A Novel Area-Efficient Binary Adder

S. B. Furber and J. Liu†
Department of Computer Science, The University of Manchester,
Oxford Road, Manchester M13 9PL, UK.
sfurber@cs.man.ac.uk
†now with Intel Corporation, Austin, Texas, USA

Abstract

A novel circuit for binary addition based on a parallel-prefix carry structure is presented. This circuit uses a recoding of the conventional carry kill and generate terms to yield a number of improvements over previous designs. In particular, a single circuit produces both the carry signals and the Sum, Sum + 1 data that is required for a carry selection circuit, supporting a range of possible implementations all of which have high performance, regular layout and good area-efficiency. The simple design also leads to good power-efficiency.

Binary adders based on this technique have been used in the ARM9TDMI, the ARM Piccolo DSP coprocessor, and the AMULET3 asynchronous ARM processor.

1. Introduction

The AMULET group in the Department of Computer Science at the University of Manchester, U.K., has spent a decade researching the commercial potential of asynchronous design techniques. The focus of this work has been the design of asynchronous implementations of the ARM 32-bit RISC architecture [1].

AMULET1 used a ripple-carry adder but exploited its asynchronous timing environment to deliver good average performance, having a variable delay that depends on the length of the longest carry propagation path in the current addition [2]. AMULET2 accelerated this scheme using a 4-bit carry look-ahead adder [3]. The most recent design, AMULET3 [4], required a significantly higher performance than its predecessors and this, in turn, led to a requirement for a higher performance binary adder [5].

1.1. Binary addition

Binary addition circuits are important in many applications and much has been written regarding their optimization for performance and area [6] and, more recently, power consumption. Ultimately the problem boils down to that of generating the carry signals efficiently. Since the carry out from the top bit of the adder is a function of every input bit (including, frequently, a carry into the least significant bit) there is an irreducible fan-in problem to be addressed. The carry out from a 32-bit adder is a boolean function of 65 input variables.

With such a large fan-in, and given the fan-in restrictions of CMOS circuits, carry generation circuits are always compound designs comprising multiple stages. The carry logic equations have the properties of associativity and idempotency [7], so solutions can be found from highly sequential ripple-carry circuits to maximally parallel-prefix tree circuits [8].

The objective of binary adder design is to identify a carry generation technique that integrates well into the complete adder (or ALU) and that has high regularity (particularly when used in a custom datapath, for example in a high-performance microprocessor), appropriate performance, low power and uses minimum area.

The binary adder circuit described in the remainder of this paper meets these requirements and has been used, in various forms, in the AMULET3 asynchronous ARM-compatible processor [4], ARM9 [9] and the ARM Piccolo DSP coprocessor.

2. Adder circuit

The carry generation circuit uses the carry generate (G) and kill (K) terms, computed using a parallel-prefix tree [8].

The generate and kill terms can be combined according to the formula:

$$(G,K) \bullet (G',K') = (G + \overline{K} \cdot G', K + \overline{G} \cdot K') \qquad \text{(Eq. 1)}$$

However, a simpler circuit is produced if the kill term is propagated in its inverse form. (Alternatively, the generate term may be inverted with similar effect.) Then we can rewrite equation 1 as:

$$(G,\overline{K}) \bullet (G',\overline{K}') = (\overline{K} \cdot (G + G'), \overline{K} \cdot (G + \overline{K}')) \qquad \text{(Eq. 2)}$$

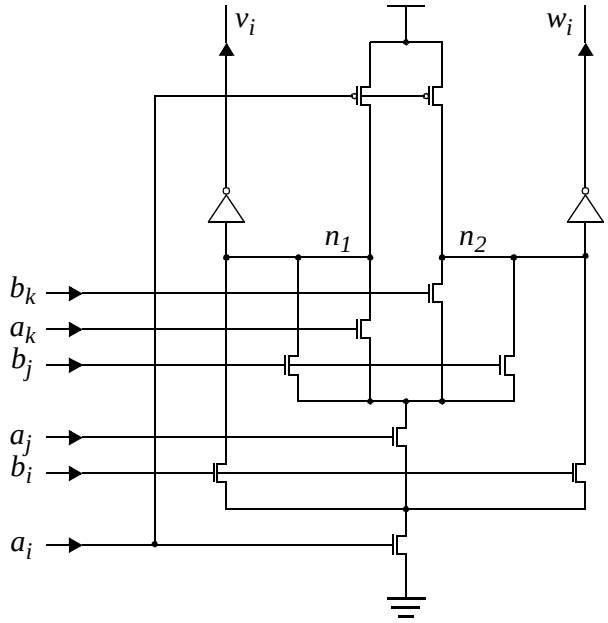

Figure 1. Dynamic 3-input carry circuit.

Note that equation 2 strictly enforces the relation $G \Rightarrow \overline{K}$ at all levels. Some other parallel-prefix schemes operate on the basis that if G then $\overline{K}$ is 'don't care', but this will prevent the use of the parallel-prefix tree for generating group *Sum* and *Sum + 1* outputs as used in the carry-select scheme later.

The implementation of equation 2 as a CMOS circuit is particularly efficient due to the symmetry of the pair of equations and the commonality of factors in the two terms.

2.1. Circuit fan-in and fan-out

Although using equation 2 to combine two generate and kill terms is feasible, it is possible to combine more than two terms in a single circuit [10]. Generating the carries for an N-bit adder using 2-input carry circuits requires a logic depth of $\log_2 N$. The use of 3-input circuits gives a depth of $\log_3 N$, and so on. Higher fan-in circuits are slower in CMOS, and optimization requires an analysis of the trade-off between logic depth and individual circuit speed.

An analysis of the principles of circuit fan-in suggests that for a constant fan-out circuit with a fan-in of n and word-width N the delay varies as $n\log_n N$. The optimum value of n is *e*, the natural logarithm base, and the optimum integer value is 3. However, when implemented as a tree the present circuit has a fan-out equal to its fan-in, so the delay varies as $n^2\log_n N$. The optimum value of n is now less than 2, and the optimum integer value is 2. If inverter buffers are included, these isolate the fan-in and fan-out and the delay is again $n\log_n N$ with an optimum integer solution of n=3.

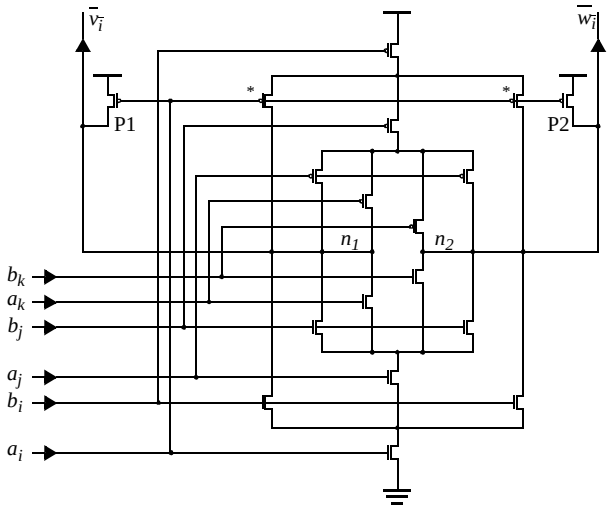

Figure 2. AMULET3 3-input carry circuit.

Additional factors may come into play, for example the bit pitch selected for the full-custom datapath in AMULET3 provides sufficient space for a 3-input carry circuit; a 2-input circuit could not use the available area as efficiently, and a 4-input circuit would not fit. It was therefore decided to base the design upon a 3-input carry circuit.

2.2. Circuit implementation

A 3-input dynamic implementation of equation 2 is shown in figure 1. The *a* inputs represent 'not kill' terms, the *b* inputs 'generate' terms. The v_i output is the group 'not kill'; the w_i output is the group 'generate'. The transistor circuit is symmetrical and exploits the shared terms fully to minimize circuit complexity. The circuit is designed in a self-precharging form, so it assumes that the inputs are all zero before the circuit is activated. (In fact, only the a_i input really needs to be zero provided the other inputs do not change after a_i has risen.) Nodes n1 and n2 are precharged when a_i is low and then selectively discharged during evaluation. The inverters increase the output drive and ensure that the output transitions are suitable for driving subsequent stages in the carry computation.

The dynamic circuit is simple and fast but has poor noise immunity. A fully static version of the circuit can be designed, but the inclusion of correctly ratioed p-transistors will slow the circuit significantly as they represent 2 to 3 times the capacitive load of the n transistors. To avoid this problem the 3-input carry circuit used in AMULET3 is neither fully static nor fully dynamic; it is as shown in figure 2 (with the output buffers omitted). This circuit combines the speed of a dynamic circuit with the noise-immunity of a static circuit. The fully static circuit is implemented, but the

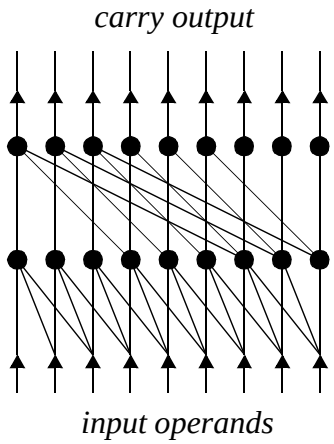

Figure 3. 9-bit carry computation.

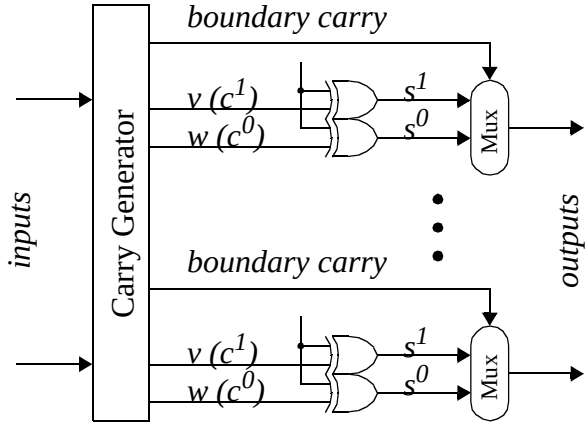

Figure 4. Carry-select scheme.

p transistors are minimum size rather than being ratioed. Additional larger p transistors (P1 and P2) are used to accelerate the precharge process between evaluations, and the evaluation process employs selective discharge just as the dynamic circuit does.

(The p transistors marked '*' in figure 2 are redundant as their function is replaced by P1 and P2, so they may safely be omitted. They have been included here to clarify the base form of the static circuit.)

The 3-input carry circuit can be cascaded into a carry computation tree as shown in figure 3. The black dots in the figure represent 3-input carry circuits. Two layers of 3-input carry circuits can complete the carry computation for a 9-bit addition.

2.3. Carry-select adder

It is possible to merge the proposed parallel-prefix carry tree with a carry select scheme [3] as illustrated in figure 4. An interpretation of the *v* and *w* outputs is that *v* represents

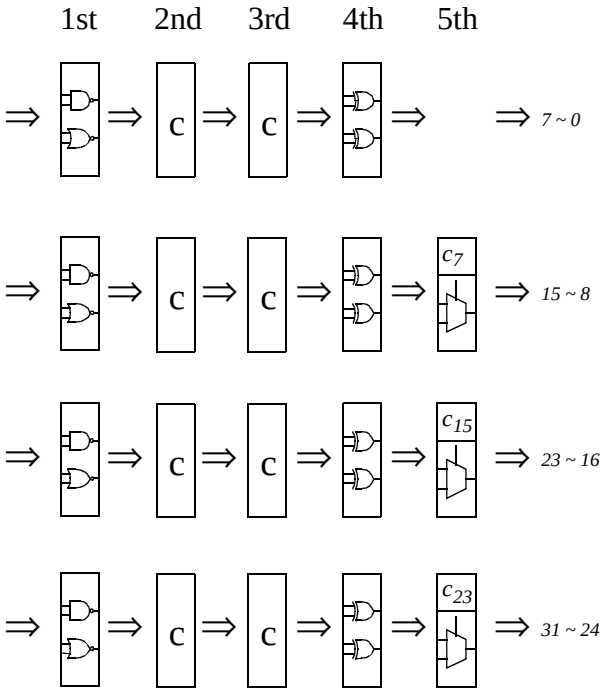

Figure 5. AMULET3 adder organization.

the carry out of a block if the carry in is 1 and *w* represents the carry out if the carry in is 0. The *Sum* and *Sum + 1* can therefore be formed and the input carry used to select the appropriate answer.

Note that separate group adders are not required since the group carries are produced by the same carry generator, exploiting the idempotency of the carry logic.

3. AMULET3 adder

The adder circuit used within the AMULET3 ALU is based upon the techniques described above. The organization of the adder is shown in figure 5.

In the 1st logic layer the inputs are converted into *G* and $\overline{K}$ form using AND and OR gates respectively. These gates have enables, forcing their outputs to zero to precharge the following carry circuits.

There are then two layers of 3-input carry circuits, ('c' in figure 5) together resolving the carry information across all groups of 9 bits. In the 4th logic layer the *Sum* and *Sum + 1* results are formed across groups of 8 bits, and two additional carry circuits (not shown in figure 5) are used to generate c_{15} and c_{23} for the final carry select layer.

3.1. Physical layout

The technology on which the AMULET3 adder is based is a 0.35 μm triple metal CMOS process. The minimum

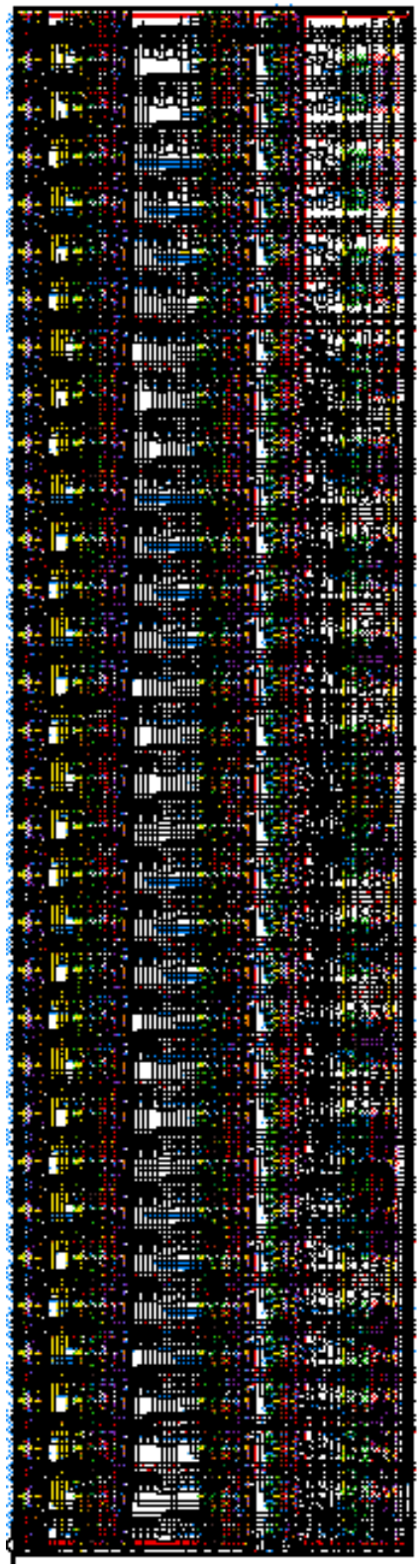

Figure 6. AMULET3 adder layout.

drawn width is 0.4 μm.

The layout of the adder uses a full-custom style for the datapath. The datapath bit pitch is 82 λ. Data is routed horizontally in metal 3; control signals are routed vertically in metal 2. Metal 1 is used for local interconnections within the cells. The global power rails use metal 1 and metal 3, the local power rails use metal 2.

The layout of the AMULET3 adder is illustrated in figure 6. The regularity and compactness of the resulting layout are clearly visible.

3.2. Performance

The adder has a typical delay for a 32-bit addition of 1.4 ns with a worst-case delay of 1.8 ns. It consumes 80 μW per MHz with random data.

The silicon area of the adder datapath is 686 λ × 2624 λ (137.2 × 524.8 μm^2 in 0.35 μm technology).

4. AMULET3 and DRACO

The adder circuitry described above is used in AMULET3, an asynchronous ARM-compatible processor employed in the DRACO chip. First DRACO silicon was received in September 2000 and is highly functional.

The DRACO (DECT Radio Communications Controller) chip is a telecommunications controller intended for ISDN (*Integrated Services Digital Network*) DECT (*Digital European Cordless Telephone*) base station applications. The chip area is divided equally between the AMULET3H asynchronous processing subsystem and a synchronous telecommunications peripheral subsystem. The layout of the chip is shown in figure 7, where the position of the adder within the AMULET3 processor core is indicated.

DRACO's asynchronous processing subsystem is based around the MARBLE self-timed on-chip bus [11]. This bus is a full functionality multi-master on-chip interconnect with a central arbiter and address decoder. It supports split transactions and, in its current manifestation, can perform up to 83 million 32-bit data transfers per second.

The processor core is a 120 MIPS AMULET3 32-bit core with on-chip debug hardware support. 8 Kbytes of dual-port high-speed RAM are connected to the processor's instruction and data memory interfaces and both memory ports then connect to the MARBLE bus. The bus also hosts a 32-channel DMA controller (synthesized using Balsa, an asynchronous sythesis tool [12]), 16 Kbytes of ROM and an external memory interface. The programmable external memory interface supports the direct connection of SRAM, DRAM and flash memory.

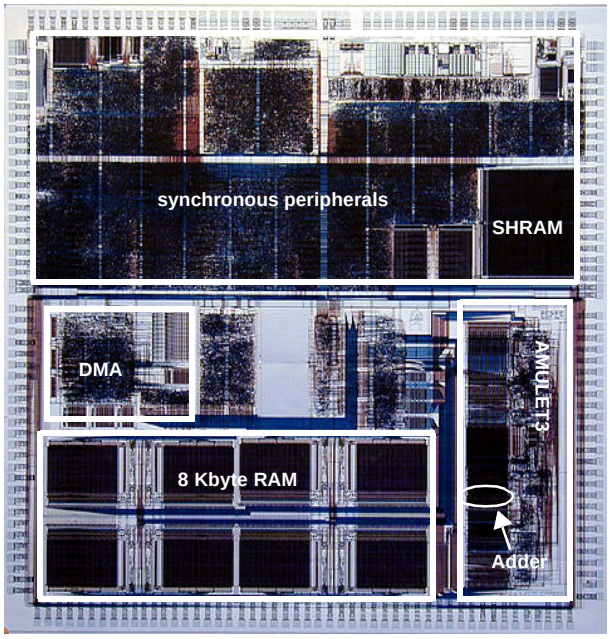

Figure 7. The DRACO chip

5. Conclusions

A hybrid carry scheme based on a fan-in 3 parallel-prefix carry tree and a final carry-select stage has been successfully employed in AMULET3 and implemented in silicon within the DRACO chip. The adder design gives good performance and dense, regular layout.

Related carry circuits have also been used in ARM9 [9] and the ARM Piccolo DSP coprocessor [1].

As with other parallel-prefix schemes, a wide range of area and performance options are available through selective exploitation of the associativity and idempotency of the carry generation logic.

6. Acknowledgments

The development of AMULET3 was supported primarily within the EU-funded OMI-DE2 and OMI-ATOM projects, and authors are grateful to the European Commission for their continuing support for this work. ARM Limited coordinated these projects; their support, and that of the other project partners, is also acknowledged.

Aspects of the work have benefited from support from the UK government through the EPSRC.

The VLSI design work has leant heavily on CAD tools from Compass Design Automation (now part of Avant!) and EPIC Design Technology, Inc. (now part of Synopsis).

7. References

[1] S.B. Furber, *ARM System-on-Chip Architecture*, Addison Wesley Longman, 2000. ISBN 0-201-67519-6

[2] J.D. Garside, “A CMOS VLSI Implementation of an Asynchronous ALU”, *IFIP Working Conference on Asynchronous Design Methodologies*, April 1993. Ed. Furber, S. B. and Edwards, M. D. Pub. North Holland.

[3] D. Goldberg, “Computer Arithmetic”, Appendix A of: J.L. Hennessy and D.A. Patterson, “Computer Architecture: A Quantitative Approach” (2nd edition), Morgan Kaufmann, San Francisco, 1996.

[4] S.B. Furber, D.A. Edwards and J.D. Garside, “AMULET3: a 100 MIPS Asynchronous Embedded Processor”, *Proc. ICCD 2000*, Austin, Texas, September 2000.

[5] J. Liu, *Arithmetic and Control Components for an Asynchronous System*, PhD thesis, The Universioty of Manchester, 1997.

[6] B.W.Y. Wei and C.D. Thompson, “Area-Time Optimal Adder Design”, *IEEE Trans.*, C-39(5), May 1990, pp. 666-675.

[7] T. Lynch and E.E. Swartzlander Jr, “A Spanning Tree Carry Lookahead Adder”, *IEEE Trans.*, C-41(8), August 1992, pp. 931-939.

[8] R.P. Brent and H.T. Kung, “A Regular Layout for Parallel Adders”, *IEEE Trans.*, C-31(3), March 1982, pp. 260-264.

[9] S. Segars, “The ARM9 Family - High Performance Microprocessors for Embedded Applications”, *Proc. ICCD’98*, Austin, October 1998, pp. 230-235.

[10] N.T. Quach and M.J. Flynn, “High-Speed Addition in CMOS”, *IEEE Trans.*, C-41(12), December 1992, pp. 1612-1615.

[11] W.J. Bainbridge and S.B. Furber, “Asynchronous Macrocell Interconnect using MARBLE”, *Proc. Async’98*, San Diego, April 1998.

[12] A. Bardsley and D.A. Edwards, “Compiling the Language Balsa to Delay Insensitive Hardware”, *Proc. CHDL’97*, Toledo, April 1997. Published in Kloos, C.D. and Cerny, E. (eds.) *Hardware Descriptions Languages and their Applications*, IFIP & Chapman Hall, ISBN 01412 78810 1, 1997 pp. 89-91.