iJET EXPLORE International Journal of Engineering & Technology
CENTRE OF PROFESSIONAL RESEARCH PUBLICATIONS
ISSN 2049-3444

International Journal of Engineering and Technology Volume 2 No. 5, May, 2012

Architecture of a Floating Point Register for an Experimental RISC CPU

[1]Ajay A. Joshi , [2]Siew Lam, [2]Yee Chan
[1]Department of Electrical & Computer Engineering,
University of the West Indies, St. Augustine Campus, Trinidad & Tobago
[2]Multimedia University,
Jalan Multimedia, Cyberjaya, Malaysia.

ABSTRACT

An 8-bit RISC-CPU is designed at gate level using completely custom chip approach. CPU has an 8-bit integer unit and 16-bit floating point unit. The circuits are optimized by using more efficient algorithms. The algorithm discussed in this paper was applied for an 8-bit CPU design, however there is no reason that this couldn't be used for more powerful and serious CPU development. This paper discusses the architecture & design of Floating point register, which includes its VHDL implementation & simulation results as well. Entire project is implemented using VHDL and simulated using Altera MaxPlus II simulation software which can map the design into an Altera CPLD.

Keywords: *Floating point register, CPU, simulation, algorithm, Floating point unit, VHDL.*

1. INTRODUCTION

Paper focuses on the design and implementation of a 16-bit floating point register and decoder unit, which is a part of CPU with 8-bit integer unit. CPU has 4x16bit FPU registers, 16 bit data, address busses and 16-bit program counter. Data path is where most of the operations are done on by the processor's control unit. There are seven functional units, out of which 3 for FPU. This paper will discuss the design of a a 16-bit floating point register and decoder unit. Logic is discussed in detail and implementation block diagram along with the VHDL code and simulation results. The design and methodology is different than a generic one[3]. Main focus is on improved algorithm and relevant design. Amazing designs [2][4][8] helped us think different. This part of work is in line with other important blocks of the same CPU [1][5][6]

2. ARCHITECTURE

The figure 1 shows the internal architecture of the register file. FA and FB are 3-bit signal taken from the operand field of instruction register. They are used to select the relevant registers to datapath. Coupled with write back signal, FA signal will be used to determine which register to be stored with result from datapath. LT is an interrupt signal. This signal will be set to active high when there is an interrupt request from an Input/Output device. When this signal is active, all ALU and FPU registers will be loaded into temporary registers for storage while processing the interrupt routine. Besides, program counter (PC) will also be saved into a 16-bit temporary register.

RT is a return from interrupt signal. This signal is active when an interrupt service routine has ended. When RT is active, the processor will be returned to its previous state before the interrupt by restoring ALU and FPU registers data from temporary register file. PC will also be restored

3. FLOATING POINT REGISTER FILE

From the overall register file block diagram, now we take a look at the internal architecture of floating-point register file shown figure 1 the overall block diagram. The control signals are the same as what have been described earlier. We have a total of 8 16-bit registers, which are FW, FX, FY, FZ and another 4 temporary registers. On the left, we have a 'load register logic circuit' and a decoder. It produces appropriate signal for write back or storing immediate value to one of the registers. On the right, we have two blocks of 16-bit 4:1 multiplexer to select two sources to the datapath as specified in the instruction. All registers are 16-bit in length. You might ask why they are 16-bit in length when our objective is just to design a simple

8-bit Processor. As we will soon see, these registers contain three fields and each of them is operated separately. Therefore, we will only be working on data which is equal or less than 8-bit.

Decoder Design

Detail circuit for the decoder is shown in figure 2. It is a simple logic circuit which none of its output signals

will be active when either WB or S2 is '0'. Each of the output signals goes to its designated register.

As 4-input AND gate is extremely slow, I have converted it to NAND gates with extra NOT gates. More transistors will be used but it is significantly faster. Figure 3 shows transistor count and relevant delays.

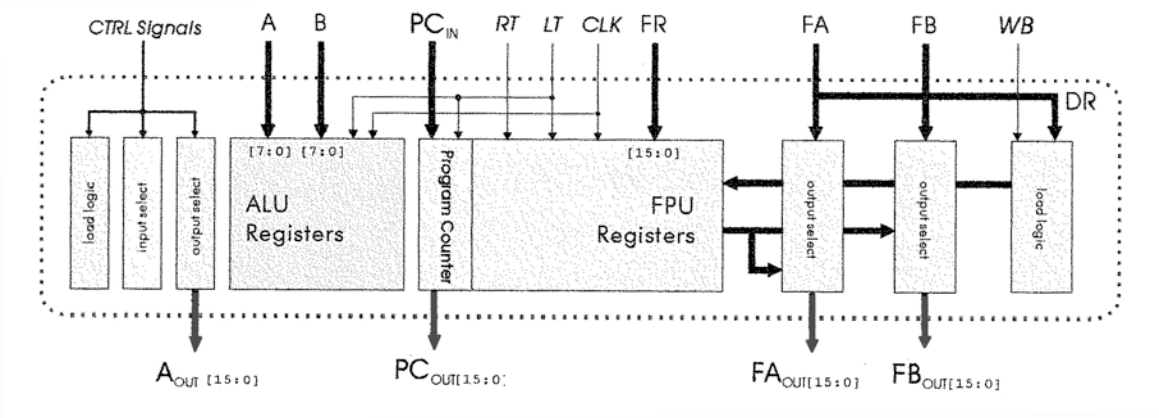

Fig. 1 the Internal architecture of floating-point register file

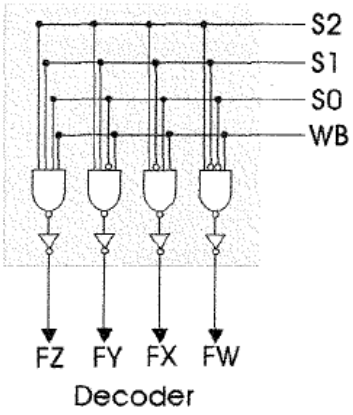

Fig. 2 Decoder circuit

Transistor:

Gates	Quantity	0.50 micron		0.18 micron	
		Transistors	Delay	Transistors	Delay
NOT	6	12	1.0ns	12	18ps
NAND4	4	40	2.2ns	32	60ps
Total Transistors		52	-	44	-

Fig. 3 Transistor count details

Delay (based on 0.18 micron): Critical path delay = (18 + 60 + 18)ps = 96ps.
Delay (based on 0.50 micron): Critical path delay = (1.0 + 2.2 + l.0)ns = 4.2ns.

VHDL Code for Decoder

```
library ieee;
use ieee.std_logic_1164.all;
entity decoder is
port( s          :in std_logic_vector(2 downto 0);
    WB           :in std_logic;
    FW,FX,FY,FZ  :out std_logic);
end decoder;
architecture dataflow of decoder is
```

```
begin
FW <=S(2) AND WB AND (NOT S(1)) AND (NOT
S(0));
FX <=S(2) AND WB (NOT S(1)) S(0);
FY <=S(2) AND WB S(1) AND (NOT S(0));
FZ <=S(2) WB S(1) S(0);
end dataflow;
```

Simulation and Comment

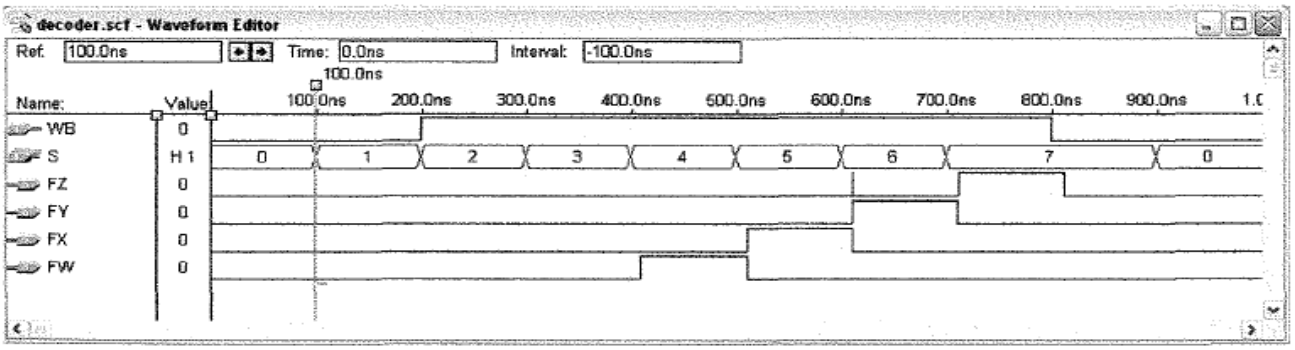

Fig. 4 Simulation results of Decoder

As we can see from the timing figure 4, only one signal will be active at any one time and when WB or S2 is inactive, all output signals are inactive too.

16-bit Floating Point Register

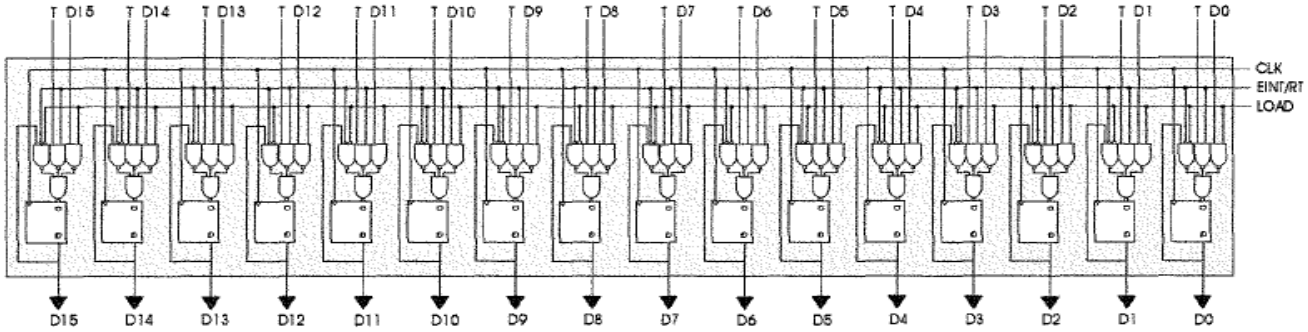

Fig. 5 16-bit Floating point register

The circuit in figure 5 shows a 16-bit register as storage location for floating-point data. D flip-flop without enable signal is used. Each register has two sources of input data, which are from the temporary floating-point register and immediate data or result from datapath. The control signals are RT and Load. Only one of them will be active at any one time. When RT is active, source from temporary register file will be restored into the main register file. On the other hand, source :from immediate data or datapath will be stored when Load signal is active. Load signal is connected to one of the output of decoder discussed earlier. Figure 6 shows transistor count and relevant delays for 16-bit FP register.

Transistors:

Gates	Quantity	0.50 micron		0.18 micron	
		Transistors	Delay	Transistors	Delay
D flip flop	16	448	5.6ns	416	100ps
NOT	2	4	1.0ns	4	18ps
NAND2	32	128	1.4ns	128	25ps
NAND3	32	256	1.8ns	192	41ps
Total Transistors		836	-	740	-

Fig. 6 Transistor count details for 16-bit FP register

Since there are 4 registers of this type:

Total transistors (for 0.18micron) = 740 * 4 = 2960.
Total transistors (for 0.50micron) = 836 * 4 = 3344.
(0.18micron) Critical path delay = (18 + 41 + 41)ps =l00ps.
(0.50micron) Critical path delay = (1.0 + 1.8 + 1.8)ns = 4.6ns.

VHDL Code for 16-bit Floating Point Register

```
library ieee;
use ieee.std_logic_1164.all;
entity fregister is
port( D,T              :in std_logic_vector(15 downto 0);
      CLK,RT,LOAD  :in std_logic;
      O              :in std_logic_vector(15 downto 0);
end fregister;
architecture behavioral of fregister is
signal temp:std_logic_vector(15 downto 0);
begin
process(CLK)
begin
if(CLK'event and CLK='1') then
          if RT='1' then temp<=T;
          elsif Load='1' then temp <=D;
          else temp<=temp;
end if; endif;
end process;
O<=temp;
end behavioral;
```

Simulation and Comment

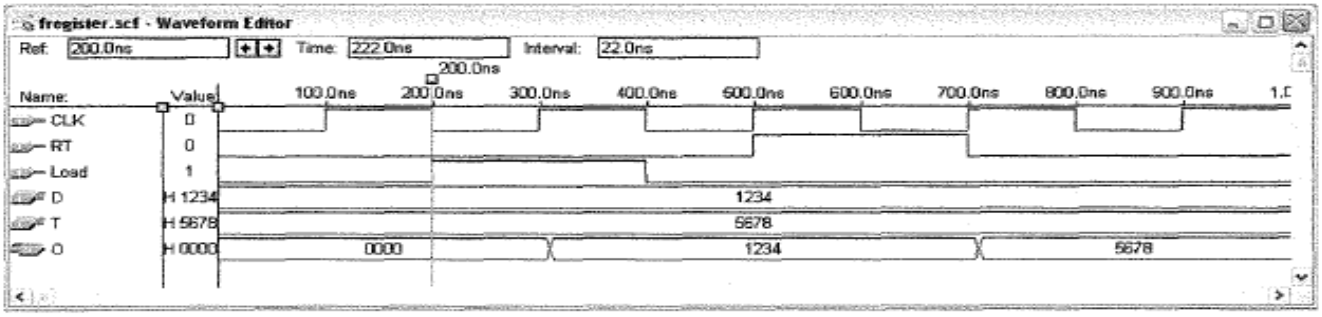

Fig. 7 Simulation results for 16-bit FP register

From figure 7 it clear that source from D input will be stored when Load signal is active. When RT is '1 ', source from T (temporary) will be restored into the main register file.

4. CONCLUSION

The decoder & 16-bit FP register simulation results are satisfactory. The results obtained are very encouraging. We successfully optimized the designs. More streamlined design can be implemented with some effort. We are also happy that the functional units are flexible enough to be used as a base for more powerful CPU with sufficient effort and time.

Currently an effort is being made to optimize and make sure that the current design be used for more powerful implementations.

REFERENCES

[1] Joshi, A., Lam S.L. and Chan, Y.Y., Algorithm & design of an efficient floating point ADD/SUB unit for an experimental CPU, International Journal of Intelligent Information Technology Application, 2009, 2(6), pg. 273-278

[2] 2. F. Fang, Tsuhan Chen, Rob A. Rutenbar, “Lightweight Floating-Point Arithmetic: Case Study of Inverse Discrete Cosine Transform” in EURASIP Journal on Signal Processing, Special Issue on Applied Implementation of DSP and Communication Systems.

[3] 3. Stallings, W., Computer Organization and Architecture, sixth edition, Pierson &Prentice-Hall, 2003.

[4] 4 L. Lacassagne, D. Etiemble, S.A. Ould Kablia, “16-bit floating point instructions for embedded multimedia applications”, in Proc. CAMP’05, July 2005, Palermo 5. Hida, Y., Li, X. S.,and Bailey, D. H., Algorithms for Quad-Double Precision Floating Point Arithmetic, Proceedings of the 15th IEEE Symposium on Computer Arithmetic,pg.155, 2001

[5] A.A.Joshi, S.L.Lam, and Y.Y.Chan, “"Design & simulation of a Barrel shifter for the 16-bit floating point unit of an experimental CPU", Journal of Information and Communication Technology,ISSN 2072-1471, Volume 2 [5] 2009.

[6] A.A.Joshi, S.L.Lam, and Y.Y.Chan, "Design of an Improved Multiplier Unit for an Experimental RISC CPU", Electrical Power Sys-tems and Computers, Lecture notes in Electrical Engineering 99 Vol. 3, pp. 323–330. Springer-Verlag Berlin Heidelberg 2011, ISBN 978-3-642-21746-3

[7] J. L. Hennessy, and D. A. Patterson, “Computer Architecture A Quantitative Approach”, 4th Edition; 2006

[8] 8. L. Lacassagne and D. Etiemble, “16-bit floating point operations for low-end and high-end embedded processors”, in Digests of ODES-3, March 2005, San Jose.. Available at http://www.ece.vill.edu/~deepu/odes/odes-3_digest.pdf