Sequential Logic and HCL

CMPU 224 – Computer Organization
Jason Waterman
Sequential Logic Intro

• Recall: 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

\[ \text{Vin} \rightarrow V_1 \rightarrow V_2 \]

![Graph showing V1 and V2 as functions of Vin](image-url)
Storing 1 Bit

V\textsubscript{in} \rightarrow V\textsubscript{1} \rightarrow V\textsubscript{2}

Stable 0

Stable 1

Metastable

V\textsubscript{in} \rightarrow V\textsubscript{2} \rightarrow V\textsubscript{in}

0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

Vin

V\textsubscript{in}

V\textsubscript{2}

0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

Stable 1
Physical Analogy

Stable 0

Stable 1

Stable left

Stable right

Vin

Metastable
Storing and Accessing 1 Bit

Bistable Element

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

Resetting

Setting

Storing

Reset-Set Latch

R-S Latch
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>

Latching

Storing
Clocking

D Latch

RISING EDGE OF CLOCK

Data

Clock

D

R

S

Q+

Q–

Period

R | S | State
---|---|---
1 | 0 | Reset to 0
0 | 1 | Set to 1
0 | 0 | Storing
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

Latching

Changing D

10/25/2018

CMPU 224 -- Computer Organization
Edge-Triggered Latch

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

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

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

![Diagram of register operation](image)

State = x
Input = y
Output = x
Rising clock

State = y
Output = y

10/25/2018
CMPU 224 -- Computer Organization
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
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’ll 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
    • $a \land b$, $a \lor b$, $!a$
  • Word Comparisons
    • $A == B$, $A != B$, $A < B$, $A <= B$, $A >= B$, $A > B$
  • Set Membership
    • $A$ in { $B$, $C$, $D$ }
      • Same as $A == B$ || $A == C$ || $A == D$

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

HCL Expression

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

Word-Level Representation

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

HCL Representation

\[ \begin{align*}
A & \rightarrow \text{Eq} \\
B & \rightarrow \text{Eq} \\
\text{Eq} & \rightarrow \text{Eq}
\end{align*} \]
Bit-Level Multiplexor

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

HCL Expression

$$ \text{bool } out = (s \&\& a) || (!s \&\& b) $$
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

<table>
<thead>
<tr>
<th>s</th>
<th>A</th>
<th>B</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>31</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>32</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>33</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>34</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>35</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>36</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>37</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>38</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>39</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>41</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>42</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>43</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>44</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>45</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>46</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>47</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>48</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>49</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>51</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>52</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>53</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>54</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>55</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>56</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>57</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>58</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>59</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>60</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>61</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>62</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>63</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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

```c
int Min3 = [
    A < B && A < C : A;
    B < C : B;
    1 : C;
];
```

4-Way Multiplexor

```c
int Out4 = [
    !s1&&!s0: D0;
    !s1 : D1;
    !s0 : D2;
    1 : D3;
];
```
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