Sequential Logic and HCL

CMPU 224 – Computer Organization
Jason Waterman
Sequential Logic Intro

• Last time: **Combinational Circuits**
  • Acyclic Network of Logic Gates
  • Continuously responds to changes on primary inputs
  • Primary outputs become (after some delay) Boolean functions of primary inputs
Sequential Logic

• Sequential logic circuits are those whose outputs are also dependent upon past inputs
• In other words, the output of a sequential circuit may depend upon its previous outputs
• In effect, it has some form of "memory"
Storing 1 Bit

Vin → V1

Vin → V2

V1

V2

Vin

V1

V2

V1

V2

Vin

V1

V2

Vin

V1

V2

Vin

V1

V2

Vin

V1

V2

Vin
Storing 1 Bit

Vin
V1
V2

Stable 0
Metastable
Stable 1

V2
Vin

0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

Vin

0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

V2

0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Physical Analogy

- Stable 0
- Stable 1
- Metastable
- Stable left
- Stable right
Storing and Accessing 1 Bit

Bistable Element

\[ q = 0 \text{ or } 1 \]

R-S Latch

Reset-Set Latch

Storing

Resetting

Setting
1-Bit Latch

D Latch

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>Reset to 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Set to 1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Storing</td>
</tr>
</tbody>
</table>

Storing

Latching
Clocking

D Latch

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>Reset to 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Set to 1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Storing</td>
</tr>
</tbody>
</table>

11/9/2022
Transparent 1-Bit Latch

- When in latching mode, combinational propagation from D to Q+ and Q–
- Value latched depends on value of D as C falls
Edge-Triggered Latch

- Only in latching mode for brief period
  - Rising clock edge
- Value latched depends on the data as the clock rises
- Output remains stable at all other times

CLOCK LOW
STORAGE MODE
INPUT D IS IGNORED

Data

Clock

Trigger

Q+
Q−
Edge-Triggered Latch

- Only in latching mode for brief period
- Rising clock edge
- Value latched depends on the data as the clock rises
- Output remains stable at all other times
Edge-Triggered Latch

- Only in latching mode for brief period
  - Rising clock edge
- Value latched depends on the data as the clock rises
- Output remains stable at all other times
Registers

- Stores a word of data
- Collection of edge-triggered latches
- Loads input on rising edge of clock
Register Operation

- Stores data bits
- For most of the time acts as barrier between input and output
- As clock rises, loads input
Random-Access Memory

- Stores multiple words of memory
  - Address input specifies which word to read or write
- Register file
  - Holds values of program registers
  - $%rax$, $%rsp$, etc.
  - Register identifier serves as index into the register file
- Multiple Ports
  - Can read and/or write multiple words in one cycle
    - Each has separate address and data input/output
Register File Timing

• Reading
  • Like combinational logic
  • Output data generated based on input address
    • After some delay

• Writing
  • Like register
  • Update only as clock rises
Hardware Control Language (HCL)

- Very simple hardware description language
  - Can only express limited aspects of hardware operation
    - Parts we want to explore and modify
  - Boolean operations have syntax similar to C logical operations
  - We will use it to describe control logic for processors

- Data Types
  - bool: Boolean
    - a, b, c, ...
  - int: words
    - A, B, C, ...
    - Does not specify word size---bytes, 64-bit words, ...

- Statements
  - bool a = bool-expr ;
  - int A = int-expr ;
HCL Operations

• Classify by type of value returned

• Boolean Expressions
  • Logic Operations
    • \( a \land b, a \lor b, \neg a \)
  • Word Comparisons
    • \( A = B, A \neq B, A < B, A \leq B, A \geq B, A > B \)
  • Set Membership
    • \( A \in \{ B, C, D \} \)
      • Same as \( A = B \lor A = C \lor A = D \)

• Word Expressions
  • Case expressions
    • \( [ a : A; b : B; c : C ] \)
    • Evaluate test expressions \( a, b, c, \ldots \) in sequence
    • Return word expression \( A, B, C, \ldots \) for first successful test
Bit Equality

HCL Expression

```c
bool eq = (a&&b) || (!a&&!b)
```
Word Equality

Word-Level Representation

\[
\text{bool Eq} = (A == B)
\]

HCL Representation
Bit-Level Multiplexor

- Control signal s
- Data signals a and b
- Output a when s=1, b when s=0

HCL Expression

```cpp
bool out = (s & a) || (!s & b)
```

Diagram of Bit MUX with inputs s, a, b and output out.
Word Multiplexor

- Select input word A or B depending on control signal s
- HCL representation
  - Case expression
  - Series of test : value pairs
  - Output value for first successful test

Word-Level Representation

![Multiplexor Diagram]

HCL Representation

```c
int Out = [
  s : A;
  1 : B;
];
```
HCL Word-Level Examples

- Find minimum of three input words
- HCL case expression
- Final case guarantees match

- Select one of 4 inputs based on two control bits
- HCL case expression
- Simplify tests by assuming sequential matching

Minimum of 3 Words

\[
\text{int Min3} = \begin{cases} 
A < B \land A < C & : A; \\
B < C & : B; \\
1 & : C; 
\end{cases}
\]

4-Way Multiplexor

\[
\text{int Out4} = \begin{cases} 
\neg s1 \land \neg s0 & : D0; \\
\neg s1 & : D1; \\
\neg s0 & : D2; \\
1 & : D3; 
\end{cases}
\]
Summary

• **Computation**
  - Performed by combinational logic
  - Computes Boolean functions
  - Continuously reacts to input changes

• **Storage**
  - Registers
    - Hold single words
    - Loaded as clock rises
  - Random-access memories
    - Hold multiple words
    - Possible multiple read or write ports
    - Read word when address input changes
    - Write word as clock rises

• **Hardware Control Language (HCL)**
  - Simple hardware description language to describe the control logic for a processor