Design of the ARM VFP11 Divide and Square Root Synthesisable Macrocell

Neil Burgess
Cardiff School of Engineering,
Cardiff University,
CARDIFF UK
scenb@cf.ac.uk

Chris N. Hinds
ARM Ltd.,
Fulbourn Road,
CAMBRIDGE UK
chris.hinds@arm.com

Abstract

This paper presents the detailed design of the ARM VFP11 Divide and Square Root synthesisable macrocell. The macrocell was designed using the minimum-redundancy radix-4 SRT digit recurrence algorithm, and this paper describes a novel acceleration technique employed to achieve the required processor clock frequency of up to 750MHz in 90nm CMOS. Logical Effort theory is used to provide a delay analysis of the unit, which demonstrates the balanced nature of the two critical paths therein.

1. Introduction

The VFP11 coprocessor [1] is an implementation of the ARM Vector Floating-point Architecture for integration with ARM11-family cores. The VFP11 implements IEEE 754 compliant single-precision and double-precision operations with software support for operations and data types rarely used in an embedded context. The arithmetic pipeline is optimized for 3D graphics processing [2] on a chained multiply-accumulate engine with a throughput of one FMAC instruction per cycle for single-precision data and one per two cycles for double-precision data. Hardware divide and square root are optimized for 2 bits per cycle throughput and parallel execution with data transfer operations and operations on the FMAC pipeline. Vector operations provide the capability to issue up to 8 single-precision operations in a single instruction. A vector operation will run in parallel with data transfer operations and divide/square root operations and result in nearly optimum utilization of the FMAC pipeline for graphics operations and other high data throughput computations. The VFP11 is designed for low power consumption and small die size.

This paper presents the detailed design of the VFP11 divide and square root synthesisable macrocell. The major design requirement was to achieve a logic depth of as close to 15 logic levels as possible so as to meet a variety of performance targets for the whole chip. This led to the minimum-redundancy radix-4 SRT algorithm being used because multiplicative solutions were unattractive at the required high clock rate for two reasons:

- fast multipliers are large and power-hungry – it was infeasible to use the extant VFP11 multiplier-accumulator chain for performing division and square root operations, and wasteful in area and power terms to build a second multiplier dedicated to square root and division
- in a deeply-pipelined processor design, Newton-Raphson and Goldschmidt iterations take many cycles to complete due to dependencies between and within successive iterations – see [3], for example

By contrast, using radix-4 SRT, VFP11 takes 15 cycles (single precision) or 29 cycles (double precision) to compute either correctly rounded quotients or square roots.

Some details of this block were presented previously in [4], including:

- the digit selection Table constants
- logic for partial remainder m.s.b. compression that is needed to preserve sign information while discarding the m.s.b.'s of the partial remainder between iterations (the unit was designed using signed-digit representation of the partial remainder)
- the use of comparators instead of a look-up table to implement digit selection
- an adaptation of "on-the-fly" conversion to derive updated square root estimates in minimal logic depth

This paper reveals further parallelisation of the SRT recurrence so as to remove one of two "back-to-back" carry-propagate additions from the digit selection logic [5].

18th IEEE Symposium on Computer Arithmetic(ARITH'07)
0-7695-2854-6/07 $20.00 © 2007 IEEE

2. VFP11 Divide and Square Root Macrocell Implementation

The recurrence equation for SRT division is:

$$R_{i+1} = r \cdot R_i - D \cdot q_{i+1} \qquad \text{—(1)}$$

and that for SRT square root is:

$$R_{i+1} = r \cdot R_i - 2Q_i \cdot q_{i+1} - q_{i+1}^{\ 2} \cdot r^{-(i+1)} \qquad \text{—(2)}$$

where R_x is the remainder after the x^{th} iteration, r is the radix of the SRT algorithm, D is the divisor, q_x is the x^{th} digit of R_x, and Q_x is the x-digit result computed after the x^{th} iteration. In SRT division, the divisor is assumed to be in the range $1 \le D < 2$ in keeping with the significand range of the IEEE floating-point standard. For consistency between the SRT division implementations, the root estimate is constrained to satisfy $1 \le 2Q_i < 2$, implying that the radicand must be in the range $0.25 \le R_i < 1$. This range of radicand ensures that the exponent can always be even. To initialise the square root recurrence, q_0 is forced to 1 so that if $Q_i > u$, the redundancy factor defined as $q_{max}/(r-1)$, the result is still obtainable. For minimum-redundancy radix-4 SRT iterations, $u = 2/3$. These two equations are frequently combined into one unified expression by writing:

$$R_{i+1} = r \cdot R_i - F_i \cdot q_{i+1} \qquad \text{—(3)}$$

where F_i is the appropriate value for division or square root derived from (1) and (2).

A block diagram of the divide and square root synthesisable macrocell is shown in Figure 1, and Figures 2 - 4 provide more detail of the three critical logic blocks.

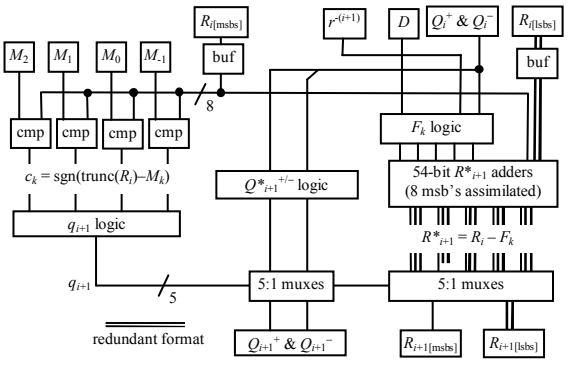

Figure 1 Block diagram of divide and square root macrocell

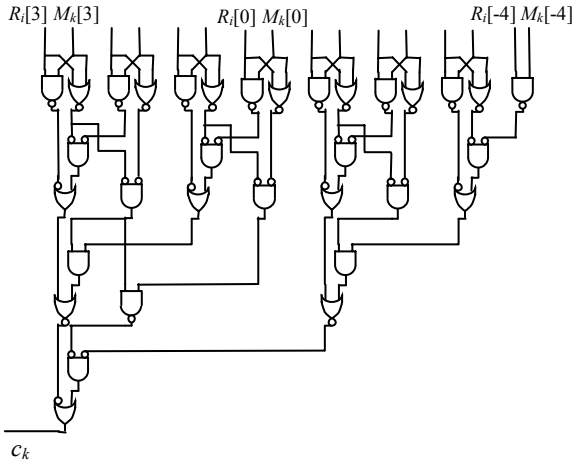

Figure 2 8-bit comparator: c_k = sign(R_i[3:-4] – M_k[3:-4])

Before the first iteration, the four selection constants used to select q_{i+1}, denoted M_k, are loaded into registers, and the D, R_i, Q_i^+ and Q_i^- registers are initialised with the appropriate values. Then, at each subsequent iteration, the top eight bits of the partial remainder, R_i, are compared with the four selection constants, M_k (Figure 2), and the four 1-bit results of these comparisons, c_k, are combined to derive the value of the next result digit, q_{i+1}, in a 1-hot encoding.

In parallel with these comparisons, the five possible updated remainders $R^*_{i+1} = R_i - F_k$ for $k = -2 \ldots +2$ are computed, but with the top 8 bits in non-redundant format, as shown in Figure 3. In this way, no 3:2 reduction is needed ahead of the M_k comparators on the next iteration, thus minimising the logic depth of the critical path through the comparators.

As discussed in [4], these short carry-propagate additions across the m.s.b.'s also have the effect of "compressing" the speculative signed-digit remainders, so that sign information is not lost when the top two bits of the remainder are discarded between iterations. Also as discussed in [4], the four possible updated subtrahends, F_k, (for $k \in \{-2, -1, +1, +2\}$ only) are formed by a circuit whose logic depth comprises a NOR gate driving into the data input of a 2:1 multiplexer. The multiplexer is needed to select the correct set of F_k values depending on whether a division or a square root operation is being executed.

Finally, the new value of q_{i+1} as derived from the four values of c_k selects the appropriate value of R_{i+1} (in redundant format except for the 8 m.s.b.'s) and the updated range estimates of the square root, $Q_{i+1}{}^+$ and $Q_{i+1}{}^-$, and the iteration is complete (see Figure 4, where the logic that derives and concatenates the 2 l.s.b.'s of

the updated square root range estimates Q_{i+1}^{+} and Q_{i+1}^{-} is not shown).

The performance of the macrocell is summarised in Table 2 for three different technology nodes. The ranges of frequency at 130nm and 90nm reflect different process technologies at those nodes that trade off speed for leakage power.

Table 2 Operating frequencies of macrocell

Technology	Frequency	Delay / CMOS gate
180nm	270 MHz	142 ns
130nm	350 – 550 MHz	78 – 122 ns
90nm	450 – 750 MHz	57 – 95 ns

There were 18 stages of CMOS logic and buffers along the critical path (that ran through the q_{i+1} logic), which after allowing for clock insertion, translated to 142 ns per logic stage at 180nm. This was deemed near enough to the initial specification (of 15 CMOS stages) to be acceptable.

3. Logical Effort analysis of VFP11 divide and square root macrocell

In this Section, the macrocell is analysed using Logical Effort [6] to provide further insight into the balanced nature of the two critical paths through the design. Logical Effort is a design methodology for estimating the number of CMOS stages (including buffers) required to implement a given logic function. While Logical Effort is not a substitute for detailed simulation, it is excellent at comparing different CMOS digital designs. The method described below has been applied to Knowles' "Family of Adders" [7], where it identified the same trade-offs between delay and area as were described in that paper, even to the extent of providing the same ranking of the adders for speed. There was also no greater than a 5% difference in delay estimate between the Logical Effort model and Knowles' reported delays [8].

Logical Effort uses a small number of basic concepts, which are:

logical effort, g: total FET gate capacitance of a CMOS logic gate relative to that of a minimum-sized inverter

electrical effort, h: ratio of output capacitance to input capacitance for each CMOS logic gate along a critical path

branching effort, b: ratio of total capacitative load on one CMOS logic gate's output along the critical path to the FET gate capacitance of the next CMOS gate on the critical path

parasitic delay, p: total diffusion capacitance on the output node of a CMOS logic gate relative to the input FET gate capacitance of a minimum-sized inverter

Logical Effort operates by calculating the total Path Effort along the critical path of a digital CMOS circuit as:

$$F = GBH \quad \text{—— (4)}$$

where $G = \Pi g$, $B = \Pi b$, and $H = \Pi h$. The last term reduces to the ratio of the output capacitance loading the last CMOS logic gate to the FET gate capacitance of the first CMOS logic gate along the critical path. Usually, H is forced to 1 by assuming that the circuit being modelled is connected to a copy of itself. This allows input branching effort to be incorporated in a delay estimate. Values of Logical Effort and Parasitic Delay for a selection of cells (taken from [6]) are listed in Table 3.

Once the path effort has been calculated, a near-optimum design for the CMOS circuit can be determined by deriving the number of CMOS stages (including buffers) required in the circuit as:

$$N = \log_4 F \quad \text{—— (5)}$$

N is then rounded to the nearest integer to give the parameter, α:

$$\alpha = F^{1/N} \quad \text{—— (6)}$$

Table 3 Values of Logical Effort and Parasitic Delay for selected CMOS cells

CMOS gate	g	p
NOT	1	1
NOR2	5/3	2
NAND2	4/3	2
NAND3	5/3	3
AOI21	6/3	7/3
OAI21	6/3	8/3
XOR2	12/3	5

The FET's along the critical path are now sized such that the electrical effort of each logic gate (i.e. the ratio of the total output load capacitance to the input FET gate capacitance), $h = \alpha/g$. Then the total delay of the CMOS circuit under consideration may be written as:

$$D = N\alpha + \sum p \quad \text{—— (7)}$$

in arbitrary delay units. Dividing this expression by 5 yields an approximation to the delay in terms of fan-out = 4 ("FO4") inverter delays.

Thus, from (4), the Path Effort of a logic circuit, F, can be calculated by multiplying together the fan-out loads ($g \cdot b$) at each node along the critical path. In computing the fan-outs, track capacitance per logic gate fan-out has been assumed to be equivalent to one minimum-geometry p-FET, or 2/3 of the gate

capacitance of a minimum-size inverter. Hence, a track that has a fan-out of three cells is allocated a track fan-out of 6/3 = 2 minimum size inverters. However, in a datapath component, laid out with two cells per bit or column, a track fan-out of 1 p-FET per bit traversed is allocated. Hence, a track that traverses 4 bits of a comparator would have a lateral distance of four columns and would be allocated a track fan-out of $4 \times 2/3 = 8/3$ minimum size inverters [8]. Note that in the following analysis, the logic diagrams presented are indicative only because the macrocell was synthesised from a pseudo-behavioural RTL description.

There are two parallel critical paths through the macrocell: one starts at the R_i register and goes through one of four 8-bit comparators followed by the 1-hot encoding of q_{i+1} and into the select input of the 5:1 muxes that return the updated values R_{i+1}, Q_{i+1}^{+}, and Q_{i+1}^{-}. The other path starts at the Q_i registers and goes through the logic that derives the five possible values of F_k, then through one of five 56-bit carry-save adders with 8-bit carry-propagate subtractors, ("R_{i+1} adders"), and finally into the data inputs of the 5:1 multiplexer. The logic comprising the q_{i+1} 1-hot encoding logic and 5:1 multiplexer, presented in Figure 4, shows that although the logic depths of the R_{i+1} and Q_{i+1} update 5:1 muxes are essentially the same, the path through $q_{i+1=0}$ has the largest fan-out of four NAND2 gates.

Figures 5 and 6 present logic diagrams of the 8-bit subtractor (with its carry input held at logic '1') and full adder and also found on the critical paths of the macrocell. Note that the i1 input of the full adder has a shorter critical path through the full adder than the other two.

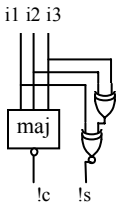

Figure 5 Logic diagram of full adder

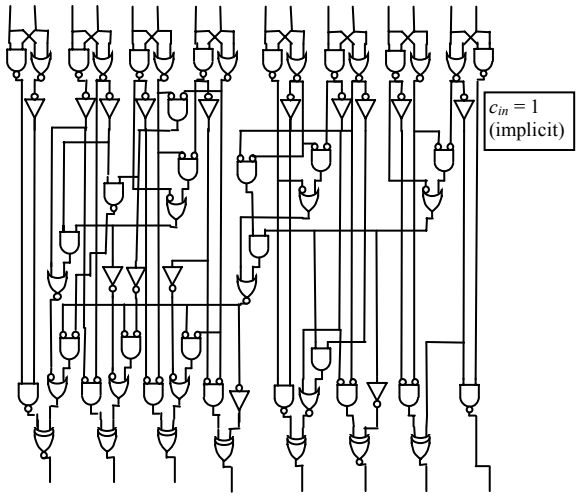

Figure 6 Logic diagram of 8-bit subtractor

The Logical Effort analyses of the two critical paths through the macrocell are laid out below, and show that the estimated delays of the two paths are 16.0 and 15.4 FO4. Thus, the design is well balanced between the two critical paths (only a 3.8% difference), with the path delay through the slower subtractor matched by that through the comparator and the multi-stage buffer.

Critical path 1: through F_k logic (output connected back onto input)

gate	load	$g \cdot b$	p
buf*	4 × nor2's + wire	4×5/3 + 4×2/3	2
nor2	mux2 + wire	6/3 + 2/3	2
mux2	xnor2 + min + wire	12/3 + 12/3 + 4/3	4
xnor	nand2 + nor2 +wire	4/3 + 5/3 + 4/3	5
nor2	oai21 + wire	6/3 + 2/3	2
oai21	inv + 2×aoi21 + wire	1 + 2×2 + 6/3	8/3
aoi21	inv + 3×oai21 + wire	1 + 3×2 + 8/3	7/3
oai21	xnor + wire	12/3 + 2/3	8/3
xnor	aoi22 + wire	6/3 + 2/3	5
aoi22	nand3 + wire	5/3 + 2/3	4
nand3	buf + wire	1 + 2/3	3

*Buf is multi-stage due to large fan-out loading

$F = \Pi g \cdot b = 8.8 \times 10^6$; $P = \Sigma p = 34.7$
$N = \text{rnd}(\log_4 F) = 12$; $\alpha = 3.79$
$D = (N\alpha + P)/5 = (12 \times 3.79 + 34.7)/5 = 16.0$ FO4 delays

Critical path 2: through M_k comparators (output connected back onto input)

gate	load	$g \cdot b$	p
buf*	4×(nand2+nor2) + 5×(min+xor2) + wire	4×9/3 + 5×24/3 + 36/3	2
nand2	oai21 + wire	6/3 + 2/3	2
oai21	aoi21 + wire	6/3 + 4/3	8/3
aoi21	oai21 + wire	6/3 + 8/3	7/3
oai21	2×buf + wire	2×1 + 4/3	8/3
inv	nor2 + wire	5/3 + 2/3	1
nor2	2×buf + wire	2×1 + 4/3	2
buf*	56×(4×nand2) + wire	896/3 + 448/3	3
nand2	nand3 + wire	5/3 + 2/3	2
nand3	buf + wire	1 + 2/3	3

*Bufs are multi-stage due to large fan-out loading

$F = \Pi g \cdot b = 1.2 \times 10^8$; $P = \Sigma p = 22.67$
$N = \text{rnd}(\log_4 F) = 13$; $\alpha = 4.18$
$D = (N\alpha + P)/5 = (13 \times 4.18 + 22.67)/5 = 15.4$ FO4 delays

Interestingly, the synthesis results showed that the second path (through the comparators) was slightly the slower of the two, whereas the Logical Effort analysis has the path through the 8-bit subtractors as slightly slower. This reflects inaccuracies in the Logical Effort modelling approach (discussed later), but may also be due to less than optimal placement in the synthesised circuit resulting in longer wires and more buffers than assumed in the analysis. Indeed, in the synthesised macrocell, the sets of comparators were actually duplicated to help achieve timing closure.

4. Timing and Area comparisons

Fandrianto [9] was the first to publish a combined radix-4 SRT divide and square root unit, but this design employed a complicated digit selection algorithm and a PLA to provide an initial estimate of the radicand. Ercegovac and Lang [10] showed how the initial estimate PLA could be dispensed with and refined Fandrianto's digit selection technique. The cycle time of their design comprised the following delays:

- register load (& clocking)
- 4-to-1 multiplexer
- 3:2 carry-save adder
- digit selector (8-bit carry-propagate adder & 12-input logic network)

The 12-input logic network contains a short wordlength comparison which requires a carry-propagate subtraction, so that each SRT iteration contains two sequential short wordlength carry-propagate additions, considerably impacting on the cycle time. Harris *et al* described a variety of methods for accelerating radix-2 and radix-4 SRT division, achieved by overlapping non-dependent elements of successive iterations [11]. However, none of these methods can be readily extended to square root calculations because of the need to generate updated root estimates every cycle (Q_{i+1}^+ and Q_{i+1}^- in Figures 1 and 4 above) which have a dependency on the new result digit produced that same cycle. Most recently, two new approaches to designing low-power combined divide / square root units have both focussed on using radix-4 SRT [12,13]: however, these proposals have the equivalent of two back-to-back short-wordlength carry-propagate adders on their critical paths. The proposal of [13] also uses a retiming technique to reduce power consumption, and a block diagram of the critical path of the retimed architecture is presented in Figure 7. It comprises three main blocks:

- DSMUX / FGEN, which is akin to the F_k logic of Figure 1;
- CSA, which is a 3:2 carry-save adder;
- SEL, which provides the next quotient digit, q_{i+1}, and in turn comprises two sub-blocks – an 8-bit carry propagate adder and 4 6-bit comparators followed by a small logic network to return q_{i+1} in a zero/one-hot format.

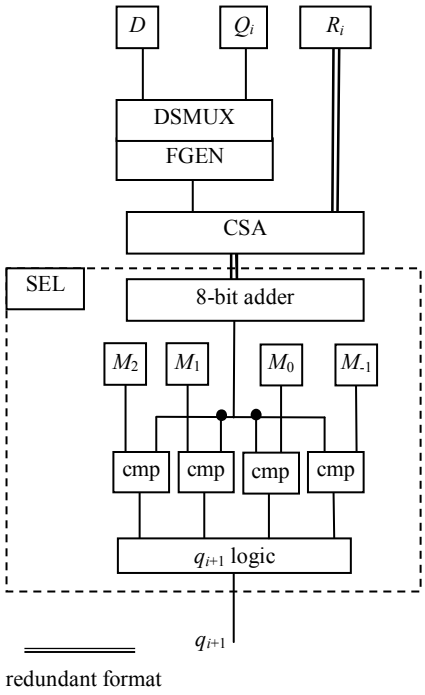

Figure 7 Block diagram of retimed low-power combined SRT divide / square root unit [13]

The FGEN unit comprises a 2-to-1 multiplexer (to select between D or Q_i depending on whether a division or square root operation is being performed) driving a 4-to-1 multiplexer (to return the new value of F_i depending on the most recent quotient digit). The output of the FGEN unit is connected to the fast input of the CSA unit. Finally, in SEL, the most significant 8 output digits of the CSA unit (in redundant format) are assimilated in an 8-bit carry-propagate adder and then broadcast to 4 6-bit comparators to derive the next value of q_{i+1} using logic similar to that in Figure 4. Logic diagrams of circuits unique to this architecture are presented below in Figures 8 and 9.

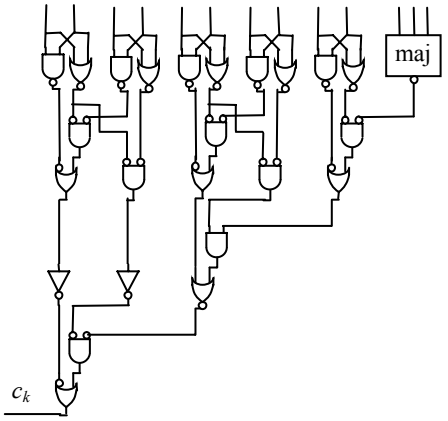

Figure 8 6-bit comparator

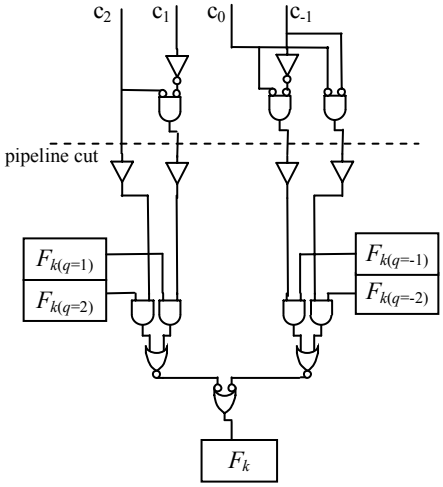

Figure 9 1-hot logic and 4:1 multiplexer for SRT divider

The Logical Effort analysis of this architecture proceeds as before by computing the capacitative load on each cell along the critical path of the logic circuit, assuming 2/3 inverter wire load per fan-out or per bit across a datapath element. The full analysis is laid out below:

Critical path: through FGEN, CSA, and SEL (output connected back onto input)

gate	load	$g \cdot b$	p
buf*	56 × mux2's + wire	56×2 + 56×2/3	3
mux2	4 × aoi22 + wire	4 × 6/3 + 8/3	4
aoi22	nand2 + wire	6/3 + 2/3	4
nand2	xnor2 + min + wire	12/3 + 12/3 + 4/3	2
xnor	nand2 + nor2 +wire	4/3 + 5/3 + 4/3	5
nor2	oai21 + wire	6/3 + 2/3	2
oai21	inv + 2×aoi21 + wire	1 + 2×2 + 6/3	8/3
aoi21	inv + 3×oai21 + wire	1 + 3×2 + 8/3	7/3
oai21	xnor + wire	12/3 + 2/3	8/3
xnor	4×min + wire	4×6/3 + 8/3	5
min	oai21 + wire	2 + 2/3	2
oai21	aoi21 + wire	2 + 4/3	8/3
aoi21	oai21 + wire	2 + 8/3	7/3
oai21	2×nor2 + wire	2×5/3 + 6/3	8/3
nor2	mux2 + wire	6/3 + 2/3	2

*Buf is multi-stage due to large fan-out loading

$F = \Pi g \cdot b = 9.1 \times 10^{11}$; $P = \Sigma p = 41.33$

$N = \text{rnd}(\log_4(F) = 20$; $\alpha = 3.96$

$D = (N\alpha + P)/5 = (20 \times 3.96 + 44.33)/5 = 24.7$ FO4 delays

The low-power design is more than 50% slower than the VFP11 design, mostly because of the two successive carry-propagate structures (8-bit adder and 6-bit comparator). Incidentally, [13] reported a logic delay (excluding flop and clock insertion delays) of 6.2ns in 0.6um CMOS, equivalent to 28.7 FO4 (assuming 1 FO4 = 360 × 0.6 = 216ps [14]). The theoretical result given by Logical Effort is optimistic compared to the synthesis result of [13], probably due to Logical Effort's ignoring of slew effects and its implicit assumption that logic gates are available in an infinite number of logic strengths.

By contrast, the ARM VFP11 unit has two well-matched critical paths, each with only one carry-propagate operation on them, allowing the design to fit in a processor with a shallow pipeline. This was achieved by using the next result digit, q_{i+1}, to select one of five speculative results; moreover, by performing an 8-bit carry-propagate addition across the msb's of the speculatively updated partial remainders, the result digit comparisons needed to derive the next result digit were also accelerated.

However, the speculative computations needed to accelerate the SRT iteration have come at a major hardware cost. Table 4 compares the cell counts of the VFP11 design with the low-power design counting 2 for XOR cells, 5 for flops, and assuming that the q_{i+1} and Q_{i+1} logic is negligible. The area of the low-power design is much smaller than that of the high-performance divide and square root macrocell because the 5 full-length carry-save adders that formed the speculative values of the new remainder in the VFP11 macrocell have been replaced by a single carry-save adder, and the four full-length multiplexers used to return the speculative values of F_k in the VFP11 macrocell have been replaced by one multiplexer in the low-power design. Moreover, the number and sizes of the required registers have been reduced in the low-power design. Excluding buffers, the overall hardware saving is around a factor of 4.5. This is an excellent illustration of the perennial trade-off in high-performance VLSI design between delay and area: in this case, in order to meet the required processor clock rate, quintuple parallel speculation was required; if the required clock rate were relaxed, considerable hardware savings could have been implemented because speculation would not have been necessary.

Table 4 Estimated cell counts for combined and divide-only macrocells

Logic block	# CMOS cells	ARM macrocell	Low-power unit [13]
F_k / q_kD logic (Figs. 4 & 9)	10 or 3 / bit	10×56 = 560	3×56 = 168
Comparators (Figs. 2 & 8)	26 or 20	26×4 = 104	20×4 = 104
8-b subtractor (Fig. 6)	65	65×5 = 325	65
(3:2) adders (Fig. 5)	5	5×(5×48) = 1200	5×56 = 280
half adders	3	3×(5×6) = 90	-
5-to-1 mux (Fig. 4)	14	54×14 = 756	-
Flops	5	5×(54×6 + 10×4) = 1820	5×(56 + 24 + 4) = 420
TOTAL (exc. buffers, etc)		4855	1037

5. Summary

This paper has presented the ARM VFP11 divide and square root unit, which makes use of partial remainder speculation to achieve a cycle time that is 50% faster than the best previous proposal. However, the amount of speculation required led to a significant increase in area when compared with a recently-proposed low-power design making use of a retiming technique. It may yet prove possible to combine the retiming technique with the speculation employed in this unit: after a small number of iterations, the top few bits of the root estimate do not change, thus facilitating both short-wordlength speculation and delayed long-wordlength reduction, as described recently in a hardware-efficient yet comparably fast divide-only architecture [15].

6. Acknowledgements

The authors wish to acknowledge the efforts of the reviewers in helping make the published version of this paper markedly better than the original submission.

7. References

[1] ARM. *VFP11 Vector Floating-Point Coprocessor Technical Reference Manual*, DDI 0274B, 2002.

[2] C.N. Hinds and D.R. Lutz, "Accelerating Floating-Point 3D Graphics for Vector Microprocessors", Proc. Asilomar Conference, Pacific Grove, CA, Nov. 2003, pp. 355-359.

[3] R.C. Agarwal, F.G. Gustavson, and M.S. Schmookler, "Series Approximation Methods for Divide and Square Root in the Power3TM Processor", Proc. IEEE Symp. Comp. Arith., Adelaide, Australia, April 1999, pp. 116 – 123

[4] N. Burgess and C. Hinds, "Design Issues in Radix-4 SRT Square Root and Divide Unit", Proc. 35th Asilomar Conference, Pacific Grove, CA, Nov. 2001, pp. 1646-1650

[5] C.N. Hinds and N. Burgess, "Apparatus and Method for Performing Operations Implemented by Iterative Execution of a Recurrence Equation", ARM Ltd., U.S. patent 7016930, 2006

[6] I.E. Sutherland, R.F. Sproull and D. Harris, *Logical Effort: Designing Fast CMOS Circuits*. San Francisco, CA: Morgan-Kaufman, 1999

[7] S. Knowles, "A Family of Adders", Proc. IEEE Symp. Comp. Arith., Vail, CO, June 2001, pp. 277 – 284

[8] N. Burgess, "New Models of Prefix Adder Topologies", J. VLSI Sig. Proc., 40, pp. 125-141 (2005)

[9] J. Fandrianto, "Algorithm for High Speed Shared Radix 4 Division and Radix 4 Square Root", Proc. IEEE Symp. Comp. Arith., Como, Italy, May 1987, pp. 73 – 79

[10] M.D. Ercegovac and T. Lang, "Radix-4 Square Root Without Initial PLA", IEEE Trans. Comp., 39, pp. 1016 - 1024 (1990)

[11] D.L. Harris, S.F. Oberman, M.A. Horowitz, "SRT Division Architectures and Implementations", Proc. IEEE Symp. Comp. Arith., Asilomar, CA, July 1997, pp. 18 – 25

[12] M. Kuhlmann, and K.K. Parhi, "Fast low-power shared division and square-root architecture", Proc. ICCD, Austin, TX, Oct. 1998, pp. 128-135

[13] A. Nannarelli and T. Lang, "Low-power radix-4 combined division and square root", Proc. ICCD, Austin, TX, Oct. 1999, pp. 236-242

[14] R. Ho, K.W. Mai and M.A. Horowitz, "The future of wires", Proc. IEEE, 89, pp. 490-504 (2001)

[15] E. Antelo et al, "Digit-recurrence Dividers with Reduced Logical Depth", IEEE Trans. Comp., 54, pp. 837-851 (2005)

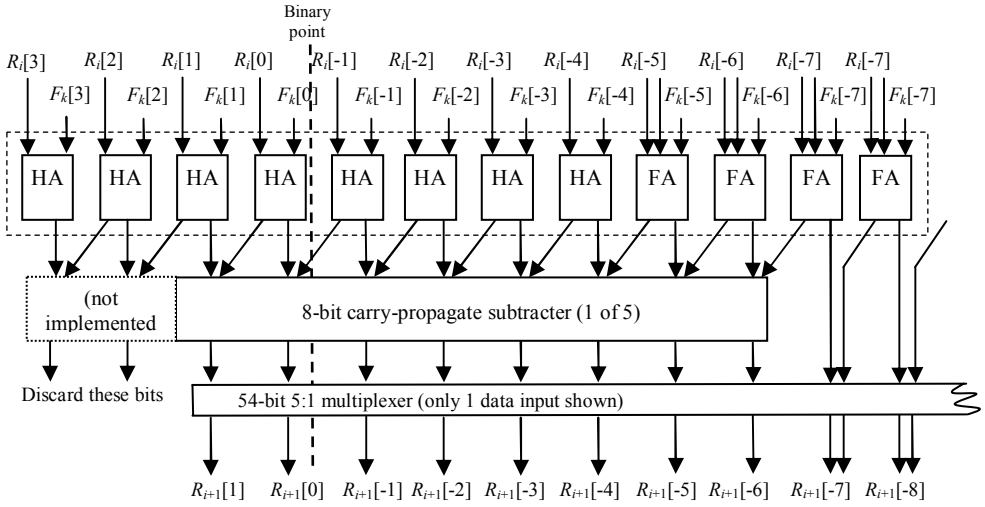

Figure 3 M.s.b.'s of R_{i+1} adder and 5:1 multiplexer

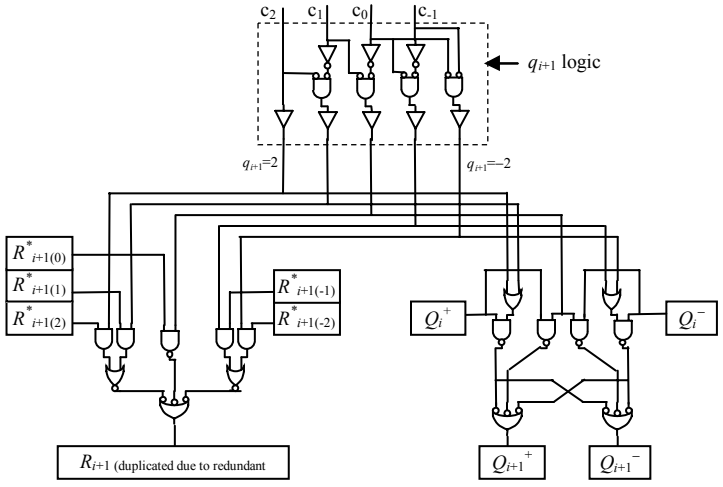

Figure 4 Logic diagram of q_{i+1} 1-hot encoding and 5:1 multiplexers