International Journal of Enhanced Research in Science, Technology & Engineering
ISSN: 2319-7463, Vol. 5 Issue 9, September-2016

Advanced Microprocessor and its Software

Shevernadze Dutt

Oregon State University, Corvallis, Oregon, USA

ABSTRACT

Here the differential CMOS gates are improvised for low power consumption. They are used to design a reduced instruction set code (RISC) processor, ALU and on chip memory. This modifies the way algorithms are written resulting faster execution time.

1.0 INTRODUCTION

The computer has gone through a rapid development due to progress in CMOS technology in last twenty five years. This rate of growth is so visible that by 1995 the microprocessor based computer became absolute necessity in modern life. Mainframe computers are replaced with multiprocessor system based on microprocessors. This article is about the gate level realization of high speed circuit, design of microprocessor and its influence on the software language as we implement the parallel algorithms.

Arithmetic functions such as binary addition, subtraction, multiplication and division are very basic need of many VLSI design. We implement two's complement binary arithmetic and avoid using subtraction operation. Division operation is done by an algorithm using subtraction and multiplication repeatedly.

Is believed that parallel processor is the single most important factor in deciding the processing speed. Here we found a method to increase the speed of uni-processor with RISC architecture and thus overall increase in performance of multiprocessor system.

Section II describes a method of passing of charge between two capacitors (one is high and another capacitor is low) through a inductance. They are connected in differential mode. Similar techniques are used with CMOS devices to realize NAND, XOR and D latch circuits.

Section III introduces the microprocessor which has ALU (ADDER and MULTIPLIER), the program memory and variable finite state machines with jump instruction. The distribution of instruction is such that program memory takes one or two bits for each instruction. The microprocessor has on chip data memory.

Section IV describes the content of FSM, program memory and data memory for matrix multiplication and inverse of a square matrix.

Section V and section VI deals with parallel microprocessors and their application in different DSP based algorithm. We conclude the paper in section VII. Performance evaluation of this processor and the parallel processor quantitatively are done here.

2.0 HIGH SPEED GATES

We use differential gate (NOT, NAND, XOR, D latch) and transfer the charge from capacitor at higher voltage to output capacitor at lower voltage through an inductor using natural oscillation. This is explained in Figure 1, where capacitor C_a has V voltage and the other output capacitor C_b is at 0 voltage. The inductance L is connected through switch S_1, such that as the gate voltage of S_1 is high C_a should go low at 0 and C_b will be high. The solution of voltage is given in Laplace domain as $Vy(s)=V/LC/(s^2+2/LC)$ which is a sine wave of frequency $w_o=(2/LC)^{0.5}$. The current will be $Iy(s)=Vy(s)*sC_b$ and it will be a cosine wave. To trap the charge at C_b we need to disconnect the switch S_1 at Iy(t) equals to zero. But the inductance has a induced voltage of $L\ di/dt= -V\ w_o*\sin(w_o*t)v$ which will be maximum (negative) at that time.

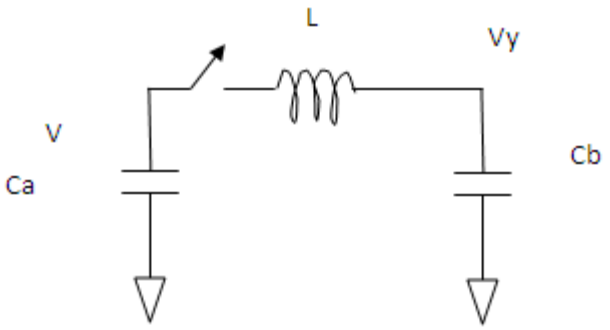

Figure 1. Charge transfer from capacitor Ca to Cb

To reduce the damage on switch S_1 we put a back to back diode with higher threshold voltage (>100 volt). This makes the current as triangular with less induced voltage at chop off time. Figure 2 shows an inverter as buffer which will transfer the charge at high speed with less power dissipation. Similarly we can get NAND gate, XOR gate and D latch.

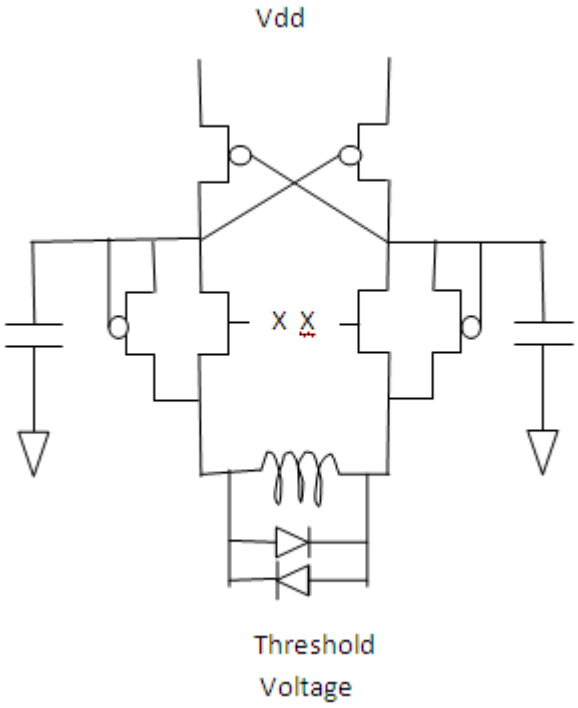

Figure 2. Buffer circuit.

3.0 MICROPROCESSOR DESIGN

The arithmetic logic unit (ALU) consists of ADDER and MULTIPLIER. Both are implemented using bit serial techniques. Here we considered a data width of 8 bits which will be 9 bits after addition and 15 bits after 8 x 8 multiplications (could be 16 bit also). For DSP processor, the data could be in fraction, maximum less than 1 and with exponent. In multiplication it is a normal procedure with exponents are added. In addition, it's the exponents are kept same. ALU is eight time faster than CPU FSM.

This is extremely reduced instruction set with + or * in operation. Subtraction is done by two's complement addition and the division is done by algorithm using addition and multiplication.

The finite state machine implements the variable counter. It calculates when to come out of the loop as stated by written program. It also finds JMP condition and to move N steps UP or DN of the program memory and data memory.

Thus the written program has four arithmetic operations, for loop with odd number of loop number and JMP on condition to which steps in written program and data memory. The FSM stop data flow if the steps to be jumped do not match. This is the reduced instruction set for the most simplistic RISC processor.

If the variable FSM is implemented in software it will have a variable counter and a comparator. This could be avoided by using hardware based FSM. Both cases, FSM should generate UP/DN or hang pulse matching with the delay it gets executed. Figure 3 explains the microprocessor.

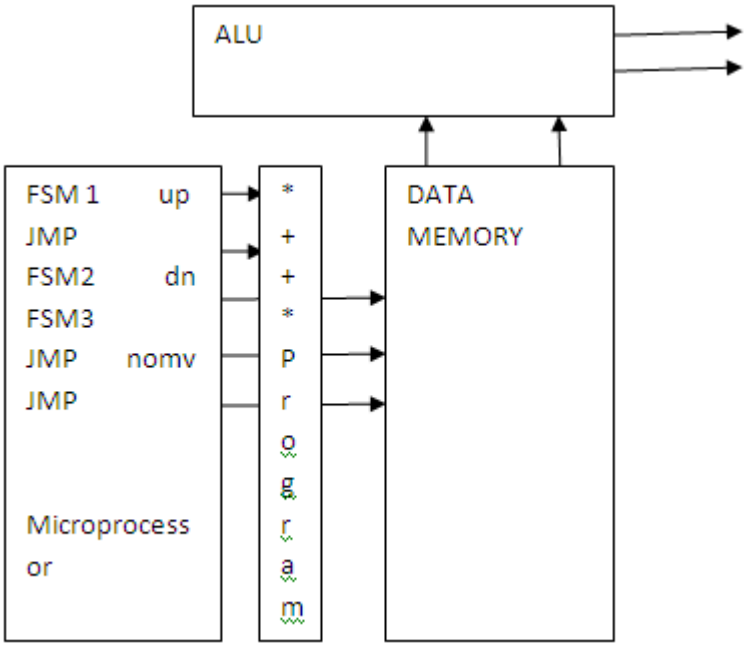

Figure 3. Microprocessor

4.0 EXAMPLES

Y=AB
The matrix multiplication is needed almost in every field of science and technology. Assume, A and B are square matrices of dimension n x n. The dimension are such that they can be put into 1 x n2 parallel bits of on chip memory which moves in both directions The algorithm is
For i=1:n
For j=1:n
For k=1:n.
Y(I,j)=Y(I,j)+A(I,k)*B(k,j);
End
End
End
The implementation in FSM, program and data are as follows: We need two memories for A (a1,a2,a3,….) and for B(b1,bn+1,b2n+1,…..). The FSM is given by
Counter1
Counter2
Counter 3
*
+
+
*
Counter3
Counter2
Counter1
Now we do the same with data and roll it forward and back. It is faster at least by a factor of 2 and it is the speed of shift register which limits the operation of FSM.

```
Y=inv(A)
        We now look into the problem of matrix inversion of n x n dimension. The algorithm is as follows:
         For k=1:n
                b(k)=1/a(k,k)
                for i=1:2n
                        a(k,i)=a(k,i)*b(k)
                end
                for i=1:n ~k
                        for j=1:2n
                                a(I,j)=a(I,j)-a(i,k)*a(k,j)
                end
        end
        a(k,k)~0
end
We have to find FSM, the program and the data flow.
                        Counter1
                                Division
                                Counter2
                                        *
                                End
                                Counter3 CMP  JMP
                                        Counter4
                                                *
                                                +
                                                +
                                                *
                                        End
                                End
                        End

The compare and jump will move the data by n data.

Y=inv(X)Z';, where X is a square matrix and Z is row vector.
For k=1:n
                b(k)=1/a(k,k)
                for i=1:n+1
                        a(k,i)=a(k,i)*b(k)
                end
                for i=1:n ~k
                        for j=1:n+1
                                a(I,j)=a(I,j)-a(i,k)*a(k,j)
                end
        end
        a(k,k)~0
end

We have to find FSM, the program and the data flow.
                        Counter1
                                Division
                                Counter2
                                        *
                                End
                                Counter3 CMP  JMP
                                        Counter4
                                                *
                                                +
                                                +
```

```
                                        *
                                    End
                                End
                            End
```

5.0 PARALLEL MICROPROCESSORS

The multiprocessor based systems are mainly processors working parallelly the same problem to make it faster. It could be the processors with individual memory along with a shared memory. Whatever is in shared memory can be accessed by all by the same bus. The opposite of this idea is separate bus connecting any two of the processors. Here, the buses will be problem as space is not there.

6.0 PARALLEL ALGORITHMS WITH SHARED MEMORY

Here we look at the problem of matrix multiplication when a single memory cannot accommodate the data matrix A. The algorithm of the problem is given in section 4. Here we choose the architecture with 4 data memories of 4 microprocessor have accommodated the matrix A as D1, D2, D3, D4. The shared memory has a portion of matrix B which is E1. Hence we get D1*E1, D2*E1, D3*E1 and D4*E1 at the same time and save to main memory. Then we load E2 to shared memory and the multiplication continues.

CONCLUSION

Here we explained a new method of implementing gates with higher speed and lower power consumption. It does allow the charge to get to the ground but changes the position in capacitor using an inductor for oscillation. It was first explained with the help of LC circuit and then with an inverter. Other gates can be improvised as proposed. This allows a higher clocked microprocessor design, which was subsequently discussed for single and parallel processor.

REFERENCES

[1]. M.C. kan and D.I. Pulfrey, “A Comparison of CMOS Circuit Techniques: Differential Cascode Voltage Switch Logic Versus Conventional Logic,” IEEE Journal of Solid State Circuits, SC-22, No. 4, August 1987
[2]. D. Das, “VLSI Design,” Oxford University Press, 2010
[3]. J.P. Uyemura, “Introduction to VLSI Circuits and Systems,” Wiley India Edition, 2002
[4]. Z. Kohavi and N. K. Jha, “Switching and Finite Automata Theory,” Cambridge, 3rd Edn.