International Journal of Enhanced Research in Science Technology & Engineering, ISSN: 2319-7463
Vol. 4 Issue 1, January-2015, pp: (9-12), Impact Factor: 1.252, Available online at: www.erpublications.com

Design of High Performance Microprocessor

Amit Kumar Dutta
Calcutta Institute of Technology, WB, India

ABSTRACT: High performance microprocessor design using Q-Dot technology addresses the key design issues like ALU design, memory management required to run parallel algorithms in solving Fast Fourier Transform algorithm, matrix inversion algorithm, Barnes problem, Ocean problem, sorting and graphical processing unit. We design the processor such that it can implement parallel algorithm in real time operation using error detection for addition and multiplication. Current mode logic is used to get the speed.

1.0 INTRODUCTION

The major challenge of today's computer is how to improve on microprocessor's performance in terms of its speed of execution while implementing various algorithms. The design of computer depends on how it will be used and the available technology. Trends in memory management by operating system have lasting impact on its usage. Here we use shift register banks instead of DRAM and shared memory to achieve high parallelism in hardware as well as in software. Lastly the recent trend is to replace Assembly language programming by higher level programming but we stick to the former. So the design procedure is to define the ALU in the first step with adder, subtractor and multiplier except their numbers and connections are kept unknown which is to be decided by the algorithm to be implemented. Similarly, stack memory, register memory and on chip cache are to be decided by the requirement of the algorithm as well as the shift register bank size.

Another area of problem is the speed of data bus which decides the speed of execution of ALU for most of the algorithm. We change it by sending four data parallel at a time to the CPU. Lastly we solve the data latency problem in READ/WRITE cycle by making the data independent of the address. Here we make the data available to the registers all the time and storing the result of computation in the shift register or in the shared memory. Thus there is data flow to the CPU and data flow out of CPU to the shift register with same speed.

In Fast Fourier transform algorithm, we choose the radix-2 Decimation in Time/Frequency algorithm. We solve the memory management problem by MUX-ing the memory out going data stream and changing the path of input data stream by putting switch to toggle. The shift register memory gives two data output and we do butterfly structure with them. The weighted coefficients are calculated earlier and kept in shared memory or in stack.

We formulate the matrix inversion along with a vector multiplication as in X=inv(A)*Y. Here the Y vector is kept in shared memory and row vectors are multiplied and subtracted. In the Barnes algorithm, a new method is invented which requires N^2 additions. We explain these three algorithms in detail.

The section-I introduces the subject. In the section-II we design the ALU and in the following section we modify ALU to accommodate error detection over addition and multiplication. Section IV discusses the various parallel algorithms to be implemented by the microprocessors. We study three main algorithms for memory management purpose. In section V we design the microprocessor and the mother board. In the last section we conclude the paper.

2.0 ALU

ALU normally has four arithmetic operations (ADD/SUB/MULT/DIV) and other logical operations [5]. Here we consider ADD/MULT to be the back bone of ALU and SUBtractor as a modified circuit used for ADDition. The divide operation is done using multiplier and subtrator circuit. Now the ADD operation can be of X=X+A(i) kind or X=A(i)+B(i) kind. Similarly, the multiplier can be of X=A.B or X=X+A.B or X=X-A.B type. Here we keep all the provisions. Floating point addition, subtraction and multiplication are also possible.

A modification of adder circuit over bit serial is that it is done by two XOR gates and a D f/f gate as shown in Figure 1. This is modified to keep provision of subtraction. The rough power dissipation in adder, subtractor and multiplier circuits are calculated as follows:

For the adder circuit: $\frac{1}{2}*C*V^2*(2*3*2)$.

For the multiplier circuit: 16*Adder circuit.

The voltage swing (V) is kept low so that statistically the charging current creates less error in voltage with a variation in capacitor.

3.0 ERROR DETECTION

Error often takes place in adder, subtractor and multiplier circuits during computation due to misalignment of signals which results in not charging of capacitor correctly. This error has to be detected and if possible be corrected. Here we consider a error detection scheme based on mod-2 addition where individual code byte is protected by even parity checker. Also in between additions, the carry byte is checked for error by parity.

Example: 1010 even
\+ 0011 even

1001 even
Carry + 0100 odd

1101 odd

4.0 PARALLEL ALGORITHMS

Here, the emphasis is kept on different applications and one such application is in parallel multiprocessing environment [1]. Fast Fourier Transform, matrix solution of linear and bilinear problems, the Barnes problem, the Ocean application, sorting of very large database and graphics are the areas we are trying the microprocessor to work. Here we check the first three problems and find their solution in terms of algorithm and memory management associated with them.

FAST FOURIER TRANSFORM

The Fast Fourier Transform (FFT) is a signal processing method used for spectral estimation. Here we study the application of one dimensional complex variable FFT. It has complexity of multiplication $n*\log_2 n$ except that it is difficult to manage the memory. The algorithm for Decimation in Frequency radix-2 is given below:

$$X(2k) = \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) + x\left(n + \frac{N}{2}\right)\right] W_{N/2}^{kn} \quad k = 0,1,\ldots,\frac{N}{2} - 1.$$

$$X(2k+1) = \sum_{n=0}^{\frac{N}{2}-1} \{\left[x(n) - x\left(n + \frac{N}{2}\right)\right] W_N^n\} W_{N/2}^{kn} \quad k = 0,1,\ldots,\frac{N}{2} - 1.$$

For a single processor with a single memory as shift register bank, the data (2 FP numbers) will be out at a time from memory to the up. If the memory size is 1024 then memory will be 32 *32 and we can have numbers placed as
0 1 2 3 4 5 6 7 8 …….31
512 513 514 515 543
32 33 34 35 36 …….63
544 545 546 547…….574
……
256 257 ……287
768 ……799

So the first data set is (0,512) and second data set (256,768) and after butterfly operation their memory place will be shuffled to (0,256) and (512,768). This continues through a MUX in the output and input of the memory.

Similarly in multiprocessor environment, we can find FFT of longer length like 1024*8 for 8 processors. If the data speed is 100Kbps parallel and ALU is faster enough to multiply and add in 10 microsecond then total 1024 point FFT will be performed in roughly 100 milliseconds.

SOLUTION OF LINEAR SYSTEM

Linear equations can be formed as AX=Y where A is square matrix of dimension (N x N) and X, Y are unknown and known vectors. Here we consider a case where N=512. We consider the memory size as 512*64 words for each processor. We consider a shared memory of 2*512 words. The algorithm is as follows:

Step 1: Load Y to shared memory.
Step 2: Load A to each processor's data memory.
Step 3: Pivot the Nth row for N=1, and load it to shared memory.
Step 4: load A to CPUs and multiply by the first data in row to the pivoted row and subtract. Store it.
Step 5: Goto Step 3.
Step 6: Put the values and subtract in backward direction from 511 and find the solutions X.

It has $0.5*N^3$ multiplications and similar numbers of subtraction. The memory requires MUX in the output and input. It needs 512*512*256/(8*400000) or 21 seconds.

SOLUTION OF BILINEAR EQUATIONS

Here the equations can be written as

$$\begin{bmatrix} a_{11}X_1 & a_{12}X_1 & \dots & a_{1N}X_1 \\ a_{21}X_2 & a_{22}X_2 & \dots & a_{2N}X_2 \\ \dots & \dots & \dots & \dots \\ a_{N1}X_N & a_{N2}X_N & \dots & a_{NN}X_N \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \dots \\ X_N \end{bmatrix} = K$$

This can be solved by using the algorithm X[n+1]=0.5*[X[n]+K/X[n]];
Each iteration takes roughly 21 seconds for N=512.

THE BARNES PROBLEM

Barnes is an implementation of an algorithm for n-body problem in galaxy evolution [1]. N-body algorithm simulates the interaction among a large number of bodies that have forces interacting among them. Here the bodies represent collections of stars and planets and the force is gravity. The repulsive force is the electromagnetic action. So for a given distance we have to find the current to get the steady state solution. The force due to electromagnetic action is **F=j** x **B,** where **j** stands for current density and **B** is the magnetic field [2]. Now the magnetic field due to a current at a distance will be dependent of the distance vector. Hence the total force acting on the body should be $\sum_{n=1,n\neq i}^{N} J_i \times B_n + \sum_{n=1,n\neq i}^{N} G_{in} = 0$.
So in matrix form it can be written as,

$$\begin{bmatrix} a_{11}J_1 & a_{12}J_1 & \dots & a_{1N}J \\ a_{21}J_2 & a_{22}J_2 & \dots & a_{2N}J_2 \\ \dots & \dots & \dots & \dots \\ a_{N1}J_N & a_{N2}J_N & \dots & a_{NN}J_N \end{bmatrix} \begin{bmatrix} J_1 \\ J_2 \\ \dots \\ J_N \end{bmatrix} = -G$$

Where the coefficients depend on inverse of distance and coil's diameter. Now we can use the algorithm mentioned in the method of solution of bilinear equation which has N^3 complexity. We have a different solution which has N^2 complexity with number of iteration yet to be found out.

We consider the initial starting point as x_0 and consider $y=G-x^2$ as the curve where the square-root value will be the x axis where y=0. We take a tangent at x_0 and find x_1 where y=0; and find x_1 and y_1 point on the curve. And so on until we reach where the y_{n+1} will be 0.

The algorithm is $x_{n+1}=x_n-y_n/2x_n$ where $y_n=C-x_n^2$;

Now in our application the equations are given in the last page. We consider 3 bodies. The slopes will be m_1=-(a_{12}*J_2+a_{13}*J_3); m_2=-(a_{21}*J_1+a_{23}*J_3) and m_3=-(a_{31}*J_1+a_{32}*J_2)

So the algorithm will be $J1_{n+1}$=$J1_n$-$Y1_n$/m1; $J2_{n+1}$=$J2_n$-$Y2_n$/m2 ; $J3_{n+1}$=$J3_n$-$Y3_n$/m3. This avoids matrix inversion. For single value, square-root algorithm and this algorithm are same in each step, but for more variables they will differ.

5.0 THE MICROPROCESSOR

The microprocessor here is thought of as a math-processor engine with ADD/SUB/MULT/DIV as the primary operations [3][4]. It has a Instruction Register set and we choose reduced instruction set. It has data memory stack and output data memory stack. It also has a stack in Instruction Register. The data flow with the Instructions and gets executed and flow out of CPU to the data memory automatically if not changed by programming. The control unit in microprocessor makes the arithmetic/logical operation possible. There is the central controller which controls the memory management of all parallel microprocessors and feeds the instructions from instruction (program) memory. So we find that the data bus speed decides the processor speed. The ALU has multiple 4/4 adders/multipliers. The multipliers are as given in reference [5]. If data speed is 100 kbps then we allow four data in parallel.

The mother board has memory for data, program and shared memory. The size of the memory is decided by data size. The size of stack in CPU will be minimum 8(x 16) and it keeps the shared memory outputs like W_N^{kn} in FFT. So the power dissipation in microprocessor will be less than 50 mW at 2 MHz clock speed though ALU can operate at much higher frequency. It is because the capacitance values are very less and voltage swing is limited.

6.0 CONCLUSION

Reference [6] introduces a very high frequency oscillator using Q-Dot technology. If the voltage swing is stable and time period is stable with low jitter then we can use the similar digital and analog circuit at same or at 1/4th of that frequency. This is the principle we followed in designing the logic gates and that culminates into a microprocessor which may work at very high frequency. Unfortunately, the data bus is not that fast and cannot be made more than four parallel lines which limit the microprocessor clock. We attempt to solve this problem by using addressless data stream through a shift register bank as memory using same technology and provide suitable memory management techniques for the main parallel algorithms.

REFERENCES

[1]. J.L Hennessy and D. A. Patterson, “Computer Architecture A Quantitative Approach,” Morgan Kaufmann, 2nd Edition
[2]. R.P. Feynman, R.B. Leighton and M. Sands, “ The Feynman Lectures on Physics,” Addison Wesley
[3]. B. Parhami, “Computer Architecture-from Microprocessor to Supercomputer,” Oxford Series, 2005
[4]. K. Hwang, “Advanced Computer Architecture with parallel programming,” McGraw Hill 1993
[5]. A. K. Dutta, “Vector Arithmetic Logic Unit,” IJERSTE Vol. 2 Issue 12, December -2013, pp: 75-80
[6]. Accepted, A. K. Dutta, “Current Mode Logic Gates Using Q-Dot Technology,” IJERSTE Vol. 3 Issue 11, November-2014

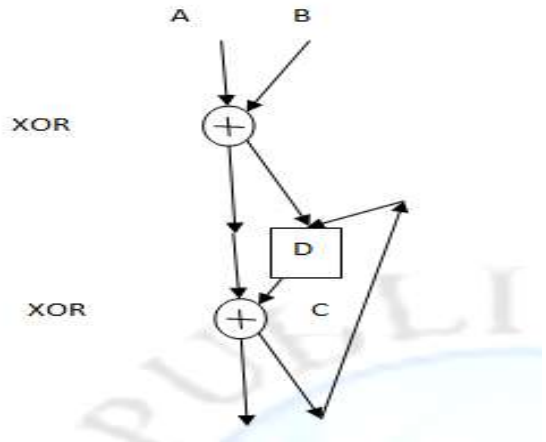

Figure 1. Modified bit serial adder.