System Designs

• Introduction
  – Methodology and Framework
• Components
• Specification
• Implementation
Introduction

• Methodology
  • Approach with success stories.
  • Hierarchical designs with interface between the modules (BSV).
• Data Subsystem and Control Subsystem
  • For n-bit data, each operation takes n times or more in hardware complexity.
  • Data subsystem carries out the data operations and transports.
  • Control system sequences the data subsystem and itself.
I. Introduction: Framework

Data Subsystem

Control Subsystem

Data Inputs

Control Inputs

Conditions

Control Signals

Start/Request

Done/Acknowledgement

n=128

n=64

Data Outputs

Control Outputs
I. Introduction: Handshaking

Master that calls module S

Data Inputs

Control Inputs

Conditions

Control Signals

Start/Request

n=128

Data Subsystem

n=64

Control Subsystem

Done/Acknowledgement

Data Outputs

Control Outputs

Inputs

Outputs

Start/Request

module S

Z

X

Y

Z

X

Y

start

done

start

done
Handshaking
Handshaking: iClicker

The master module that calls module S controls the following signals

A. Signal start
B. Signal done
C. Signals start and done
D. None of the above
## II. Components

<table>
<thead>
<tr>
<th>Components</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Subsystem</td>
<td></td>
</tr>
<tr>
<td>Storage Modules</td>
<td>Data storage</td>
</tr>
<tr>
<td>Operators</td>
<td>Data operations</td>
</tr>
<tr>
<td>Interconnections</td>
<td>Data transport</td>
</tr>
<tr>
<td>Control Subsystem</td>
<td></td>
</tr>
<tr>
<td>Sequential machines</td>
<td>Control of data operations</td>
</tr>
<tr>
<td></td>
<td>Control of data transports</td>
</tr>
<tr>
<td></td>
<td>Control of the sequential system</td>
</tr>
</tbody>
</table>
II. Data Subsystem Components

- Storage: Register, RAM, FIFO, LIFO, Counter, Shifter
- Operator: ALU, Floating Point Operators
- Interconnect: Wire, Buses, Crossbars
II. Components: Storage Modules, Register

LD: Load
CLR: Clear

\[
Q(t+1) = \begin{cases} 
(0, 0, \ldots, 0) & \text{if CLR} = 1 \\
D & \text{if LD} = 1 \text{ and CLR} = 0 \\
Q(t) & \text{if LD} = 0 \text{ and CLR} = 0 
\end{cases}
\]
Modulo-n Counter

\[ Q(t+1) = \begin{cases} 
(0, 0, \ldots, 0) & \text{if CLR = 1} \\
D & \text{if LD = 1 and CLR = 0} \\
(Q(t)+1) \mod n & \text{if LD = 0, CNT = 1 and CLR = 0} \\
Q(t) & \text{if LD = 0, CNT = 0 and CLR = 0} 
\end{cases} \]

\[ \text{TC} = \begin{cases} 
1 & \text{if } Q(t) = n-1 \text{ and CNT = 1} \\
0 & \text{otherwise} 
\end{cases} \]
Storage Component: Registers, Array of Registers

Registers: If $C$ then $R \leftarrow D$

Register Array: If $C$ then $R_{\text{address}} \leftarrow D$
Sharing connections and controls
Storage Components: RAM, FIFO, LIFO

RAM

Decoder

Address

RAM

Size of RAM larger than registers
Performance is slower

FIFO (First in first out)

LIFO (Last in first out: Stack)
CASE Op-Sel Is
  When F1, Z <= A op1 B
  When F2, Z <= A op2 B
  .
  .
End CASE

Example:
CASE Op-Set Is
  Z <= (A + B)mod 2^n if Op-Sel=addition,
  Z <= (A - B)mod 2^n if Op-Sel=subtraction
End CASE
Interconnect Modules (Wires and Switches)

- Single Lines
- Band of Wires
- Shared Buses
- Crossbar

1. Single line (shifting, time sharing)
2. Band of Wires (BUS)

3. Shared Bus

Switches
4. Crossbar (Multiple buses running horizontally) in simultaneous transfers are possible, but more expensive.
III. Specification: Program

1. Objects (Registers, Outputs of combinational logic)
2. Operation (Logic, Add, Multiplication, DSP, and etc.)
3. Assignment
4. Sequencing

Example:

Signal S1, S2, R[15:0]: FFs, Registers, wires
Z \( \leftarrow \) A + B: Registers, Adder, Interconnect
R1 \( \leftarrow \) R2: Registers and Interconnect
Begin, End: Control
if ( ) then ( ), ENDIF: Control
Ex. If C then $R_1 \leftarrow S_1$
Else $R_2 \leftarrow S_2$
Endif;

If $C_1$ then $X \leftarrow A$
Else $X \leftarrow B + C$
Endif
If $C_2$ then $R_g \leftarrow X$
Endif
VI. Implementation

- Example
- Handshaking
  - Request and Acknowledgement
- Datapath Subsystem
  - Data Operators
  - Data Transporters
- Control Subsystem
  - One Hot Machine Design
VI. Implementation: Example

AddModule(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] type bit-vector;
    Output Z[15:0] type bit-vector,
    done type Boolean;
S0: If start’ goto S0 || done ← 1;
S1: A ← X || B ← Y || done ← 0;
S2: Z ← Add(A, B) || goto S0;
}
Exercise: Go through the handshaking, data subsystem and control subsystem designs.
AddModule(X, Y, start, done)
Handshaking

start

done
Suppose that each step (Si) takes one clock cycle. How many clock cycles does the AddModule take to complete one handshaking iteration?
A. One cycle
B. Two cycles
C. More than two cycles