Implementing a MIPS processor using SME

Carl-Johannes Johnsen (grc421)

6th February 2017

Abstract

asoenuthnsoaeuhtt

Contents

1 todo introduction?

SME terminology!

Bit1	Bit2	AND	OR	NOT	XOR
false	false	false	false	true	false
false	true	false	true	true	true
true	false	false	true	false	true
true	true	true	true	false	false

Table 1: The truth table for the four basic logic gates. Note: `NOT` is only considering `Bit1`.

2 Basic logic circuits

In this section, I will be looking at some basic combinatorial circuits. I start by looking at some logic gates, which implement some boolean functions. All of the considered values in the system are binary, e.g. the logic gates works on `1`s and `0`s.

2.1 Basic logic gates

Start by defining the basic logic gates, which are common for most circuits. A logic gate is a circuit abstraction, which has inputs and outputs. Its output values are based upon the input values.

`AND` - outputs `1` iff. all of its inputs are `1`, otherwise it outputs `0`.

`OR` - outputs `1` if one or more of its inputs are `1`, otherwise it outputs `0`.

`NOT` -outputs the inverse of its input, i.e. `1` becomes `0` and `0` becomes `1`.

`XOR` - outputs `1` iff exactly one of its inputs are `1`, otherwise it outputs `0`

The full truth table for all of the four logic gates can be seen in Table 1

2.1.1 Implementation

Implementing each of these four logic gates is quite simple: There is an input bus with two 1-bit values, a process for each of the gates, and an output bus with a 1-bit value for each of the logic gates.

2.1.2 Testing

To test the four processes, I have made a process, which sets the bits to all of the values in the truth table, and checks whether or not each process outputs the expected value from the truth table. How the processes are connected can be seen in Figure 1.

2.2 Decoder

The first combinatorial circuit I make is the decoder. An decoder takes an n-bit input, and produces an 2^n-bit output, where the bit corresponding to the input numbers value is set to `1`. E.g. if the input value is the binary representation of the number 5, then the 5th output bit will be `1`, and the rest will be `0`.

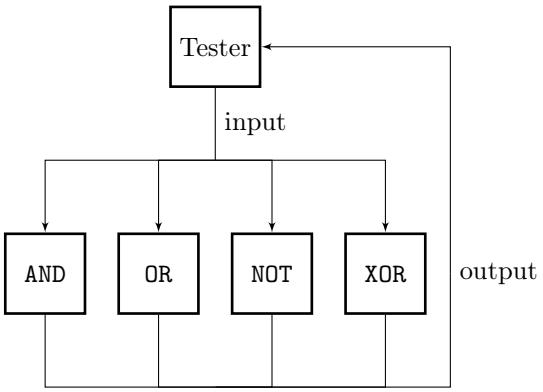

Figure 1: The structure of the test of the logic gates

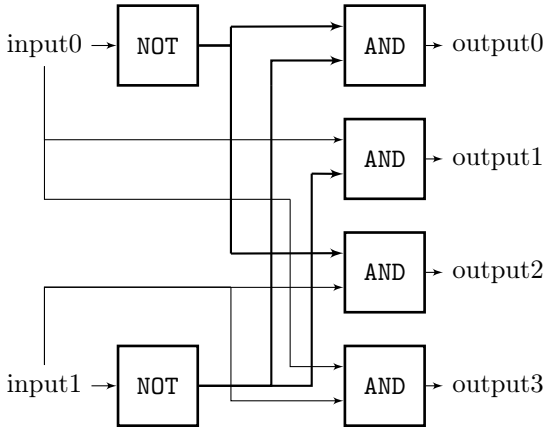

Figure 2: An 2-bit decoder made by `AND` and `NOT` gates.

A decoder can be made from a set of `NOT` and `AND` gates. We need to have n `NOT` gates, and 2^n `AND` gates. For each input, we split it into two, and send the copy to a `NOT` gate. Then for each output, we attach an `AND` gate, and give it inputs corresponding to the binary representation of the number E.g. if we get the number 5, the binary representation is 101, i.e. the 5th `AND` gate gets input from Bit0, `NOT` Bit1 and Bit2. An example of a 2-bit decoder can be seen in Figure 2.

2.2.1 Testing

I have written both a 2-bit decoder, and an n-bit decoder. They are tested in the same fashion as the logic gates, i.e. a tester process is connected to the input and output bus. The test inputs every combination of numbers, i.e. 2^n, and checks after each input, that the correct output is set to `1` and `0` respectively.

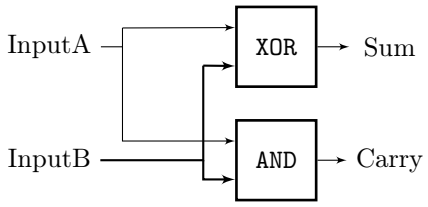

Figure 3: An half adder composed of `XOR` and `AND` gates

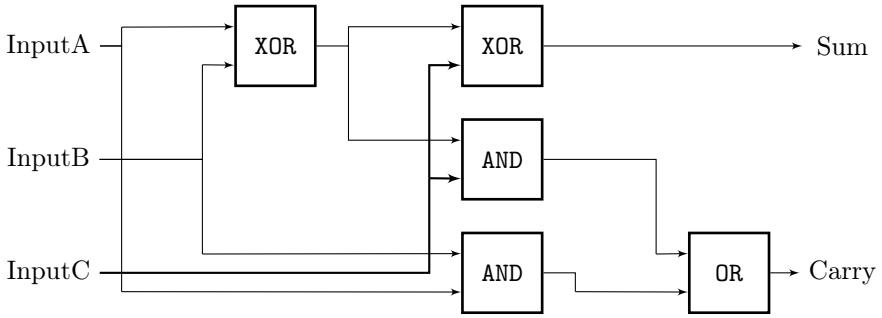

Figure 4: A full adder composed of `AND`, `OR` and `XOR` gates

2.3 Adder

As with the decoder, an adder can be constructed by a combination of `AND`, `OR` and `XOR` gates. An n-bit adder is a chain of two major components: an half adder and a full adder.

The half adder is the initial component in the chain. It takes two binary inputs, and outputs the sum and the carry of the addition (See Figure 3.

The rest of the n-bit adder consists of a chain of full adders, that take three inputs, A, B, and the carry from the previous link in the chain, and outputs the sum and the carry of the addition (See Figure 4).

The combination of the components can be seen in Figure 5.

2.3.1 Testing

I have tested both the half adder and the full adder, by giving each every possible input, and for each, compared them to their expected output. For the n-bit adder, I have made a function, that takes an integer as input, sends it along the corresponding input wires. I have also made a similar function, that takes the values from the output wires and packs it into an integer. I start by testing if it can make one of the simplest additions: 2+2. Then I check if it can overflow, i.e. set the Carry $n - 1$ to `1`. Finally, I run a series of random numbers through the adder, and check if the output is the expected sum of the two numbers.

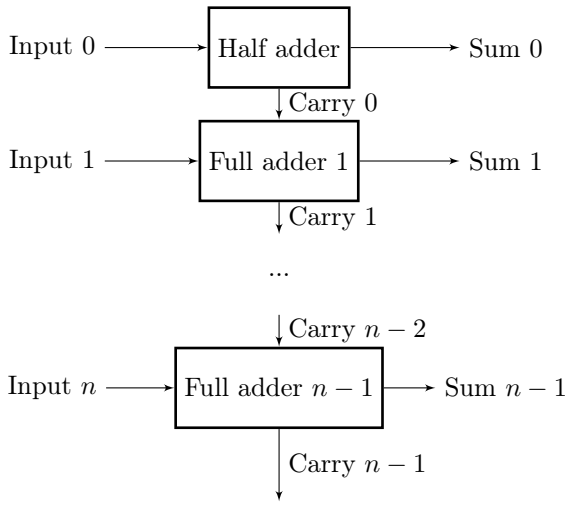

Figure 5: An n-bit adder composed of a half adder, and $n-1$ full adders. Note: Input A and B are both inside the inputs for simplicity.