Chapter # 1: Introduction

Contemporary Logic Design

Randy H. Katz
University of California, Berkeley

May 1993
The Process of Design

Bottom Up Assembly

Primitives composed to build more and more complex assemblies

e.g., a group of rooms form a floor

e.g., a group of floors form a bldg.

a group of transistors form a gate

a group of gates form an addition circuit

addition circuits plus storage circuits form a processor datapath
Digital Hardware Systems

Digital Systems

Digital vs. Analog Waveforms

Digital: only assumes discrete values

Analog: values vary over a broad range continuously
Digital Hardware Systems

Advantages of Digital Systems

Analog systems: slight error in input yields large error in output

Digital systems more accurate and reliable
  Readily available as self-contained, easy to cascade building blocks

Computers use digital circuits internally

Interface circuits (i.e., sensors & actuators) often analog

This course is about logic design, not system design (processor architecture), not circuit design (transistor level)
Digital Hardware Systems

Digital Binary Systems

- **Two discrete values:**
  yes, on, 5 volts, current flowing, magnetized North, "1"
  no, off, 0 volts, no current flowing, magnetized South, "0"

- **Advantage of binary systems:**
  rigorous mathematical foundation based on logic

  IF the garage door is open
  AND the car is running
  THEN the car can be backed out of the garage

  IF N-S is green
  AND E-W is red
  AND 45 seconds has expired since the last light change
  THEN we can advance to the next light configuration

  the three preconditions must be true to imply the conclusion
Digital Hardware Systems

Boolean Algebra and Logical Operators

*Algebra*: variables, values, operations

In Boolean algebra, the values are the symbols 0 and 1
If a logic statement is false, it has value 0
If a logic statement is true, it has value 1

*Operations*: AND, OR, NOT

\[
\begin{array}{c|c|c|c|c|c}
X & Y & X \text{ AND } Y & X & Y & X \text{ OR } Y \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c|c}
X & \text{ NOT X} & X & \text{ NOT X} \\
0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 \\
\end{array}
\]
**Digital Hardware Systems**

**Hardware Systems and Logical Operators**

IF the garage door is open AND the car is running THEN the car can be backed out of the garage

<table>
<thead>
<tr>
<th>door open?</th>
<th>car running?</th>
<th>back out car?</th>
</tr>
</thead>
<tbody>
<tr>
<td>false/0</td>
<td>false/0</td>
<td>false/0</td>
</tr>
<tr>
<td>false/0</td>
<td>true/1</td>
<td>false/0</td>
</tr>
<tr>
<td>true/1</td>
<td>false/0</td>
<td>false/0</td>
</tr>
<tr>
<td>true/1</td>
<td>true/1</td>
<td>TRUE/1</td>
</tr>
</tbody>
</table>
Digital Hardware Systems

The Real World

Physical electronic components are continuous, not discrete!

These are the building blocks of all digital components!

![Graph showing transition from logic 1 to logic 0](image)

Transition from logic 1 to logic 0 does not take place instantaneously in real digital systems.

Intermediate values may be visible for an instant.

Boolean algebra useful for describing the steady state behavior of digital systems.

Be aware of the dynamic, time varying behavior too!
Digital Hardware Systems

Digital Circuit Technologies

*Integrated circuit technology*
choice of conducting, non-conducting, sometimes conducting ("semiconductor") materials

whether or not their interaction allows electrons to flow forms the basis for electrically controlled switches

*Main technologies*

MOS: Metal-Oxide-Silicon

Bipolar
  - Transistor-Transistor Logic
  - Emitter Coupled Logic
MOS Technology

Transistor
basic electrical switch

three terminal switch: gate, source, drain

voltage between gate and source exceeds threshold
switch is conducting or "closed"
electrons flow between source and drain

when voltage is removed,
the switch is "open" or non-conducting
connection between source and drain is broken
Circuit that implements logical negation (NOT)

1 at input yields 0 at output
0 at input yields 1 at output

Inverter behavior as a function of input voltage
input ramps from 0V to 5V
output holds at 5V for some range of small input voltages
then changes rapidly, but not instantaneously!

Remember distinction between steady state and dynamic behavior
Combinational vs. Sequential Logic

Network implemented from switching elements or logic gates. The presence of feedback distinguishes between sequential and combinational networks.

**Combinational logic**
no feedback among inputs and outputs
outputs are a pure function of the inputs
e.g., full adder circuit:
(A, B, Carry In) mapped into (Sum, Carry Out)

```
X_1 \rightarrow Z_1
X_2 \rightarrow Z_2
\vdots
X_n \rightarrow Z_m
```

```
A \rightarrow \text{Sum}
B \rightarrow \text{Sum}
Cin \rightarrow \text{Cout}
```

Full Adder
Digital Hardware Systems

Sequential logic

inputs and outputs overlap
outputs depend on inputs and the entire history of execution!

network typically has only a limited number of unique configurations
these are called states
e.g., traffic light controller sequences infinitely through four states

new component in sequential logic networks:
storage elements to remember the current state

output and new state is a function of the inputs and the old state
i.e., the fed back inputs are the state!

Synchronous systems

period reference signal, the clock, causes the storage elements to accept new values and to change state

Asynchronous systems

no single indication of when to change state
**Digital Hardware Systems**

**Combinational vs Sequential Logic**

**Traffic Light Example**

- **Next State Logic**: Maps current state and alarm events into the next state.

- **Current State**: Storage elements replaced by next state when the clock signal arrives.

- **Output Logic**: Current state mapped into control signals to change the lights and to start the event timers.

**IF**

controller in state N-S green, E-W red

AND the 45 second timer alarm is asserted

**THEN**

the next state becomes N-S yellow, E-W red when the clk signal is next asserted.

**New Traffic Light Controller Configuration**
Representations of a Digital Design

Switches

A switch connects two points under control signal.

Normally Open
- when the control signal is 0 (false), the switch is open
- when it is 1 (true), the switch is closed

Normally Closed
- when control is 1 (true), switch is open
- when control is 0 (false), switch is closed

![Switch Diagrams]
Representations of a Digital Design: Switches

Examples: routing inputs to outputs through a maze

EXAMPLE:
IF car in garage
AND garage door open
AND car running
THEN can back out car

EXAMPLE:
IF car in driveway
OR (car in garage
AND NOT garage door closed)
AND car running
THEN can back out car

Floating nodes:
what happens if the car is not running?
outputs are floating rather than forced to be false

Under all possible control signal settings
(1) all outputs must be connected to some input through a path
(2) no output is connected to more than one input through any path
Representations of a Digital Design: Switches

Implementation of AND and OR Functions with Switches

**AND function**
Series connection to TRUE

**OR function**
Parallel connection to TRUE
Representations of a Digital Design

Truth Tables

tabulate all possible input combinations and their associated output values

Example: half adder adds two binary digits to form Sum and Carry

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Sum</th>
<th>Carry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Example: full adder adds two binary digits and Carry in to form Sum and Carry Out

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Cin</th>
<th>Sum</th>
<th>Cout</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

NOTE: 1 plus 1 is 0 with a carry of 1 in binary
Representations of a Digital Design

Boolean Algebra

values: 0, 1
variables: A, B, C, . . ., X, Y, Z
operations: NOT, AND, OR, . . .

NOT X is written as $\overline{X}$
X AND Y is written as $X \& Y$, or sometimes $X \cdot Y$
X OR Y is written as $X + Y$

Deriving Boolean equations from truth tables:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Sum</th>
<th>Carry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Sum = $\overline{A} \cdot B + A \cdot \overline{B}$

OR'd together product terms for each truth table row where the function is 1

if input variable is 0, it appears in complemented form;
if 1, it appears uncomplemented

Carry = $A \cdot B$
**Representations of a Digital Design: Boolean Algebra**

*Another example:*

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Cin</th>
<th>Sum</th>
<th>Cout</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
\text{Sum} = \overline{A} \overline{B} \overline{Cin} + \overline{A} B \overline{Cin} + A \overline{B} \overline{Cin} + A B \overline{Cin}
\]

\[
\text{Cout} = \overline{A} B \overline{Cin} + A B \overline{Cin} + A \overline{B} \overline{Cin} + A B \overline{Cin}
\]
Representations of a Digital Design: Boolean Algebra

Reducing the complexity of Boolean equations

Laws of Boolean algebra can be applied to full adder's carry out function to derive the following simplified expression:

\[ \text{Cout} = A \text{ Cin} + B \text{ Cin} + A B \]

Verify equivalence with the original Carry Out truth table:

place a 1 in each truth table row where the product term is true

each product term in the above equation covers exactly two rows in the truth table; several rows are "covered" by more than one term
Representations of a Digital Design

Gates
most widely used primitive building block in digital system design

Standard Logic Gate Representation

Half Adder Schematic

Net: electrically connected collection of wires

Netlist: tabulation of gate inputs & outputs and the nets they are connected to
Representations of a Digital Design: Gates

**Full Adder Schematic**

- **Fan-in**: number of inputs to a gate
- **Fan-out**: number of gate inputs an output is connected to

Technology "Rules of Composition" place limits on fan-in/fan-out
Representations of a Digital Design

Blocks
- structural organization of the design
- black boxes with input and output connections
- corresponds to well defined functions
- concentrates on how the components are composed by wiring

Full Adder realized in terms of composition of half adder blocks

Block diagram representation of the Full Adder