Arithmetic Circuits

CS255

Chris Pollett

Feb. 22, 2006.

Outline

- Introduction
- Combination Circuits
- Full Adders
- Circuits for Addition (start)

Introduction

- We are now going to consider our second model of parallel computations: circuits.
- The algorithms we are going to demonstrate can be parallelized, are ones for addition, multiplication, etc.
- One motivation in trying to make these parallel is for speeding up the implementation circuits of these operations in hardware.
- Running time here will be depth of the circuit.

Combinational Elements

- Our circuits will be built out of combinational elements interconnected by wires.
- A **combinational element** is any circuit element that has a constant number of inputs and outputs and that computes a well defined function.
- If the inputs and outputs are boolean ({0,1}) then we will call an element a **boolean combinational element**.
- Example, could have ADD(x,y) whose inputs are single bits and whose output is two wire which has there sum in binary. Could view the high order and low order outputs as separate functions.
- If our circuit computes just one boolean function, we call it a **logic gate**. For example, AND, OR, NOT, XOR gates.

Truth table of a Logic Gate

- Is a table which says what the logic gate does on each possible input:

X	Y	AND
0	0	0
0	1	0
1	0	0
1	1	1

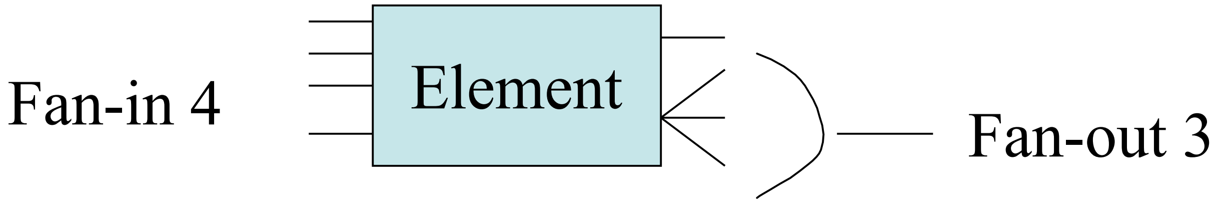
- How many logic possible n input boolean functions?

Time to Compute a Logic Gate

- Real logic gates take some time to settle and become stable on their output values after inputs values have been applied.
- This fixed amount of time we will call the **propagation delay** of the element.
- We will assume it is constant.

Combinational Circuits

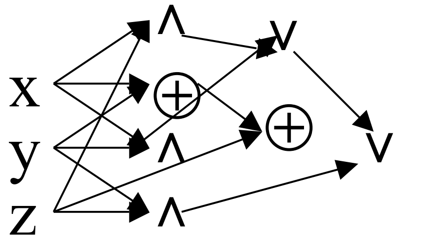
- A combinational circuit will consist of one or more combinational elements interconnected in an acyclic fashion.
- Interconnections will be called **wires**.
- A given output of an element can be connected to several other elements. the number of other elements is its **fan-out**:



- Fan-in is the number of wires going into an element.
- A wire which connects does not connect to an element input is called a **circuit output**.

Full Adders

- Consider the full adder on three boolean inputs x, y, z.
- This functions takes the three inputs and outputs two bits which represent the sum of the inputs in binary.
- The low order output could be computed as the parity on the inputs s=parity(x, y, z) = $x \oplus y \oplus z$.
- The high order output is on if the majority of the inputs are on: c=majority(x,y,z) = $(x \wedge y) \vee (y \wedge z) \vee (x \wedge z)$.
- Here is what a circuit might look like:



The time to compute this function is the length of the longest path = 3.

Size and Depth

- The **depth** of an input wire is 0.
- The **depth** of an element is the maximum depth of its inputs wire + 1.
- The **depth** of an output wire is the depth of the element it come from.
- The **size** of a circuit is the number of elements it contains.
- So in the last slide the depth of the full adder was 3 and its size was 7.
- Elements of the same depth can operate in parallel.

Addition Circuits

- We now try to develop efficient circuits for addition of two n bit numbers.
- For example, we will first look at ripple carry addition which can add two n-bit numbers in $\Theta(n)$ size and depth.
- This kind of addition thus does not give us any parallelism.
- We will then look at carry-lookahead addition which also has $\Theta(n)$ size but not O(1og n) depth. So very parallel.