Overview of Logic Design

- Fundamental Hardware Requirements
  - Communication
    - How to get values from one place to another
  - Computation
  - Storage
- Bits are Our Friends
  - Everything expressed in terms of values 0 and 1
  - Communication
    - Low or high voltage on wire
  - Computation
    - Compute Boolean functions
  - Storage
    - Store bits of information

Digital Signals

- Use voltage thresholds to extract discrete values from continuous signal
- Simplest version: 1-bit signal
  - Either high range (1) or low range (0)
  - With guard range between them
- Not strongly affected by noise or low quality circuit elements
  - Can make circuits simple, small, and fast

Computing with Logic Gates

- Outputs are Boolean functions of inputs
- Respond continuously to changes in inputs
  - With some, small delay

Combination Circuits

- Acyclic Network of Logic Gates
  - Continuously responds to changes on primary inputs
  - Primary outputs become (after some delay) Boolean functions of primary inputs
  - Outputs of two or more logic gates cannot be connected together

Bit Equality

- Generate 1 if a and b are equal
- Hardware Control Language (HCL)
  - A very simple hardware description language (HDL) used in this textbook
  - Boolean operations have syntax similar to C logical operations
  - We’ll use it to describe control logic for processors
Word Equality

- 32-bit word size
- Equality operation
  - Generates Boolean value
  - “=” for assignment, “==” for equality
- Thin lines and dashed lines

Word-Level Representation

Bit-Level Multiplexor

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

HCL Representation

bool out = (a == b)

HCL Expression

4-Word Multiplexor

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

HCL Word-Level Examples

Minimum of 3 Words

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

4-Way Multiplexor

- Find minimum of three input words
- All test cases are nonexclusive (unlike switch statement in C
- First matching case selected

HCL case expression
- Select one of 4 inputs based on two control bits

Set Membership

- HCL allows membership in a set
- Example
  - bool s1 = code == 2 || code == 3;
  - bool s2 = code == 1 || code == 3;

It is equivalent to
  - bool s1 = code in {2, 3};
  - bool s2 = code in {1, 3};

- In general form,
  - iexp in {iexp1, iexp2, iexp3...iexpk}

Arithmetic Logic Unit

- Combinational logic
  - Continuously responding to inputs
- Control signal selects function computed
  - Corresponding to 4 arithmetic/logical operations in Y86: add, sub, and, or
- Also computes values for condition codes
Storing and Accessing 1 Bit

Bistable Element

\[ Q^+ \quad Q^- \]

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

Resetting

\[ R = 1 \] and \[ S = 1 \] are not allowed, i.e. output non-deterministic

Setting

Hold

1-Bit D-Latch with Clock

1-Bit D-Latch

\[ D \quad Data \]

\[ C \quad Clock \]

Latching

\[ Latching \]

Holding

\[ Holding \]

Latching

1-Bit D-Latch

\[ D \quad Data \]

\[ C \quad Clock \]

Output remains stable at all other times

Changing D

• 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

\[ D \quad Data \]

\[ C \quad Clock \]

\[ C' \quad Trigger \]

Only in latching mode for a brief period
• Rising clock edge
• Value latched depends on data as clock rises

Register Operation

\[ State = x \]

\[ Input = y \rightarrow Output = x \]

Rising clock

\[ State = y \]

\[ Input = y \rightarrow Output = y \]

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

Registers

\[ I \quad O \]

\[ I \quad O \]

\[ I \quad O \]

\[ I \quad O \]

\[ I \quad O \]

\[ I \quad O \]

\[ I \quad O \]

\[ I \quad O \]

Structure

• Stores word of data
• Different from program registers seen in assembly code
• Collection of edge-triggered latches
• Loads input on rising edge of clock
State Machine Example

- Accumulator circuit
- Load or accumulate on each cycle

Register File (Random-Access Memory)

- Stores multiple words of memory
- Address input specifies which word to read or write
- Register file
  - Holds values of program registers
  - %eax, %esp, etc.
  - Register identifier serves as address
  - (ID 0xF) implies no read or write performed
- 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

- Very simple hardware description language
- Can only express limited aspects of hardware operation
- Parts we want to explore and modify
- Data Types
  - bool: Boolean
    - A, B, C, ...
  - int: words
    - A, B, C, ...
    - Does not specify word size—bytes, 32-bit words, ...
- Statements
  - bool a = bool-expr ;
  - int A = int-expr ;

HCL Operations

- Classify by type of value returned
- Boolean Expressions
  - Logic Operations
    - a && b, a || b, !a
  - Word Comparisons
  - 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

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