A Comparison of Power Consumption in Some CMOS Adder Circuits

D.J. Kinniment*, J.D. Garside+, and B. Gao*

*Electrical and Electronic Engineering Department,
The University, Newcastle upon Tyne, NE1 7RU, UK

+Department Of Computer Science,
The University, Oxford Road, Manchester, M13 9PL, UK

Abstract

Addition is representative of many arithmetic processing operations that must be carried out in portable digital systems, and the speed and power consumption trade-offs in adder hardware are of interest to portable digital system designers.

In this paper we compare static and dynamic circuits, and synchronous and asynchronous architectures for speed, power per add, and transistor count. Three adder circuits chosen for the comparison are: a synchronous static ripple carry adder, a static Manchester carry adder, and an asynchronous dynamic adder. The analysis and simulation results show that both the lowest power and best time-energy product per addition are given by the simple synchronous static adder based on the Manchester carry path.

Introduction

Addition is an operation common in circuits designed for portable equipment, and is typical of the digital processing carried out in computer systems. With the current interest in obtaining high performance together with long battery life, it is useful to compare the energy consumption per addition and the speed of operation of the adder designs commonly in use today.

In CMOS circuits most of the energy consumed is due to switching activity, with the number of nodes in the circuit, the stored energy per node, and the number of switching operations per second all contributing to the total power consumption. Techniques to improve addition speed, such as carry look ahead, increase both the number of nodes in the circuit and the number of transitions per node, and hence increase the energy dissipated. On the other hand asynchronous design techniques may reduce the number of unnecessary switching actions because an operation is requested only when it is required, and because the number of transitions per node can be limited when the

operation takes place. At the same time, a completion signal is generated which allows an operation to terminate when the correct result has been obtained, and, on average, can produce a high speed from a relatively simple circuit.

This paper will compare the speed, and power consumption of three basic adder circuits:

1. A dynamic asynchronous circuit [Gars93].
2. A simple static ripple carry adder.
3. A static circuit based on the Manchester ATLAS adder [Kilb59].

Whilst the asynchronous circuit shows a good performance on average, its energy dissipation per addition is inferior to the other two circuits, and in terms of a combined energy and speed measure, the synchronous static adder based on the Manchester carry path is best. The reasons for this will be discussed.

Low Power

In a CMOS circuit the majority of the energy dissipation is determined by:

$$Energy = \sum_{N} \frac{1}{2} \times C \times V^2 \times No_of_Transitions$$

In this equation there are four factors that can be reduced to achieve low energy per addition, and these are: the total number of nodes, the nodal capacitance, the power supply voltage and the number of transitions on each node.

In this paper we will assume that the process and power supply voltage are fixed, since the circuits compared all have a maximum of three transistors in series with the power rails, and respond in a similar way to power supply variation. We will therefore compare adders only on the basis of the circuit and architectural design factors which affect the power consumption.

Static and Dynamic Logic

In a static CMOS logic design each logical function is implemented twice, once in the n transistor stack, and once in the p transistor stack. Dynamic logic reduces the total number of transistors, and can improve the overall speed by eliminating the p stack and replacing it by a timed precharge transistor. Disadvantages include additional timing complexity introduced by the need to precharge every dynamic node, and frequency management to ensure that nodes are refreshed regularly.

In comparing the low power properties of static and dynamic logic, Chandrakasan et al [Chan92] highlighted a number of areas for consideration. These include:

Spurious transitions: In static logic spurious transitions due to different delay paths in the circuit occur when the function is evaluated, sometimes giving rise to many different values on the nodes during that time. These hazards can contribute to between 9% and 38% of the power consumption of the circuit according to Benini et al. [Beni94]. On the other hand the output nodes in a dynamic circuit may be discharged once and precharged once, or not at all during one evaluation. While the probability of a change in node value can never be greater than one, the number of transistions per node during evaluation is potentially unlimited**.** If the probability of a change in the final value of all the output nodes in a circuit between one evaluation and the next is 0.5, we might expect the static version to dissipate an average of between 0.545 and 0.69 units of energy per node because of the additional spurious transitions, but the dynamic version will require an average of 1 unit since there may be either zero or two transitions per node. It is likely, however, that the energy per node will be less in the dynamic circuit, and the number of spurious transitions on some output nodes of the adder (for example in the sum outputs) will be significantly greater than 38%.

Of particular interest in this analysis is the number of transitions on carry and sum nodes in a static adder whose inputs are fed by random operands, i.e. where the probability of an input change is 0.5. In this case the probability of a net change on the sum and the carry is also 0.5 at the conclusion of an addition. Unfortunately during the addition a carry may be forced to a 0 (inputs both 0) or a 1 (inputs both 1) or equal to the previous carry. In the last case, a transition from the previous carry will be propagated. An analysis of the number of transitions leads to a maximum of 0.75 on any carry and 1.25 on any sum node.

Input Capacitance: The duplication of the logic function in static logic represents a significant extra load on each output node.

Precharge: The major disadvantage of dynamic logic is the cost in energy dissipation of the precharge phase. All output nodes discharged in the previous evaluation must be precharged, an average of 50%, and then during evaluation, 50% of the precharged output nodes are discharged.

Clock gating and clock frequency management: Power consumption in synchronous systems can be reduced by disabling the clock to idle circuits and thereby preventing changes. In static circuits this has no effect on the functionality, since the output nodes are driven statically from the inputs.

Dynamic circuits on the other hand rely on timed signals such as clock to precharge the output nodes.

Synchronous and Asynchronous Circuits

Asynchronous circuit design can reduce power consumption because there is no conventional clock signal distributed to all parts of a system, individual functions are simply invoked when necessary, and a completion signal generated which indicates the availability of the correct data. Thus if there is no need for an operation, in some technologies, e.g. CMOS, the hardware needs to dissipate little energy, and if the operation is required, it can take the minimum time needed rather than a full clock period. There are potential advantages here in both power consumption and performance, but any comparison must include the effects of:

The spacer: To make use of the self timing property of an asynchronous circuit, a completion signal must be derived from data which distinguish between a valid output and an invalid output. At the beginning of the operation the input nodes to the completion circuit are set to values which cannot occur with valid output data. Typically, two rail logic can be used in which both true and complementary outputs represent the slowest signals. If both are set to 0 initially, and then allowed to take up their final complementary values when one rail of every pair becomes 1, the output is complete. This initial state can be called a 'spacer' between two evaluations. The spacer requires that there be some redundancy in the circuit nodes to allow for non-valid outputs as well as all possible valid outputs, that there be more transitions than strictly necessary to move between spacer and valid output, and then back again, and that some time be spent in setting up the spacer before evaluation.

Clock management in a synchronous circuit: Many of the advantages of asynchronous circuits can also be obtained in a synchronous system, for example, clock gating can be used to prevent the clock signal being distributed to functional units where operations are not required, thus eliminating one source of redundant energy dissipation. Additionally, systems can be selected for operation at different clock speeds, allowing for the effects of slow or fast silicon.

The addition time in a system: In an asynchronous system which includes addition as one of its operations, the average time for an addition may not be the same as a typical addition time performed in a system for two reasons. Firstly, the input data may not be random in nature. Garside [Gars93], has shown that the additions performed in the AMULET processor designed by

Furber et al. [Furb93] using micropipelining concepts described by Sutherland [Suth89], had an average maximum carry propagate path almost twice as long as that expected from purely random input data, leading to a longer than expected asynchronous addition time. Secondly, additions which take a long time in a micropipelined stage will cause the following stages to be held up, and similarly additions which require only a short time will be held up waiting for the following stages to become free. The addition time in a system with more than one micropipelined stage will therefore be greater than the average asynchronous addition time for the adder in isolation, and can only approach it if the addition stage is itself a bottleneck.

Fortunately, the requirements for a spacer in an asynchronous system largely coincide with the characteristics of the precharge phase in a dynamic system, allowing a good dynamic logic implementation of an asynchronous adder.

Adder circuits

We have analysed the performance of four different adder circuits in a 2µm CMOS process to show how the factors given above might affect the speed, energy per addition, and circuit area in a system**.** In each case the circuits have been optimised for speed by using p transistors wider than n transistors where necessary. Alternative designs optimised for power consumption at a given speed may give slightly different results but are unlikely to affect the comparison between circuit configurations**.** The circuits are:

1. A synchronous 32 bit static ripple carry adder (RCA) whose basic 1-bit design is shown in Figure 1.

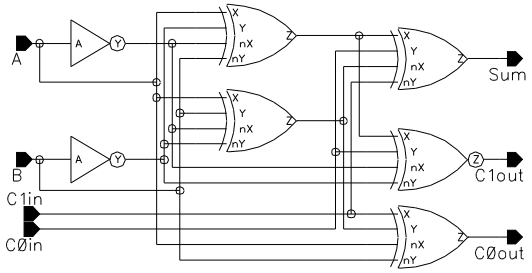

Figure 1. 1-bit RCA adder

This circuit has 9 statically driven nodes per bit including the inputs A and B which also contribute to the power dissipation, and computes complementary carry signals in a worst case situation of one gate delay per bit. An analysis of the number of transitions required for a particular set of 1000 32 bit

additions with randomly chosen operands and using a pure delay with infinitely fast edges to model the gates, shows that with the A and B inputs changing 0.5 times per addition, the carry changed on average 0.72 times, and the sum 1.21 times. This agrees with the analysis given earlier. In practice many of these transitions would merge together because of the finite rise time of the gate outputs, and the energy dissipation associated with them is therefore limited. Nevertheless, the result of the analysis shows that the number of energy dissipating hazards can be very high in some parts of the 32 bit version of this circuit and the overall average at 0.66 is similar to that quoted by Benini [Beni94]. On the other hand, because the average maximum carry path using operands collected from actual computations is significantly longer than that for random data, it is necessary to look at real data when computing energy per addition. Therefore, we have also constructed a list of 1000 32 bit additions with the distribution of average maximum carry paths adjusted to reflect the statistics gathered by Garside [Gars93], and the results from this show the sum changing 1.75 times per addition, and the carry 1.29 times, bringing the overall average up to more than 0.9. Again, in practice this 50% increase in transitions is unlikely to be reflected in a 50% increase in power consumption because of the limited rise times of the circuits, but a significant increase might well be expected.

2. An asynchronous dynamic adder (ADA) based on the design described by Garside [Gars93] is shown in Figure 2. The schematic on the left is the design of the 1-bit adder.

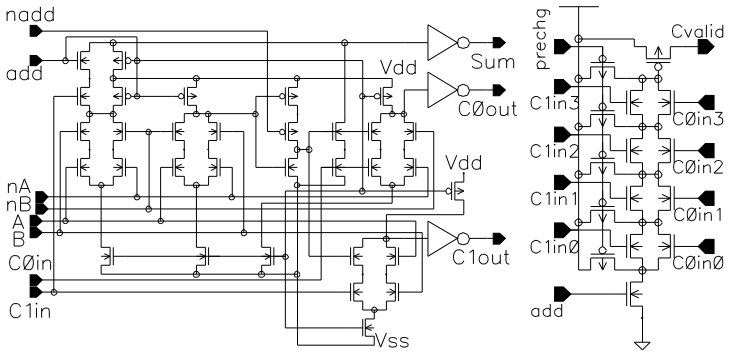

Figure 2. 1-bit ADA adder and a 4-input dynamic AND tree

This asynchronous dynamic adder relies on dual rail propagation of the carry signal. The addition begins when the *add* signal goes high and the precharge released allowing the sum and carry values to be evaluated. The sum

generator applies both XOR and XNOR functions to *A* and *B*, then conditionally raises *Sum* when the dual rail carry signal (*C1out* and *C0out*) arrives. Dual rail carry out signals are generated dynamically by either propagating the carry in, or *A*, subject to whether *A* and *B* are equal. The *Cvalid* signal for every 4 stages of the adder is generated by feeding the logical OR of the dual rail carry out signal into a 4-input dynamic AND gate as shown above on the right hand side in figure 2. Then 8 *Cvalid* signals are ANDed to signal the completion of a 32 bit addition. The precharge buffer tree is not shown in the figure. The precharge phase and the "spacer" which resets the dual rail carry signals after each addition are overlapped under the control of one *add* signal.

There are total of 7 dynamically charged and 7 statically driven nodes per bit in the circuit. If the probability of A and B inputs changing is 0.5 times per addition, the number of transitions on 5 of these dynamic nodes is 1 because there may be either zero or two transitions per node. However for each addition, there are always two transitions on the input node to all the precharge p and n devices and the dynamic nodes in the *Cvalid* AND gate. These can bring the overall average up to 1.29 per node. All the nodes in the precharge buffer tree also have two transitions in each addition.

3. A synchronous dynamic adder (SDA), which is modified from ADA, is shown in Figure 3.

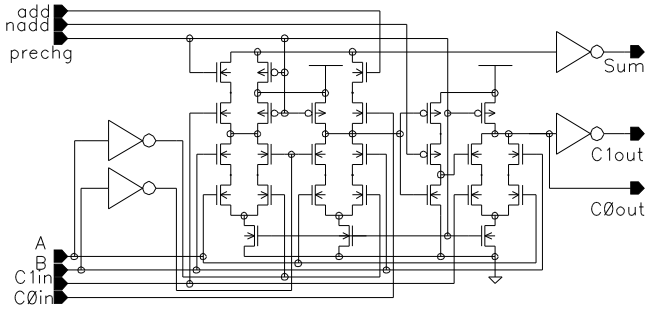

Figure 3. 1-bit SDA adder

A proportion of the energy dissipated in the asynchronous dynamic adder described above is used in generating some functionally redundant signals required to indicate completion. These include half of the dual rail carry path, and the *Cvalid* signal. If the dynamic asynchronous design is modified as shown in figure 3, to retain the dynamic nature of the circuit, but to delay the discharge of the *Sum* node until all the carries have propagated, the resulting

circuit will now have 5 dynamic nodes, and hence a reduced power dissipation. The penalty is that the addition time is now fixed because the *add* signal cannot be raised until after the worst case carry propagation time. Because two transitions on the nodes in the precharge buffer tree and the input node to the precharge transistors are still required for each addition, the power dissipated in each addition can well be expected higher than the static ripple carry adder. However, the speed of this dynamic adder will be improved over the static ripple carry adder.

4. A synchronous 32 bit static adder (ATLAS) based on the Manchester ATLAS adder [Kilb59] is shown in Figure 4. The schematic on the left is an 1-bit carry generation and propagation path, the sum design is on the right.

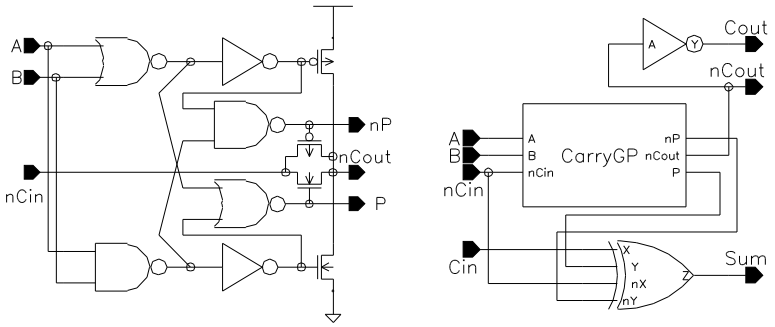

Figure 4. ATLAS: carry path and 1-bit adder schematic

Carry propagation is achieved in this circuit by a CMOS pass transistor switch which is enabled by the condition *A* XOR *B* , and the carry path is forced to a 1 by *A* AND *B*, or a 0 by *A* OR *B*. It is entirely static, and so does not require either a spacer, or an energy intensive precharge, but suffers from the disadvantages of spurious transitions in the same way as other static adder circuits. In the original design, the pass transistors were symmetrical bipolar germanium devices with a very low on resistance, and hence carry propagation was fast. CMOS devices do not have this characteristic, and therefore it is necessary to restandardize the carry signal with an inverter made from double strength transistors every two carry stages with inverse logic used for the subsequent two stages in the carry path, in order to provide an adequate drive. There are 9 statically driven nodes per bit in this circuit, but the circuit complexity is lower than the static ripple carry adder (less load) and the carry propagation speed is faster than the ripple carry adder. Because of its simplicity, and relatively fast carry path, its power dissipation characteristics are relatively good.

Power Consumption Estimation

With random input operands, we analysed the probability of transition on each node in 1-bit circuit for each of the four adders to obtain an overall average number of transitions per node per addition. To simplify the analysis, the total node capacitance including input/gate and output/diffusion, is associated with inputs only**.** One n device and one p device load are then normalized to 1 unit and 2 units of load respectively. Other simulations we have done with Hspice(93A) on a MIETEC 2µm double poly, double metal, n-well CMOS process suggest a typical power dissipation of 0.08pJ per load unit per transition so that we compute a theoretical figure for power dissipation. Table 1 lists for comparison the total number of nodes per bit, the average number of transitions per node per addition and the estimated power consumption per 32 bit addition for the four adders. The estimated power is calculated by adding the products of the average node transition and the corresponding normalized load units in 1-bit circuit which is then multiplied by 0.08 and 32 for 32 bit versions of each of the four adders.

Table 1

Adder Circuit Analysis	Total Nodes per bit	Average transitions per node per add	Estimated energy per 32bit add(**pJ**)
1. RCA	9	0.68	90.24
2. ADA	14	1.29	179.2
3. SDA	12	1.2	151.04
4. ATLAS	9	0.53	55.68

Simulation Results

32 bit versions of each of the four adder circuits were simulated with two lists of operands, one generated form an unbiased random number generator, and one consisting of operands chosen to give the same A inputs as the previous list, but with B inputs selected to match the statistics quoted by Garside in [Gars93]. Measurements were also based on Hspice(93A) simulations of the same MIETEC 2µm CMOS process using typical parameters at 5V. In Table 2, the average energy dissipated per addition for 4 different 32 bit adders is given, and it can be seen that the longer average carry paths expected from the biased data gives a slightly greater energy consumption in the static adders as expected because of the larger number of transitions in the carry path, and the sum outputs. In the dynamic circuits, the difference between random and biased input lists is less marked, but still exists because there are more transitions on the XOR node for the biased set. In the dynamic adders, we estimate that the precharge fan out drivers and the

charge/discharge of the gate capacitance associated with precharge transistors account for approximately 40% of the total energy of the self timed adder.

Table 2

Energy per 32bit addition (pJ)	Random Operands	Biased Operands	Transistors per bit
1. RCA	75.05	77.13	44
2. ADA	205.70	207.42	45
3. SDA	158.70	161.14	34
4. ATLAS	65.50	66.54	34

The simulated power dissipations per addition for the four adder circuits are in line with the estimated power consumptions given in Table 1 except that the static ripple carry adder does not consume as much power as we estimated from probability of transitions. This is because the hazards in the static ripple carry adder might not contribute to energy dissipation in proportion to their numbers in the actual circuit.

The maximum addition time has been measured from Hspice simulation results as the time taken from presentation of the A and B operands to the time of arrival of C_{32} in the worst case. For the self timed adder, a typical addition time has also been calculated by taking the average completion time for the biased set of operands. For the modified dynamic adder, we assume that the add signal is raised at the earliest possible time after evaluation of all the carries.

Table 3

Time per 32 bit addition	Max. Addition Time(**nS**)	Average Typical Addition Time(**nS**)	Cycle Time (**nS**)	Energy×Time
1. RCA	23.08	23.08	30.004	2314.12
2. ADA	16.85	8.076	9.826	2038.14
3. SDA	15.1	15.1	21.38	3445.17
4. ATLAS	16.97	16.97	22.061	1468.02

In a system, it would be necessary to allow some tolerance on the addition time in order to obtain reliable operation over a range of temperatures and supply voltages, though variations in the process parameters could to some extent be accommodated by selecting circuits for different clock speeds. We have therefore added 30% to the add times for the synchronous adders, to obtain a notional cycle time. It will also be necessary to add the precharge time of 1.75 nS to the dynamic adders since this must also be fitted within the cycle time. The results are presented in Table 3.

Finally, there are several measures which could be used to compare the circuits in a low power application. These include the number of operations obtained from a single battery charge, in which case only the energy per add is important, or some measure which also includes speed or silicon area and regularity may be used. We have chosen to compare adder circuits which are comparatively simple and regular in layout. If performance is the only objective, it is fairly easy to show that more complex adders, such as the conditional sum adder described by Sklansky [Skla60], can deliver a faster result, but consume more energy, since more nodes are required in the circuit, with at least the same number of total transitions as the ripple carry adder. Here we have chosen to give the energy - time product as a measure since it reflects the desire to maintain performance together with low energy consumption.

Conclusions

The analysis and simulation results clearly show that the precharge phase of dynamic logic consumes a considerable amount of energy, and that the advantages of eliminating transition hazards gained by the dynamic, self timed adder are outweighed by the increase in the number of transitions per node from an average of about 0.6 to 1.29. In fact, it is likely that the hazards do not contribute to energy dissipation in proportion to their numbers, since many of them will be comparable to the circuit rise times, and hence may never reach the threshold voltage of input gates. In the simulation results, the dynamic self-timed adder is clearly the fastest, and since addition power is often a very small proportion of the total system power, may well be chosen on those grounds alone. The modified dynamic version is of interest, because it shows that a synchronous version using dynamic logic can also be produced with an adequate performance and reduced power dissipation, but in terms of absolute energy consumption per addition, area, and regularity, the ATLAS adder would be the best choice which also offers an adequate clock speed, and the best speed - power compromise.

References

[Beni94] Benini, L., Favalli, M., and Ricco, B., "Analysis of Hazard Contributions to Power Dissipation in CMOS ICs." 1994 International Workshop on Low Power Design, NAPA Valley, April 1994

[Chan92] Chandrakasan, A.P., Sheng, S., and Broderson, R.W., "Low Power CMOS Digital Design." IEEE Journal of Solid State Circuits, Vol 27-4 April 1992.

[Farn94] Farnsworth, C., Edwards, D.A., and Sikand, S.S., "Utilising Dynamic Logic for Low Power Consumption in Asynchronous Circuits." Proceedings, Symposium on Advanced Research in Asynchronous Ciruits and Systems", Salt Lake City, pp 186 - 194, Nov 1994

[Furb94] Furber, S.B, Day, P, Garside, J.D, Paver, N.C, and Woods, J.V, "AMULET1: A Micropipelined ARM", IEEE CompCon 94, San Francisco, March 1994.

[Gars93] Garside, J.D, "A CMOS VLSI Implementation of an Asynchronous ALU." Proceedings of the IFIP Conference on Asynchronous Design Methodologies, Manchester, UK, 1993.

[Kilb59] Kilburn, T., Edwards, D.B.G., and Aspinall, D., "Parallel addition in Digital Computers: A New Fast "Carry" Circuit" IEE Proc 106, Pt B 464-466

[Skla60] Sklansky, J, "Conditional-Sum Addition Logic.", IRE Trans on Electronic Computers, EC-9; pp 226-231.

[Suth89] Sutherland, I.E, "Micropipelines", Communications of the ACM, 32(6): pp720-738, January 1989.