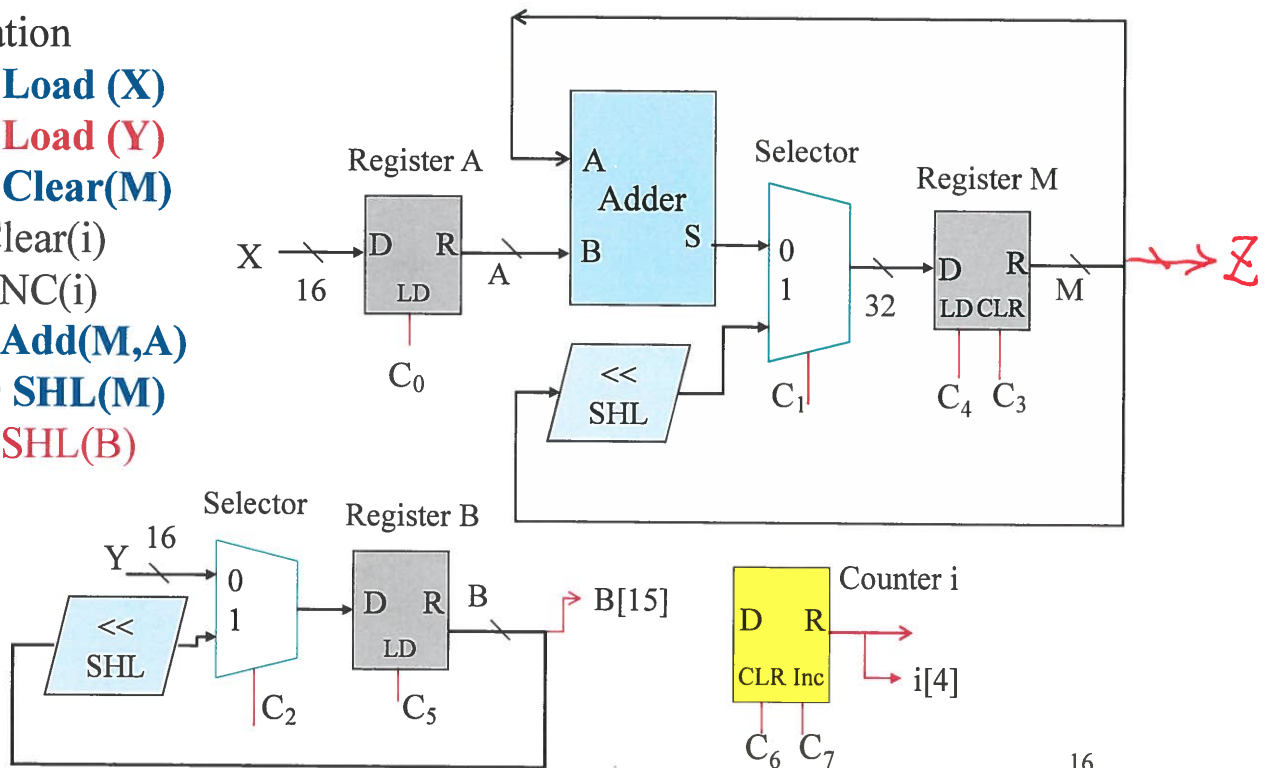
i $\leftarrow$ Clear(i)

i $\leftarrow$ INC(i)

M $\leftarrow$ Add(M,A)

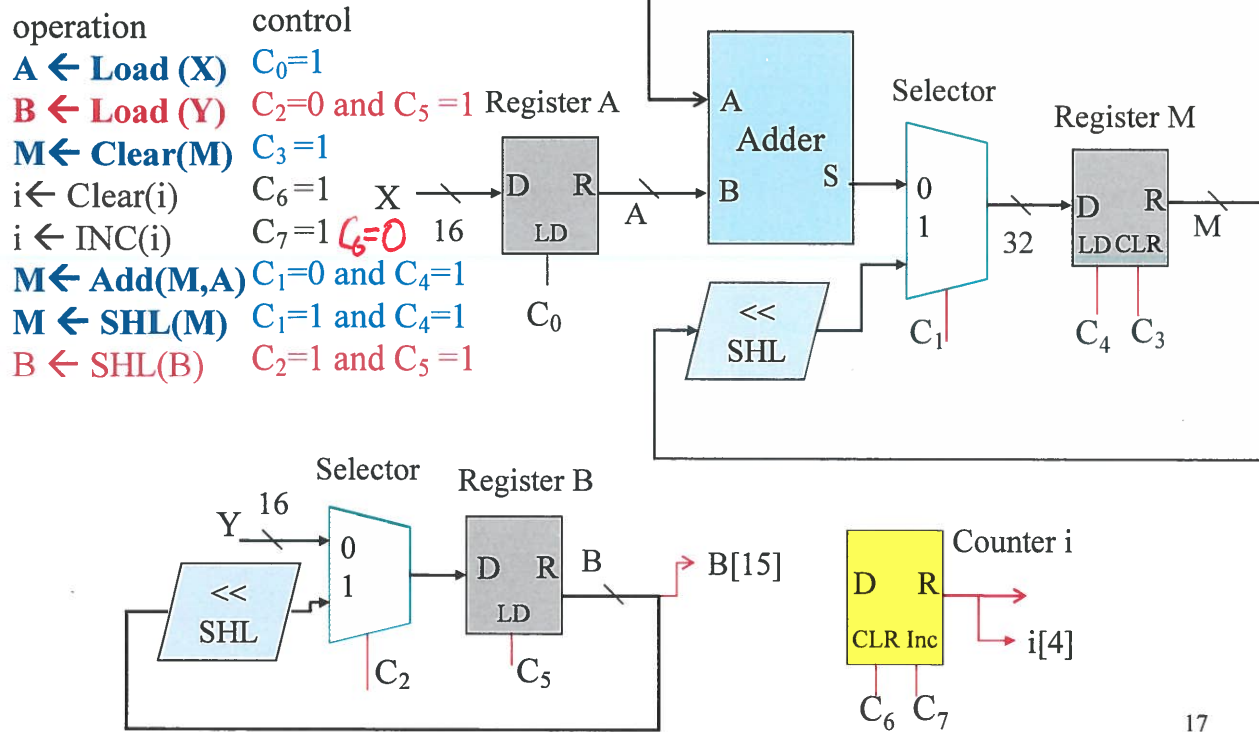
M $\leftarrow$ SHL(M)

B $\leftarrow$ SHL(B)



16

Step 6: Map Control Signals to Operations



Step 7: Identify Control Path Components

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

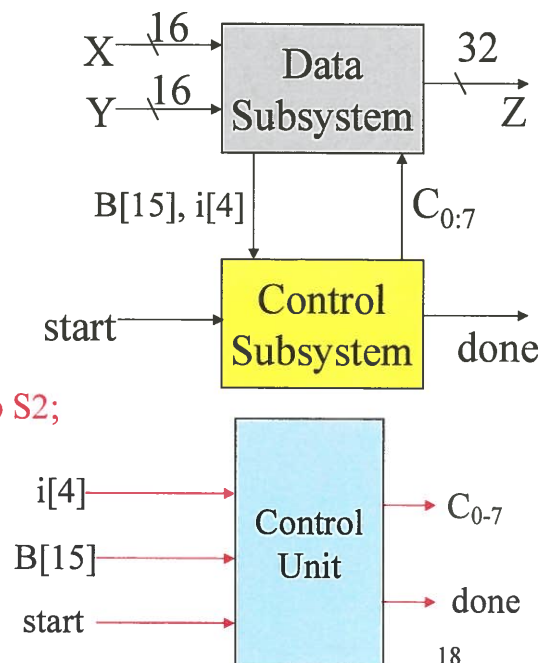
{ Input: X[15:0], Y[15:0] type bit-vector;
start type boolean;

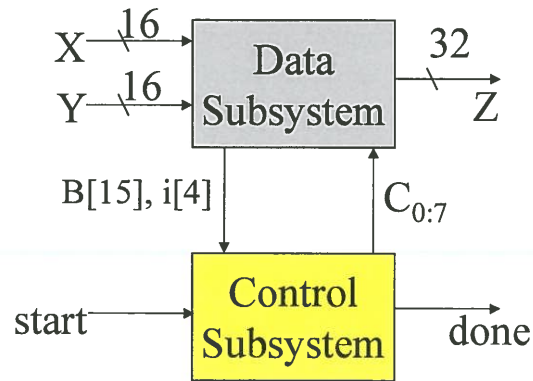
Local-Object : A[15:0], B[15:0], M[31:0],
i[4:0] type bit-vector;

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 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;
}

Z=XY





19

operation control

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

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;}

	C0 Load A	C1 Feed M	C2 Feed B	C3 Clr M	C4 Load M	C5 Load B	C6 Clr i	C7 Inc i	done
S0	0	—	—	0	0	0	0	0	1
S1	1	—	○	1	0	1	1	0	0
S2	0	—	—	0	0	0	0	1	0
S3	0	○	—	0	1	0	0	0	0
S4	0	—	—	0	0	0	0	0	0
S5	0	1	1	0	1	1	0	0	0
S6	0	—	—	0	0	0	0	0	1

20

operation	control
A $\leftarrow$ Load (X)	C ₀ =1
B $\leftarrow$ Load (Y)	C ₂ =0 and C ₅ =1
M $\leftarrow$ Clear(M)	C ₃ =1
i $\leftarrow$ Clear(i)	C ₆ =1
i $\leftarrow$ INC(i)	C ₇ =1
M $\leftarrow$ Add(M,A)	C ₁ =0 and C ₄ =1
M $\leftarrow$ SHL(M)	C ₁ =1 and C ₄ =1
B $\leftarrow$ SHL(B)	C ₂ =1 and C ₅ =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;
}

```

	C0 Load A	C1 Feed M	C2 Feed B	C3 Clr M	C4 Load M	C5 Load B	C6 Clr i	C7 Inc i	done
S0	0	X	X	0	0	0	0	0	1
S1	1	X	0	1	0	1	1	0	0
S2	0	X	X	0	0	0	0	1	0
S3	0	0	X	0	1	0	0	0	0
S4	0	X	X	0	0	0	0	0	0
S5	0	1	1	0	1	1	0	0	0
S6	0	X	X	0	0	0	0	0	1

21

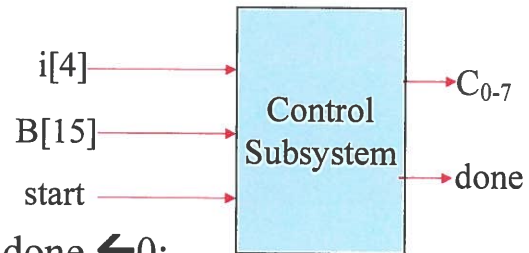
Design of the Control Subsystem

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
}

```

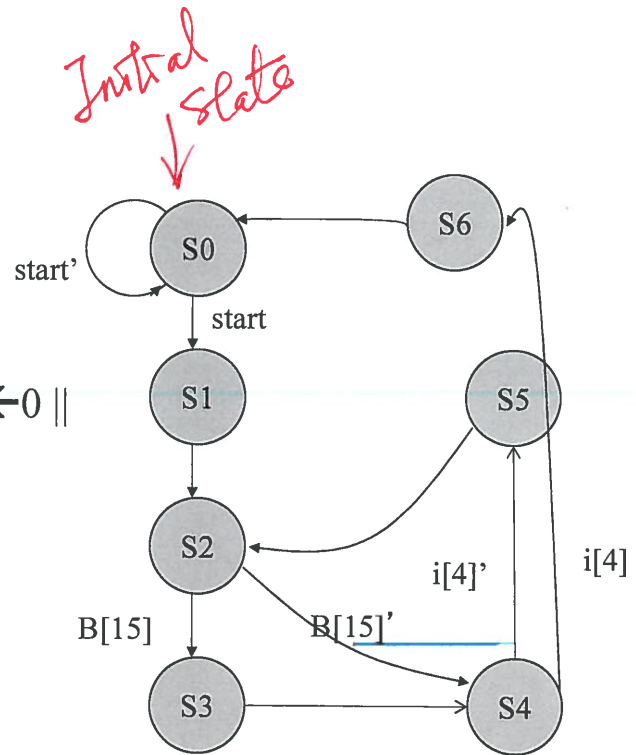


22

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
}
```



23

State Assignment

Binary	$b_2b_1b_0$
S0	000
S1	001
S2	010
S3	011
S4	100
S5	101
S6	110
S7	111

Gray	$b_2b_1b_0$
S0	000
S1	001
S2	011
S3	010
S4	110
S5	111
S6	101
S7	100

One Hot	$b_7b_6b_5b_4b_3b_2b_1b_0$
S0	0 0 0 0 0 0 0 1
S1	0 0 0 0 0 0 1 0
S2	0 0 0 0 0 1 0 0
S3	0 0 0 0 1 0 0 0
S4	0 0 0 1 0 0 0 0
S5	0 0 1 0 0 0 0 0
S6	0 1 0 0 0 0 0 0
S7	1 0 0 0 0 0 0 0

One Hot Encoding: n bits for n states. Bit $i=1$ for state i .

24

Control Subsystem: One-Hot State Machine Design

Input: State Diagram

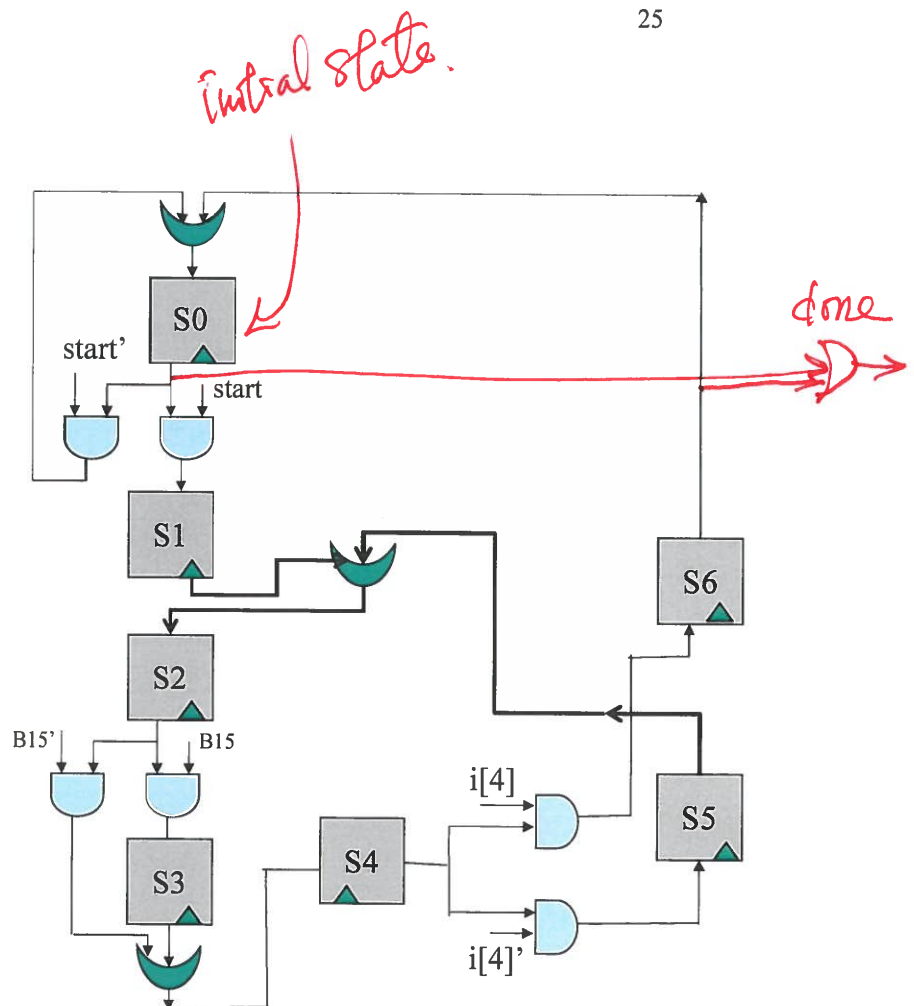
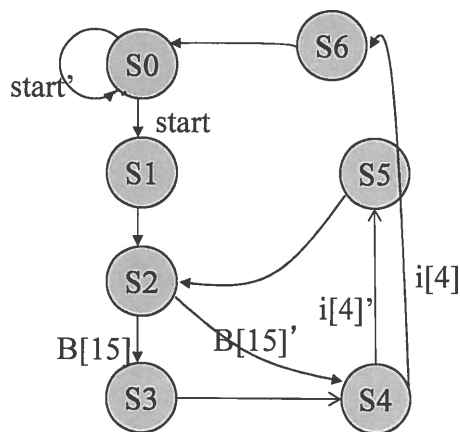
1. Use a flip flop to replace each state.

Set the flip flop which corresponds to the initial state and reset the rest flip flops.

2. Use an OR gate to collect all inward edges.

3. Use a Demux to distribute the outward edges.

One-Hot State Machine



25

26

Control Subsystem: One-Hot State Machine Design

Input: State Diagram

1. Use a flip flop to replace each state.

Set the flip flop which corresponds to the initial state and reset the rest flip flops.

2. Use an OR gate to collect all inward edges.

3. Use a Demux to distribute the outward edges.

27

Implementation: Example

Given a hardware program, implement data path and control subsystems

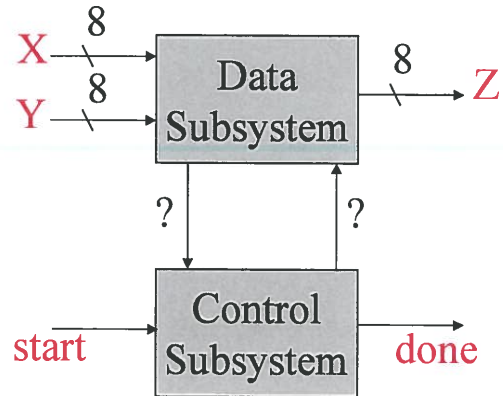
```
{ Input X[7:0], Y[7:0] type bit-vector, start type boolean;  
  Local-Object A[7:0], B[7:0] type bit-vector;  
  Output Z[7:0] type bit-vector, done type boolean;  
  Wait: If start' goto Wait || done ← 1;  
  S1: A ← X || B ← Y || done ← 0;  
  S2: If B >= 0 goto S4;  
  S3: B ← -B;  
  S4: If A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

28

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 || done ← 1;
    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 Wait;
}
```

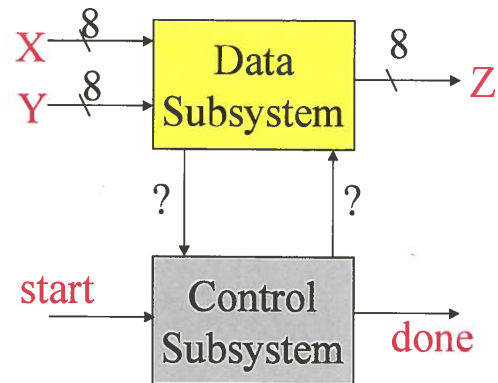


29

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 || done ← 1;
    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 Wait;
}
```



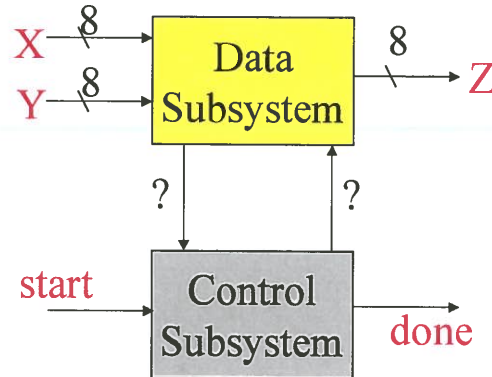
$$Z = \begin{cases} 4 \text{ Ceiling}[(X + |Y|)/2] & \text{if } X < |Y| \\ 4X & \text{otherwise} \end{cases}$$

30

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 || done ← 1;
  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 Wait;
}
```



31

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 || done ← 1;
  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 Wait;
}
```

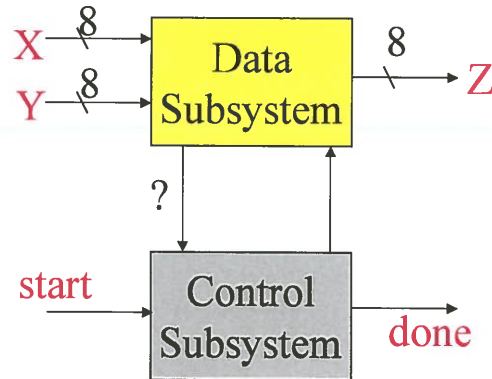
	operations
A ← X	A ← Load (X)
B ← Y	B ← Load (Y)
B ← -B	B ← CS (B)
A ≥ B	Comp (A, B)
A ← A + 1	A ← INC (A)
B ← B - 1	B ← DEC (B)
Z ← 4A	Z ← SHL(A) } twice

32

Step 3: Tag each Data Operations with a Control Signal

Operation

	operation
$A \leftarrow X$	$A \leftarrow \text{Load}(X)$
$B \leftarrow Y$	$B \leftarrow \text{Load}(Y)$
$B \leftarrow -B$	$B \leftarrow \text{CS}(B)$
$A \geq B$	$\text{Comp}(A, B)$
$A \leftarrow A + 1$	$A \leftarrow \text{INC}(A)$
$B \leftarrow B - 1$	$B \leftarrow \text{DEC}(B)$
$Z \leftarrow 4A$	$Z \leftarrow \text{SHL}(A)$



33

Step 4: Identify Condition Bits to Control Subsystem

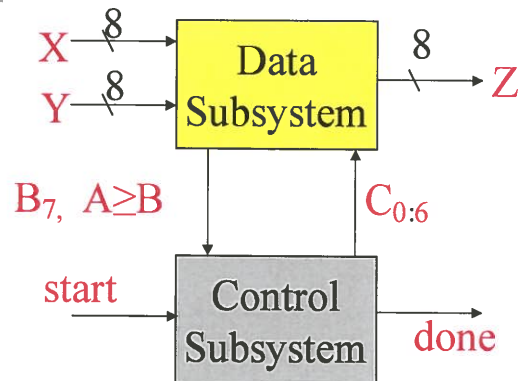
{Input $X[7:0]$, $Y[7:0]$ type bit-vector,
start type boolean;

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 || done ← 1;
S1:  $A \leftarrow X$  ||  $B \leftarrow Y$  || done ← 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 ← 1 || goto Wait;
}
  
```



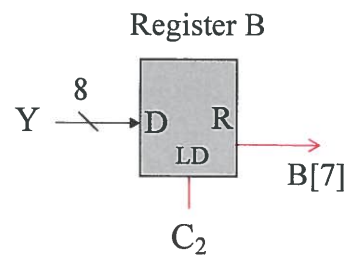
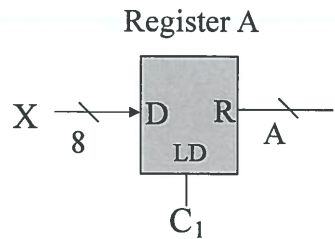
34

Step 5: Implement the Data Subsystem from Standard Modules

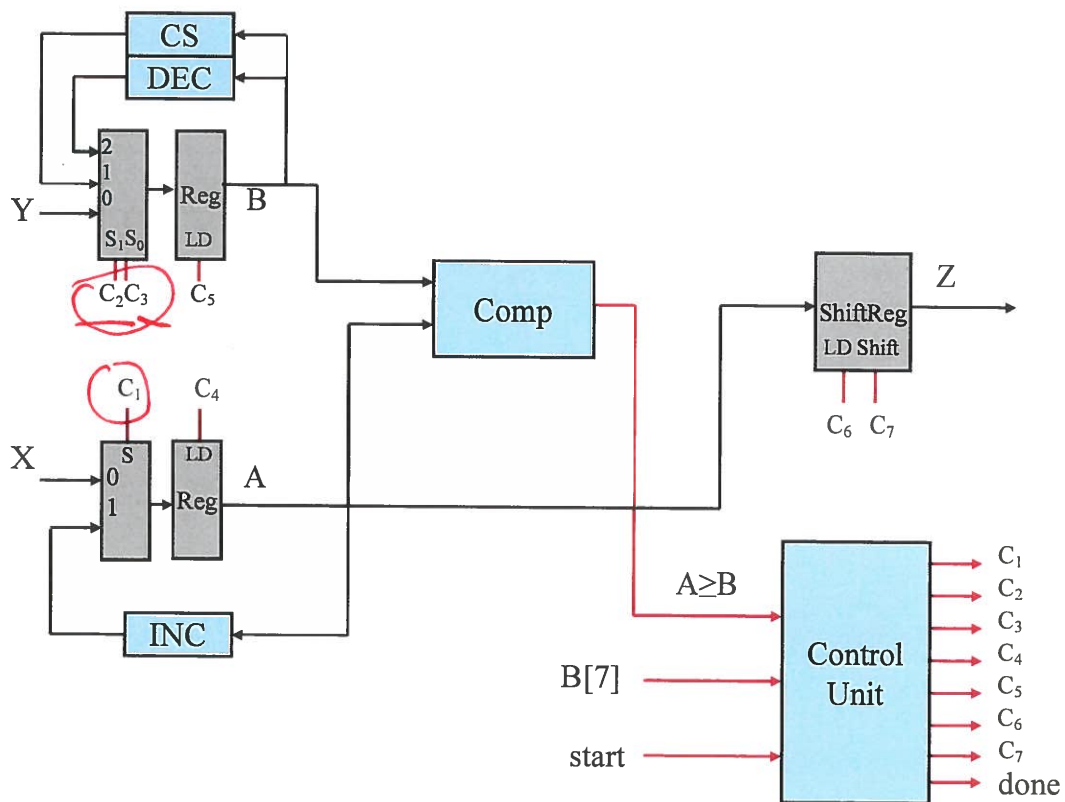
operation

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

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

$$B \leftarrow CS(B)$$
$$\text{Comp (A, B)}$$
$$A \leftarrow \text{INC}(A)$$
$$B \leftarrow \text{DEC}(B)$$
$$Z \leftarrow \text{SHL}(A)$$


35



36

Step 6: Map Control Signals to Operations

Step 7: Identify Control Path Components

S0: If start', goto S0, else goto S1 || done $\leftarrow$ 1

S1: $A \leftarrow X$ || $B \leftarrow Y$ || done $\leftarrow$ 0 || goto S2

S2: If $B' < 7$ goto S4, else goto S3

S3: $B \leftarrow CS(B)$ || goto S4

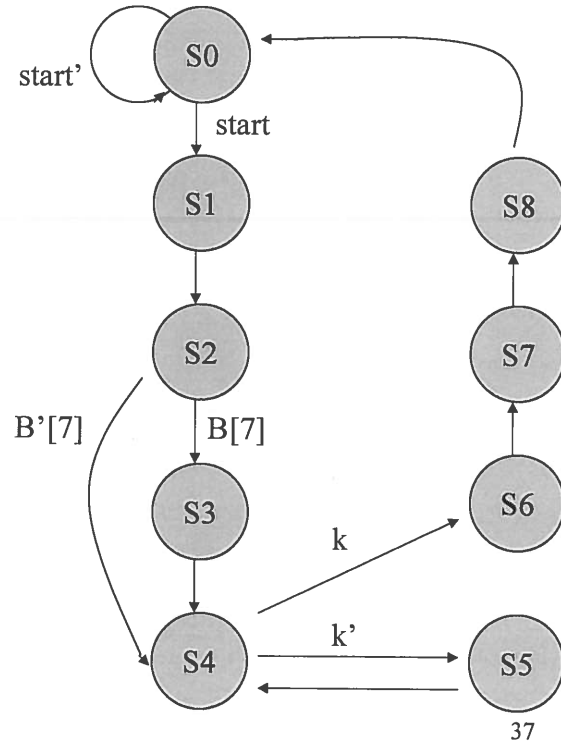
S4: If k goto S6, else goto S5

S5: $A \leftarrow INC(A)$ || $B \leftarrow DEC(B)$ || goto S4

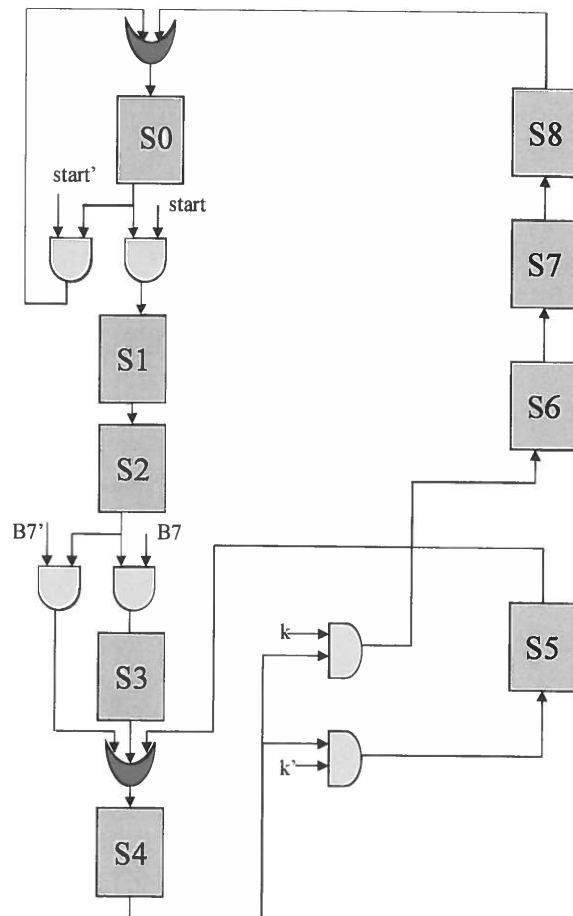
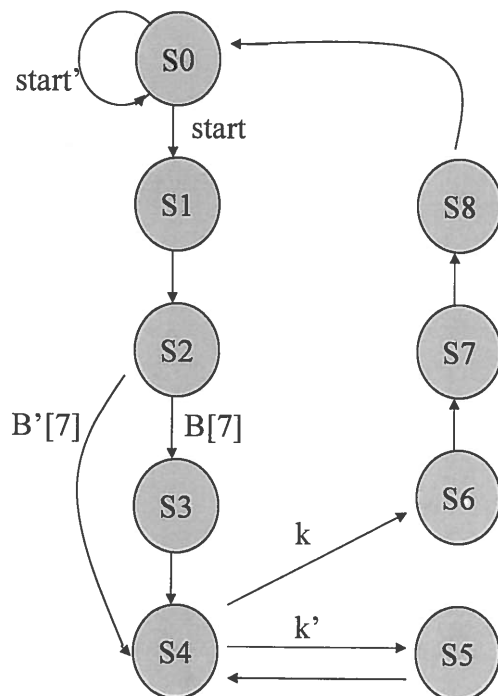
S6: $Z \leftarrow A$ || goto S7

S7: $Z \leftarrow SHL(z)$ || goto S8

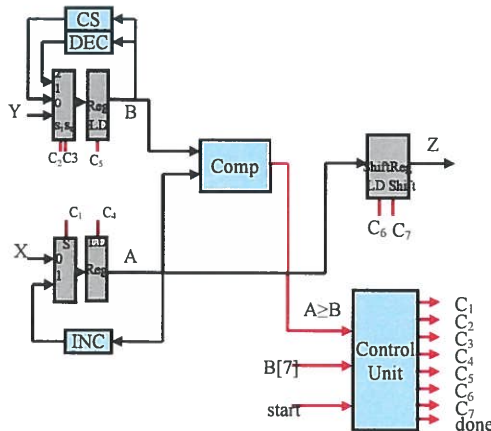
S8: $Z \leftarrow SHL(z)$ || done $\leftarrow$ 1 || goto S0



One-Hot State Machine



S0: If start', goto S0, else goto S1 || done<=1
 S1: $A \leftarrow X$ || $B \leftarrow Y$ || done $\leftarrow 0$ || goto S2
 S2: If $B' < 7 >$ goto S4, else goto S3
 S3: $B \leftarrow CS(B)$ || goto S4
 S4: If k goto S6, else goto S5
 S5: $A \leftarrow INC(A)$ || $B \leftarrow DEC(B)$ || goto S4
 S6: $Z \leftarrow A$ || goto S7
 S7: $Z \leftarrow SHL(z)$ || goto S8
 S8: $Z \leftarrow SHL(z)$ || done<=1 || goto S0



	C1	C2	C3	C4 A	C5 B	C6 Z	C7	done
S0				0	0	0	0	1
S1				1	1	0	0	0
S2				0	0	0	0	0
S3				0	1	0	0	0
S4				0	0	0	0	0
S5				1	1	0	0	0
S6				0	0	1	0	0
S7				0	0	0	1	0
S8				0	0	0	1	1

39

Summary

→ Hierarchical Decomposition (4-phase handshaking)

- Hardware Allocation
 - Balance between cost and performance
- Resource Sharing and Binding
 - Map operations to hardware
- Interconnect Synthesis
 - Convey signal transports
- Operation Scheduling
 - Sequence the process

Remarks:

1. Implement the control subsystem with one-hot state machine design.
2. Try to reduce the latency of the whole system.