Principles Of Digital Design

Chapter 8

Register Transfer Specification And Design
Register-transfer design

• Each standard or custom IC consists of one or more datapaths and control units.

• To synthesize such IC we introduce the model of a FSM with a datapath (FSMD).

• We demonstrate synthesis algorithms for FSMD model, including component selection, resource sharing, pipelining and scheduling.
Design Model

High-level block diagram

Control unit
Datapath

Control inputs
Datapath inputs

Control signals
Status signals

Control outputs
Datapath outputs

Datapath inputs

Datapath

Register-transfer-level block diagram

Control inputs
Datapath inputs

Control signals
Status signals

Control outputs
Datapath outputs

Datapath

Selector
Register
RF
Mem

ALU
*/-

Register

Next-state logic
State register

Control unit
Output logic

Copyright © 2004-2005 by Daniel D. Gajski

Slides by Xi Cheng, University of California, Irvine
Ones-counter specification

Start = 0

\[ s_0 \]

Start = 1

\[ s_1 \]
Done = 0; Data = Input

\[ s_2 \]
Done = 0; Ocount = 0

\[ s_3 \]
Done = 0; Mask = 1

\[ s_4 \]
Done = 0; Temp = Data \& Mask

Data = 0

\[ s_5 \]
Done = 0; Ocount = Ocount + Temp

\[ s_6 \]
Done = 1; Data = Data >> 1

Data = 0

\[ s_7 \]
Done = 1; Output = Ocount
FSDM Definition

In Chapter 6 we defined an FSM as a quintuple \(< S, I, O, f, h >\)
where \(S\) is a set of states, \(I\) and \(O\) are the sets of input and output
symbols: \(f: S \times I \rightarrow S\), and \(h: S \times I \rightarrow O\).

More precisely, \(I = A_1 \times A_2 \times \ldots A_k\),
\(S = Q_1 \times Q_2 \times \ldots Q_m\),
\(O = Y_1 \times Y_2 \times \ldots Y_n\).

Where \(A_i, 1 \leq i \leq k\), is an input signal, \(Q_i, 1 \leq i \leq m\) is the flip-flop output
and \(Y_i, 1 \leq i \leq n\) is an output signal.

To define a FSMD, we define a set of variables \(V = V_1 \times V_2 \times \ldots V_q\)
which defines the state of the datapath by defining the values of all
variables in each state.

\[
\begin{align*}
I &= U_{i=1} U \{ (W) \in W \} \\
&= \{ V \in \{ \leq p = f \geq \} \}
\end{align*}
\]

\[I = I_C \times I_D\]

where \(I_C = A_1 \times A_2 \times \ldots A_k\) as before and \(I_D = B_1 \times B_2 \times \ldots B_p\).

\[O = O_C \times O_D\]

Where \(O_C = Y_1 \times Y_2 \times \ldots Y_n\) as before and \(O_D = Z_1 \times Z_2 \times \ldots Z_r\).
FSDM Definition

With formal definition of expressions and relations over a set of variables we can simplify function $f : (S \times V) \times I \rightarrow S \times V$ by separating it into two parts: $f_C$ and $f_D$. The function $f_C$ defines the next state of the control unit:

$$f_C : S \times lC \times STAT \rightarrow S$$

while the function $f_D$ defines the values of datapath variables in the next state:

$$f_D : S \times V \times ID \rightarrow V$$

$$f_D := \{ f_{Di} : V \times ID \rightarrow V : \{ V_j = e_j \mid V_j \in V, e_j \in \text{Expr} (V \times ID) \} \}$$

Also,

$$h_C : S \times lC \times STAT \rightarrow Oc$$

and

$$h_D : S \times V \times ID \rightarrow OD$$
### FSMD specification of Ones-counter

#### State and output table

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next state (Start. Data=0)</th>
<th>Control Output</th>
<th>Datapath output</th>
<th>Data Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>00 01 10 11</td>
<td>Done</td>
<td>Outport</td>
<td></td>
</tr>
<tr>
<td><strong>s0</strong></td>
<td>s0 s0 s1 s1</td>
<td>0</td>
<td>Z</td>
<td>Data = Inport</td>
</tr>
<tr>
<td><strong>s1</strong></td>
<td>s2 s2 s2 s2</td>
<td>0</td>
<td>Z</td>
<td>Ocount = 0</td>
</tr>
<tr>
<td><strong>s2</strong></td>
<td>s3 s3 s3 s3</td>
<td>0</td>
<td>Z</td>
<td>Mask = 1</td>
</tr>
<tr>
<td><strong>s3</strong></td>
<td>s4 s4 s4 s4</td>
<td>0</td>
<td>Z</td>
<td>Temp = Data AND Mask</td>
</tr>
<tr>
<td><strong>s4</strong></td>
<td>s5 s5 s5 s5</td>
<td>0</td>
<td>Z</td>
<td>Ocount = Ocount + Temp</td>
</tr>
<tr>
<td><strong>s5</strong></td>
<td>s6 s6 s6 s6</td>
<td>0</td>
<td>Z</td>
<td>Data = Data &gt;&gt; 1</td>
</tr>
<tr>
<td><strong>s6</strong></td>
<td>s7 s7 s7 s7</td>
<td>0</td>
<td>Z</td>
<td></td>
</tr>
<tr>
<td><strong>s7</strong></td>
<td>s0 s0 s0 s0</td>
<td>1</td>
<td>Ocount</td>
<td></td>
</tr>
</tbody>
</table>

#### Datapath Variables

<table>
<thead>
<tr>
<th>Data</th>
<th>Ocount</th>
<th>Temp</th>
<th>Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Inport</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Data</td>
<td>0</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Data</td>
<td>Ocount</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>Data</td>
<td>Ocount</td>
<td>Data AND Mask</td>
<td>Mask</td>
</tr>
<tr>
<td>Data</td>
<td>Ocount</td>
<td>X</td>
<td>Mask</td>
</tr>
<tr>
<td>Data &gt;&gt; 1</td>
<td>Ocount</td>
<td>X</td>
<td>Mask</td>
</tr>
<tr>
<td>Data</td>
<td>Ocount</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

#### State-action table

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next state (condition, state)</th>
<th>Control and Datapath actions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>s0</strong></td>
<td>Start = 0 s0</td>
<td>Done = 0</td>
</tr>
<tr>
<td><strong>s1</strong></td>
<td>Start = 1 s1</td>
<td>Data = Inport</td>
</tr>
<tr>
<td><strong>s2</strong></td>
<td>s2</td>
<td>Data = Inport</td>
</tr>
<tr>
<td><strong>s3</strong></td>
<td>s3</td>
<td>Ocount = 0</td>
</tr>
<tr>
<td><strong>s4</strong></td>
<td>s4</td>
<td>Mask = 1</td>
</tr>
<tr>
<td><strong>s5</strong></td>
<td>s5</td>
<td>Temp = Data AND Mask</td>
</tr>
<tr>
<td><strong>s6</strong></td>
<td>Data 0 s4</td>
<td>Ocount = Ocount + Temp</td>
</tr>
<tr>
<td><strong>s7</strong></td>
<td>Data = Data &gt;&gt; 1</td>
<td>Data = Data &gt;&gt; 1</td>
</tr>
</tbody>
</table>

---

Copyright © 2004-2005 by Daniel D. Gajski

Slides by Xi Cheng, University of California, Irvine
Algorithmic-State-Machine

- Graphic representation of FSMD model
- Equivalent to state-action table
- Similar to a flowchart used for program description
### ASM Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>State box</td>
<td>Unconditional variable and output assignment</td>
<td>$s_3$</td>
</tr>
<tr>
<td>Decision Box</td>
<td>Conditional variable assignment</td>
<td>Data = 0</td>
</tr>
<tr>
<td>Condition Box</td>
<td></td>
<td>Ocount = Ocount + 1</td>
</tr>
<tr>
<td>ASM Block</td>
<td></td>
<td>Done = 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Data = Input</td>
</tr>
</tbody>
</table>
**ASM rules**

- Rule 1: The chart must define a unique next state for each state and set of conditions.

- Rule 2: Every path defined by the network of condition boxes must lead to another state.

---

**Undefined next state**

**Undefined exit path**

---

Copyright © 2004-2005 by Daniel D. Gajski

Slides by Xi Cheng, University of California, Irvine
ASM chart for Ones-counter

(a) State-based (Moore) chart

(b) Input-based (Mealy) chart
State-action tables for Ones-counter

<table>
<thead>
<tr>
<th>Present State</th>
<th>State Name</th>
<th>Next state Condition</th>
<th>Next state State</th>
<th>Datapath actions condition</th>
<th>Datapath actions Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>s0</td>
<td>Start = 0, Start = 1,</td>
<td>s0</td>
<td></td>
<td>Done = 0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>s1</td>
<td>Data = 0,</td>
<td>s2</td>
<td></td>
<td>Data = Inport</td>
</tr>
<tr>
<td>0 1 0</td>
<td>s2</td>
<td>DataLSR = 1,</td>
<td>s3</td>
<td></td>
<td>Ocount = 0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>s2</td>
<td>DataLSR = 0,</td>
<td>s4</td>
<td></td>
<td>Ocount = Ocount + 1</td>
</tr>
<tr>
<td>1 0 0</td>
<td>s4</td>
<td>Data ≠ 0, Data = 0,</td>
<td>s4</td>
<td></td>
<td>Data = Data &gt;&gt; 1</td>
</tr>
<tr>
<td>1 0 1</td>
<td>s5</td>
<td></td>
<td>s0</td>
<td></td>
<td>Done = 1, Output = Ocount</td>
</tr>
</tbody>
</table>

State-based table

- \( = = \) = + + ≠
- \( = \)  + + ≠
- \( = = + \) + ≠
- \( = + + \) ≠
- \( = = + + \) + ≠
- \( = + + \) ≠
- \( = = \) =
- \( = + = + \)
- \( = = \)
- \( = = \)
- \( = = \)
## State-action tables for Ones-counter

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next state Condition</th>
<th>Datapath actions</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Q_1 Q_0 )</td>
<td>State</td>
<td>Operation</td>
<td>Next state</td>
</tr>
<tr>
<td>0 0</td>
<td>( s_0 )</td>
<td>( \text{Start} = 0, \text{Start} = 1 )</td>
<td>( s_0 )</td>
</tr>
<tr>
<td>0 1</td>
<td>( s_1 )</td>
<td>Data ( \neq 0 ), Data = 0,</td>
<td>( s_2 ), ( s_3 )</td>
</tr>
<tr>
<td>1 0</td>
<td>( s_2 )</td>
<td>Data LSR = 1, Data ( \neq 0 ),</td>
<td>( s_0 )</td>
</tr>
<tr>
<td>1 1</td>
<td>( s_3 )</td>
<td>Data LSR = 1, Data ( \neq 0 ),</td>
<td>( s_0 )</td>
</tr>
</tbody>
</table>

### Input-based table

\[
\begin{align*}
\text{Output} &= \text{Ocount} \\
\text{Ocount} &= 0 \\
\text{Start} &= 1, \quad \text{Done} = 0 \\
\text{Start} &= 0, \quad 0 0 \\
\text{Ocount} &= \text{Ocount} + 1 \\
\text{Data LSR} &= 1, \quad \text{Data} = \text{Inport} \\
\text{Data} &= \text{Data} >> 1 \\
\text{Done} &= 1 \quad \text{Output} = \text{Ocount} \\
\end{align*}
\]
Logic schematics for Ones-counter

- $D_2 = Q_2(next) = s_2\text{Data}_{LSB} + S_3 + S_4(\text{Data} \neq 0)'$
  $= Q_1Q'_0\text{Data}_{LSB} + Q_1Q_0 + Q_2Q'_0(\text{Data} \neq 0)'$
- $D_1 = Q_1(next) = s_1 + s_2\text{Data}_{LSB} + s_4(\text{Data} \neq 0)$
  $= Q'_2Q'_1Q'_0 + Q_1Q'_0\text{Data}_{LSB} + Q_2Q'_0(\text{Data} \neq 0)$
- $D_0 = Q_0(next) = s_0\text{Start} + s_2\text{Data}_{LSB} + s_4(\text{Data} \neq 0)'$
  $= Q'_2Q'_1Q'_0\text{Start} + Q_1Q'_0\text{Data}_{LSB} + Q_2Q'_0(\text{Data} \neq 0)'$

- $S_1 = s_4 = Q_2Q'_0$
- $S_0 = s_2 + s_4 = Q_1Q'_0 + Q_2Q'_0$
- $E = s_3 = Q_1Q_0$
- Load = $s_4 = Q'_2Q'_1Q_0$
- Done = Output enable = $s_5 = Q_2Q_0$
Logic schematics for Ones-counter

- \( D_1 = Q_1 \text{ (next)} = s_1 + s_2 = Q'_1Q_0 + Q_1Q'_0 \)
- \( D_0 = Q_0 \text{ (next)} = s_0 \text{Start} + s_2(\text{Data} \neq 0)' \\
  = Q'_1Q'_0\text{Start} + Q_1Q'_0(\text{Data} \neq 0) \)
- \( S_1 = s_2(\text{Data} \neq 0) = Q_1Q'_0(\text{Data} \neq 0) \)
- \( S_0 = s_1 + s_2(\text{Data} \neq 0) = Q'_1Q_0 + Q_1Q'_0(\text{Data} \neq 0) \)
- \( E = s_2\text{Data}_{LSB} = Q_1Q'_0\text{Data}_{LSB} \)
- \( \text{Load} = s_1 = Q'_1Q_0 \)
- \( \text{Done} = \text{Output enable} = s_3 = Q_1Q_0 \)
Register-transfer synthesis

- Register sharing
- Functional unit sharing
- Bus sharing

Block diagram of Square-root approximation

ASM Chart of Square-root approximation
Resource usage in square-root approximation

**Block diagram**

```
Start
   | Control
   v
In 1  In 2
   | Out
   v
Block diagram

Start

s0
  a = In 1
  b = In 2

s1
  t1 = |a|
  t2 = |b|

s2
  x = max( t1, t2 )
  y = min( t1, t2 )

s3
  t3 = x >> 3
  t4 = y >> 1

s4
  t5 = x - t3

s5
  t6 = t4 + t5

s6
  t7 = max( t6, x )

s7
  Done = 1
  Out = t7

ASM Chart of Square-root approximation
```

**Variable usage**

<table>
<thead>
<tr>
<th>t1</th>
<th>t2</th>
<th>x</th>
<th>y</th>
<th>t3</th>
<th>t4</th>
<th>t5</th>
<th>t6</th>
<th>t7</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>s2</td>
<td>s3</td>
<td>s4</td>
<td>s5</td>
<td>s6</td>
<td>s7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>t1</th>
<th>t2</th>
<th>x</th>
<th>y</th>
<th>t3</th>
<th>t4</th>
<th>t5</th>
<th>t6</th>
<th>t7</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

No. of live variables:
- s0: 2
- s1: 2
- s2: 3
- s3: 3
- s4: 2
- s5: 2
- s6: 1
- s7: 1

**Operation usage**

<table>
<thead>
<tr>
<th>Operation</th>
<th>No. of operations</th>
<th>s1</th>
<th>s2</th>
<th>s3</th>
<th>s4</th>
<th>s5</th>
<th>s6</th>
<th>s7</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>min</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>max</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Max. no. of units:
- abs: 2
- min: 1
- max: 1
- >>: 2
- -: 1
- +: 1

No. of operations:
- abs: 2
- min: 1
- max: 1
- >>: 2
- -: 1
- +: 1
Simple library components

(a) Absolute value unit
   (version 1)

(b) Absolute value unit
   (version 2)

(c) Min unit

(d) Max unit

(e) Min/Max unit

(f) 1-bit right shifter

(g) 3-bit right shifter

(h) 1-bit/3-bit right shifter

(i) Adder

(j) Subtractor

(k) Adder/Subtractor

Copyright © 2004-2005 by Daniel D. Gajski

Slides by Xi Cheng, University of California, Irvine
Connectivity requirements

Block diagram

ASM Chart of Square-root approximation

Connectivity table
Register sharing (Variable merging)

- Grouping of variables with nonoverlapping lifetimes
- Each group shares one register
- Grouping reduces number of registers needed in the design

- Two algorithms:
  - \textit{left-edge}
  - \textit{graph-partitioning}
Left-edge algorithm

1. Start
2. Determine variable lifetimes
3. Sort variables by writing state and life length
4. Allocate a new register
5. Assign to the register all non-overlapping variables starting from the top of the list
6. Remove all assigned variables from the list
7. If the list is empty, go to End; otherwise, go back to Step 3.

End
Register sharing by left-edge algorithm

<table>
<thead>
<tr>
<th>Sorted list of variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a )</td>
</tr>
<tr>
<td>( X )</td>
</tr>
</tbody>
</table>

\[ R_1 = \{ a, t_1, x, t_7 \} \]
\[ R_2 = \{ b, t_2, y, t_4, t_6 \} \]
\[ R_3 = \{ t_2, t_5 \} \]

Register assignments

Datapath schematic

ASM Chart

\[ a = \text{In 1} \]
\[ b = \text{In 2} \]

\[ t_1 = |a| \]
\[ t_2 = |b| \]

\[ x = \max(t_1, t_2) \]
\[ y = \min(t_1, t_2) \]

\[ t_3 = x \gg 3 \]
\[ t_4 = y \gg 1 \]

\[ t_5 = x - t_3 \]

\[ t_6 = t_4 + t_5 \]

\[ t_7 = \max(t_6, x) \]

Done = 1
Out = \( t_7 \)
Merging variables with common sources and destination

\[ x = a + b \]
\[ y = c + d \]

Partial ASM Chart

Datapath without register sharing

Datapath with register sharing
Graph partitioning algorithm

Start

Create compatibility graph

Merge highest priority nodes

Upgrade compatibility graph

All nodes incompatible

no

yes

Stop

(a) Initial compatibility graph
Graph partitioning algorithm for SRA

(a) Initial compatibility graph
(b) Compatibility graph after merging t₃, t₅ and t₆
(c) Compatibility graph after merging t₁, x and t₇
(d) Compatibility graph after merging t₂ and y
(e) Final compatibility graph

ASM Chart

Start

\[ a = \text{In 1} \]
\[ b = \text{In 2} \]

0

1

x = max( t₁, t₂ )
y = min( t₁, t₂ )

\[ t₃ = x >> 3 \]
\[ t₄ = y >> 1 \]

\[ t₅ = x - t₃ \]
\[ t₆ = t₄ + t₅ \]

\[ t₇ = \text{max} ( t₆, x ) \]

Done = 1
Out = t₇

Copyright © 2004-2005 by Daniel D. Gajski

Slides by Xi Cheng, University of California, Irvine
Register assignment generated by the graph-partitioning algorithm

- \( R1 = [a, t1, x, t7] \)
- \( R2 = [b, t2, y, t3, t5, t6] \)
- \( R3 = [t4] \)

Register assignments
Functional unit sharing (operator merging)

- Group non-concurrent operations
- Each group shares one functional unit
- Sharing reduces number of functional units
- Prioritized grouping by reducing connectivity
- Clustering algorithm used for grouping
Functional unit sharing

Partial ASM Chart

Non-shared design

Shared design

\[ x = a + b \]
\[ y = c + d \]
Complex library components

Unit for computing minimum, maximum and absolute value

Unit for computing addition, subtraction, minimum and maximum

Unit for computing addition, subtraction, and absolute value

Unit for computing addition, subtraction, minimum, maximum and absolute value
Operator merging for SRA implementation

(a) Compatibility graph

(b) Cost table

(c) Merging alternative

(d) Cost table

(e) Merging alternative

(f) Cost table

ASM Chart

Copyright © 2004-2005 by Daniel D. Gajski

Slides by Xi Cheng, University of California, Irvine
Datapath connectivity

ASM Chart

(a) Datapath schematic for unit allocation from figure 8.22 (c)

(b) Datapath schematic for unit allocation from figure 8.22 (e)
Priorities in unit merging

(a) Partial ASM Chart

(b) Design without merged units

(c) Design with merged units
Unit merging for SRA datapath

(a) Compatibility graph

(b) Compatibility graph after merging of + and _

(c) Compatibility graph after merging of min, + and _

(d) Final graph partitions

ASM Chart

Start

0

S0

S1

S2

S3

S4

S5

S6

S7

a = In 1
b = In 2

t1 = |a|
t2 = |b|
x = max(t1, t2)
y = min(t1, t2)
t3 = x >> 3
t4 = y >> 1
t5 = x - t3
t6 = t4 + t5
t7 = max(t6, x)
Done = 1
Out = t7
SRA datapath generated by prioritized partitioning

- R₁ = [ a, t₁, x, t₇ ]
- R₂ = [ b, t₂, y, t₃, t₅, t₆ ]
- R₃ = [ t₄ ]

- AU₁ = [ |b| / min / + / - ]
- AU₂ = [ |a| / max / ]
- SH₁ = [ >>1 ]
- SH₂ = [ >>3 ]

(a) Register and functional unit allocation

(b) Datapath schematic
Bus sharing (connection merging)

- Group connections that are not used concurrently
- Each group forms a bus
- Connection merging reduces number of wires
- Clustering algorithm is demonstrated
Connection merging in SRA datapath

- Bus1 = \{ A, C, D, E, H \}
- Bus2 = \{ B, F, G \}
- Bus3 = \{ I, K, M \}
- Bus4 = \{ J, L, N \}

(a) Datapath for SRA

(b) Connectivity usage table

(c) Compatibility graph for input buses

(d) Compatibility graph for output buses

(e) Bus assignment
Connection merging in SRA datapath

Datapath for SRA

Bus assignment

- Bus1 = [A, C, D, E, H]
- Bus2 = [B, F, G]
- Bus3 = [I, K, M]
- Bus4 = [J, L, N]

(f) Bus oriented datapath
Register merging

- Group register with nonoverlapping accesses
- Each group assigned to one register file
- Register grouping reduces number of ports, and therefore number of buses
- Demonstration with clustering algorithm
Register merging

- \( R_1 = \{ a, t_1, x, t_7 \} \)
- \( R_2 = \{ b, t_2, y, t_3, t_5, t_6 \} \)
- \( R_3 = \{ t_4 \} \)

(a) Register assignment

(b) Register access table

(c) Compatibility graph

(d) Datapath schematic
Chaining and multicycling

- Chaining allows serial execution of two or more operations in each state
- Chaining reduces number of states and increases performance
- Multicycling allows one operation to be executed over two or more clock cycles
- Multicycling reduces size of functional units
- Chaining and multicycling are used on noncritical paths to improve resource utilization and performance
SRA datapath with chained units

(a) ASM Chart

(b) Datapath schematic
SRA datapath with multicycle units

(a) ASM Chart

Start = 1

s0

a = In 1
b = In 2

s1

t1 = |a|
t2 = |b|

s2

x = max(t1, t2)
t3 = max(t1, t2)>>3
t4 = min(t1, t2)>>1

s3

t5 = x - t3

s4

t6 = t4 + t5

s5

t7 = max(t6, x)

s6

Done = 1
Out = t7

(b) Datapath schematic

In 1

R1

Bus 1

[abs/max]

Bus 2

R2

[abs/+/-]

Bus 3

R3

min

Bus 4

Out

>>3

>>1

Copyright © 2004-2005 by Daniel D. Gajski

Slides by Xi Cheng, University of California, Irvine
Pipelining

- Pipelining improves performance at a very small additional cost

- Pipelining divides resources into stages and uses all stage concurrently for different data (assembly line principle)

- Pipelining principles works on several levels:
  
  (a) Units pipelining
  
  (b) Control pipelining
  
  (c) Datapath pipelining
Pipelined arithmetic unit

- Latches
- Adder
- Sign bit
- Selectors

Copyright © 2004-2005 by Daniel D. Gajski

Slides by Xi Cheng, University of California, Irvine
(a) ASM Chart

- S0:
  - $a = \text{In 1}$
  - $b = \text{In 2}$

- S1:
  - Start = 1

- S2:
  - $t_1 = |a|$
  - $t_2 = |b|$
  - $x = \max(t_1, t_2)$
  - $t_3 = \max(t_1, t_2) >> 3$
  - $[t_3] = \min(t_1, t_2) >> 1$

- S3:
  - $t_4 = \min(t_1, t_2) >> 1$
  - $x = \max(t_1, t_2)$
  - $t_5 = x - t_3$

- S4:
  - $t_6 = t_4 + t_5$

- S5:
  - $t_7 = \max(t_6, x)$

- S6:
  - Done = 1
  - Out = $t_7$

(b) Datapath schematic

- Bus 1:
  - $R_1$

- Bus 2:
  - $R_2$

- Bus 3:
  - $>>3$

- Bus 4:
  - $R_3$

- AU

- Out
Datapath with pipelined functional unit

(a) Datapath with pipelined AU

(b) Timing diagram
Datapath pipelining

(a) ASM Chart

\[
\begin{align*}
S_0 &\quad a = \text{In 1} \\
S_1 &\quad t_1 = |a| \\
S_2 &\quad t_2 = |b| \\
S_3 &\quad t_4 = \min (t_1, t_2) \gg 1 \\
S_4 &\quad x = \max (t_1, t_2) \\
S_5 &\quad t_5 = x - t_3 \\
S_6 &\quad t_6 = t_4 + t_5 \\
S_7 &\quad t_7 = \max (t_6, x) \\
S_8 &\quad \text{Done} = 1 \\
&\quad \text{Out} = t_7
\end{align*}
\]

(b) Pipelined datapath

\[
R_1 = [a, t_1] \\
R_2 = [b, t_2] \\
R_3 = [t_3, t_5, t_6, t_7] \\
R_4 = [x] \\
R_5 = [t_4] \\
\text{AU1} = [\text{abs/min/max}] \\
\text{AU2} = [+/\text{-}/\text{max}]
\]

(c) Register and functional unit assignment
Datapath pipelining

(b) Pipelined datapath

(d) Timing diagram
# Timing diagram for datapath pipeline with pipelined units

<table>
<thead>
<tr>
<th></th>
<th>$s_0$</th>
<th>$s_1$</th>
<th>$s_2$</th>
<th>$s_3$</th>
<th>$s_4$</th>
<th>$s_5$</th>
<th>$s_6$</th>
<th>$s_7$</th>
<th>$s_8$</th>
<th>$s_9$</th>
<th>$s_{10}$</th>
<th>$s_{11}$</th>
<th>$s_{12}$</th>
<th>$s_{13}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read R1</td>
<td>a</td>
<td>$b$</td>
<td>$t_1$</td>
<td>$t_1$</td>
<td>$t_2$</td>
<td>$t_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read R2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AU1 stage 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AU1 stage 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shifters</td>
<td>a</td>
<td>$t_1$</td>
<td>$t_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write R1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write R2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read R3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read R4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read R5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AU2 stage 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AU2 stage 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write R3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write R4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write R5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Out</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$|a| |b| |minmax| >>1 >>3 | + max | x | + max | x | t_7$

Copyright © 2004-2005 by Daniel D. Gajski

Slides by Xi Cheng, University of California, Irvine
Pipelined FSMD implementation

(a) Standard FSMD implementation

(b) FSMD implementation with control and datapath pipelining
ASM charts for pipelined FSMDs

(b) FSMD implementation with control and datapath pipelining

(a) ASM chart

(b) ASM chart for control pipeline with status register

(c) ASM chart for control pipeline with status register and control registers

(d) ASM chart for control and datapath pipeline

Copyright © 2004-2005 by Daniel D. Gajski

Slides by Xi Cheng, University of California, Irvine
Scheduling

- RT description such as ASM chart specifies data operations in each state
- Flowcharts or programming languages do not have states, but only specify order in which operations are executed.
- Scheduling transforms flowcharts or programs with RT descriptions
- Two types of scheduling
  - (a) resource constrained
    (resource given, minimize time)
  - (b) time constrained
    (time given, minimize resources)
Control/dataflow graph for SRA

(a) Flowchart

1. $a = \text{In 1}$
2. $b = \text{In 2}$
3. $a > b$
4. $t_1 = |a|$ 
5. $t_2 = |b|$
6. $x = \max(t_1, t_2)$
7. $y = \min(t_1, t_2)$
8. $t_3 = x \gg 3$
9. $t_4 = y \gg 1$
10. $t_5 = x - t_3$
11. $t_6 = t_4 + t_5$
12. $t_7 = \max(t_6, x)$
13. $\text{Done} = 1$
14. $\text{Out} = t_7$

(b) Control/Data flow graph

Start
1. $a = \text{In 1}$
2. $b = \text{In 2}$
3. $a > b$
4. $t_1 = |a|$ 
5. $t_2 = |b|$
6. $x = \max(t_1, t_2)$
7. $y = \min(t_1, t_2)$
8. $t_3 = x \gg 3$
9. $t_4 = y \gg 1$
10. $t_5 = x - t_3$
11. $t_6 = t_4 + t_5$
12. $t_7 = \max(t_6, x)$
13. $\text{Done} = 1$
14. $\text{Out} = t_7$
Basic schedules

(a) ASAP schedule

(a) ALAP schedule
List scheduling algorithm

1. Perform ASAP
2. Perform ALAP
3. Determine mobilities
4. Create ready list
5. Sort ready list by mobilities
6. Schedule ops from ready list
7. Delete scheduled ops from ready list
8. Add new ops to ready list
9. Increment state index
10. All ops scheduled?

Flowchart continues with decision nodes for 'no' and 'yes'.

Copyright © 2004-2005 by Daniel D. Gajski

Slides by Xi Cheng, University of California, Irvine
Resource-constrained scheduling

1. Perform ASAP
2. Perform ALAP
3. Determine mobilities
4. Create ready list
5. Sort ready list by mobilities
6. Schedule ops from ready list
7. Delete scheduled ops from ready list
8. Add new ops to ready list
9. Increment state index

(a) ASAP
(b) ALAP
(c) Ready list with mobilities
(d) RC schedule
Time-constrained scheduling

- Perform ASAP
- Perform ALAP
- Determine mobilities ranges
- Create probability distribution graphs

All ops scheduled?

Schedule ops from ready list

All ops scheduled?

yes

yes

no

All ops scheduled?
TC schedule for SRA algorithm

(a) ASAP  (b) ALAP  (c) TC schedule
Probability distribution graph before, during and after TC scheduling

(a) Initial probability distribution graph

(b) Distribution graph after max, + and – were scheduled

(c) Distribution graph after max, + and –, >>3 and >>1 were scheduled

(c) Distribution graph for final scheduled
Chapter summary

We introduced RT design:
- FSMD model
- RT specification with
  - Static-action tables
  - ASM charts
- Procedure for synthesis from RT specification
- Design Optimization through
  - Register sharing
  - Unit chaining
  - Functional unit sharing
  - Multiclocking
  - Bus sharing
- Design Pipelining
  - Unit pipelining
  - Control pipelining
  - Datapath pipelining
- Scheduling of flowcharts