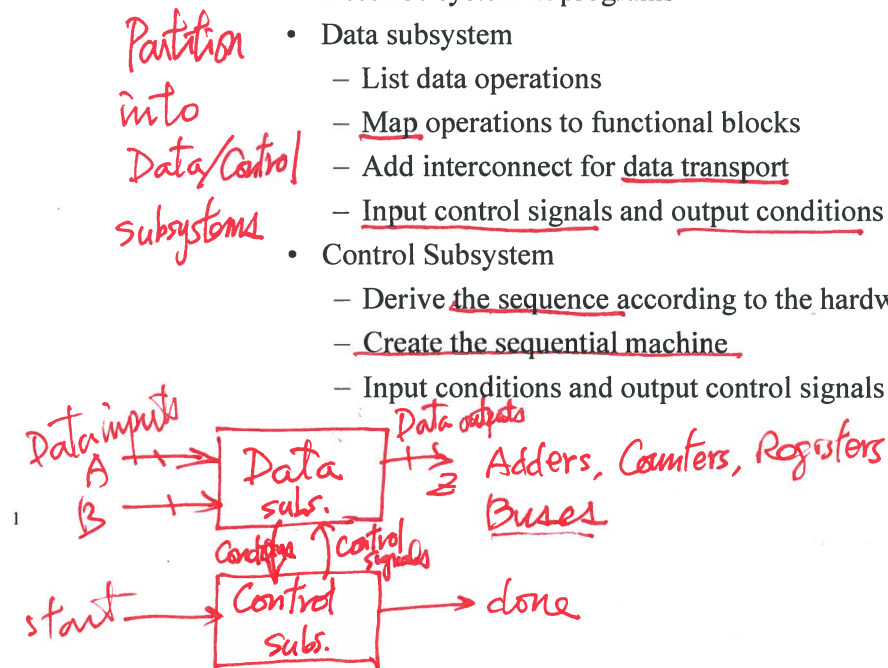


Design Process

CSE 140 Lecture 17 System Design II

CK Cheng
CSE Dept.
UC San Diego

- 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



2
IOWA

Example: Multiplication

Arithmetic

$Z = X \times Y$ *two operands*

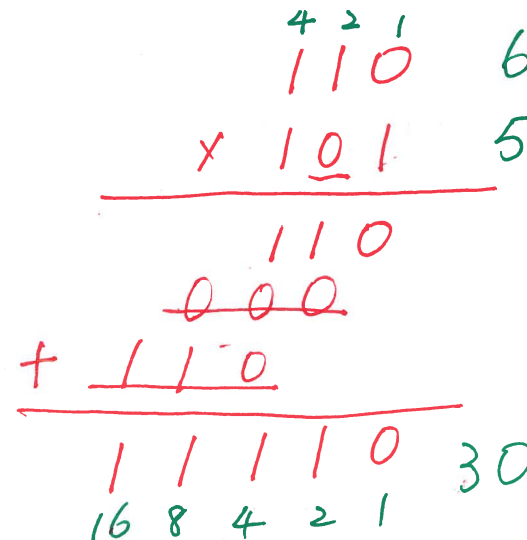
- $M = 0$
- For $i = n-1$ to 0
 - If $Y_i = 1$, $M = M + X \times 2^i$
- $Z = M$ *partial product*

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 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;
}

```

State

Data
Structure

parallel
operations

Instruction
Default Sequence

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;
}

```

6

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

```

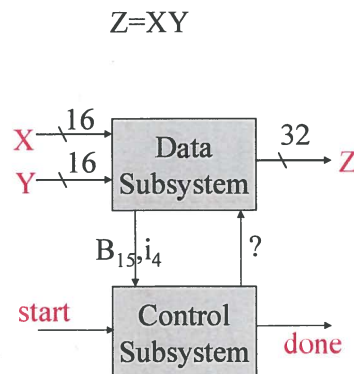
Multiply(X, Y, Z, start, done)
{ Input: X[15:0], Y[15:0] type bit-
vector,    start type boolean;
  Local-Object : A[15:0], B[15:0],
M[31:0], i[4:0] type bit-vector;
  Output Z[31:0] type bit-vector,
done type boolean;

```

```

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;
}

```



7

Step 2: Identify Condition Bits to Control Subsystem

```

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

```

```

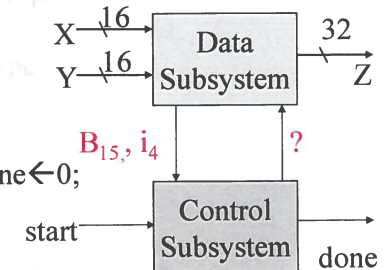
{ Input: X[15:0], Y[15:0] type bit-vector,
start type boolean;
  Local-Object : A[15:0], B[15:0], M[31:0],
i[4:0] type bit-vector;
  Output Z[31:0] type bit-vector, done type
boolean;

```

```

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 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 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[1:0] 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: $M \leftarrow M+A$;

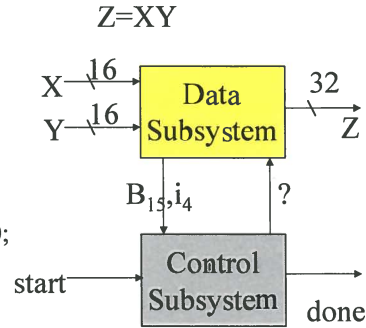
S4: if $i \geq 16$, goto S6

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

~~S2;~~

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

}



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

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

}

Instruction

operation

$A \leftarrow X$ A $\leftarrow$ Load (X)

$B \leftarrow Y$ B $\leftarrow$ Load (Y)

$M \leftarrow 0$ M $\leftarrow$ Clear(M)

$i \leftarrow 0$ i $\leftarrow$ Clear(i)

$i \leftarrow i+1$ i $\leftarrow$ INC(i)

$M \leftarrow M+A$ M $\leftarrow$ Add(M,A)

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

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

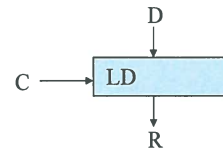
$Z \leftarrow M$ Wires

9

10

Step 5: Implement the Data Subsystem from Standard Modules

Registers: If C then $R \leftarrow D$



operation

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

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

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

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

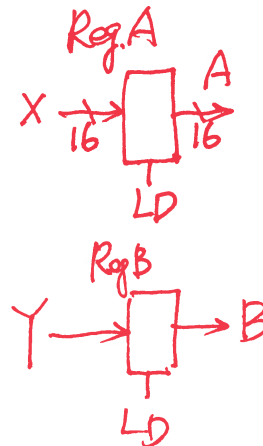
$i \leftarrow \text{INC}(i)$

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

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

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

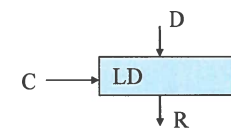
$(Z \leftarrow M)$ Wires



11

Storage Component: Registers with control signals

Registers: If C then $R \leftarrow D$



operation

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

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

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

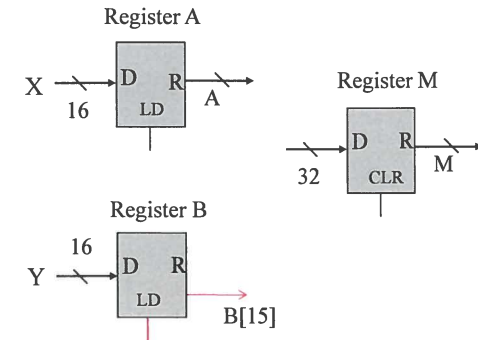
$i \leftarrow \text{Clear}(i)$

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

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

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

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



12

Data Subsystem

operation

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

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

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

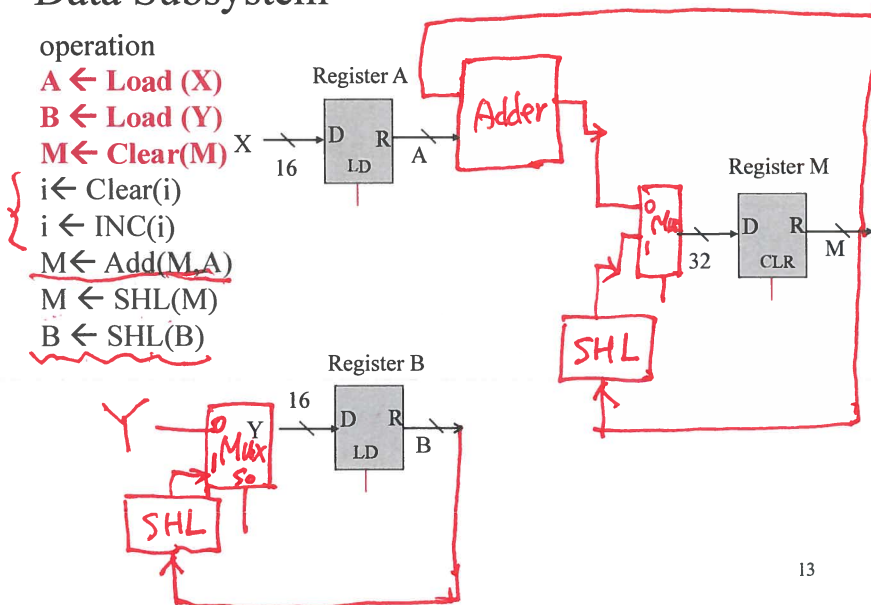
$i \leftarrow \text{Clear}(i)$

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

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

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

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



13

Function Modules: Adder, Shifter

operation

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

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

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

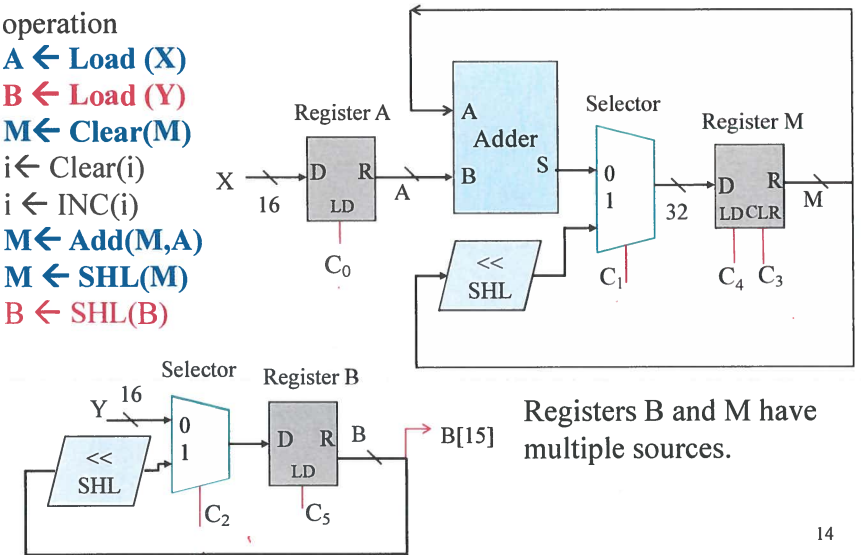
$i \leftarrow \text{Clear}(i)$

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

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

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

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



Registers B and M have multiple sources.

14

Function Modules: Adder, Shifter, Counter

operation

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

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

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

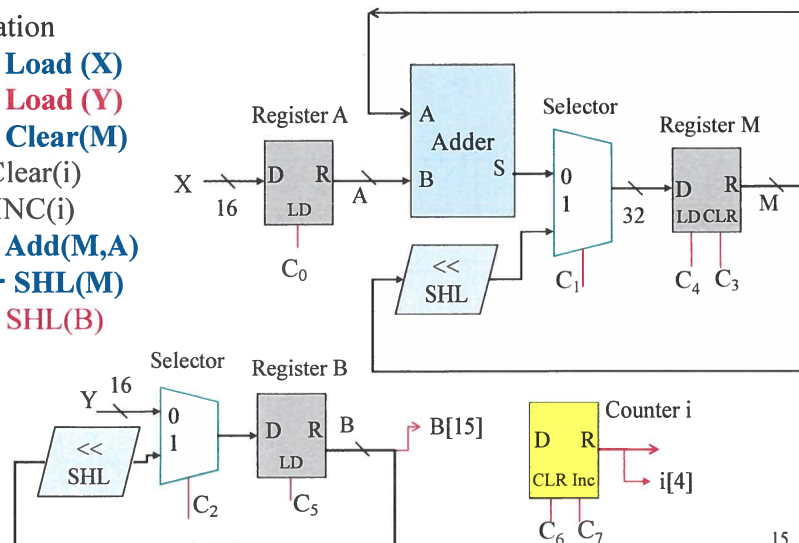
$i \leftarrow \text{Clear}(i)$

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

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

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

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



15

Step 6: Map Control Signals to Operations

operation

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

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

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

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

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

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

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

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

control

$C_0=1$

$C_2=0$ and $C_5=1$

$C_3=1$

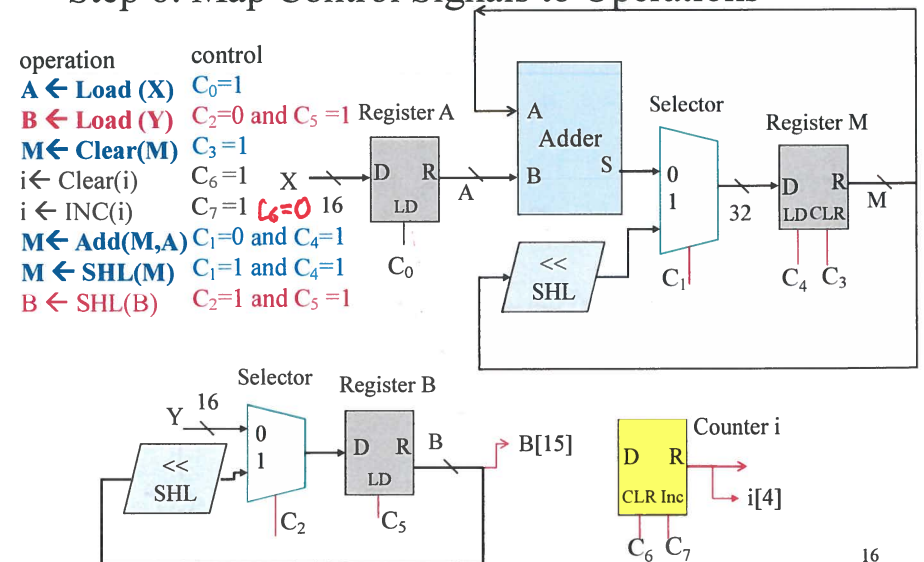
$C_6=1$

$C_7=1$

$C_1=0$ and $C_4=1$

$C_1=1$ and $C_4=1$

$C_2=1$ and $C_5=1$



16

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 ← 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

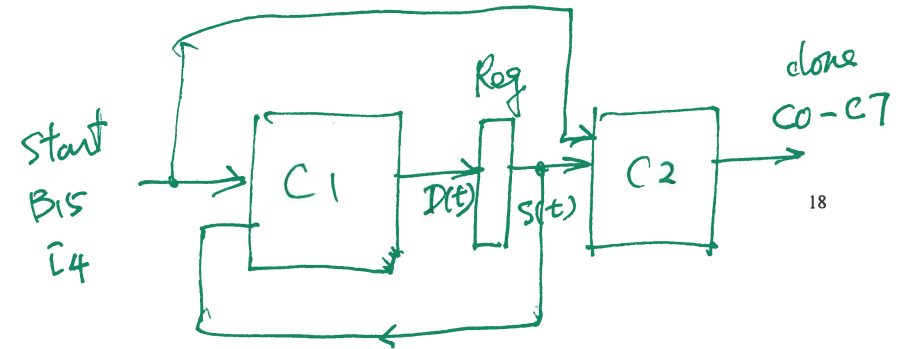
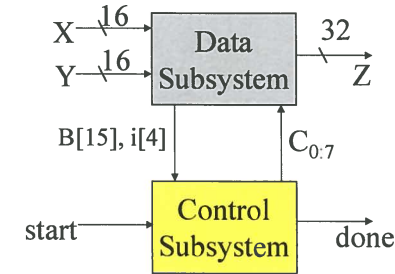
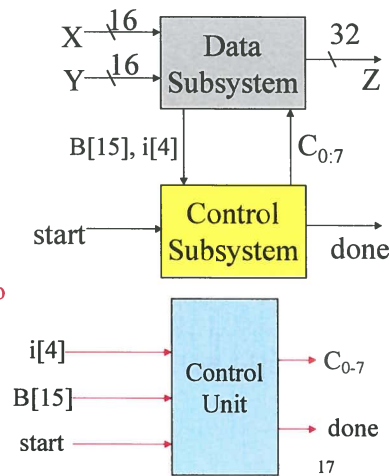
S5: M ← Shift(M,L,1) || B ← Shift(B,L,1) || goto S2;

S2;

S6: Z ← M || done ← 1 || goto S0;

}

Z=XY



operation control

A ← Load (X) C₀=1

B ← Load (Y) C₂=0 and C₅=1

M ← Clear(M) C₃=1

i ← Clear(i) C₆=1

i ← INC(i) C₇=1 C₆=0

M ← Add(M,A) C₁=0 and C₄=1

M ← SHL(M) C₁=1 and C₄=1

B ← SHL(B) C₂=1 and C₅=1

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;}

operation control

A ← Load (X) C₀=1

B ← Load (Y) C₂=0 and C₅=1

M ← Clear(M) C₃=1

i ← Clear(i) C₆=1

i ← INC(i) C₇=1

M ← Add(M,A) C₁=0 and C₄=1

M ← SHL(M) C₁=1 and C₄=1

B ← SHL(B) C₂=1 and C₅=1

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;}

	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	—	0	1	0	1	1	0	0
S2	0	—	—	0	0	0	1	1	0
S3	0	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

19

FSM

	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

S2 → C7 S1 → C6 S1 → C5

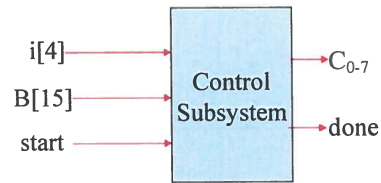
Done
S0 → Done
S6 → Done
S1 → C0
S5 → C1
S5 → C2
S1 → C3
S3 → C4
S5 → C4

20

Design of 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
}
```

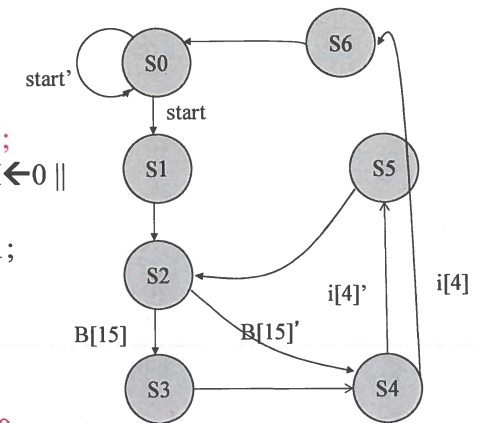


21

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
}
```



22

State Assignment

Use one F-F for one state.

At any time, one & only

one F-F holds a true value.

Binary	b ₂ b ₁ b ₀	Gray	b ₂ b ₁ b ₀	One Hot	b ₇ b ₆ b ₅ b ₄ b ₃ b ₂ b ₁ b ₀
S0	000	S0	000	S0	0 0 0 0 0 0 0 1
S1	001	S1	001	S1	0 0 0 0 0 0 1 0
S2	010	S2	011	S2	0 0 0 0 0 1 0 0
S3	011	S3	010	S3	0 0 0 0 1 0 0 0
S4	100	S4	110	S4	0 0 0 1 0 0 0 0
S5	101	S5	111	S5	0 0 1 0 0 0 0 0
S6	110	S6	101	S6	0 1 0 0 0 0 0 0
S7	111	S7	100	S7	1 0 0 0 0 0 0 0

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

23

Control Subsystem: One-Hot State Machine Design

Input: State Diagram

1. Use a flip flop to replace each state.

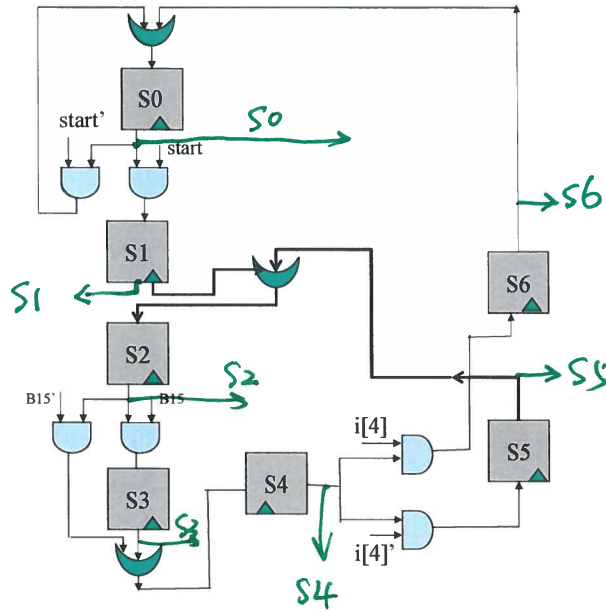
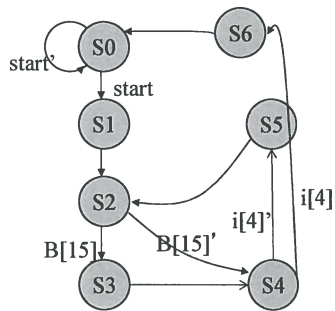
Set the flip flop which corresponds to the initial state and reset the rest flip flops.

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

3. Use a Demux to distribute the outward edges.

24

One-Hot State Machine



25

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.

26

Implementation: Example

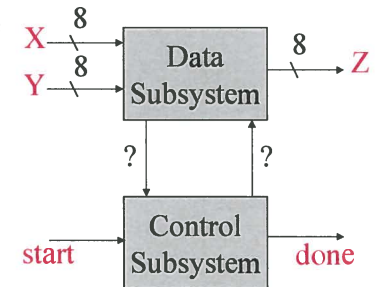
Given a hardware program, implement data path and control subsystems

```
{ 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;
}
```

27

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;
}
```

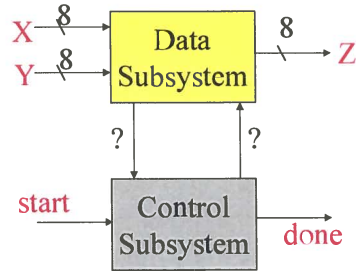


28

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;
}
```



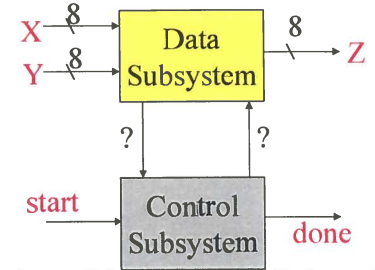
$$Z = \begin{cases} 4 \text{ Ceiling}[(X + |Y|) / 2] & \text{if } X < |Y| \\ 4X & \text{otherwise} \end{cases}$$

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;
}
```



30

Step 2: Map Data Operations to Implementable functions

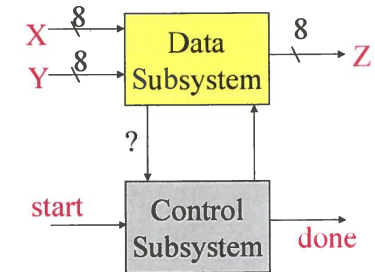
```
{Input X[7:0], Y[7:0] type bit-vector,
  start type boolean;
  Local-Object A[7:0], B[7:0] type bit-
vector;
  Output Z[7:0] type bit-vector,
  done type boolean;
```

```
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;
}
```

	operation
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)

Step 3: Tag each Data Operations with a Control Signal

	operation
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)



32

31