VLSI DESIGN
2001, Vol. 12, No. 3, pp. 349–363
Reprints available directly from the publisher
Photocopying permitted by license only

Exploiting Data-dependencies in Ultra Low-power DSP Arithmetic*

V. A. BARTLETT[a,†] and E. GRASS[b,‡]

[a]*University of Westminster, London, England;* [b]*IHP-GmbH, Frankfurt (Oder), Germany*

(Received 20 June 2000; In final form 3 August 2000)

Strategies for the design of ultra low power multipliers and multiplier-accumulators are reported. These are optimized for asynchronous applications being able to take advantage of data-dependent computation times. Nevertheless, the low power consumption can be obtained in both synchronous and asynchronous environments. Central to the energy efficiency is a dynamic-logic technique termed Conditional Evaluation which is able to exploit redundancies within the carry-save array and deliver energy consumption which is also heavily data-dependent.

Energy efficient adaptations for handling two's complement operands are introduced. Area overheads of the proposed designs are estimated and transistor level simulation results of signed and unsigned multipliers as well as a signed multiplier-accumulator are given.

Normalized comparisons with other designs show our approach to use less energy than other published multipliers.

Keywords: Low-power; Array multiplier; DSP; Asynchronous systems; Data-dependent computation

1. INTRODUCTION

In many recently emerging portable applications of DSP circuits, maximizing battery life can be of paramount importance. In recent years, this has resulted in considerable research effort being directed at the development of VLSI design techniques for circuits with reduced power consumption. Such techniques span the design hierarchy from the algorithmic level down to process technology [1]. This paper addresses power minimization at the logic and circuit levels through reductions in circuit activity and switched capacitance.

Asynchronous circuits have also been the subject of a growing body of research. Amongst

*This work was supported by a grant from the U.K. Engineering and Physical Sciences Research Council, EPSRC.

†Address for correspondence: Dept. of Electronic Systems, University of Westminster, 115 New Cavendish Street, London W1W 6UW, UK. Tel.: +44 (0) 20 7911 5146, Fax: +44 (0) 20 7580 4319, e-mail: v.bartlett@westminster.ac.uk

‡Tel.: +49 335 5625 731, Fax: +49 335 5625 671, e-mail: grass@ihp-microelectronics.com

their attractions is the potential to achieve low-power operation – attributable largely to the absence of a global clock and economies of circuitry which become possible. Another of their properties, which has attracted much attention, is the potential to exploit data-dependent delays. This gives rise, in principle, to a computational throughput which is average-case limited rather than worst-case limited.

Although relatively few asynchronous VLSI designs have been reported, in many cases data-dependent delays have been realized at the expense of low-power due to the adoption of differential logic styles such as DCVSL [2]. Whilst such styles readily facilitate Completion-Detection (CD), they are known to offer poor energy efficiency [1].

This paper reports strategies for the design of the building blocks of DSP computation – namely multipliers and multiplier-accumulators – which simultaneously achieve both ultra low-power operation and data-dependent delays. This is achieved by a *synergy* of architecture and circuit-level implementation, neither of which alone, is responsible for the low energy consumption of the circuits.

The architecture adopted for the basic multiplication operation is based on the carry-save array (CSA) proposed in [3] which exploits data-dependent delays. This architecture has the property that much of its logic becomes redundant in proportion to the number of zeros in the multiplier operand.

An issue not investigated in [3], however, is the potential for power reduction offered by this redundancy which can be exploited by inhibiting circuit activity within the redundant logic, as a function of input data. An implementation based on a hybrid static/dynamic logic CMOS circuit style lends itself readily to such exploitation. The feasibility of such an implementation was demonstrated by the authors in the design of an unsigned multiplier [4] using a technique we term 'Conditional Evaluation' illustrating both low-power operation as well as other benefits in terms of low device-count and high-regularity.

In this paper we extend that work by demonstrating that the synergy of architecture and implementation can be equally well applied to the design of signed multipliers and multiplier-accumulators to similar advantage.

It should be noted that whilst the data-dependency in terms of propagation delay is best exploited within an asynchronous framework, the high energy-efficiency can be obtained in both synchronous and asynchronous applications.

Section 2 describes background material underpinning the main design considerations. Section 3 describes the data-dependent carry-save architecture and its application to multiplier and Concurrent Multiplier-Accumulator (CMAC) designs. Section 4 deals with adaptations to allow operation with 2's complement operands. Implementation and timing issues are central to obtaining the maximum energy savings of the proposed approach – these are discussed in Section 5. Simulation results and comparisons with other designs are presented in Section 6. Finally, conclusions are presented in Section 7.

2. BACKGROUND

In this paper, the goal of ultra low-power multiplication and multiply-accumulation has been achieved by the application of a number of different techniques, which are discussed below.

2.1. Carry-save Array Multiplication

Traditional carry-save array (CSA) multipliers comprise, in principle, an array of gated full-adders cells. In any single row of such cells, the 'gates' perform the AND of the multiplicand with a single bit of the multiplier creating a 'bit-product'. The sum of all bit-products emerges from the last row of the array as Sum and Carry vectors which must be added to produce the final result – an operation involving full carry-propagation. This is carried out in a 'vector-merging' or Carry-Resolution Adder (CRA).

The terminology used in this paper is as follows. An n-bit multiplicand, MD, whose MSB is MD_{n-1} is multiplied by an m-bit multiplier, MR, whose MSB is MR_{m-1}. The product of one bit of MR with MD produces a bit-product of n-bits. The kth row of the array adds the incoming partial product to its bit-product thereby producing the kth partial product.

The simple carry-save array is one of several approaches commonly used in multipliers to perform the summation of bit-products. Also widely used are tree structures, including those using 4 : 2 counters, and modified-Booth encoding which is applicable to both carry-save arrays and tree structures. Such strategies are known to offer improved worst-case latency over the simple carry-save array. However, as well as having less regularity, which complicates VLSI implementation, their architectures make the application of the energy saving conditional-evaluation technique more complex and less advantageous. Their use here, particularly in view of the attendant overheads, is not clearly justified since the primary design goal is low power.

2.2. Concurrent Multiply-Accumulation

In DSP algorithms, multiplication is very often followed by accumulation and time to perform a multiply-accumulate operation is often quoted as a performance measure of DSP hardware. Most commonly, multiply and addition operations are carried out separately in two cascaded hardware structures. Considerable benefits can, however, be obtained by carrying out the multiply and addition functions concurrently in the same structure, making use of the unused inputs around the periphery of a non-minimized CSA. An implementation of such a Concurrent Multiply-Accumulate (CMAC) structure was described in [5]. Since this approach exploits the otherwise unused inputs at the periphery of the CSA, the overall gate cost is considerably less than would be required to implement the addition separately. In comparison with conventional multiply-accumulate structures a reduction in area of 20% has been reported [5]. Furthermore a reduction in latency of 50% (in a synchronous environment) was obtained.

2.3. Dynamic Logic

Use of dynamic logic for low-power operation inherently offers some advantages. In particular since each output can undergo, at most, one transition per evaluation, spurious transitions, which in static logic multipliers can account for as much as 50% of the energy consumption [6], are eliminated. Although spurious transitions can be addressed by other static-logic methods such as delay balancing [7], dynamic logic has the additional advantage of considerably reduced input capacitances due to the absence of a complementary logic tree.

On the other hand, in comparison with static logic, dynamic CMOS circuits have two drawbacks which tend to offset the energy advantages. The first of these is the need to charge and discharge the precharge/evaluate lines. This normally takes place once per cycle although with the conditional-evaluation technique reported in this paper, on average it happens less frequently. The second is the increased probability of output activity which results from the fact that the precharge voltage precedes each valid output voltage. Whilst for some combinational functions this puts dynamic logic at a significant disadvantage [1], for others the increase in activity is small.

In the implementations reported here the energy benefits considerably outweigh the drawbacks. Indeed, the fact that evaluation of dynamic logic can be inhibited by a single input recommends it naturally to activity reducing schemes.

To avoid the race problem, dynamic-logic carry-save arrays usually employ differential circuitry [2, 3] despite its relatively poor energy efficiency [1]. Alternatively, the power benefits of single-ended logic can be obtained by using self-timing to avoid the race [8] and this was the approach adopted here. According to our own investigations, the technique has the additional advantage that

charge-sharing problems, which with conventional dynamic logic can lead to increased transistor count, are eliminated.

2.4. Low-power Through Data-dependent Activity Reduction

Several techniques have been proposed for achieving low power by inhibiting circuit activity in combinational logic as a function of input data [6, 9, 10]. In general, these methods involve insertion of additional logic in the critical path, incurring some energy and delay penalty. The approach reported here differs from these methods. For example, the transformation of the standard carry-save-array architecture into that proposed in [3] entails the production of several additional outputs. Also, inhibition of activity in the redundant logic can be achieved with no delay penalty.

3. DATA-DEPENDENT ARCHITECTURES

3.1. Data-dependent Carry-save Array Multiplier Architecture

In the CSA multiplier architecture of [3] the array of AND gates normally used for bit-product generation is not needed. Instead, each cell in the array is fitted with a pair of multiplexers (MUXs) which, when a row's bit-product is zero, pass on unaltered, the Partial-Product (PP) and Carry-In (CI) inputs from the row above, to the row below. For rows whose bit-product is non-zero, the MUXs select the adder outputs to feed to the row below. Since the propagation delay through the bypass path is less than that through the adders, values of MR which are heavily populated with zeros produce less delay through the array than those heavily populated with ones. An additional advantage of this architecture is its good testability due to the easy internal access provided by the MUXs. A comparison of the standard and data-dependent architectures is shown in Figure 1.

3.2. Concurrent Multiply-Accumulate Structure

As outlined above, the CMAC structure allows the CSA's multiplication function to be augmented by accumulation and it offers a number of advantages over an implementation in which the two functions are carried out in separate hardware blocks. The reduction in gate count as well as the increase in regularity (which reduces interconnect capacitances) both have a beneficial impact on energy consumption.

For synchronous applications, a pipelined implementation of the two separated functions can deliver some increase in throughput albeit at the cost of latency, energy and area. In asynchronous applications, however, pipelining has the effect of pushing average-case performance closer to the worst-case due to starvation and blocking effects [11]. The argument in favor of using the CMAC structure in low-power applications, is therefore valid for both synchronous and asynchronous circuits, but particularly so in the latter case.

The general architecture of an $8+4\times4$-bit unsigned CMAC structure, based on the data-dependent CSA, is shown in Figure 2.

In order to accommodate growth in wordlength it is common to provide guard-bits in the accumulator. These can be included by extending the most significant end of the CRA. As with the multiplier discussed above, the aim here is to produce an asynchronous CMAC with reduced propagation delays and energy consumption by utilizing the data-dependency of the operation.

3.3. Carry-resolution Adder

Conventional unsigned CSA multipliers require an n-bit adder for carry resolution. However, with the data-dependent CSA, the number of bits in the CRA is greater than n, requiring additional bits at the least significant end. This is because carries into the LSB of a row of the array can, if the row is in bypass mode, emerge from the array unresolved. In a stand-alone multiplier the first two rows of the CSA cannot produce (non-zero)

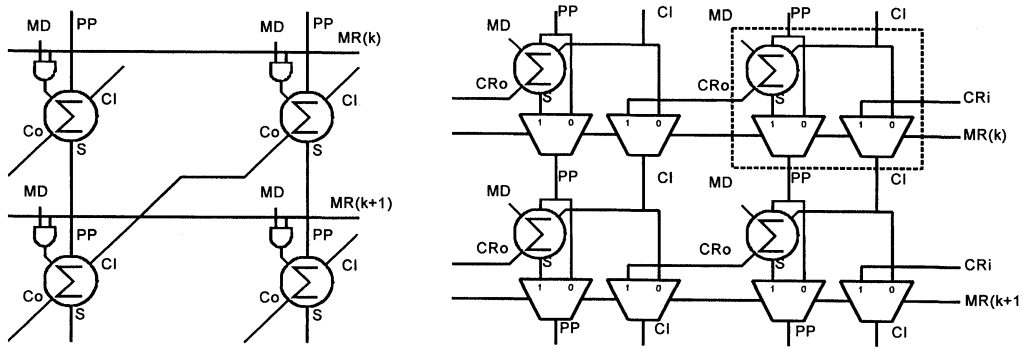

FIGURE 1 Carry-save array (CSA) architectures: standard (left) and data-dependent (right).

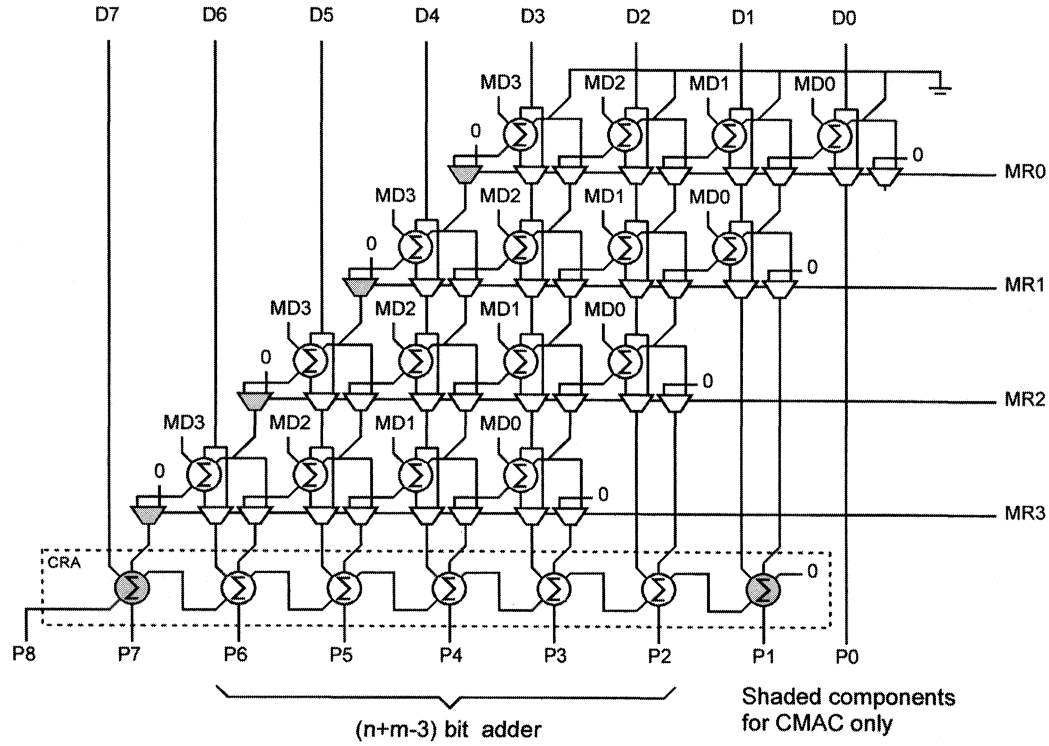

FIGURE 2 Unsigned Concurrent Multiply-Accumulate (CMAC) structure using data-dependent CSA.

unresolved carries and the number of bits required in the CRA is n+m−3 as shown in Figure 2. In the CMAC, (and in certain 2's complement modifications to the multiplier described later) an additional CRA bit is required at the most significant end. Also, with the CMAC, since all rows except the first can produce unresolved carries, one further adder-bit is needed at the least significant end. These additional bits are shown shaded in the CRA of Figure 2.

The CRA used is a ripple-carry adder, chosen because of its minimal power dissipation. It is fitted with completion-detection circuitry and therefore the increase in average latency, due to

the extra bits, is small. The Activity-Monitoring Completion-Detection (AMCD) method [12] is used. AMCD has certain advantages over other completion-detection methods. In particular, its silicon and energy overhead is small in comparison with other techniques especially when applied to domino CMOS [13]. A brief explanation of AMCD is given in Section 5.

4. HANDLING SIGNED OPERANDS

The structure of Figure 2 performs multiplication and multiply-accumulation on unsigned numbers. In many applications, however, the 2's complement data representation is used. The following section presents a method of adapting the above architecture to handle such operands.

4.1. Two's Complement Multiplier, MR

Array multipliers commonly use a simple modification to accommodate a 2's complement representation of MR: since the MSB of a 2's complement number carries a negative weight, instead of performing an addition of the bit-product to the partial-product in the mth (last) row of the array, a subtraction is carried out. The subtraction is often implemented by adding the 1's complement of MD together with a 1 in its LSB position.

The same algorithm can be implemented with the data-dependent CSA albeit with some alternative method of adding the extra 1. Since there are no unused inputs to the CRA at the appropriate position, the addition of the 1 is carried out in the last row of the array by tying the unused input of the LSB's carry-bypass MUX to a logic 1. The 1 is therefore only selected when the row is performing an evaluation, as required.

4.2. Two's Complement Multiplicand, MD

In order to accommodate a 2's complement representation of MD, several strategies can be adopted. The simplest involves sign extension of the MSBs of the two vectors representing a partial product. In a row of the data-dependent array, the most significant carry-out represents the sign of the carry vector, only if the row evaluates – in bypass mode the carry vector's sign is represented by the row's most significant carry-in. Therefore, sign extension can be implemented using an additional MUX to select the appropriate bit to extend. The sign-bit of the sum vector extends as normal. This method implies an increase in the fanout of the two extended signals, thereby adding delays to the critical path and increasing energy consumption. Furthermore, use of the PP input to the MSB cell for sign extension, precludes its use in the CMAC structure. For these reasons, this method was not used.

An alternative which incurs no such penalties uses an arithmetic transform to convert each negative vector into a positive vector plus a (negative) correction term and has some similarities to the algorithm described in [14]. It is illustrated below by means of an example.

For any single binary digit, d, the Boolean identity: $d = 1 - \bar{d}$ or more usefully $-d = \bar{d} - 1$ can be used to transform the negatively weighted MSB of each bit-product into its positively-weighted complement minus 1. Hence the summation of bit-products in the carry save array can be transformed into the representation shown in Figure 3. Here, a row of 5 dots represents a 5 bit, bit-product and an overbar represents the complement of a bit (shown with its -1 correction term).

By elimination of all negatively weighted bits within the array, addition of the sum and carry vectors from each row can take place without sign-extension, the MSB of the partial product being simply the most-significant Carry-out of the row.

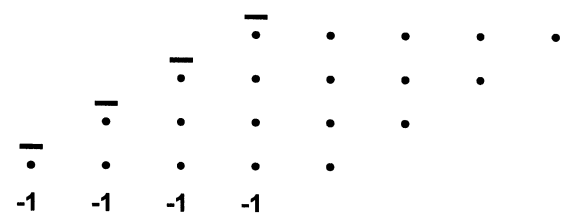

FIGURE 3 Addition of transformed bit-products and correction term.

The correction term is dealt with at the periphery of the array.

This algorithm can be implemented in the data-dependent CSA multiplier provided that care is taken with rows in bypass mode. Inversion of its sign-bit results in a bit-product which can take one of two possible values: $\overline{MD_4}$, MD_3, MD_2, MD_1, MD_0 and 1 0 0 0 0 in evaluate and bypass mode respectively. Since the latter case corresponds to no computation in the row, the non-zero bit must be entered through one of the bypass MUXs. Two methods of achieving this were investigated. One uses a reconfiguration of the inputs to each row's MSB cell as shown for the kth row in Figure 4(a) whilst the other requires some modification to the MSB cell itself, shown in Figure 4(b).

Since both MD_4 and MR_k are available in both logic polarities anyway, neither method involves any increase in gate count. The circuit of Figure 4(a) has the advantage of using the unmodified cell at the cost of a slightly increased switched capacitance on $\overline{MR_k}$. The circuit of Figure 4(b), although modified, has no such associated energy cost and was chosen for that reason.

Although the logic of both these circuits can be minimized, their structure is retained here for clarity. Whilst these solutions are adequate for a stand-alone multiplier, it should be noted that both preclude the combining of multiply and addition functions in the CMAC structure.

The $-1\ -1\ -1\ -1$ 'correction term' can be dealt with most efficiently by Booth encoding it into the equivalent form: $-1\ +0\ +0\ +0\ +1$ whose LSB, has a weight of 2^{n-1} (the same as the MSB of the multiplicand). Therefore this bit's addition can most simply be performed in the most significant cell of the CSA's first row, by tying its Carry-in to a logic 1. By doing so, however, a Carry-out can be produced which, in a stand-alone multiplier, would require extra circuitry down the left-hand side of the array. To avoid this, instead of applying the 1 with weight 2^{n-1}, a string of 1s can be injected with weights 2^{n-2} through 2^2 inclusive (thereby avoiding the most significant cell), together with a 1 of weight 2^2 via the unused Carry-in input to the CRA. In other words the correction term is applied as:

$$\begin{array}{r} -1\ 0\ 0\ 0\ 0\ +1\ +1\ 0\ 0 \\ \hline +1 \end{array}$$

The negatively weighted MSB of the correction term can be dealt with in the CRA by tying one of its MSB inputs to 1, ignoring any Carry thereby produced *i.e.*, a modulo-two addition.

4.3. Two's Complement Multiplicand in the CMAC

For reasons outlined above, simple sign-extension is undesirable. The arithmetic transform method can be applied to the CMAC structure with some modification to the implementation. The transformation of the bit-products into those with inverted

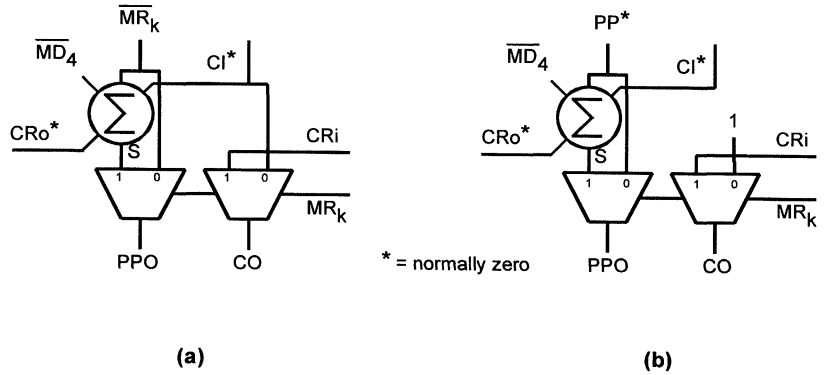

FIGURE 4 MSB cells for bit-product sign-bit inversion (a) unmodified (b) modified.

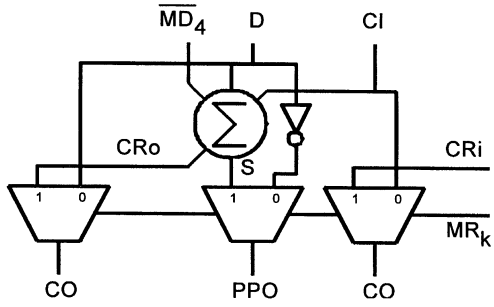

FIGURE 5 MSB cell in row of CMAC for partial-product sign-bit inversion.

sign-bits cannot be carried out as in the case of the multiplier because the unused adder inputs are now required for summand, D. Therefore, bit-product sign-bit inversion is carried out by replacing each row's most significant cell with that shown in Figure 5.

Here, the 1 required to be injected at the MSB position when the row is in bypass-mode is combined with another input (here chosen to be the D input since the increase in capacitive load is placed off the critical path). By doing so, the injected 1 becomes a two bit binary number of value (0 1) when D is zero and (1 0) when D is 1; in other words, of value: $(D, \bar{D})$.

Unlike the multiplier, the CMAC structure must anyway have circuitry (*i.e.*, a MUX) to deal with the Carry-out from the most significant full-adder of each row. Consequently, addition of the positive bit of the correction term can be done more simply with the CMAC structure than with the multiplier, merely requiring that the top row's most significant Cin input be tied to a logic 1. The negative bit of the correction term is dealt with, as before, in the MSB of the CRA.

5. IMPLEMENTATION AND TIMING

5.1. Interface to the Environment

The multiplier and CMAC circuits fit into the asynchronous framework for dynamic logic proposed in [15] *i.e.*, the leading and trailing edges of the START input, initiate the evaluation and precharge phases respectively. Similarly, the leading edge of the DONE signal indicates validity of the output data and its trailing edge indicates completion of the precharge phase. Such a protocol requires that the input operands are stable before the leading edge of the START signal and remain so throughout the computation.

To ensure low-power operation, the circuits should not be left in the evaluation phase for longer than the dynamic storage time, otherwise charge leakage from dynamic storage nodes could cause short-circuit power dissipation in the domino inverters. This requirement is easily met in both synchronous and asynchronous applications, a handshake circuit suitable for the latter being described in [15].

5.2. Carry Save Array Using Conditional Evaluation

In the implementation of [4] significant energy savings derive from using a CSA comprising a combination of static logic for the MUXs and dynamic logic for the adders. The inclusion of static MUXs allows a row to function correctly in bypass mode with no transition on the precharge/evaluate lines – these remain in precharge mode until addition is called for. Assuming equal probabilities of ones and zeros in the bits of MR, on average only 50% of the precharge/evaluate lines will be driven for each multiplication, yielding a corresponding reduction in energy consumption. This, strategy, termed Conditional-Evaluation in [4], was shown to be an important mechanism for power reduction – a row in bypass mode typically using less than one quarter of the energy of a row in which addition is performed.

The structure of the CSA cell is shown in Figure 6(a). The inverters provide buffering to prevent excessive rise/fall times through the MUXs, (particularly important in bypass-mode). As outlined above, to maximize energy savings, single ended dynamic-logic was employed unlike the differential implementation of [3]. The

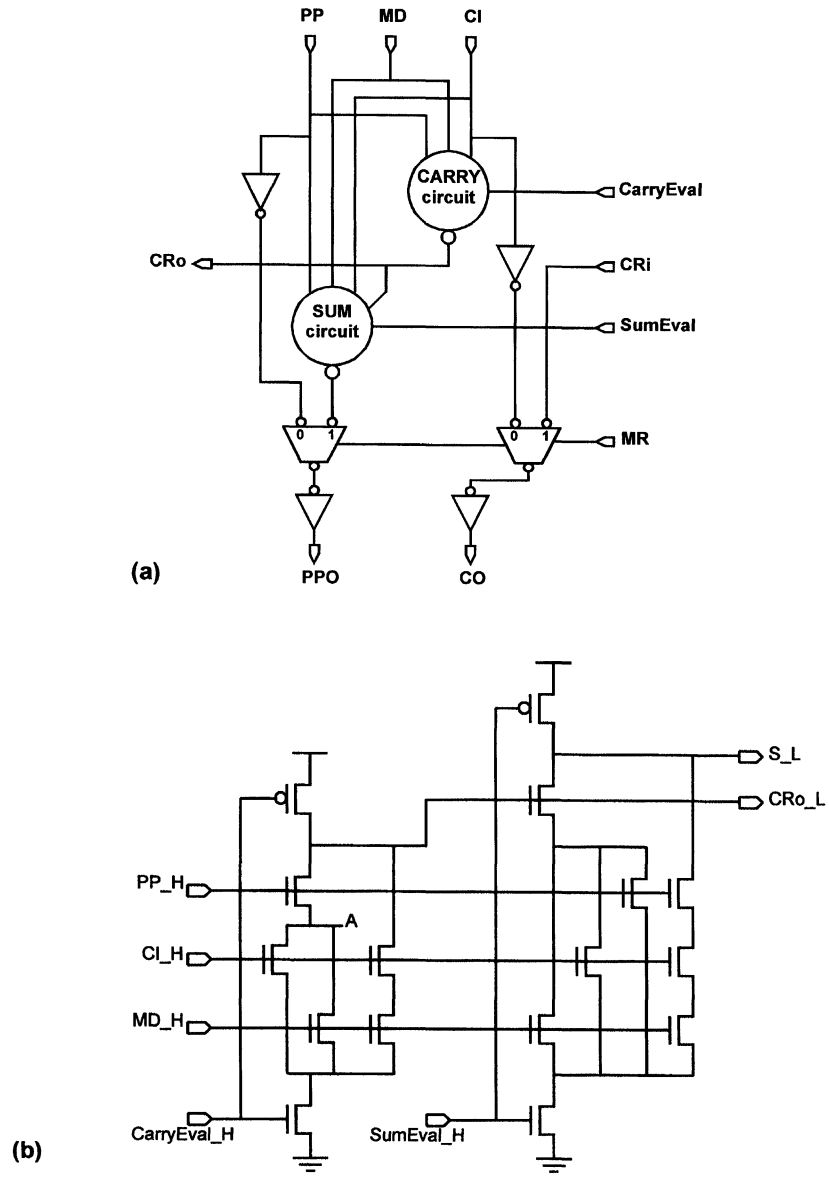

FIGURE 6 Carry-save array cell showing structure (a) and full-adder circuit (b).

full-adder is itself composed of a cascade of two dynamic n-blocks, one which evaluates the Carry-Out signal, Co, and one which uses Co to evaluate the Sum output, S. This circuit was used because of its low device count and hence low switched-capacitance. The precharge/evaluate lines for the carry and sum circuits are labeled CarryEval and SumEval respectively and are common to all cells in the same row. The transistor-level schematic of the two dynamic blocks is shown in Figure 6(b).

The self-timed mechanism used to eliminate the dynamic-logic race uses inserted delays to post-pone evaluation of each dynamic stage until all of its inputs have stabilized [8]. Each row of the CSA is controlled by a Conditional Evaluation timer circuit whose structure is shown in Figure 7.

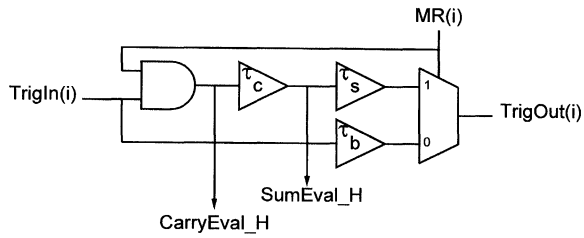

FIGURE 7 Conditional evaluation timer for data-dependent CSA.

For rows whose multiplier bit is one, evaluation of the carry and sum circuits, takes place when low-high transitions occur on CarryEval_H and SumEval_H, separated in time by τ_c. The duration of τ_c is chosen to allow Co to settle before switching the sum circuit into evaluation mode. Similarly, τ_s and τ_b are chosen to postpone evaluation in the following row until it is safe to do so.

TrigIn of the first row's timer is driven by the START signal which initiates computation in the array. It is a requirement of this type of circuit that the input operands are stable before the leading edge of the START signal and remain so throughout the evaluation phase. The TrigOut signal is connected to TrigIn of the following row's timer. In rows whose multiplier bit is zero, it is passed on through the MUX after delay τ_b to the following row, without firing the precharge/ evaluate lines.

Delays for the timer structure of Figure 7 can most simply be implemented with a minimum-sized inverter whose pull-down is connected to ground *via* a permanently-on n-type device. The channel length of this latter device is selected to produce the required delay. Whilst such a circuit has a very low switched capacitance, selection of the channel length requires careful simulation. Furthermore, the structural differences between the delay and the data-path it is intended to model, will demand a greater margin of safety to ensure correct operation under a spread of operating conditions.

An alternative to this simple delay is to use a matched delay circuit. This presents a higher switched capacitance but has the advantage that the delay more closely tracks that found in the data-path – the delay is produced using a circuit identical to that of the data path but hard-wired to its slowest configuration.

A characteristic of the structure of Figure 7 is that rise times on CarryEval and SumEval add to the delays through the timer. If these rise times are known accurately then the timer delays can be adjusted accordingly. However, given the relatively large capacitance on these two signals, their rise times may be subject to significant variation and therefore the timer may require more conservative timing margins than would otherwise be necessary. To avoid this uncertainty the structure can be modified slightly such that the capacitances of SumEval and CarryEval are decoupled from the actual timing path by buffering. This gives tighter control of the timer delays but incurs the added risk that a large difference between the rise times of CarryEval and SumEval could cause the timing requirements to be violated. In practice, since the load and interconnect of these two signals is very similar, differences between their rise times can be kept small. The row-timer adopted for the designs reported here, uses this decoupled timer structure with matched circuits for the delays.

5.3. Asynchronous Carry Resolution Adder

The carry-resolution adder uses a similar full-adder circuit to that shown in Figure 6(b). All CarryEval lines are connected together as are all SumEval lines. With the inclusion of the domino inverters, the carry circuits are cascaded, creating a standard domino configuration. This, unlike the adders in the CSA, is susceptible to charge sharing effects, requiring an additional precharge transistor connected to node A.

The CarryEval signal of the CRA is connected to the TrigOut signal from the mth (last) row's timer thereby initiating evaluation as soon as stable inputs are signalled. The CRA uses the AMCD method of completion detection which is based on the premise that a CMOS circuit will have completed its computation when all transi-

tions on its wires have ceased. With AMCD, Activity-Monitors (AMs) are connected to certain points within the circuit, each AM generating a narrow pulse on its output when it detects a transition on its input. The presence of the pulse signifies that the circuitry immediately downstream of the AM, is likely to be not yet settled since a transition has just occurred on one of its inputs. Assuming appropriate placement (granularity) of the AMs and sufficient pulse-widths to ensure overlap, the logical OR of the individual pulses indicates that the whole circuit is still in transition. This signal is generated with a wired-OR configuration and its trailing edge indicates that the circuit has completed its operation.

Granularity of the AMs is set according to [13] *i.e.*, such that one AM covers two cascaded full-adders. To deal with the case when there are no transitions on the monitored signals, a minimum-delay-generator (MDG) is required; an AM connected to CarryEval is used in order to produce this required minimum delay.

To avoid the dynamic race, evaluation of the sum circuits is delayed until the output from the AMCD circuit indicates that all carries have settled. Therefore SumEval should rise following a rising edge on ACT_L (the signal indicating completion), but fall when the multiplier/CMAC enters its precharge phase (during which period ACT_L remains high). To produce these dependencies, an interlock circuit as shown in Figure 8 is used.

To produce the DONE signal, whose leading edge indicates validity of the output data, an additional delay is used, DONE being merely the CRA's SumEval signal delayed by at least the settling time of the sum circuit. Its trailing edge indicates that the precharge phase is complete. The delay was implemented here using the matched delay approach described above.

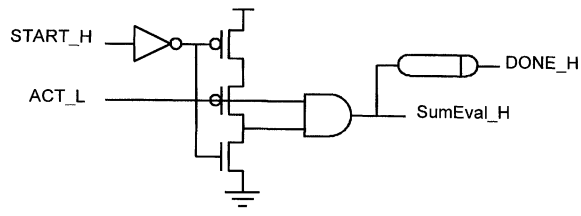

FIGURE 8 CRA controller for sum circuit.

5.4. Precharge Strategy and Spurious Transitions

The time to precharge the CMAC adds directly to its cycle time and should be kept as small as possible. One of the attractions of dynamic logic for low power circuits is its immunity from spurious transitions during evaluation. Dynamic circuits using the delayed-evaluation technique can, however, be susceptible to spurious transitions when entering precharge unless the relative timing of each stage's precharge phase is appropriately controlled. This is because some logic-tree transistors are turned on during precharge and therefore an upstream gate precharging before a downstream gate, could result in a glitch on the output of the downstream gate.

To avoid this possibility, the safest, most general approach is to ensure that all stages precharge in the reverse order to their evaluation. (First-Evaluate, Last-Precharge). However, this demands a more complex control circuit than the simple cascade of delays described above and furthermore, a fairly long precharge delay is produced.

Another possible approach is to ensure that the time for a H-L transition to propagate through delay τ_c (Fig. 7) is sufficiently short that the sum circuit does not have time to react to the precharged Carry-out signal before it too, is precharged. Therefore τ_c becomes an asymmetric delay and all stages precharge in the same order as they evaluate.

A third possibility is to use a scheme in which all stages precharge simultaneously. This requires modest modifications to the control circuit but results in a shorter precharge delay than the other methods. Although this may result in slightly higher electromagnetic emissions than the other precharge strategies (since all charge required to precharge the circuits will be drawn at the same instant), emmisions are anyway likely to be

significantly lower than with synchronous implementations.

The possibility of spurious transitions is also introduced by the embedding of static-logic (the bypass MUXs) within the array as required by the conditional-evaluation technique. Although all dynamic logic outputs are precharged high, the MUXs provides a path along which addend values can propagate into the array giving rise to possible spurious transitions when MR and D change. The multiplicand can take no part in the creation of spurious transitions. The likelihood of spurious transitions in the unsigned multiplier is less because during the precharge phase both inputs to the bypass MUXs return to the same value. Therefore, the only mechanism that can produce glitches arises when MR arrives before the state left by the previous evaluation, has flushed through. The precise energy cost of these glitches is difficult to ascertain, however two factors tend to lessen their effect in comparison with a fully static CSA. Firstly, propagation of glitches is blocked by rows whose MR bit remains 1 between computations. Secondly, propagation is only possible down the columns of the array – in the fully static case a glitch can propagate both down the columns as well as across them *via* the carry circuit.

6. RESULTS

6.1. Area

The silicon area required to implement the described structures is dominated by that required for the CSA. For minimization of energy consumption, a hand-crafted full-custom layout was undertaken in a 0.35 μm 3-layer metal CMOS technology.

In comparison with a standard, static-logic based carry-save array, the data-dependent structure benefits from certain economies of area. These are largely brought about by the elimination of the bit-product generation circuitry and by the significant reduction in transistor-count associated with the domino logic. On the other hand, the two buffered bypass-MUXs used in each cell require additional area. Overall, the CSA cell area of the two approaches is therefore similar.

The asynchronous CRA cell also benefits from a domino implementation but requires an activity monitor in every second bit. The net silicon area per cell is therefore comparable to that of a conventional static ripple-carry adder. (The latter, being unable to exploit data-dependencies would however, be considerably slower).

The principal silicon overheads of the proposed approach result from the extra full-adders used at the LS end of the CRA (approaching one per CSA row) and from the Conditional Evaluation timers (one per row). The use of the matched-delay technique gives one of the latter cells a similar area to the CSA cell.

In summary therefore, a multiplier implemented using the data-dependent approach can be expected to require an approximate area overhead of the equivalent of two CSA cells per CSA row *i.e.*, $\sim 25\%$ for an 8×8 structure.

6.2. Simulation Results

Transistor level simulations of three structures – an 8×8 bit unsigned multiplier (USMULT) an 8×8 bit 2's complement multiplier (SMULT) and a $16+8 \times 8$ bit 2's complement CMAC, were run on a SPICE-based analog simulator. Devices were taken from a 0.35 μm n-well CMOS process using a 3.3 V power supply. Typical models at 27°C were used. In order to reduce switched capacitances, all transistors in the data-path are of minimum size. Layout parasitics are not included in the models.

In assessing the energy efficiency of such structures, it is important that the energy costs associated with the switched capacitance of the MUX select lines and other circuit inputs is not overlooked. All input operands were therefore buffered through inverters thereby ensuring that these costs are included in the simulation results.

The highly data-dependent evaluation time of these circuits depends principally on two factors:

the number of ones in MR and the length of the carry-ripple in the CRA. With fairly conservative timing margins, a row of the CSA takes approximately 0.25 ns and 0.9 ns in bypass and addition modes respectively. The CRA takes approximately 1.3 ns fixed-delay overhead plus 0.5 ns per carry-ripple bit-pair. Precharge delay is largely constant at 0.8 ns.

Table I shows a comparison of the minimum, average and maximum propagation delays including the precharge delay. Energy consumption per computation is also shown. Since increased circuit activity requires more delay and more energy, these two parameters have a strong correlation. However, the computation yielding worst case delay is not in fact that which uses the most energy.

Averages are calculated from 50 computations using uniformly distributed random operand values. Maximum and minimum measurements are taken from this set. Operand values which yield the worst-case performance depend on the carry-ripple speed relative to the row delays. Their derivation, particularly with the CMAC, is non-trivial. Critical path analysis indicates that the worst case delays have an upper bound of approximately 11.9 ns, 12.1 ns and 12.4 ns for the three circuits respectively.

6.3. Comparisons

For comparison with other published work, account must be taken of differences in supply voltage, technology dimensions and operand wordlengths. An algorithm for normalizing multiplier energy consumption was advocated in [16] and is based on the assumption that energy is proportional to the square of the supply voltage but linearly related to the multiplicand-word-length, multiplier-wordlength and technology dimensions. This algorithm was applied to several recently published designs, normalizing to a 16×16-bit structure on a 1.2 V, 0.35 μm technology. A supply voltage of 1.2 V was chosen since it represents the approximate limit below which Vdd can no longer be reduced to obtain energy savings. The designs were selected because low energy consumption was a primary design goal. The results of this normalization are shown in Table II along with the simulation results of a 16×16-bit implementation of our CMAC, also normalized to $\text{Vdd} = 1.2\,\text{V}$. Simulations of parasitic-extracted, back-annotated layouts show an allowance of 100% for parasitics for our circuit to be a realistic estimate. The energy-consumption and delay given for our CMAC have therefore been doubled to take this into account. (Note: simulations of the full, back-annotated 16×16-bit layout were prohibitively time consuming).

In the case of computation delays, normalization has been carried out using a different algorithm. Delays taken from [8, 16] and [17] have been linearly extrapolated to a 16-bit multiplier operand. The circuits of [8, 16] and [18] have been scaled linearly to a 0.35 μm process technology (from 2 μm, 0.5 μm and 0.25 μm respectively). Delays have also been adjusted on the assumption that they are inversely proportional to supply voltage. After these normalizations the figures given in the table are obtained.

TABLE I Simulation results for 8×8 and $16+8 \times 8$ structures

		Min.	Avge.	Max.
USMULT	Delay (ns)	4.68	7.11	10.0
	Energy (pJ)	7.23	19.1	38.2
SMULT	Delay (ns)	5.77	7.50	12.0
	Energy (pJ)	17.7	24.4	34.2
CMAC	Delay (ns)	4.79	7.74	8.98
	Energy (pJ)	15.1	24.9	35.8

TABLE II Energy and delay of various multipliers after normalization to 16 × 16-bit, 0.35 μm, $Vdd = 1.2$ V

	Energy	Delay	Comments
[8]	31 pJ	~800 ns	
[18]	50 pJ	21 ns	Multiplier-Accumulator. Non-standard process.
[17]	42 pJ	32 ns	CSA only. CRA not included.
[16]	32 pJ	62 ns	Energy benefits from non-random operands.
This work	23 pJ	74 ns	Delay is an average value.

As can be seen, our design yields the lowest normalized energy consumption per multiplication. The energy given for the multiplier of [16] is taken from a FIR filter and uses other, filter-specific techniques to reduce operand activity. In a similar environment our circuit could be expected to consume significantly less energy.

The design of [17] is a CSA only and the significant energy and delay contributions of resolving the sum and carry vectors are omitted. A non-standard CMOS process adapted to maintain performance at reduced supply voltage was used in [18], a fact reflected in the given delay. The other designs were based on standard CMOS processes and a performance comparison is more meaningful. In this respect our design is, on average, 19% slower than [16] but shows at least a 14% improvement in Energy*Delay. In terms of Energy*Delay^2 our design is approximately equivalent to [16].

7. CONCLUSIONS

Self-timed circuits for multiplication and multiply-accumulation using a data-dependent architecture have been implemented, laid-out and simulated at transistor-level. Adaptation of the architecture to handle 2's complement operands has been discussed and an efficient solution is provided. Measurements indicate that the proposed design style offers considerable benefits in terms of energy consumption. These are achieved by a synergy of data-dependent architecture and implementation. Central to the energy efficiency of the implementation is the Conditional-Evaluation technique, which enables redundant dynamic-logic activity to be inhibited at very low overhead. Comparisons show this approach can achieve lower energy consumption than other reported designs.

The designs also benefit from low device count, high regularity and good testability, which facilitate VLSI implementation. Whilst the data-dependent computation times of the proposed structures can be best exploited within an asynchronous environment, the high energy-efficiency can be obtained in both synchronous and asynchronous applications.

References

[1] Chandrakasan, A., Sheng, S. and Brodersen, R. W. (1992). "Low-Power CMOS Digital Design", *IEEE Journal of Solid-State Circuits*, **SC-27**, 473–484.

[2] Meng, T. H.-Y., Brodersen, R. W. and Messerschmitt, D. G. (1992). "Asynchronous Design for Programmable Digital Signal Processors", *IEEE Transactions on Signal Processing*, **39**(4), 939–952.

[3] Kearney, D. and Bergmann, N. W. (1997). "Bundled Data Asynchronous Multipliers with Data Dependent Computation Times", *Proc. 3rd International Symposium on Advanced Research in Asynchronous Circuits and Systems*, IEEE Computer Society Press, pp. 186–197.

[4] Bartlett, V. A. and Grass, E. (1998). "A Self-Timed Multiplier using Conditional Evaluation", *Proc. PATMOS'98, 8th International Workshop on Power, Timing, Modeling, Optimization and Simulation*, Lyngby, Denmark, pp. 429–438.

[5] Poornaiah, D. V., Haribabu, R. and Ahmad, M. O. (1993). "Design and VLSI Implementation of a Novel Concurrent 16-bit Multiplier-Accumulator for DSP Applications", *Proc. International Conference on Acoustics, Speech and Signal Processing ICASSP*-93, pp. I: 385–388.

[6] Lemonds, C. and Shetti, S. S. (Apr., 1994). "A low power 16 by 16 multiplier using transition reduction circuitry", *Proc. 1994 International Workshop on Low Power Design*, pp. 139–142.

[7] Sakuta, T., Lee, W. and Balsara, P. T. (1995). "Delay Balanced Multipliers for Low Power/Low Voltage DSP

Core", *Proc. Int. Symp. Low Power Electronics and Design*, pp. 36–37.
[8] Sobelman, G. E. and Raatz, D. (1995). "Low-power multiplier design using delayed evaluation", *Proc. International Symposium on Circuits and Systems*, pp. 1564–1567.
[9] Alidina, M. *et al.* (Dec., 1994). "Precomputation-Based Sequential Logic Optimization for Low Power", *IEEE Trans. VLSI Syst.*, **2**, 426–436.
[10] Tiwari, V., Malik, S. and Ashar, P. (1998). "Guarded Evaluation: Pushing Power Management to Logic Synthesis/Design", *IEEE Trans. CAD of Integrated Circuits and Systems*, **17**(10), 1050–1060.
[11] Kinniment, D. J. (Mar., 1996). "An Evaluation of Asynchronous Addition", *IEEE Trans. VLSI Syst.*, **4**, 137–140.
[12] Grass, E. and Jones, S. (1996). "Activity-Monitoring Completion-Detection (AMCD): A new approach to achieve self-timing", *Electronics Letters*, **32**(2), 86–88.
[13] Bartlett, V. A. and Grass, E. (1997). "Completion-Detection Technique for Dynamic Logic", *Electronics Letters*, **33**(22), 1850–1852.
[14] Salomon, O., Green, J.-M. and Klar, H. (Jul., 1995). "General Algorithms for a Simplified Addition of 2's Complement Numbers", *IEEE J. Solid-State Circuits*, **30**, 839–844.
[15] Furber, S. B. and Liu, J. (1996). "Dynamic Logic in Four-Phase Micropipelines", *Proc. 2nd International Symposium on Advanced Research in Asynchronous Circuits and Systems*, IEEE Computer Society Press, pp. 11–16.
[16] Nicol, C. J. and Larsson, P. (1997). "Low Power Multiplication for FIR Filters", *Proc. Int. Symp. Low Power Electronics and Design*, pp. 76–79.
[17] Khoo, K.-Y., Yu, Z. and Willson, A. N. Jr. (1999). "Improved-Booth Encoding for Low-Power Multipliers", In: *Proc. Int. Symp. Circuits Syst., ISCAS'99*, **1**, 62–65.
[18] Izumikawa, M. *et al.* (1997). "A 0.25-μm CMOS 0.9-V 100-MHz DSP Core", *IEEE Journal of Solid-State Circuits*, **SC-32**, 52–61.

Authors' Biographies

Viv Bartlett graduated with B.Sc. Eng. (Hons.) from Imperial College, London in 1977. He is currently a Senior Lecturer with the Department of Electronic Systems, University of Westminster, London where he has been on the academic staff since 1984. He has recently received the award of Ph.D. after taking a two-year sabbatical to pursue his research interests on a full time basis. These include VLSI design, DSP architectures, low-power design and asynchronous systems.

Eckhard Grass received his Dipl.-Ing. degree in Electronics in 1987 and his Dr.-Ing. in 1992 from Humboldt University, Berlin. He worked as a visiting Research Fellow at Loughborough University (U.K.) from 1993–95 and as a Senior Lecturer in Microelectronics at the University of Westminster (London) from 1995–1999. Dr. Grass is now working for IHP-GmbH in Germany, leading a project in the design and implementation of a wireless broadband communication system. His research interests include data-driven signal processing structures, asynchronous circuit design methodologies and VLSI implementation of wireless communication systems.