System Designs

• Introduction
• Components
• Spec
• Implementation
Digital Designs vs Computer Architectures

• Instruction Set (H.Chapter 6, CSE141)
  – Bottleneck: Silicon Area, Power
• Data Path (H.Chapter 7.1-7.3)
• Control Subsystem (H.Chapter 7.1-7.3)
• Memory Management (Chapter 8, CSE141)
  – Bottleneck: IO, Memory Latency
Introduction

- Methodology
  - Approach with success stories
  - Hierarchical designs with interface between the levels
- Data Subsystem and Control Subsystem
  - For n-bit data, each operation takes n or more in complexity
  - Data subsystem carries out the data operations and transports
  - Control system sequences the data subsystem and itself.
I. Introduction

Data Subsystem

Control Subsystem

Conditions

Control Signals

Data Inputs

n=64

Data Outputs

Control Inputs

Start/Request

n=64

Control Outputs

Done/Acknowledgement
## Introduction

<table>
<thead>
<tr>
<th>Components</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Control Subsystem</th>
<th>Sequential machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control of data operations</td>
<td></td>
</tr>
<tr>
<td>Control of data transports</td>
<td></td>
</tr>
<tr>
<td>Control of the sequential system</td>
<td></td>
</tr>
</tbody>
</table>
Data Subsystem Components

- Storage
- Operator
- Interconnect
Components: Storage Modules, Register

LD: Load
CLR: Clear

\[
Q(t+1) = \begin{cases} 
(0, 0, \ldots, 0) & \text{if CLR} = 1 \\
D & \text{if LD} = 1 \text{ and CLR} = 0 \\
Q(t) & \text{if LD} = 0 \text{ and CLR} = 0 
\end{cases}
\]
Storage Component: Registers, Array of Registers

Registers: If C then R $\leftarrow$ D

Register Array: If C then R $\leftarrow D_{\text{address}}$

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)
Functional Modules

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
CASE F Is
When F=0, Z <= A
When F=1, Z <= B
End CASE

iClicker:
The above function can be implemented by
A. A vector of decoders
B. A vector of multiplexers
C. A RAM
D. None of the above
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) m simultaneous transfers are possible, but more expensive.
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  
Z \leftarrow A + B: Registers, Adder, Interconnect  
R1 \leftarrow R2: Registers and Interconnect  
Begin, End: Control  
if ( ) then ( ), ENDIF: Control
Ex. If $C$ then $R1 \leftarrow S1$
Else $R2 \leftarrow S2$
Endif;

If $C1$ then $X \leftarrow A$
Else $X \leftarrow B + C$
Endif

If $C2$ then $G \leftarrow X$
Endif
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.