

Logic Design

These slides derived from some provided by the authors of our textbook.

Also see the slides titled "Flops" for more information about sequential circuits and the construction of a circuit that is a 1 bit memory

Overview of Logic Design

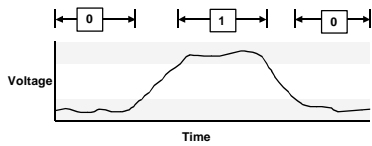
Fundamental Hardware Requirements

- Communication
 - How to get values from one place to another
- Computation
- Storage

Bits are Our Friends

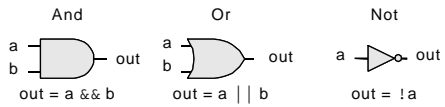
- Everything expressed in terms of values 0 and 1
- Communication
 - Low or high voltage on wire
- Computation
 - Compute Boolean functions
- Storage
 - Store bits of information

Digital Signals

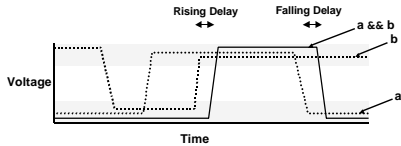


- Use voltage thresholds to extract discrete values from continuous signal
- Simplest version: 1-bit signal
 - Either high range (1) or low range (0)
 - With guard range between them
- Not strongly affected by noise or low quality circuit elements
 - Can make circuits simple, small, and fast

Computing with Logic Gates



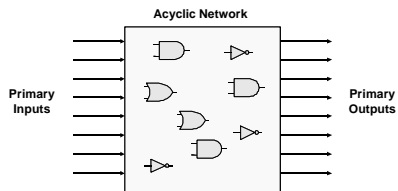
- Outputs are Boolean functions of inputs
- Respond continuously to changes in inputs
 - With some, small delay



CompOrg Fall 2002 - Logic Design

4

Combinational Circuits



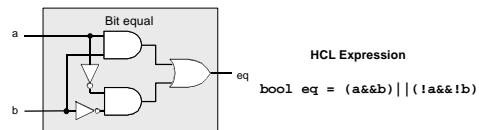
Acyclic Network of Logic Gates

- Continuously responds to changes on primary inputs
- Primary outputs become (after some delay) Boolean functions of primary inputs

CompOrg Fall 2002 - Logic Design

5

Bit Equality



- Generate 1 if a and b are equal

Hardware Control Language (HCL)

- Very simple hardware description language
 - Boolean operations have syntax similar to C logical operations
- We'll use it to describe control logic for processors

CompOrg Fall 2002 - Logic Design

6

Word Equality

Word-Level Representation

B
A

Eq

HCL Representation

```
bool Eq = (A == B)
```

- 32-bit word size
- HCL representation
 - Equality operation
 - Generates Boolean value

CompOrg Fall 2002 - Logic Design 7

Bit-Level Multiplexor

HCL Expression

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

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

CompOrg Fall 2002 - Logic Design 8

Word Multiplexor

Word-Level Representation

S
B
A

MUX

Out

HCL Representation

```
int Out = [
  s : A;
  1 : B;
];
```

- 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

CompOrg Fall 2002 - Logic Design 9

HCL Word-Level Examples

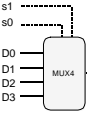
Minimum of 3 Words



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

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

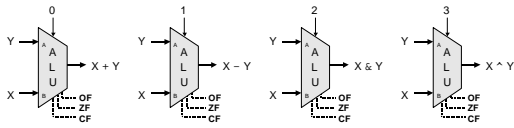
4-Way Multiplexor



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

- Selected data based on control based on two control bits
- HCL case expression
- Simplify tests by assuming sequential matching

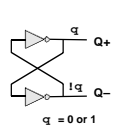
Arithmetic Logic Unit



- Combinational logic
 - Continuously responding to inputs
- Control signal selects function computed
 - Corresponding to 4 arithmetic/logical operations in Y86
- Also computes values for condition codes

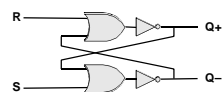
Storing and Accessing 1 Bit

Bistable Element

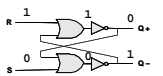


q = 0 or 1

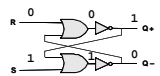
R-S Latch



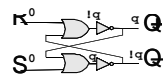
Resetting



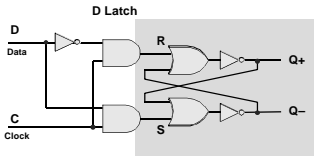
Setting



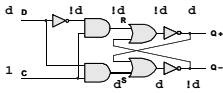
Storing



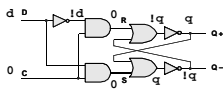
1-Bit Latch



Latching



Storing

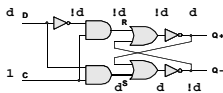


CompOrg Fall 2002 - Logic Design

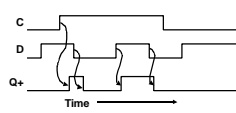
13

Transparent 1-Bit Latch

Latching



Changing D

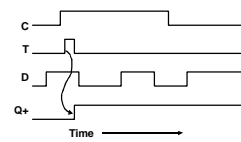
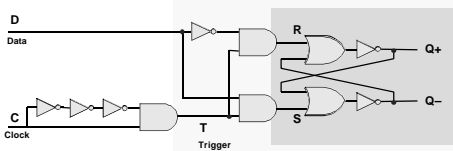


- When in latching mode, combinational propagation from D to Q+ and Q-
- Value latched depends on value of D as C falls

CompOrg Fall 2002 - Logic Design

14

Edge-Triggered Latch

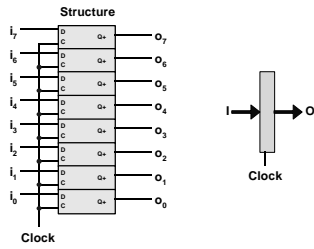


- Only in latching mode for brief period
 - Rising clock edge
- Value latched depends on data as clock rises
- Output remains stable at all other times

CompOrg Fall 2002 - Logic Design

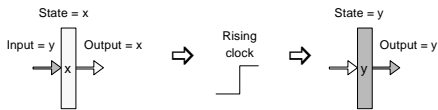
15

Registers



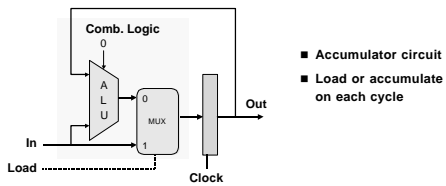
- Stores word of data
 - Different from *program registers* seen in assembly code
- Collection of edge-triggered latches
- Loads input on rising edge of clock

Register Operation

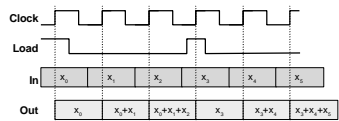


- Stores data bits
- For most of time acts as barrier between input and output
- As clock rises, loads input

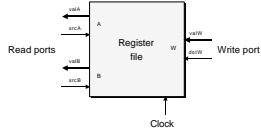
State Machine Example



- Accumulator circuit
- Load or accumulate on each cycle



Random-Access Memory

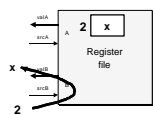


- Stores multiple words of memory
 - Address input specifies which word to read or write
- Register file
 - Holds values of program registers
 - %eax, %esp, etc.
 - Register identifier serves as address
 - ID 8 implies no read or write performed
- Multiple Ports
 - Can read and/or write multiple words in one cycle
 - Each has separate address and data input/output

CompOrg Fall 2002 - Logic Design

19

Register File Timing

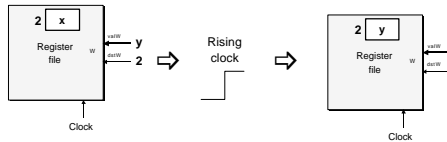


Reading

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

Writing

- Like register
- Update only as clock rises



CompOrg Fall 2002 - Logic Design

20

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 ;

CompOrg Fall 2002 - Logic Design

21

HCL Operations

- Classify by type of value returned

Boolean Expressions

- Logic Operations
 - $a \ \&\& \ b, a \ || \ b, !a$
- Word Comparisons
 - $A == B, A != B, A < B, A <= B, A >= B, A > B$
- Set Membership
 - $A \ \text{in} \ \{ B, C, D \}$
 - » Same as $A == B \ || \ A == C \ || \ A == D$

Word Expressions

- Case expressions
 - $[a : A; b : B; c : C]$
 - Evaluate test expressions a, b, c, \dots in sequence
 - Return word expression A, B, C, \dots 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
