CS 237 Meeting 12 — 10/3/12

Announcements

1. atoi available on course website.

Useful Combinational Circuits

1. All of the digital circuits we have seen in class and in lab are examples of combinational circuits — circuits whose output only depend on the current state of the circuit's input.

2. There are a few such circuits that serve as building blocks in many more complicated digital circuits. Our goal today is to talk about a few examples.

Decoders

1. A decode takes an n-bit input and has 2^n outputs. Each output is numbered going from 0 to $2^n - 1$. All of the output are zero except the one whose number equals the binary number encoded by the 1s and 0s on the n input lines.

2. If you think about each of a decoder's output lines separately you will realize that the truth tables for all the lines share an important property. The functions associated with these output lines are 1 for exactly one combination of input values. That is, in sum-of-products form, each output is described by a single product of inputs and their negations. Better yet, for every possible mix of inputs and their negations there is one output of the decoder that corresponds to that combination.

3. Decoders have many uses. If you try to imagine vaguely how the hardware that implements the load word instruction it should seem plausible that there would be a decoder to take the 5 bit instruction number in the current instruction and turn it into 32 signals (one for each register) such that just the signal for the register that should change would be 1.

Multiplexers

1. Multiplexers are closely related to decoders.

2. First, to motivate the use of a multiplexor, imagine how the hardware might fetch a number from a register to use as an operand for an instruction like ADD.

   - There would be a set of lines coming from each of the 32 registers. We would like a circuit that would take the 5-bit register number as one set of inputs, and the 32 lines for the nth bit of each of the 32 registers as additional inputs, and produce the nth bit of the register selected by the register number as output.

3. A multiplexer has n control inputs, 2^n data inputs and 1 output. The data input lines are numbered from 0 to $2^n - 1$. The output line is equal in value to the input line whose number is encoded by the control lines.

4. The wiring for a multiplexer can be seen as a slight extension of the decoder.

   - Start with a decoder.
   - Connect each output of the decoder to an and gate together with the data input line associated with the same control input as the decoder line.
   - Or together all of the outputs of these and gates.

Adders (and other arithmetic circuits)

1. Obviously, we want our computers to be able to compute, so we need to design circuits that can do arithmetic. We have previously seen that each of the bits of the $n + 1$ bit sum of two n bit binary numbers can be seen as the output of a boolean function of $2n$ inputs. Therefore, we know an adder can be built using sum-of-products form.

2. We also hinted at a more practical way to build an adder by designing a divice that does the addition of a pair of bits that form a column of a binary addition problem and then hook together enough of these circuits that we have one for each bit of each of the inputs.

- Such a circuit would have 3 inputs: one for a bit from matching positions of each of the two n-bit numbers being added and an additional input equal to the carry form the preceding column.
- It would have two outputs: 1 bit that is the sum to be written under the column and 1 bit that is the carry to the next column.

3. We even say a direct way to build such a circuit:
   - The sum output is the exclusive or of the three inputs.
   - The carry output is the majority logic function applied to the three inputs.
4. The column adder we have described is usually called a full adder. Also, it is usually described as being built from two "half adders" rather than an exclusive or and a majority logic.
   - A half adder takes two input bits and produces a sum bit and a carry bit.
   - The sum is the exclusive or of the inputs. The carry is the and.
5. we can build a full adder from two half adders and an or gate.
   - Connect the two data bits to one half adder.
   - Connect the other half adder to the sum output of the first half adder and the carry input.
   - Use the output of the second half adder as the full adder's sum output.
   - Use the or of the half adder carries as the full adder carry.

Take a Little Time

1. The structure of a circuit determines the speed with which it can "compute" the output of the function it implements.
   - When the voltage on a transistor's gate changes, it takes some finite amount of time for the transistor to turn on or off.
   - The time from when a new set of input values arrives at a gate and it is safe to assume that the output value(s) have changed and are stable is called the gate delay (it varies from one type of gate to another).
   - A device's overall delay is determined by the path from some input to some output with the highest combined gate delay.
2. In our adder built from full adders, the longest path will be the path from the first bits (and the carry of 0 into the low-order column) to the carry-out and sum of the high order bits.
   - The total delay will be roughly two times the number of digits in each input.
3. We know we can do better than this! Each output bit can be expressed in sum-of-products form where the maximal path will be three gates long (one not, one and, and one or).

   Unfortunately, the number of gates in such a circuit will be enormous! The "average" min-term for the higher order bit of the sum of a two input n-bit adder would require n not gates and an 2^n input and gate. There could be as many as 2^2n min-terms.
4. There are options between the simple carry ripple adder and the sum-of-products term.