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⇐1;
S1: A⇐ X || B⇐ Y || i⇐0 || M⇐0 || done⇐0;
S2: If B_{15} = 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
}
RTL Design (contd)

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

\[
\text{Multiply}(X, Y, Z, \text{start}, \text{done})
\]
\[
\{
\text{S0: If start' goto S0 } \parallel \text{ done } \leftarrow 1;
\text{S1: } A \leftarrow X \parallel B \leftarrow Y \parallel i \leftarrow 0 \parallel M \leftarrow 0 \parallel \text{ done } \leftarrow 0;
\text{S2: If } B_{15} = 0 \text{ goto S4 } \parallel i \leftarrow i+1;
\text{S3: } M \leftarrow M + A;
\text{S4: if } i \geq 16, \text{ goto S6}
\text{S5: } M \leftarrow \text{Shift}(M,L,1) \parallel B \leftarrow \text{Shift}(B,L,1) \parallel \text{ goto S2};
\text{S6: } Z \leftarrow M \parallel \text{ done } \leftarrow 1 \parallel \text{ goto S0}
\}
\]

operation
\[
\text{A } \leftarrow \text{ Load (X)}
\text{B } \leftarrow \text{ Load (Y)}
\text{B } \leftarrow \text{ SHL(B)}
\text{M } \leftarrow \text{ Clear(M)}
\text{M } \leftarrow \text{ Add(M,A)}
\text{M } \leftarrow \text{ SHL(M)}
\text{i } \leftarrow \text{ Clear(i)}
\text{i } \leftarrow \text{ INC(i)}
\]
Step 2d: Map Control Signals to Operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Control Signal Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>A ← Load (X)</td>
<td>$C_0 = 1$</td>
</tr>
<tr>
<td>B ← Load (Y)</td>
<td>$C_5 = 0$ and $C_3 = 1$</td>
</tr>
<tr>
<td>B ← SHL(B)</td>
<td>$C_5 = 1$ and $C_3 = 1$</td>
</tr>
<tr>
<td>M ← Clear(M)</td>
<td>$C_2 = 1$</td>
</tr>
<tr>
<td>M ← Add(M, A)</td>
<td>$C_4 = 0$ and $C_1 = 1$</td>
</tr>
<tr>
<td>M ← SHL(M)</td>
<td>$C_4 = 1$ and $C_1 = 1$</td>
</tr>
<tr>
<td>i ← Clear(i)</td>
<td>$C_6 = 1$</td>
</tr>
<tr>
<td>i ← INC(i)</td>
<td>$C_7 = 1$</td>
</tr>
</tbody>
</table>

Source: CK Cheng
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: «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.

Source: CK Cheng
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.

Source: CK Cheng
One-Hot State Machine

Is the next state logic for S1 correct?
A. Yes
B. No

Source: CK Cheng
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

Source: CK Cheng
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: ← M || done ← 1 || goto S0
}

<table>
<thead>
<tr>
<th></th>
<th>C0</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4 (mux)</th>
<th>C5 (mux)</th>
<th>C6</th>
<th>C7</th>
<th>done</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Multiply**$(X, Y, Z, \text{start, done})$

{
S0: If start’ goto S0 || done $\leftarrow 1$;  
S1: $C_0=1$ || $C_5=0$ and $C_3=1$ || $C_6=1$ || $C_2=1$ || done $\leftarrow 0$;  
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 $\leftarrow 1$ || goto S0

<table>
<thead>
<tr>
<th></th>
<th>C0</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4 (mux)</th>
<th>C5 (mux)</th>
<th>C6</th>
<th>C7</th>
<th>done</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>S1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>S3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>C0</td>
<td>C1</td>
<td>C2</td>
<td>C3</td>
<td>C4 (mux)</td>
<td>C5 (mux)</td>
<td>C6</td>
<td>C7</td>
<td>done</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---------</td>
<td>---------</td>
<td>---</td>
<td>---</td>
<td>------</td>
</tr>
<tr>
<td>S0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>S1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>S3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Source: CK Cheng
Control subsystem

S0 \rightarrow S1 \rightarrow S2 \rightarrow S3 \rightarrow S4 \rightarrow S5 \rightarrow S6

B[15] \rightarrow S3 \rightarrow S4

i[4] \rightarrow S2

start \rightarrow S0

B[15]’ \rightarrow S4

S0 \rightarrow S1 \rightarrow S2 \rightarrow S3 \rightarrow S4 \rightarrow S5 \rightarrow S6

source: CK Cheng
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 ← 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;

}
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 ≜ 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;
}
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 ← 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;
}

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 ← 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;
}
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 ← 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;
}
Step 3: Implement the Data Subsystem from Standard Modules

- Operation:
  - Load (X)
  - INC(A)
  - Load (Y)
  - SUB (0,B)
  - DEC(B)
  - Comp (A, B)
  - SHL(A,2)
  - B[7]

Diagram: Register B and Register A
operation
A ← Load (X)
A ← INC(A)
B ← Load (Y)
B ← SUB (0,B)
B ← DEC(B)
Z ← SHL(A,2)
Cmp (A, B)
B[7]

start

Control Unit
done

Cmp (A, B)
A_gt_B

<< 2

X

A

INC

Y

B

SUB DEC

Z

Cmp

C1

C2

C3

C4

C5

C6

C0
operation
A \leftarrow \text{Load} (X) \quad C_4 = 0 \text{ and } C_1 = 1
A \leftarrow \text{INC}(A) \quad C_4 = 1 \text{ and } C_1 = 1
B \leftarrow \text{Load} (Y) \quad C_5 = 1, \quad C_3 = 0, \quad C_2 = 1
B \leftarrow \text{SUB} (0, B) \quad C_5 = 0, \quad C_3 = 1, \quad C_2 = 1
B \leftarrow \text{DEC}(B) \quad C_5 = 0, \quad C_3 = 0, \quad C_2 = 1
Z \leftarrow \text{SHL}(A, 2) \quad C_6 = 1,
Cmp (A, B)
B[7]

\begin{align*}
A \text{ \_gt\_} B \\
\text{B[7]} \\
\text{start} \\
\text{Control Unit} \\
\text{done}
\end{align*}
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  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;
}
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;
}

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

operation
A ← Load (X) C₄ =0 and C₁ =1
A ← INC(A) C₄ =1 and C₁ =1
B ← Load (Y) C₅ =1, C₃ =0, C₂ =1
B ← SUB (0,B) C₅ =0, C₃ =1, C₂ =1
B ← DEC(B) C₅ =0, C₃ =0, C₂ =1
Z ← SHL(A,2) C₆ =1,
Cmp (A, B) A_gt_B
B>=0 B[7]
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;
}

Some_function
{
    S0: If start’ goto Wait;
    S1: C_4 =0, C_1 =1 || C_5 =1, C_3 =0, C_2 =1 || done ← 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|| C_5 =0, C_3 =0, C_2 =1 || goto S4;
    S6: C_6 =1 || done ← 1 || goto S0;
}
Some_function
{
    S0: If start’ goto Wait;
    S1: C_4 =0, C_1 =1 || C_5 =1, C_3 =0, C_2 =1 || 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|| C_5 =0, C_3 =0, C_2 =1 || goto S4;
    S6: C_6 =1 || done \leftarrow 1 || goto S0;
}
Some_function
{
    S0: If start’ goto Wait;
    S1: C_4 = 0, C_1 = 1 || C_5 = 1, C_3 = 0, C_2 = 1 || 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 || C_5 = 0, C_3 = 0, C_2 = 1 || goto S4;
    S6: C_6 = 1 || done \leftarrow 1 || goto S0;
}
One-Hot State Machine