CSE140: System Design

Xinyuan Wang

05/31/2019
Example: SP18 Final

(System Designs) Implement the following algorithm:

Alg(X, Y, Z, start, W, done);
Input X[7:0], Y[7:0], Z[7:0], start;
Output W[7:0], done;
Local-object A[7:0], B[7:0], C[7:0];
S0: If start' goto S0 || done <= 1;
S1: A <= X || B <= Y || C <= Z || done <= 0;
S2: if (A[7]) goto S5 || B <= B+C;
S3: A <= A+B;
S4: if C'[7] goto S2 || C <= Inc(C);
S5: A <= Inc(A) || if C[7] goto S2;
S6: W <= C || goto S0;
End Alg
Framework

Data Subsystem

Control Subsystem

Data Inputs

Control Inputs

Conditions

Control Signals

Start/Request

Done/Acknowledgement

n=128

n=64
Handshaking

Master that calls module S

Start/Request

Done/Acknowledgment

Data Subsystem

Control Subsystem

Data Inputs

Control Inputs

Conditions

Control Signals

Data Outputs

Control Outputs

Start done

Data n=128

Control n=64

module S

X

Y

Z

X

Y

Z

n=128

n=64
Before implementing the data and control subsystems

With the given description of system

- Identify Input and Output of data and control subsystems
- Identify Condition Bits to Control Subsystem
- Identify Data Subsystem Operations
- Map Data Operations to Implementable functions
Before implementing the data and control subsystems

With the given description of system

• Identify Input and Output of data and control subsystems

Alg(X, Y, Z, start, W, done);
Input X[7:0], Y[7:0], Z[7:0], start
Output W[7:0], done
Local-object A[7:0], B[7:0], C[7:0]
S0: If start’ goto S0 || done \(\Leftarrow\) 1;
S1: A \(\Leftarrow\) X || B \(\Leftarrow\) Y || C \(\Leftarrow\) Z || done \(\Leftarrow\) 0;
S2: if (A[7]) goto S5 || B \(\Leftarrow\) B+C;
S3: A \(\Leftarrow\) A+B;
S4: if C’[7] goto S2 || C \(\Leftarrow\) Inc(C);
S5: A \(\Leftarrow\) Inc(A) || if C[7] goto S2;
S6: W \(\Leftarrow\) C || goto S0;
End Alg
Before implementing the data and control subsystems

With the given description of system

• Identify Input and Output of data and control subsystems

• Identify Condition Bits to Control Subsystem

Alg(X, Y, Z, start, W, done);
Input X[7:0], Y[7:0], Z[7:0], start;
Output W[7:0], done;
Local-object A[7:0], B[7:0], C[7:0];
S0: If start’ goto S0 || done <=> 1;
S1: A <= X || B <= Y || C <= Z || done <= 0;
S2: if (A[7]) goto S5 || B <= B+C;
S3: A <= A+B;
S4: if [C’7] goto S2 || C <= Inc(C);
S5: A <= Inc(A) || if C[7] goto S2;
S6: W <= C || goto S0;
End Alg
Before implementing the data and control subsystems

With the given description of system

• Identify Input and Output of data and control subsystems
• Identify Condition Bits to Control Subsystem

• Identify Data Subsystem Operations

Alg(X, Y, Z, start, W, done);
Input X[7:0], Y[7:0], Z[7:0], start;
Output W[7:0], done;
Local-object A[7:0], B[7:0], C[7:0];
S0: If start’ goto S0 || done \leftarrow 1;
S1: \[A \leftarrow X \parallel B \leftarrow Y \parallel C \leftarrow Z\] || done \leftarrow 0;
S2: if (A[7]) goto S5 || B \leftarrow B+C;
S3: A \leftarrow A+B;
S4: if C’[7] goto S2 || C \leftarrow Inc(C);
S5: A \leftarrow Inc(A) || if C[7] goto S2;
S6: W \leftarrow C || goto S0;
End Alg
Before implementing the data and control subsystems

With the given description of system

- Identify Input and Output of data and control subsystems
- Identify Condition Bits to Control Subsystem
- Identify Data Subsystem Operations

- **Map Data Operations to Implementable functions**

```plaintext
Alg(X, Y, Z, start, W, done);
Input X[7:0], Y[7:0], Z[7:0], start;
Output W[7:0], done;
Local-object A[7:0], B[7:0], C[7:0];
S0: If start’ goto S0 || done <= 1;
S1: A ← X || B ← Y || C ← Z || done ≤ 0;
S2: if (A[7]) goto S5 || B ← B+C;
S3: A ← A+B;
S4: if C’[7] goto S2 || C ← Inc(C);
S5: A ← Inc(A) || if C[7] goto S2;
S6: W ← C || goto S0;
End Alg
```

<table>
<thead>
<tr>
<th>state</th>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>A ← X</td>
<td>A ← Load(X)</td>
</tr>
<tr>
<td>S1</td>
<td>B ← Y</td>
<td>B ← Load(Y)</td>
</tr>
<tr>
<td>S1</td>
<td>C ← Z</td>
<td>C ← Load(Z)</td>
</tr>
<tr>
<td>S2</td>
<td>B ← B+C</td>
<td>B ← Add(B,C)</td>
</tr>
<tr>
<td>S3</td>
<td>A ← A+B</td>
<td>A ← Add(A,B)</td>
</tr>
<tr>
<td>S4</td>
<td>C ← Inc(C)</td>
<td>C ← Inc(C)</td>
</tr>
<tr>
<td>S5</td>
<td>A ← Inc(A)</td>
<td>A ← Inc(A)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>state</th>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>done ← 1</td>
<td></td>
</tr>
<tr>
<td>S6</td>
<td>W ← C</td>
<td>Wire</td>
</tr>
</tbody>
</table>
Implement the data subsystem

• List data operations
• Map operations to functional blocks
• Add interconnect for data transport
• Input control signals and output conditions
Implement the data subsystem

- List data operations
- **Map operations to functional blocks: registers**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A [\leftarrow] Load(X)</td>
<td></td>
</tr>
<tr>
<td>B [\leftarrow] Load(Y)</td>
<td></td>
</tr>
<tr>
<td>C [\leftarrow] Load(Z)</td>
<td></td>
</tr>
<tr>
<td>B [\leftarrow] Add(B,C)</td>
<td></td>
</tr>
<tr>
<td>A [\leftarrow] Add(A,B)</td>
<td></td>
</tr>
<tr>
<td>C [\leftarrow] Inc(C)</td>
<td></td>
</tr>
<tr>
<td>A [\leftarrow] Inc(A)</td>
<td></td>
</tr>
</tbody>
</table>

Registers: If C then R \[\leftarrow\] D
Implement the data subsystem

• List data operations

• **Map operations to functional blocks: adders**

• Add interconnect for data transport

<table>
<thead>
<tr>
<th>Operation</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A $\leftarrow$ Load(X)</td>
<td></td>
</tr>
<tr>
<td>B $\leftarrow$ Load(Y)</td>
<td></td>
</tr>
<tr>
<td>C $\leftarrow$ Load(Z)</td>
<td></td>
</tr>
<tr>
<td>B $\leftarrow$ Add(B,C)</td>
<td>✔</td>
</tr>
<tr>
<td>A $\leftarrow$ Add(A,B)</td>
<td></td>
</tr>
<tr>
<td>C $\leftarrow$ Inc(C)</td>
<td></td>
</tr>
<tr>
<td>A $\leftarrow$ Inc(A)</td>
<td></td>
</tr>
</tbody>
</table>
Implement the data subsystem

• List data operations

• **Map operations to functional blocks:** *mux for multiple sources*

• Add interconnect for data transport

<table>
<thead>
<tr>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A ← Load(X)</td>
</tr>
<tr>
<td>B ← Load(Y)</td>
</tr>
<tr>
<td>C ← Load(Z)</td>
</tr>
<tr>
<td>B ← Add(B, C)</td>
</tr>
<tr>
<td>A ← Add(A, B)</td>
</tr>
<tr>
<td>C ← Inc(C)</td>
</tr>
<tr>
<td>A ← Inc(A)</td>
</tr>
</tbody>
</table>
Implement the data subsystem

- List data operations
- Map operations to functional blocks
- Add interconnect for data transport
- **Input control signals and output conditions**
Implement the control subsystem

• Map control signals to operations
• Identify control path components
• Express the control subsystem with FSM
• Implement the control subsystem
Implement the control subsystem

- Map control signals to operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Control Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>A ← Load(X)</td>
<td>( C_0 = 0, L_0 = 1 ) and ( L_1 = 0 )</td>
</tr>
<tr>
<td>B ← Load(Y)</td>
<td>( C_1 = 0 ) and ( L_2 = 1 )</td>
</tr>
<tr>
<td>C ← Load(Z)</td>
<td>( L_3 = 1 ) and ( L_4 = 0 )</td>
</tr>
<tr>
<td>B ← Add(B,C)</td>
<td>( C_1 = 1 ) and ( L_2 = 1 )</td>
</tr>
<tr>
<td>A ← Add(A,B)</td>
<td>( C_0 = 1, L_0 = 1 ) and ( L_1 = 0 )</td>
</tr>
<tr>
<td>C ← Inc(C)</td>
<td>( L_3 = 0 ) and ( L_4 = 1 )</td>
</tr>
<tr>
<td>A ← Inc(A)</td>
<td>( L_0 = 0 ) and ( L_1 = 1 )</td>
</tr>
</tbody>
</table>
Implement the control subsystem

- Map control signals to operations
- **Identify control path components**

```
Alg(X, Y, Z, start, W, done);
Input X[7:0], Y[7:0], Z[7:0], start;
Output W[7:0], done;
Local-object A[7:0], B[7:0], C[7:0];
S0: If start’ goto S0 || done <= 1;
S1: A <= X || B <= Y || C <= Z || done <= 0;
S2: if (A[7]) goto S5 || B <= B+C;
S3: A <= A+B;
S4: if C[7] goto S2 || C <= Inc(C);
S5: A <= Inc(A) || if C[7] goto S2;
S6: W <= C || goto S0;
End Alg
```
Implement the control subsystem

<table>
<thead>
<tr>
<th>state</th>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>A ← X</td>
<td>A ← Load(X)</td>
</tr>
<tr>
<td>S1</td>
<td>B ← Y</td>
<td>B ← Load(Y)</td>
</tr>
<tr>
<td>S1</td>
<td>C ← Z</td>
<td>C ← Load(Z)</td>
</tr>
<tr>
<td>S2</td>
<td>B ← B+C</td>
<td>B ← Add(B,C)</td>
</tr>
<tr>
<td>S3</td>
<td>A ← A+B</td>
<td>A ← Add(A,B)</td>
</tr>
<tr>
<td>S4</td>
<td>C ← Inc(C)</td>
<td>C ← Inc(C)</td>
</tr>
<tr>
<td>S5</td>
<td>A ← Inc(A)</td>
<td>A ← Inc(A)</td>
</tr>
</tbody>
</table>

control

$$C_0=0, L_0=1 \text{ and } L_1=0$$
$$C_1=0 \text{ and } L_2=1$$
$$L_3=1 \text{ and } L_4=0$$
$$C_0=1, L_0=1 \text{ and } L_1=0$$
$$L_3=0 \text{ and } L_4=1$$
$$L_0=0 \text{ and } L_1=1$$

Alg(X, Y, Z, start, W, done);
Input X[7:0], Y[7:0], Z[7:0], start;
Output W[7:0], done;
Local-object A[7:0], B[7:0], C[7:0];
S0: If start’ goto S0 || done ← 1;
S1: A ← X || B ← Y || C ← Z || done ← 0;
S2: if (A[7]) goto S5 || B ← B+C;
S3: A ← A+B;
S4: if C'[7] goto S2 || C ← Inc(C);
S5: A ← Inc(A) || if C[7] goto S2;
S6: W ← C || goto S0;
End Alg

<table>
<thead>
<tr>
<th>state</th>
<th>L0</th>
<th>L1</th>
<th>L2</th>
<th>L3</th>
<th>L4</th>
<th>c0</th>
<th>c1</th>
<th>done</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>S1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>S2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>S3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>S4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>S5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>S6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
</tr>
</tbody>
</table>
Implement the control subsystem

- Map control signals to operations
- Identify control path components
- **Express the control subsystem with FSM**

```
Alg(X, Y, Z, start, W, done);
Input X[7:0], Y[7:0], Z[7:0], start;
Output W[7:0], done;
Local-object A[7:0], B[7:0], C[7:0];
S0: If start’ goto S0 || done <= 1;
S1: A <= X || B <= Y || C <= Z || done <= 0;
S2: if (A[7]) goto S5 || B <= B+C;
S3: A <= A+B;
S4: if C’[7] goto S2 || C <= Inc(C);
S5: A <= Inc(A) || if C[7] goto S2;
S6: W <= C || goto S0;
End Alg
```
Implement the control subsystem

- Map control signals to operations
- Identify control path components
- Express the control subsystem with FSM

- **Implement the control subsystem**

How to implement the state assignment?
State Assignment

<table>
<thead>
<tr>
<th>Binary</th>
<th>$b_2b_1b_0$</th>
<th>Gray</th>
<th>$b_2b_1b_0$</th>
<th>One Hot</th>
<th>$b_7b_6b_5b_4b_3b_2b_1b_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>000</td>
<td>S0</td>
<td>000</td>
<td>S0</td>
<td>0 0 0 0 0 0 0 1</td>
</tr>
<tr>
<td>S1</td>
<td>001</td>
<td>S1</td>
<td>001</td>
<td>S1</td>
<td>0 0 0 0 0 0 1 0</td>
</tr>
<tr>
<td>S2</td>
<td>010</td>
<td>S2</td>
<td>011</td>
<td>S2</td>
<td>0 0 0 0 0 1 0 0</td>
</tr>
<tr>
<td>S3</td>
<td>011</td>
<td>S3</td>
<td>010</td>
<td>S3</td>
<td>0 0 0 1 0 0 0 0</td>
</tr>
<tr>
<td>S4</td>
<td>100</td>
<td>S4</td>
<td>110</td>
<td>S4</td>
<td>0 0 1 0 0 0 0 0</td>
</tr>
<tr>
<td>S5</td>
<td>101</td>
<td>S5</td>
<td>111</td>
<td>S5</td>
<td>0 1 0 0 0 0 0 0</td>
</tr>
<tr>
<td>S6</td>
<td>110</td>
<td>S6</td>
<td>101</td>
<td>S6</td>
<td>0 1 0 0 0 0 0 0</td>
</tr>
<tr>
<td>S7</td>
<td>111</td>
<td>S7</td>
<td>100</td>
<td>S7</td>
<td>1 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

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

21
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.
Implement the control subsystem

- Map control signals to operations
- Identify control path components
- Express the control subsystem with FSM
- **Implement the control subsystem**