Building Blocks of Digital Circuits

- Gates as electronic components
- Storage elements
- Larger Building Blocks
- Lookup tables and Field Programmable Gate Arrays

Jon Turner
So, Why Binary?

- Electronic computers represent information as voltage levels
- To make computer hardware simple and reliable, computers represent information in binary form
  - example: voltages greater than 3V are interpreted as representing one value (called “1”), voltages less than 2V are interpreted as representing another value (called “0”)
- In principle, could use more voltage levels
  - example: 0 to .75V represents “0”, 1 to 1.75V represents “1”, 2 to 2.75V represents “2”, and so forth
- In practice, this is rarely done
  - requires more complex circuits
  - circuits are more susceptible to noise, hence less reliable
Noise in Computer Systems

- Computers (like all electronics) are affected by noise
  - various sources (nearby signal changes, thermal vibrations of molecules in semiconductor materials, . . .)
  - in computers, noise can cause binary signals to be misinterpreted

- **Noise margin** is the amount of noise that a system can tolerate and still correctly identify a logic high or low
  - smaller power supply voltages lead to smaller noise margins
  - but also improve performance and reduce power requirements

<table>
<thead>
<tr>
<th>Voltage</th>
<th>Noise Margin</th>
</tr>
</thead>
<tbody>
<tr>
<td>5V</td>
<td>High</td>
</tr>
<tr>
<td>4V</td>
<td>High</td>
</tr>
<tr>
<td>3V</td>
<td>Undefined</td>
</tr>
<tr>
<td>2V</td>
<td>High</td>
</tr>
<tr>
<td>1V</td>
<td>Undefined</td>
</tr>
<tr>
<td>0V</td>
<td>Low</td>
</tr>
</tbody>
</table>

\[
\text{noise margin} = 3 \text{ V} \\
\text{noise margin} = 1 \text{ V}
\]
Delays in Logic Circuits

- Signal transitions take time and consequently signals take time to pass through gates
  - delays add up and place limits on circuit performance
- Timing relationships expressed using *timing diagrams*
  - shows how outputs respond as inputs change
  - often used to *specify* desired circuit behavior
Tri-State Buffers

- A tri-state buffer has a data input and a control input
  - when control input is high, output equals input
  - when control input is low, output is disconnected from input – called high impedance state and denoted by Z
- Allows different components to take turns using shared output wire(s) (often called a bus)
- Also allows wire(s) to be used for both input and output
Storage Elements

- **D-latch** is basic storage element
  - when $C=1$, $Q=D$ (and $Q'=D'$)
  - when $C=0$, $Q$ and $Q'$ don’t change

- **D flip flop** stores value at $D$ input when clock rises
  - most widely used storage element for digital circuits
  - can be implemented with two latches

- If $D$ input of latch changes as $C$ goes from 1 to 0, output value is uncertain
  - similarly for flip flop
  - output may oscillate or remain at intermediate value (metastability)
  - to avoid metastability, keep $D$ stable when $C$ is changing
Exercises

1. Draw a circuit that implements the logic function $A+BC'$. Draw a timing diagram that shows the output of the circuit as a function of time. Assume that initially $ABC = "000"$, then $ABC = "001"$, "010" and so forth, up to "111".

2. Draw a circuit that has three data inputs $P$, $Q$ and $R$, plus two control inputs $C_1$ and $C_2$, which are never both high at the same time. The output of the circuit should be $Z$ when $C_1=C_2=0$. If $C_1$ is high, then the output should equal $PQR$, while if $C_2$ is high, then the output should equal $P+Q+R$.

3. In the circuit diagram shown below, note that there is one flip flop and one latch. Complete the timing diagram at the right to show how outputs $X$ and $Y$ change in response to the input changes.
Decoders and Muxes

- A \( k \rightarrow 2^k \) decoder converts a \( k \) bit binary input value to a 1 on one of \( 2^k \) outputs
  - can build larger decoders using two smaller ones, plus some additional gates
  - useful for selecting one of a number of cases based on a numerical value

- A multiplexer selects connects one of its data inputs to its output
  - can also construct larger muxes from smaller ones, or using decoders
  - useful for selecting one of several stored values
Circuits with XOR Gates

- Complement circuit

- Equality/inequality circuit
Comparison Circuit

- Compare bits from most-significant down to least-significant
  - first position where bits differ determines which is larger
    - so if $A=010110$ and $B=010011$, $A$ is larger because it is 1, in first bit where the two differ

- Circuit constructed from repeated elements in linear array
Memory

- Implements array of *n* numbered storage locations
  - index for a location is called its *address*
  - each location stores a *word* with *w* bits

- Operation
  - when *enable* and *read/write* are high, value stored in location specified by *address* appears at *data_out*
  - when *enable* is high and *read/write* is low, value on *data_in* replaces value in location specified by *address*
  - may operate synchronously, or asynchronously

- Can be implemented using latches or flip flops, but usually, uses specialized, higher density circuits
Lookup Tables

- Digital logic is often implemented using devices called Field Programmable Gate Arrays (FPGA)
  - configurable device that can implement many different circuits
  - implemented using “programmable” logic elements and wires
- A Lookup Table (LUT) can be configured for any logic function with specified number of inputs (typically 4)
  - can view LUT as hardware implementation of truth table
- Example of circuit implemented with LUTs:
How Many LUTs Does it Take?

- When using configurable logic circuits, it’s useful to have way to estimate required number of LUTs
- For circuit with $n$ outputs, need at least $n$ LUTs
  - if using LUT4s & all outputs have $\leq 4$ inputs, $n$ is enough
  - newest FPGAs use LUT6s, so for outputs with up to 6 inputs, one LUT6 is enough
- In a circuit with $k$ LUT4s, $m$ inputs and $n$ outputs with $k > n$, $k-n$ LUT outputs connect to LUT inputs
  - so, $m+(k-n)\leq 4k$ which means $k \geq \lceil (m-n)/3 \rceil$
  - so, a 4-mux requires at least $\lceil (6-1)/3 \rceil = 2$ LUT4s and an 8-mux requires at least $\lceil (11-1)/3 \rceil = 4$ LUT4s
    - in practice, need three LUT4s for 4-mux, seven for 8-mux
Exercises

1. Show how you can implement a 4 input mux using a decoder and four tri-state buffers.
2. Draw a diagram of a 3-to-8 decoder using a 1-to-2 decoder and a 2-to-4 decoder as building blocks.
3. Write a VHDL specification for a 2-to-4 decoder.
4. Show how to implement a 4 bit comparison circuit using LUT4s. How many LUT4s are needed to implement a comparison circuit for n bit numbers? How does this compare to the lower bound?
5. Write a VHDL specification of a 4 bit comparison circuit based on the circuit design on slide 10.
6. Design another version of the comparison circuit in which the low order bits are compared first and information propagates from the lower order bits to the high order bits.
Xilinx FPGA Organization

- CLBs can be connected to “passing” wires
- Wire segments connected by switch matrix
- Long wire segments used to connect distant CLBs
- Configuration information stored in SRAM bits that are loaded when power turns on
Xilinx Configurable Logic Block

[Diagram showing various LUTs and control signals related to configurable logic block]