EEP 201
Calculating a^b on Hardware

Saurabh Gupta Ankit Sagwal

December 10, 2008

1 Introduction

We intend to calculate a^b using the fast powering algorithm which involves repeated squaring and odd power accumulation to give the result of the computation in $log(b)$ time. Since power has to be raised sequentially, we can not do this in time better than this, even by doing computation in parallel.

2 Input Output Specification

We require to input two numbers and display a 32 bit output. Weighing different possibilities and keeping in mind the limited input and output ports available on the FPGA board, we decided to take parallel four bit input using the four slider switches for both the operands. A push button is used as the load button which indicates when to load the operands. The output is displayed on the 8 LEDs, wherein the user can select which portion of the output he wishes to see through the 4 push buttons. The user may also choose to enter the numbers afresh by reseting from the output state.

1. Press *b4* to start entering the first number.
2. Set the value of a on the slider switches.
3. Press *b3* to indicate load for the input a.
4. Set the value of b on the slider switches.
5. Press *b4* to indicate load for the input b.
6. Press *b1, b2, b3, b5* to see the output bits in groups of eight. Reset to idle state by pressing *b4*.

3 Modules

3.1 The Fast Power Module

The input to this module will be two 32 bit regesters containing the two operands and the output from this module will be a 32 bit regester containing the result of the computation.

The invariant maintained by the module will be as follows

$$y_o{}^{n_o} = y^n * p \tag{1}$$

The base case for this is when $n = 0$. Then the answer is given by p.
The induction step will be as follows:

$$if\ (n\%2 == 1)\ p = y * p \tag{2}$$

$$y = y * y \tag{3}$$

$$n = n/2 \tag{4}$$

Thus the two computational requirements for the module are checking oddness of power, division of the power by 2 and multiplication of p and y and y and y. We intend to do the odd checking by simply checking the last bit of the power, division by a simple right shift of the regester and the multiplication by implementing a sequential multiplier.

3.2 The Serial Multiplier

This module will simply multiply the two operands it receives as input and will place the output of the computation in another register. The invariant maintained by the module will be

$$a_o * b_o = p + a * b \tag{5}$$

The base case for the computation will be when $b = 0$ and then the answer would be p.

The induction step for it will be

$$if\ (b\%2 == 1)\ p = a + p \tag{6}$$

$$a = a * 2 \tag{7}$$

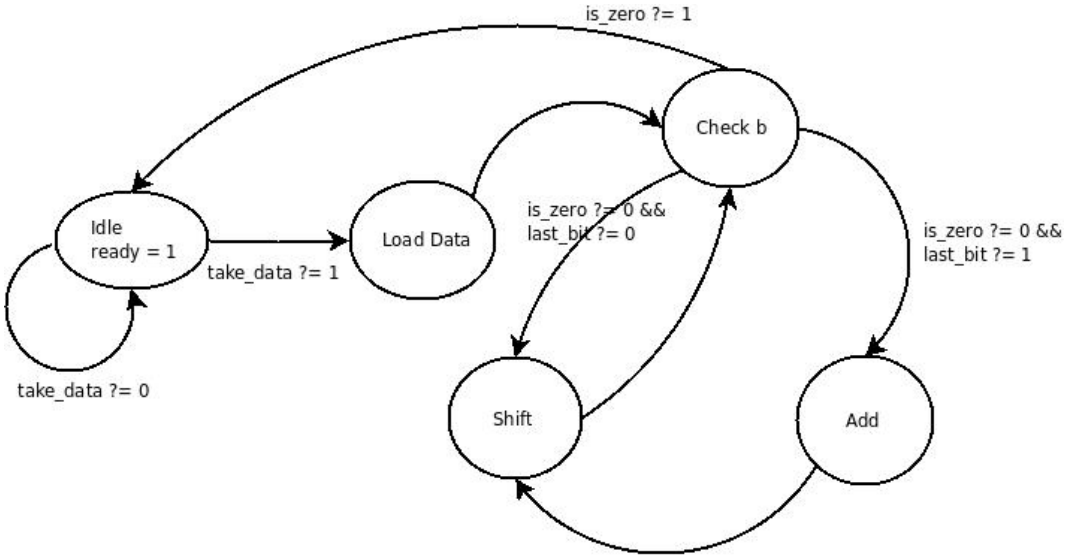
$$b = b/2 \tag{8}$$

Thus the computational requirement for this is again checking oddness of the multiplier, division of multiplier by 2, multiplication of multiplicand by 2, and addition of the multiplier to the partial answer. We implement the division and the multiplications by 2 by a simple left or right shift and the addition by a combinational adder.

4 Module Description

4.1 Algorithmic State Machine For Multiplication

Besides the relevant inputs and outputs to this module, we will also have a ready signal to indicate that the multiplier has finished with its previous computation and a take data signal, to indicate to the multiplier to take the data in for processing.



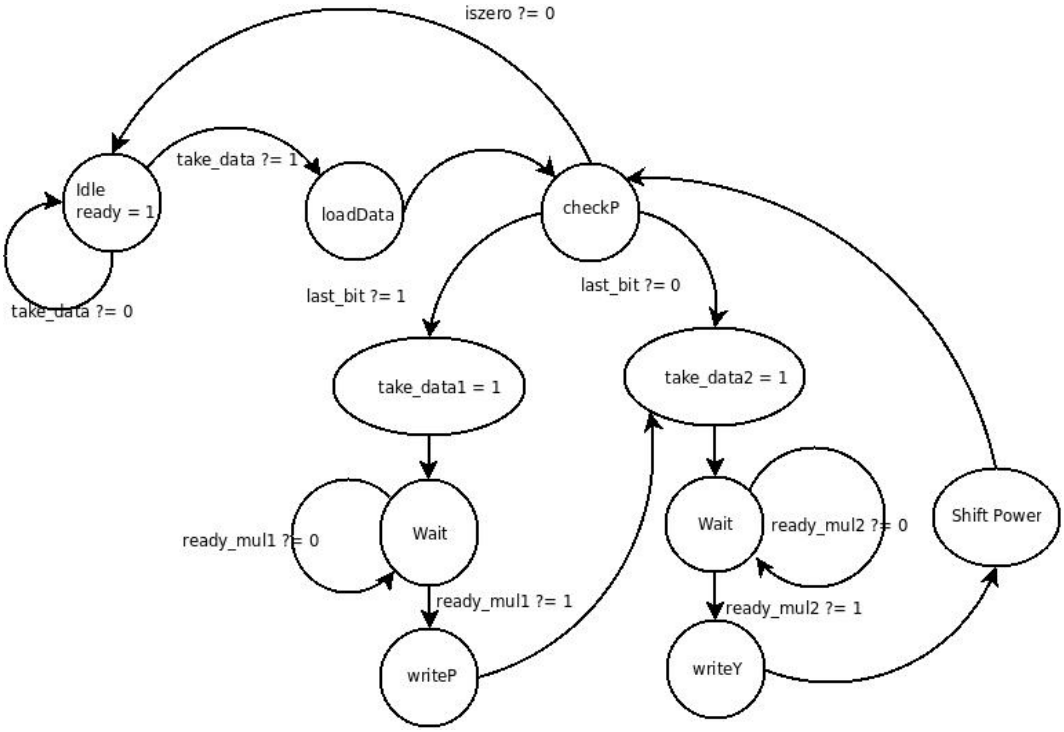
The datapath for multiplication includes a left shift register(for the multiplicand), a right shift register(for the multiplier), a accumulator register(for the partial sum and the final answer) and an adder(to add into the accumulator).

The controller will tell the datapath to load the multiplier, load the multiplicand, shift the multiplier right, shift the multiplicand right, add into the accumulator and reset the regesters. The datapath will in exchange tell the last bit of the multiplier and whether the multplier has gone to zero or not.

Inter Connections in the Data Path

4.2 Algorithmic State Machine For Fast Power

Besides the relevant inputs and outputs to this module, we will also have a ready signal to indicate that the multiplier has finished with its previous computation and a take data signal, to indicate to the multiplier to take the data in for processing.



The datapath for the fast power module includes a right shift register(for the power), a answer register(for the partial odd power product and the final answer) and the above multiplier.

The controller will tell the datapath to load the value of a, load the value of b, shift the power regester right, multiply the a by partial answer and to store back in the partial answer and the square the value of a and also reset signals. The datapath will in exchange tell the last bit of the power and whether the power has gone to zero or not.

Inter Connections in the Data Path