An Analysis of the Efficiency of Pipelined Processing Model in the Design of Integrated Devices

Eludire A. A.
Department of Computer Science, Joseph Ayo Babalola University
Ikeji Arakeji, Osun State, Nigeria, aaeludire@jabu.edu.ng

ABSTRACT

In the implementation of pipelined processing, the performance or efficiency is usually represented by two parameters, speedup and throughput. It is important to investigate how the speedup and throughput change as pipelining is implemented. Pipelining is implemented by defining a particular operation which can be executed in a number of stages and the performance is dependent on the number of stages. The main task in the execution of the operations is to identify the optimal number of pipeline stages that gives good throughput and speedup based on cost performance. The MULTIPLY operation is chosen in this work using a fixed point multiplier pipelining for the analysis of efficiency measurement. Pipelined multipliers are used in implementing pipelining arithmetic unit when performing multiplication of fixed point numbers. It is known that multiplication takes longer time, so pipelining improves the performance and speed of this operation. The work analysed pipelined processing and identified critical point of efficient hardware utilization by showing possible redundancy for the achievement of a particular threshold.

Keywords: pipelining, efficiency, integrated, devices, speedup

INTRODUCTION

In recent times a lot of attention is devoted to the design and use of pipelined processors for a wide range of highly productive computing devices. Increasing the efficiency of such computing devices can be achieved by proposing new design algorithms and methods oriented toward large-scale integration (LSI) and very large scale integration (VLSI) (Ajit Pal, 2014; Omondi, 1999). Such increment of efficiency brings about problems connected with optimization of pipelined information devices at the structural, functional and circuitry design stages (Agarwal et al, 2000). Pipelining is one of the most important and popular technique that is used to enhance the performance of a processor (Golub and Ortega, 2014). It is a variant of parallel implementation technique that exploits instruction level parallelism which is done in the hardware. The choice of pipeline structure in relation to the design of microprocessors was investigated in (Hartstein and Puzak, 2002). The work intensified the question of dependency that exists between pipeline depth and microprocessor performance.

Kunkel and Smith (1986) considered the dependency issue in the context of gate delay using scalar processors as the example. In this work we investigated the efficiency of pipelined processors in relation to the algorithm of multiplication as a means of unifying processor's operations. The selected multiply algorithm for this work is based on classic variant of receiving partial product by one bit at a time with a pipelined serial carry-save adder summing these partial products.

In analysing the circuits of pipelined devices, the followings were investigated multiplication algorithms and their register structures, basic pipelined device structures and the coefficient of hardware expenditure. These analyses are aimed at isolating the most effective algorithm for carrying out MULTIPLY operation shown in Fig.1, assuming that the efficiency of algorithms increases with an:

- increment in inserting control function into the structure of pipelined devices;
- increase in pipelining power;
- increase in ability to lengthen the parallelisation power.

In (Handler, 1977), a classification scheme was proposed for identifying the parallelism degree and pipelining degree built into the hardware of a computer system. Parallel-pipelined processing in this case can be considered at three subsystem levels (Hwang and Briggs, 1988):

- Processor control unit corresponding to one processor or central processing unit (CPU);
- Arithmetic logic unit corresponding to the processing element; and
- Bit-level circuits corresponding to the combinational logic circuitry needed to perform 1-bit operations in the arithmetic logical unit (ALU).

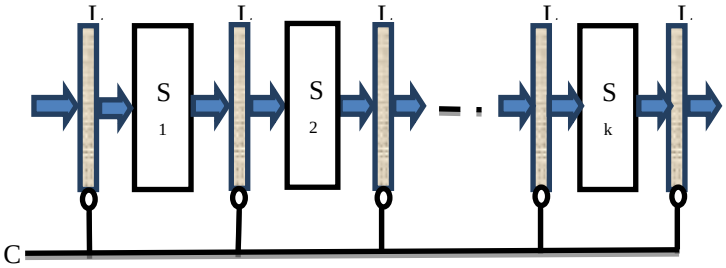
0. Start
1. Form Partial products for
   $a_i b_j \quad 0 \leq i, j \leq n-1$
2. $a_i b_j \ (i = 0, j = 0)$ are sent to the accumulator register.
3. Paired summing of odd terms is carried out
$\sum_{t=n-1}^{S} \sum_{j=S-1}^{n-1} a_i \, b_j + c_S$ and sent to the accumulator register
4. Paired summing of even terms is carried out
$\sum_{i=0}^{S} \sum_{j=S}^{n} a_i b_j + c_S$ and sent to the accumulator register
5. End.

Figure 1: MULTIPLY Algorithm

The number of pipe layer *N(S)* necessary for realising this algorithm is defined by formula (Hwang and Briggs, 1988):

$$N(S) = 3\left|\frac{N(S)-1}{2}\right| + N(S) Mod 2; \quad N(3) = 1 \qquad (1)$$

In designing pipelined integrated devices (PID), the choice of pipe stages and its optimal structure is very important. The choice of pipe layer to a large extent determines the internal structure of pipes and the system of connection amongst pipes (Weaver et al, 2002) Assuming that algorithms realised in PID allows arbitrary division into series of separate computing schemes then depending on the choice of algorithm for one or the other multiplication scheme, the efficiency of PID can change within a wide range. The choice and separation of algorithm for carrying out corresponding operations have to be done taking into consideration the complexity of each layer of PID and general number of layers in as Fig.2.



L – latch; C-control signal; $S_1 \ldots S_k$ – the i^{th} stage

Figure 2. Basic Structure of a Pipelined Processing Module

METHODS AND ANALYSIS

The analysis of how a pipelined processor's time is spent is important to determine the effective work done by the processor. If the total time is *T*, then it can be divided into the time that the processing unit is busy doing useful work - *Tb*, and the time that execution is not busy or held by any of a number of pipeline hazards, *Tnb*. This can also be regarded as the busy and not-busy times, which have been discussed in previous works (Macdougal, 1984; Emma et al, 1989). Typical pipeline holds include branch prediction errors and both decode and execution data dependencies as observed in (Emma and Davidson, 1987).

Generally, when estimating machine performance the estimated execution time, *T* is the most important measure. The impact of performance improvement can be measured in terms of the speedup *S*, expressed as the ratio of the execution time without the improvement (T_{wo}) to the execution time with the introduced improvement (T_w):

$$S = \frac{T_{wo}}{T_w} \qquad (2)$$

For example, if adding a 2MB cache module to a computer system results in lowering the execution time of some benchmark program from 24 seconds to 16 seconds, then the speedup would be 24/16, 1.5, or 50%. An equation to calculate speedup as a direct per cent can be represented as:

$$S = \frac{T_{wo} - T_W}{T_w}\ x\ 100 \qquad (3)$$

To develop a more fine-grained equation for estimating T it is necessary to have more information about the machine's clock period. In this case the total execution time for the program is given by:

$$T = IC\ x\ CPI\ x\ \tau \qquad (4)$$

where , τ …..is the machine clock period

CPI ….number of clock cycles per instruction.

IC … count of the number of instructions executed by the program during its execution

CPI and IC can be expressed either as an average over the instruction set and total count, respectively, or summed over each kind and number of instructions in the instruction set and program. Substituting the latter equation into the former we get:

$$S = \frac{IC_{wo}\ x\ CPI_{wo}\ x\ \tau_{wo} - IC_w\ x\ CPI_w\ x\ \tau_w}{IC_w\ x\ CPI_w\ x\ \tau_w}\ x\ 100 \qquad (5)$$

These equations and others derived from them are useful in computing and estimating the impact of changes in instructions and architecture upon performance. Using these equations we can also determine the coefficient of efficiency of the speedup. In the case of estimating the speedup obtained by replacing a CPU having an average CPI of 5 with another CPU having an average CPI of 3.5, with the clock period increased from 100 ns to 120ns, *S* will be 19%. Thus, without actually running a benchmark program we can estimate the impact of an architectural change upon performance. The choice of layers for PID is determined by the fact that at each time-sequence the next result bit is formed. We are applying this technique in the estimation of hardware efficiency in the design of pipelined integrated devices.

Therefore the number of pipelined integrated circuit (PIC) layer will be determined by the result bits necessary for obtaining the required computational precision. If for the purpose of separate computing step an action connected with the receipt of one next result digit is chosen then all computing processes will be the same. The realization of the same operations in every layer of PID allows the design of PIC with synchronized constant working

cycle which creates a simplified process for their production in the form of large scale integrated devices and very large scale integrated devices.

From the structural viewpoint, increasing the efficiency of PID is connected with adding some special extra registers called pipelined registers into the normal multiplying devices. The maximum number of pipelined registers that can be added is defined by formula

$$L_{max} = 3(Z - 1) \quad (6)$$

where Z = max(m, n);
m, n - bit size of the multiplicand and multiplier

The efficiency or utilisation rate of multiplication operation to a greater extent is defined by the following coefficient:

$$\propto = \frac{t_p}{t_y} \quad (7)$$

where tp – summed delay time on pipelined registers;
ty – pipeline stage execution time.

To determine the coefficient of acceleration of pipelined processing relatively to normal sequential processing firstly we determine total carry time, T_m for carrying out multiply operation in a multiplier and considering that in normal ALU multiplication the time is defined as:

$$T_m = Nt_y \quad (8)$$

where N is the length of the computed expression, but in a pipelined ALU with s-stages it is :

$$T_k = \left(\frac{t_y}{S+1} + t_p S\right)(N + S) \quad (9)$$

The coefficient of efficiency (acceleration) K_y of pipelined processing is then determined as:

$$K_y = \frac{T_k}{T_m} = \frac{\left(\frac{t_y}{S+1} + t_p S\right)(S+N)}{Nt_y} \quad (10)$$

Considering formula (2) then (10) can be expressed as:

$$K_y = \frac{T_k}{T_m} = \frac{\left(\frac{1}{S+1} + \propto S\right)(S+N)}{Nt_y} \quad (11)$$

and assuming that N is infinitely large then (11) can be rewritten as:

$$K_y = \left(\frac{1}{S+1} + \propto S\right) \quad (12)$$

We would like to compute the average number of cycles needed to execute an instruction, and the execution efficiency. If a branch is taken in a five-stage pipeline, then four cycles are needed to flush the pipeline and the branch penalty b is 4. The probability P_b that the branch is taken is .5. When the pipeline is filled and there are no branches, then the average number of cycles per instruction (CPINO_Branch) is 1. The average number of cycles per instruction when there are branches is then:

$$AvgCPI = (1 - P_b)(NoBrCPI) + P_b[P_t(1 + b) + (1 - P_t)(NoBrCPI)] = 1 + bP_bP_t \quad (13)$$

After making substitutions, we have 1.5 cycles. The execution efficiency is the ratio of the cycles per instruction when there are no branches to the cycles per instruction when there are branches. Thus we have execution efficiency = (NoBrCPI) / (AvgCPI) = 1/1.5 = 67%.

The processor runs at 67% of its potential speed as a result of branches, but this is still much better than the five cycles per instruction that might be needed without pipelining. There are techniques for improving the efficiency. As stated above, we know that loops are usually executed more than once, so we can guess that a branch out of a loop will not be taken and be right most of the time. We can also run simulations on the non-loop branches, and get a statistical sampling of which branches are likely to be taken, and then guess the branches accordingly. As explained above,

this approach works best when the pipeline is deep or the clock rate is slow.

No one doubts that there are significant differences between different types of operation run on processors in relation to the computational algorithms (Eludire, 2011). Maynard, et. al., (1984) have explored the pipelined operation differences between multi-user commercial workloads and technical workloads. The differences are associated with branch prediction accuracy, the degree of operating system calls, I/O content and dispatch characteristics (Gee et.al., 1991 and Charney et. al., 1997).

RESULTS AND DISCUSSIONS

We have simulated seven stages of register inclusion and obtained the K_y as a function of pipeline depth for each of them, and determined the optimum pipeline depth for each. In Fig. 2 we show the distribution of these optimum pipeline efficiency for different number of registers introduced into the processing module. The obtained result is shown in Table 1 and plotted in Fig. 3.

From Table 1 and the graphs shown in Figure 3, it can be deduced that when $N \geq 5$ multiply operation with dynamic introduced pipelined processing is effective. Pipelined processing is effective as far as the condition of $t_p \leq \alpha t_y$ is satisfied and from this we can determine α. The number of pipe layers necessary to ensure effective realization of pipelined processing by way of adding extra pipelined registers is determined from the following inequalities:

when $\alpha > 10^{-4}$ up to 8 layers of pipe;

$\alpha \leq 10^{-4}$ not more than 32 layers of pipe.

The following inequalities can be used in selecting an optimal number of pipe layers for achieving a given minimal efficiency T_k:

$0 < N \leq 4 \ldots\ldots 3\ to\ 4\ layers$;

$5 \leq N \leq 8 \ldots\ldots 5\ to\ 7\ layers$;

$N > 8 \ldots\ldots 7\ to\ 8\ layers$.

The choice of specific number of layers would depend on the needed α in the given range. Using a large number of layers in a pipe increases T_k and hardware expenditure which becomes comparable with realization of multiply operations without using pipelined registers shown in Fig.3.

Table 1: Coefficient of efficiency depending on number of pipeline registers introduced

S/N	1	2	4	8	16	32	64
0	4.498	8.996	17.992	35.984	71.968	143.936	287.872
1	4.673	7.010	11.683	21.029	39.722	77.108	151.879
2	5.024	6.698	10.047	16.745	30.142	56.934	110.519
3	5.549	6.937	9.711	15.260	26.359	48.556	92.949
4	6.250	7.500	10.000	15.000	25.000	45.000	85.000
5	7.126	8.314	10.689	15.440	24.941	43.944	81.949
6	8.177	9.345	11.682	16.354	25.700	44.391	81.772
7	9.404	10.579	12.930	17.632	27.035	45.843	83.457
8	10.805	12.006	14.407	19.209	28.814	48.023	86.442
9	12.382	13.620	16.097	21.049	30.955	50.766	90.389
10	14.134	15.419	17.989	23.128	33.408	53.966	95.083
11	16.061	17.400	20.077	25.430	36.138	57.553	100.383
12	18.164	19.561	22.355	27.944	39.122	61.477	106.187
13	20.441	21.901	24.821	30.662	42.342	65.704	112.427
14	22.894	24.420	27.473	33.578	45.788	70.208	119.049
15	25.522	27.117	30.307	36.688	49.449	74.971	126.015
16	28.325	29.991	33.324	39.989	53.318	79.977	133.295
17	31.304	33.043	36.521	43.477	57.390	85.215	140.866
18	34.457	36.271	39.898	47.152	61.660	90.677	148.710
19	37.786	39.675	43.454	51.011	66.126	96.354	156.812
20	41.290	43.256	47.189	55.053	70.783	102.242	165.160
21	44.969	47.013	51.101	59.278	75.630	108.335	173.745
22	48.824	50.946	55.192	63.683	80.665	114.629	182.558
23	52.853	55.055	59.460	68.269	85.886	121.122	191.593
24	57.058	59.340	63.905	73.034	91.293	127.810	200.844
25	61.438	63.801	68.527	77.979	96.883	134.691	210.307
26	65.993	68.437	73.326	83.103	102.656	141.763	219.977
27	70.724	73.249	78.301	88.405	108.611	149.025	229.852
28	75.629	78.237	83.453	93.885	114.748	156.474	239.927
29	80.710	83.400	88.781	99.542	121.065	164.110	250.201
30	85.966	88.739	94.285	105.378	127.562	171.932	260.671
31	91.397	94.253	99.966	111.390	134.240	179.938	271.335
32	97.004	99.943	105.822	117.580	141.096	188.128	282.192

S – number of pipeline stages (layers)

N1..N7 – number of registers introduced at the stages

T_K – quantified coefficient of efficiency

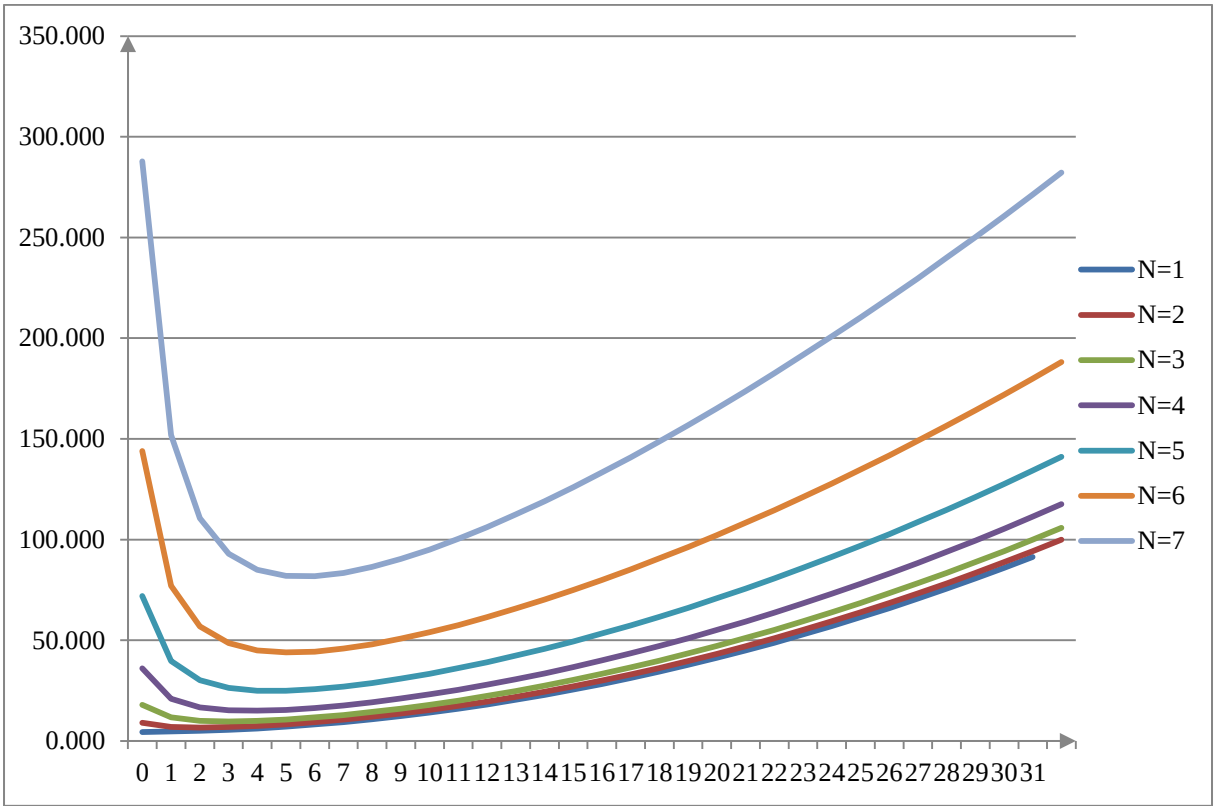


Figure 2: Dependency of Efficiency on the Number of Injected Registers and Pipeline Stages

CONCLUSION

An analysis has been presented on the choice of optimum pipeline registers for a microprocessor in the design of integrated devices. The theoretical position has been tested by simulating a variable depth pipeline model based on a pipelined multiplier, and has been found to be agreeable. It is found that the relationship between increasing the number of pipelined registers in a deeper pipeline to increase throughput, and limiting the number of pipeline stages, results in an optimum pipeline efficiency. That efficiency depends in a way on the detailed microarchitecture of the processor, details of the underlying technology used to build the processor, the algorithm and certain characteristics of the operations run on the processor.

REFERENCES

Agarwal V., Hrishikesh M. S., Keckler S. W. and Burger D. (2000). "Clock Rate versus IPC: The End of the Road for Conventional Microarchitectures", *Proc. of the 27th Annual International Symposium on Computer Architectures*, pp. 248 - 259.

Charney M. J. and Puzak T. R. (1997). "Prefetching and Memory System Behavior of the SPEC95 benchmark Suite" *IBM Journal of Research and Development* **41**, pp. 265 - 286.

Eludire A. A. (2011). Approximation of Mathematical Functions Using Horner's Scheme and Continued Fraction. *Research Journal of Information Technology*, 3(2): 68-71

Emma P., Knight J, Pomerene J, Puzak T, Rechschaffen R (1989). "Simulation and Analysis of a Pipeline Processor", *8th Winter Simulation Conference*, pp. 1047 - 1057.

Emma P. G. and Davidson E. S. (1987). "Characterization of Branch and Data Dependencies in Programs for Evaluating Pipeline Performance", *IEEE Transactions on Computers* **C-36**, pp. 859 -875.

Gee J. D., Hill M. D., Pnevmatikatos D. N, and Smith A. J. (1991). "Cache Performance of the SPEC Benchmark Suite", *Technical Report 1049, Computer Sciences Department, University of Wisconsin*.

Golub Gene H., Ortega James M. (2014) Scientific Computing - An Introduction with Parallel Computing - Academic Press / Google Books, 452 pages.

Handler, W. (1977). The impact of classification schemes on computer architecture". Proc. 1977 Int. Conf. on Parallel Proc., pp. 7-15.

Hartstein, A and Thomas R. Puzak (2002). The Optimum Pipeline Depth for a Microprocessor. ACM SIGARCH Computer Architecture News - Special Issue: Proceedings of the 29th annual international symposium on Computer architecture (ISCA '02) Volume 30 Issue 2, Pg 7-13

Hwang Kai and Briggs Faye A. (1988), Computer Architecture and Parallel Processing, McGraw Hill International Editions

Kunkel S. R. and Smith J. E. (1986). "Optimal pipelining in supercomputers", *Proc. of the 13th Annual International Symposium on Computer Architectures*, pp. 404 - 411.

Macdougal M. H. (1984). "Instruction-Level Program and Processor Modeling", *Computer*, pp. 14 - 24.

Maynard A. M. G., Donnelly C. M., and Olszewski B. R. (1994). "Contrasting Characteristics and cache performance of technical and multi-user commercial workloads", *ASPLOS VI*, pp. 145 -156.

Omondi Amos R. (1999) The Microarchitecture of Pipelined and Superscalar Computers -– Springer Science & Business Media BV.

Pal Ajit (2014). High Performance Computer Architecture Prof. Lecture - 6 Pipelining – Introduction Available at http://textofvideo.nptel.iitm.ac.in/106105033/lec6.pdf

Weaver Chris, Barr Kenneth C., Marsman Eric, Ernst Dan, and Austin Todd (2002) Performance Analysis Using Pipeline Visualization. *2001 IEEE International Symposium on Performance Analysis of Systems and Software. ISPASS.*