Review of Digital Logic Design

The design of microprocessor systems usually requires the design of logic circuits to interconnect VLSI devices. This lecture is a review of combinational and sequential logic design.

Logical Variables and Boolean Algebra

A logic variable can take on one of two values, typically called true and false (T and F) or 1 and 0. Remember that 'true' can be represented by either a high (positive-true logic) or low (negative-true logic) voltage.

Variables using negative logic are usually denoted by placing a bar over the name ($\overline{\mathrm{B}}$), or an asterisk after the variable name (B^*).

Warnings: Some authors use 1 and 0 to represent voltage levels rather than truth values. An $\overline{\text{overbar}}$ is sometimes used to indicate a logical complement operation rather than a negative-true signal.

Combinational Logic

A combinational logic circuit is one where the output is a function of the current input only. A combinational logic circuit can be represented as:

- *a truth table* that shows the output values for each possible combination of input values,
- *a boolean equation* that defines the value of each output variable as a function of the input variables, or
- *a schematic* that shows a connection of hardware logic gates (and possibly other devices) that implement the circuit.

Truth Tables

For example, the truth table for a circuit with an output that shows if its three inputs have an even number of 1's (even parity) would be:

a	b	c	p
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Sum of Products Form

From the truth table we can obtain an expression for each output as a function of the input variables.

The simplest method is to write a *sum of products* expression. Each term in the sum corresponds to one line of the truth table for which the desired output variable is true (1). The term is the product of the each input variable (if that variable is 1) or its complement (if that variable is 0). Each such term is called a *minterm* and such an equation is said to be in *canonical* form.

For example, the variable p above takes on a value of 1 in four lines (the first, fourth, sixth and seventh lines) so there would be four terms. The first term corresponds to the case where the input variables are $a = 0$, $b = 0$ and $c = 0$. So the term is $\overline{\mathrm{a}}\overline{\mathrm{b}}\overline{\mathrm{c}}$. Note that this product will only be true when a, b and c have the desired values, that is, only for the specific combination of inputs on the third line.

If we form similar terms for the other lines where the desired output variable takes on the value one and then sum all these terms we will have an expression that will evaluate to one only when required and will evaluate to zero in all other cases.

Exercise: Write out the sum-of-products equation for p. Evaluate the expression for the first two lines in the table.

Common Combinational Logic Circuits

In addition to the standard logic functions (NAND, OR, etc) some combinational logic functions are widely used and are often available as SSI ICs and as library blocks for ASIC design:

- a *decoder* is a circuit with N inputs and 2^N outputs. The output selected by the input bits (treated as a binary number) is set true and the other outputs are false. This circuit is often used for address decoding.
- a *priority encoder* does the inverse operation. The N output bits represent the number of the (highest numbered) input line.
- a *multiplexer* copies the value of one of 2^N inputs to a single output. The input is selected by an N-bit input.
- a *demultiplexer* does the inverse operation and copies one input to one of 2^N outputs.
- *adders*, *comparators* and *ALUs* (arithmetic logic units) implement the basic arithmetic and logic functions
- *drivers* and *buffers* provide high current, tri-state or open collector (OC) outputs

Exercise: Write out the truth table and the canonical sum-of-products expression for a 2-to-1 multiplexer.

Example: 7-segment display driver

LED numeric displays typically use seven segments to display a digit between 0 to 9. This example shows the design of a circuit that converts a 2-bit number into seven outputs that turn the segments on and off to show numbers between 0 and 3. We use the variables A and B for the two inputs bits and a to g for the seven outputs. We can build up a truth table for this function as follows:

B	A	a	b	c	d	e	f	g
0	0	1	1	1	1	1	1	0
0	1	0	1	1	0	0	0	0
1	0	1	1	0	1	1	0	1
1	1	1	1	1	1	0	0	1

From the truth table we can then write out the sum of products expressions for each of the outputs:

$$\begin{aligned}
a &= \overline{A}\overline{B} + \overline{A}B + AB \\
b &= 1 \\
c &= \overline{A}\overline{B} + A\overline{B} + AB \\
d &= \overline{A}\overline{B} + \overline{A}B + AB \\
e &= \overline{A}\overline{B} + \overline{A}B \\
f &= \overline{A}\overline{B} \\
g &= \overline{A}B + AB
\end{aligned}$$

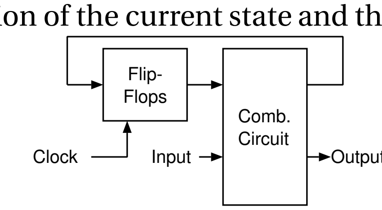
Exercise: Draw a schematic of a circuit that implements the above logic function.

Sequential Logic

Sequential logic circuits have some form of memory and so their outputs depend the current as well as past inputs.

The *state* of a sequential circuit is defined by the contents of all of the memory devices in a system. The current state of a circuit together with knowledge of the inputs allows us to determine the future behavior of that circuit.

A sequential logic circuit is a type of state machine (SM). There are two types of SM circuits. In the *Moore* SM the output is a function only of the current state. In the *Mealy* SM the output is a function of the current state and the inputs.



Design of Sequential Logic Circuits

The first step in the design of a sequential logic circuit is to specify the states, inputs, outputs, and transition conditions.

We must choose a sufficient number of memory elements (flip-flops) to represent all of the possible states. n flip-flops can be used to represent up to 2^n states (e.g. 3 flip-flops can encode up to 8 states).

We then write out the tabular form of the state machine representation with one line for each possible combination of inputs and flip-flop states. On each line we also show the the required outputs for that state and the desired next state.

The final step is to design a combinational circuit. This circuit has two sets of inputs: the outputs of the flip-flops (representing the current state) and the input to the sequential circuit. The circuit also has two sets of outputs: the inputs of the flip-flops (representing the next state) and the output of the sequential circuit. The designs of these circuits proceeds as described previously.

We also need to apply a clock signal to the clock inputs of the flip-flops . The sequential circuit will change state (although perhaps to the same state) at every positive (or negative, depending on the flip-flops used) edge of this clock signal. Practical circuits may also require some means to initialize the circuit when power is first applied ("reset" or "preset").

Flip-flops

A flip-flop can be in one of two states and is used as the memory in sequential circuits. The D (delay) flip-flop has a next-state input (D) and a clock input. The D flip-flop state changes only on the clock edge.

Usually all of the flip-flops in a circuit will have the same signal applied to their clock inputs. This *synchronous* operation guarantees that their states will change at the same time.

Common Sequential Logic Circuits

A *shift register* is a circuit of several flip-flops where the output of each flip-flop is connected to the input of the adjacent flip-flop . On each clock pulse the state of each flip-flop is transfered to the next flip-flop . This allows the data shifted in at one "end" of the register to appear at the other end after a delay equal to the number of stages in the shift register. The flip-flops of a shift registers can often be accessed directly and this type of shift register can be used for converting between parallel and serial bit streams.

A *counter* is a circuit with an N-bit output whose value increases by 1 with each clock. A *synchronous counter* is a conventional state machine and uses a combinational circuit to select the next count based on the current count value. A *ripple counter* is a simpler circuit in which the the Q output of one flip-flop drives the clock input of the next counter stage.

Example : Synchronous 2-bit Counter

A two bit counter will have four states. Two flip-flops are sufficient to implement four states. In this case there are no inputs, the circuit merely counts up at each clock signal. The transition conditions are simply to unconditionally go from one count to the next higher count.

If we use the variables Q0 and Q1 to represent the state of the system, and Q0' and Q1' as the subsequent state, the tabular representation would be as follows:

Q1	Q0	Q1'	Q0'
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

This example is particularly simple since there are no inputs or outputs. The only combinational circuit to be designed is the one to select the next state based on the current state:

$$Q1' = \overline{Q1}Q0 + Q1\overline{Q0}$$
$$Q0' = \overline{Q1}\overline{Q0} + Q1\overline{Q0}$$

Exercise: Add an up/down* input to the counter that controls the count direction.