Using the Reverse-Carry Approach for Double Datapath Floating-Point Addition*

Javier D. Bruguera
Dept. Electronic and Computer Engineering
University of Santiago de Compostela, Spain
bruguera@dec.usc.es

Tomás Lang
Dept. Electrical and Computer Engineering
University of California at Irvine, USA
tlang@uci.edu

Abstract

The double-datapath organization of a floating-point adder results in a reduced latency. One of the main characteristics of this organization is the combination of the addition/subtraction with the rounding into a single Add/Round module, which is implemented as one pipeline stage and might be responsible for the cycle time. We propose the utilization of the most-significant carry detector and the corresponding adder using the reverse-carry approach to reduce the latency of this Add/Round module. In addition, the particular organization of the reverse-carry adder is used to reduce the contribution on the delay of the row of half adders that is included in the FAR datapath to produce the sum plus two. Estimates for a 64-bit Add/Round module show a potential reduction of delay of about 15%.

1. Introduction

In contemporary processors, the latency of floating–point addition/subtraction is of several cycles and it affects the performance of important applications. Consequently, several architectural improvements have been included to reduce this latency: double datapath organization, leading zero anticipator, compound adder, fast calculation of the exponent difference, among others, as summarized for instance in [1, 12, 14]. In particular, several processors implement a double datapath organization [6, 10]. In this organization, the addition/subtraction is combined with the rounding into a single Add/Round module, which is implemented as one pipeline stage and might determine the cycle time.

*This work was supported in part by the *Secretaria Xeral de Investigacion e Desenvolvemento de Galicia* (Spain) under contract PGIDT99-PXI20602B and by NSF grant MIP-9726244.

In this paper we propose the utilization of the most-significant carry detector and of the corresponding adder using the reverse-carry approach [2, 8], to reduce the latency of the Add/Round module. In addition, we utilize the particular organization of the reverse-carry adder to reduce the contribution on the delay of the row of half adders that is included in the Add/Round module of the FAR datapath to produce the sum plus two required in the directed rounding modes [11].

To estimate the potential improvement of this proposal we estimate the delay using the characteristics of a family of standard cells. In the estimation we incorporate the effect of the load and of the delay of the connections. We compare the delay with an estimate for the traditional implementation of the Add/Round module and conclude that the proposed architecture has the potential of reducing the delay of the Add/Round module by about 15%.

2. Double–datapath floating–point adder and implementation of Add/Round

In the double-datapath floating-point adder the datapath is split into two paths [5]: the *CLOSE* datapath that computes the effective subtractions for an exponent difference $|d| \leq 1$, and the *FAR* datapath that computes all the effective additions and the effective subtractions for an exponent difference $|d| > 1$. In this way the full–length alignment shift and the full-length normalization shift are mutually exclusive: the FAR datapath requires a full alignment shift and the normalization shift is, at most, of 1 bit (right shift for effective addition and left shift for effective subtraction); whereas the CLOSE datapath requires a full normalization shift and only a 1–bit alignment shift. For implementation details and improvements see [12, 10, 14].

The latency of the floating–point addition is reduced by combining the rounding with the addi-

0-7695-1150-3/01 $10.00 © 2001 IEEE

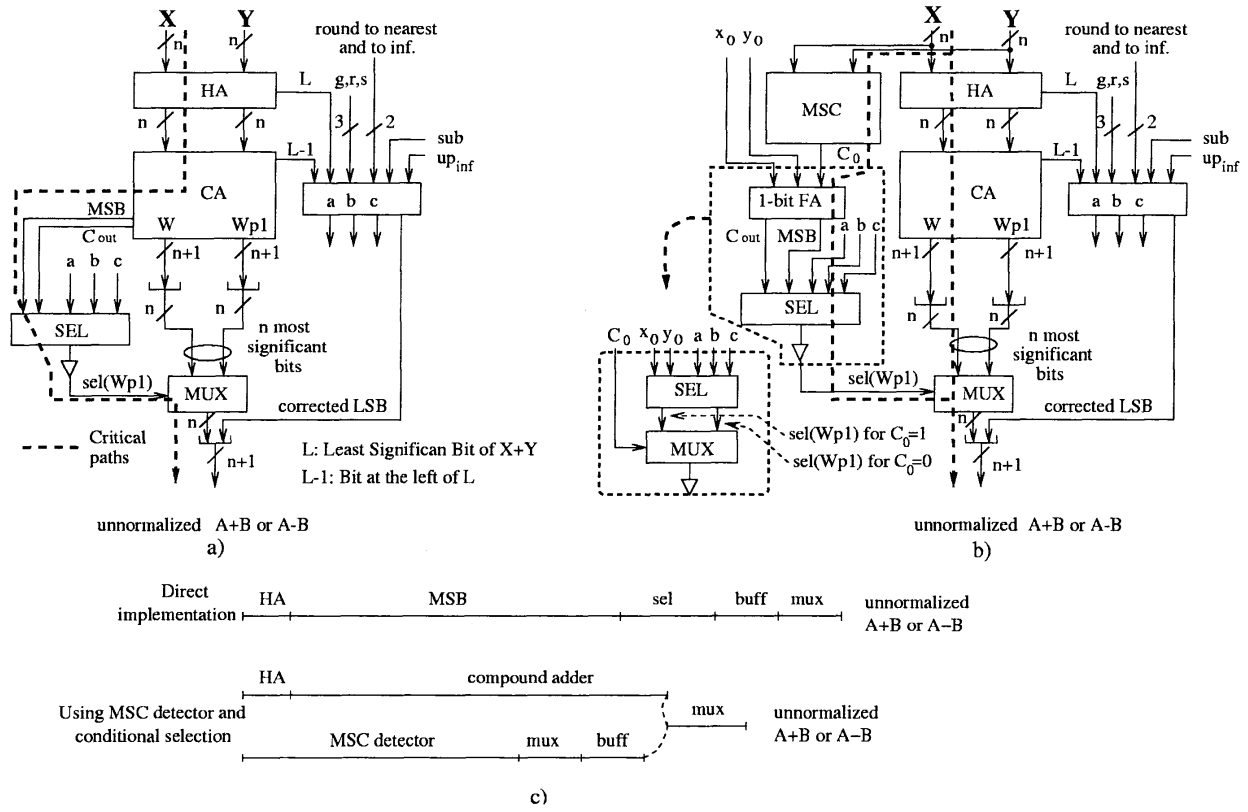

Figure 1. Add/Round for the FAR datapath. a) Direct implementation. b) Using a MSC detector. c) Time diagrams

tion/subtraction (**Add/Round**). In this way, the rounding is performed before the normalization and this fact has to be taken into account to determine the rounding bits. The combined addition uses a *compound adder* that computes simultaneously the sum and the sum plus one. A simple implementation of this adder corresponds to a prefix–tree adder plus an additional level of OR gates. Then, the correct rounded result is obtained by selecting one of the outputs according to the requirements of the rounding. In the rounding to $\pm\infty$ modes the sum plus two result is also required in the FAR datapath but it can be computed using the compound adder preceded by a row of half adders.

There are several alternatives for the pipelining of the double–datapath floating–point adder depending on the technology and clock frequency. The number of stages can be either two [1, 14], three [6, 12] or four [10]. However, in all cases, an important component in the determination of the number of stages and/or the stage delay is the Add/Round module in the FAR datapath. This module is also important in the CLOSE datapath but due to the n–bit HA (to compute the sum plus two) and the rounding complexity the delay of the FAR datapath is larger. Therefore we concentrate in reducing the delay of the Add/Round module in the FAR datapath, but similar reductions are possible in the CLOSE datapath.

2.1. Add/Round in the FAR datapath

We discuss the implementation of the Add/Round module in the FAR datapath since in the following sections we propose an alternative to reduce its delay. The operands have normalized significands of n bits, being bit 0 the most–significant bit and bit $n-1$ the least–significant bit. Figure 1(a) shows a direct implementation of the Add/Round module. We denote as X and Y the inputs to the adder after alignment and bit–inversion. Signal *sub* is set for effective subtraction and up_∞ indicates if the adder result has to be rounded–up in the rounding to $+\infty$ and $-\infty$ modes[1].

[1]For a complete description of the control signals see [3]

For the rounding it is necessary to produce either $X+Y$, $X+Y+1$, or $X+Y+2$. For this purpose, the combination of HAs and compound adder produce Wp and $Wp1$ which are either $X+Y$ and $X+Y+1$ or $X+Y+1$ and $X+Y+2$ and then the correct result is selected [11].

Note that in Figure 1 the selection logic has been organized into two levels: a first level which uses the signals that are obtained early in the FAR datapath and a second level which uses the output of the first level (a, b and c) and the C_{out} and the MSB of the addition[2]. The second level corresponds to

$$sel(Wp1) = a \cdot \overline{C_{out}} + C_{out}(b + c \cdot MSB) \quad (1)$$

Finally, the LSB of the addition/subtraction is obtained as $LSB(corrected) = sel(Sp1) \oplus L$. In this way, the slowest path is the calculation of $sel(Wp1)$. Consequently, the delay of the Add/Round module is

$$t_{a/r} = t_{HA} + t_{MSB} + t_{sel} + t_{bf} + t_{mx} \quad (2)$$

being t_{HA} and t_{MSB} the delay of the row of HAs and of the calculation of the MSB, respectively, and t_{sel}, t_{bf} and t_{mx} the delay of the selection, the buffer (for the control signal of the wide multiplexer) and the multiplexer, respectively.

This delay is reduced if the C_{out} and the MSB are calculated in parallel with the operation of the compound adder using a Most Significant Carry (MSC) detector, as shown in Figure 1(b). The MSC detector, implemented as a P,G tree, permits to know the carry of an addition/subtraction before the result of the addition. The MSC block generates C_0, the carry into position 0, which is used to compute the MSB and C_{out}. Note that its inputs are taken before the HAs; although the HAs pre–adds a 1, this does not affect the most–significant carry, since the 1 is pre–added only when the least–significant bit is 0.

If the critical path is still the *sel* signal (and not the compound adder), a further reduction can be obtained with a conditional $sel(Wp1)$ logic, as shown in the dashed box of Figure 1(b). In this way

$$t_{a/r} = max\{(t_{HA}+t_{CA}), (t_{MSC}+t_{mx}+t_{bf})\}+t_{mx} \quad (3)$$

where t_{CA} and t_{MSC} are the delay of the compound adder and the MSC block, respectively.

Figure 1(c) illustrates the delay of each of the implementations discussed in this section. Note that the critical path delay reduction obtained using a MSC detector and conditional selection depends on the delay of the compound adder and the MSC detector used.

[2]the expressions for signals a,b, and c are given in [3]. We do not include them here because these signals are not in the critical path

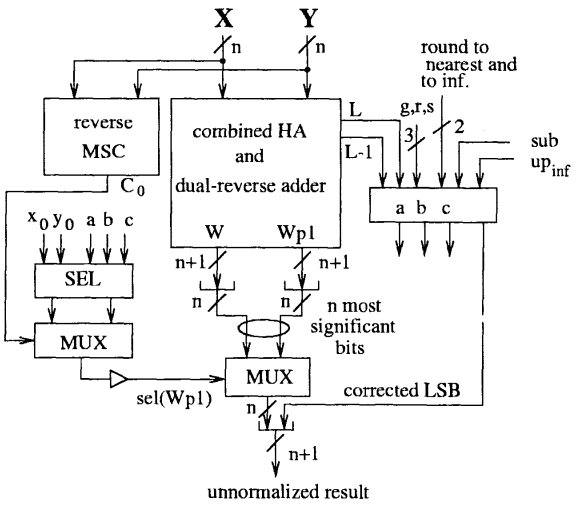

Figure 2. Add/Round in FAR datapath with reverse MSC and combined HA and dual reverse adder

3. Reverse Add/Round module

We now get to the main contribution of this paper, namely, the use of the reverse-carry approach in both the MSC and the compound adder and the incorporation of the HA array into the reverse compound adder. The multilevel reverse–carry algorithm [8] is a technique for early detection of the most–significant–carry (MSC) of an addition/subtraction. This has been then extended to implement a fast adder [2], called *reverse–carry adder*, that reduces the delay of the addition/subtraction with respect to the traditional prefix adders. We call this adder *dual reverse adder*. The indicated modifications to the implementation of Figure 1(b) result in the implementation of Figure 2. It also includes the combination of the half adders with the reverse dual adder, as presented in the next section.

The utilization of the reverse–carry algorithm in the MSC detector and the compound (or dual) adder in the Add/Round module should produce a reduction in the overall delay because these modules result in smaller delays than those of the conventional implementation. Its delay is given by the expression

$$t_{a/r} = max\{(t_{HA} + t_{DRA}), (t_{RMSC} + t_{mx} + t_{bf})\} + t_{mx} \quad (4)$$

being t_{DRA} and t_{RMSC} the delay of the dual reverse adder and the reverse MSC, respectively. Note that this equation is similar to equation (3), but replacing the compound adder and the MSC detector by the dual reverse adder (DRA) and reverse MSC detector

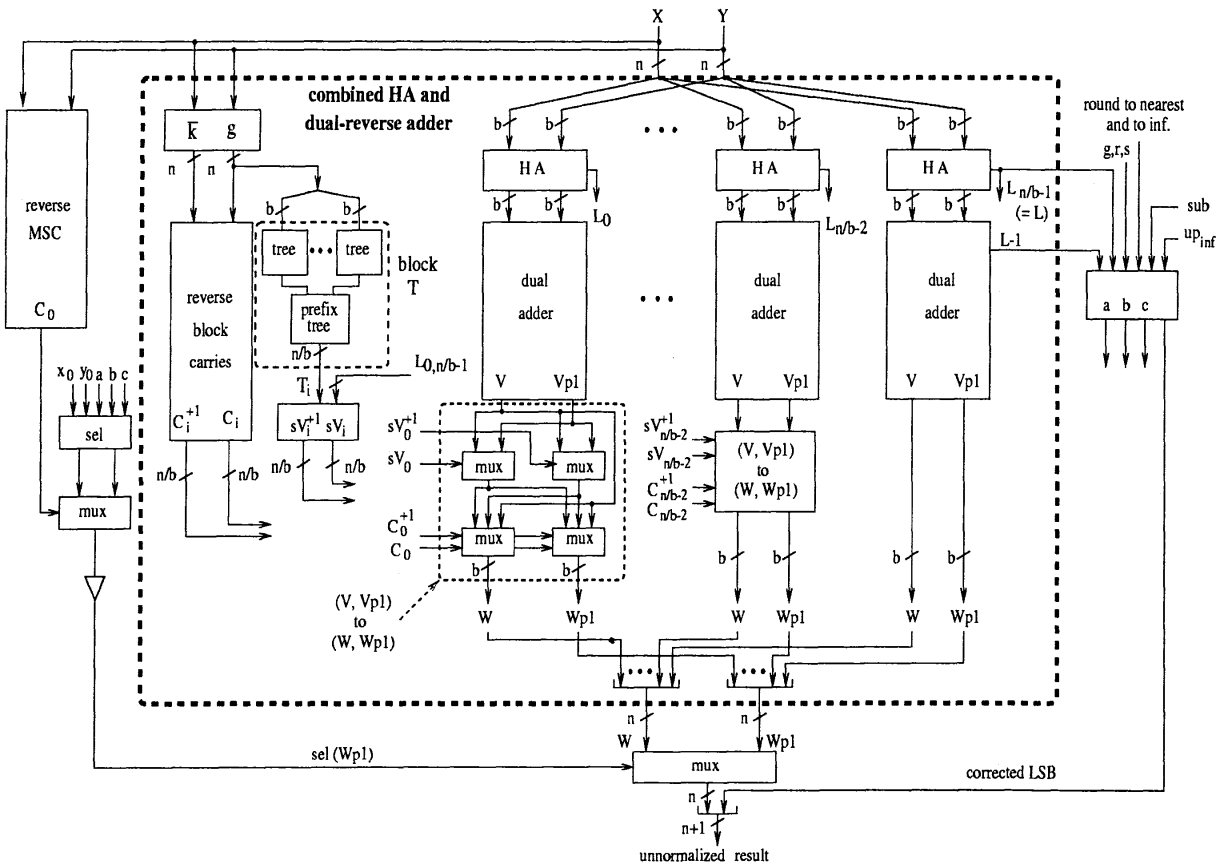

Figure 3. Reverse Add/Round module with combined HA

(RMSC), respectively.

3.1. Eliminating the HAs from the critical path

If the critical path goes through the dual reverse adder then the delay can be further reduced by eliminating the half adders from the critical path. This can be achieved by the implementation of Figure 3 in which the half adders and the dual reverse adder are combined.

To explain this implementation we summarize the pertinent characteristics of the reverse-carry adder [2]. The adder is divided into b–bit dual adder blocks, each of which produces the block sum, V, and sum plus 1, $Vp1$. Then the selection between V and $Vp1$, to produce W and $Wp1$, is controlled by the carries into the blocks, which are computed using the reverse-carry approach[3].

In this implementation the critical path goes through the module that produces the block carries. Consequently, the total delay of *HA + dual adder* is reduced if the inputs to the block-carries module comes directly from the operands X and Y, instead of from the output of the half adders. Here we give a high-level description of this combination, and in the next subsection we go into more detail.

Each block operates as the complete *HA + dual adder* in the implementation of Figure 2. Then, the block produces either $X^b + Y^b$ and $X^b + Y^b + 1$ or $X^b + Y^b + 1$ and $X^b + Y^b + 2$, being X^b and Y^b b–bit blocks of X and Y and L_i the LSB of $X^b + Y^b$, so that the three required block outputs can be generated by suitably modifying the least-significant bit[4].

[3]To achieve a dual adder, producing the sum and sum plus 1, the reverse-carry adder is modified as described in the report mentioned together with [2] in the references

[4]The requirement of the block output $X + Y + 2$ is explained below

Now, to produce the overall outputs, it is necessary to use the block carries for selection. These carries are C and C^{+1}, which select V or $Vp1$ to produce W and $Wp1$. However, now there is an additional complication because for the calculation of $X+Y+2$ there might be two carries into a block[5]. However, there are two carries into block i only when all bits of the previous blocks of X and Y are 1. Consequently, we define a variable T_i to detect this situation and select the block output $X+Y+2$ for $Wp1$ when $T_i = 1$.

The delay of this implementation is:

$$t_{a/r} = max\{(t_{comb}), (t_{RMSC} + t_{mx} + t_{bf})\} + t_{mx} \quad (5)$$

where t_{comb} is the delay of the combined HA and dual reverse adder. In this way, the delay of the reverse FAR Add/Round module is reduced if $t_{comb} < t_{HA} + t_{DRA}$ (see equation (4)).

The architecture reduces the delay with respect to a traditional implementation (n–bit HA followed by n–bit dual adder) since the carries of the dual adder are obtained from the input operands to the n–bit HA. Note that the block carries obtained before the HAs are the same as if the block carries were obtained after the HAs. In this way, considering that the carries calculation is the critical path of the dual adder, the n–bit HA are eliminated from the critical path.

Implementation

The combined HA and dual reverse adder (see Figure 3) provides two outputs (W, $Wp1$) which correspond to ($X+Y$, $X+Y+1$) or to ($X+Y+1$, $X+Y+2$). As said before, the combined HA and dual reverse adder is split into blocks and the reverse–carry is used only to compute the input carries to the blocks. Signal T_i is used to specify when $X_i^b+Y_i^b+2$ is needed as output of the block dual adder. Actually, T_i indicates that two carries are propagated to block i, and this only occurs when the dual adder is computing $X+Y+2$ and $x_j \cdot y_j = 1$ for $j = n-1$ up the most significant bits of block $i-1$. Note that when $T_i = 1$ then $C_i = C_i^{+1} = 1$.

Table 1 summarizes the block output necessary for the calculation of $X+Y$, $X+Y+1$ and $X+Y+2$. Note that $X+Y$ and $X+Y+1$ only require $X_i^b + Y_i^b$ or $X_i^b + Y_i^b + 1$, depending on the input carries, whereas $X+Y+2$ requires $X_i^b+Y_i^b+2$ too. Although the selection requires all three values, each block only provides two outputs,(V, $Vp1$), which correspond to ($X_i^b+Y_i^b$, $X_i^b+Y_i^b+1$) or ($X_i^b+Y_i^b+1$, $X_i^b+Y_i^b+2$),

[5]This problem is solved by the set of half adders in the traditional approach; here it reappears because we use directly X and Y to compute the block carries.

Table 1. Block output selection ($X_i^b + Y_i^b$ or $X_i^b + Y_i^b + 1$ or $X_i^b + Y_i^b + 2$)

T_i	C_i	C_i^{+1}	$X+Y$	$X+Y+1$	$X+Y+2$
0	0	0	$X_i^b + Y_i^b$	$X_i^b + Y_i^b$	$X_i^b + Y_i^b$
0	0	1	$X_i^b + Y_i^b$	$X_i^b + Y_i^b + 1$	$X_i^b + Y_i^b + 1$
0	1	–	$X_i^b + Y_i^b + 1$	$X_i^b + Y_i^b + 1$	$X_i^b + Y_i^b + 1$
1	–	–	$X_i^b + Y_i^b + 1$	$X_i^b + Y_i^b + 1$	$X_i^b + Y_i^b + 2$

Table 2. Selection of W and $Wp1$ in each block

T_i	C_i	C_i^{+1}	W	$Wp1$	LSB correction
0	0	0	V	V	Correct. in W and $Wp1$
0	0	1	V	V	No correction
0	1	–	V	V	No correction
1	–	–	V	$Vp1$	No correction

Since $L_i = 0$ $(V,\ Vp1) = (X^b + Y^b + 1,\ X^b + Y^b + 2)$
Since $L = 0$ $(W,\ Wp1) = (X + Y + 1,\ X + Y + 2)$

a) $L = 0$ and $L_i = 0$

T_i	C_i	C_i^{+1}	W	$Wp1$	LSB correction
0	0	0	V	V	No correction
0	0	1	$Vp1$	$Vp1$	No correction
0	1	–	$Vp1$	$Vp1$	No correction
1	–	–	$Vp1$	$Vp1$	Correction in $Wp1$

Since $L_i = 1$ $(V,\ Vp1) = (X^b + Y^b,\ X^b + Y^b + 1)$
Since $L = 0$ $(W,\ Wp1) = (X + Y + 1,\ X + Y + 2)$

b) $L = 0$ and $L_i = 1$

T_i	C_i	C_i^{+1}	W	$Wp1$	LSB correction
0	0	0	V	V	Correct. in W and $Wp1$
0	0	1	V	V	Correction in W
0	1	–	V	V	No correction
1	–	–	–	–	

Since $L_i = 0$ $(V,\ Vp1) = (X^b + Y^b + 1,\ X^b + Y^b + 2)$
Since $L = 1$ $(W,\ Wp1) = (X + Y,\ X + Y + 1)$

c) $L = 1$ and $L_i = 0$

T_i	C_i	C_i^{+1}	W	$Wp1$	LSB correction
0	0	0	V	V	No correction
0	0	1	V	$Vp1$	No correction
0	1	–	$Vp1$	$Vp1$	No correction
1	–	–	–	–	

Since $L_i = 1$ $(V,\ Vp1) = (X^b + Y^b,\ X^b + Y^b + 1)$
Since $L = 1$ $(W,\ Wp1) = (X + Y,\ X + Y + 1)$

d) $L = 1$ and $L_i = 1$

depending on the value of $L_i = LSB(X_i^b) \oplus LSB(Y_i^b)$. Namely,

$$(V,\ Vp1) = \begin{cases} (X_i^b + Y_i^b,\ X_i^b + Y_i^b + 1) & \text{if } L_i = 1 \\ (X_i^b + Y_i^b + 1,\ X_i^b + Y_i^b + 2) & \text{if } L_i = 0 \end{cases}$$

In both cases, to obtain the third result, $X_i^b + Y_i^b + 2$ when $L_i = 1$ or $X_i^b + Y_i^b$ when $L_i = 0$, the least–significant bit of the block output has to be corrected. Therefore, the block W and $Wp1$ are obtained from V and $Vp1$ depending on L_i and L. Table 2 summarizes the selection for W and $Wp1$ in each block of the combined HA and dual reverse adder. From the Table, we obtain:

$$\begin{aligned} sel(Vp1)_i^W &= L \cdot L_i \cdot C_i + \bar{L} \cdot L_i \cdot C_i^{+1} \quad (6) \\ sel(Vp1)_i^{Wp1} &= \bar{L} \cdot T_i + L_i \cdot C_i^{+1} \end{aligned}$$

Figure 3 shows the implementation of the combined HA and dual reverse–adder. The carries are calculated using the reverse–carry algorithm in parallel with the b–bit dual adders and the calculation of the T_is. Note that the implementation of equations (6) would introduce an additional delay in the carries calculation path, since these equations should be implemented after the calculation of the C_i and C_i^{+1}. To reduce the delay, equation (6) has been implemented in two levels, composed of a multiplexer each. The first level is a conditional selection, with control signals sV_i^{+1} and sV_i, whereas the carries are used as control signals in the second–level multiplexer to complete the selection. Moreover, those multiplexers implement the block LSB correction, as indicated in Table 2; note that the correction, similarly to the selection, depends only on sV_i^{+1}, sV_i, C_i and C_i^{+1}.

With these considerations the critical path of the combined HA and dual reverse adder is

$$t_{comb} = max\{(t_{\bar{k}g} + t_c), (t_{HA} + t_{DAb} + t_{mx})\} + t_{mx} \quad (7)$$

being t_c the delay of the C_i and C_i^{+1} calculation and t_{DAb} the delay of the b-bits dual adder.

4. Estimation of delay and comparison with traditional implementation

In this section we estimate the delay of the proposed reverse Add/Round architectures and compare it with traditional implementations. There are several implementation alternatives depending on the number of bits (n), the block size (b), the type of gates and cells, the technology, etc. Consequently, for a specific set of requirements and constraints, the designer should explore the design space for the best implementation. Here, to show the potential benefits of the approach, we consider 64–bits adder[6] and use gates from a 0.5μm CMOS standard-cell library [9]. We have included the load of gates and an estimate of the effect of the long wires. It has to be pointed out that this is only an *approximate* estimation, since the delay is heavily dependent on the technology and considering other technology and/or standard cells libraries the delay estimation could be different.

We use as unit of delay the delay of a 2-input NAND with a load of 2. As we are interested in obtaining only an estimation of the delay, and not in exact delays figures, the results are approximated with one fractional digit. The delay estimation is performed considering the actual load of each gate, which consists of the load produced by the gates connected to the corresponding output plus the effect of the wires. Long wires can introduce significant loads on the gates [7] and may require the use of buffers and/or high output drive strength gates. To obtain a rough estimate of the effect of the wires, we consider the delay evaluation of prefix adders given in [7]. Following that evaluation we estimate that the additional load of a connection of length[7] 4 (called L_4) is equal to 3 [2]. However, since the wire load is dependent on the technology, we consider a range of wire loads, namely from $L_4 = 0$ to $L_4 = 5$. In the following we estimate the delay of the Add/Round module for $L_4 = 0$ and $L_4 = 3$ and summarize for L_4 from 0 to 5.

4.1. Delay estimation

We estimate the delay of the reverse Add/Round module (in the FAR datapath) without ($t^{w/o}$) and with combined HA and dual reverse adder (t^w). The delay of both alternatives is given by equations (4) and (5), respectively, in section 3:

$$t^{w/o} = max\{(t_{HA} + t_{DRA}), (t_{RMSC} + t_{mx} + t_{bf})\} + t_{mx}$$

$$t^w = max\{(t_{comb}), (t_{RMSC} + t_{mx} + t_{bf})\} + t_{mx}$$

In both cases, the reverse MSC and the reverse dual adder are important components of the critical path. The delays of these two elements, for an implementation with 64 bits, have been estimated in [8] and [2] and are summarized in Table 3 for different wire loads. The table also shows the delay of the traditional implementations of the MSC detector and the compound adder. Moreover, it shows the delay of the combined

[6]The IEEE double precision format uses 53 bits in the significand, but we have done the estimations with 64 bits since this is the nearest power of two

[7]The length is measured as the number of bits positions between the beginning and the end of the line

Table 3. MSC detector and dual adder delays

	$L_4=0$	$L_4=1$	$L_4=3$	$L_4=5$
rev. MSC	8.0	9.5	10.2	11.2
trad. MSC	9.0	10.7	12.7	13.7
dual rev. adder	12.7	14.0	16.0	17.5
compound adder	15.2	17.5	19.5	21.7
combined HA and dual rev. adder	14.8	15.0	16.0	17.5

Table 4. Add/Round module delay

Add/Round mod.	$L_4=0$	$L_4=1$	$L_4=3$	$L_4=5$
reverse approach				
without comb. HA	16.7	18.0	20.0	22.3
with comb. HA	16.8	17.0	19.4	22.3
Trad. (non–rev.)	19.2	21.5	23.5	25.7

L_4 is the load of a connection of length 4

HA and dual reverse adder (see Figure 3). For low wire loads the critical path is composed of the HAs plus the 8–bits dual adder plus the multiplexers so that its delay is larger than the delay of the dual reverse adder. However, for large wire loads the critical path corresponds to the carries calculation path, resulting in the same delay as for the reverse dual adder.

With respect to the remaining components, mux and buffers, the wire length only affects the load of the buffer in the $sel(Wp1)$ line. This buffer has to drive a 64–bits multiplexer, with a gate load of 32, and a wire load due to a connection of length 64. For $L_4=0$ the load of the buffer is 32; whereas for $L_4=3$ the load is $L=32+L_{64}=80$, being $L_{64}=48$ the load introduced by the connection of length 64. With this load, the delay of the buffer is too large (6.2 t_{nd2a}), but this delay is reduced if two additional buffers are placed to distribute the load. In this way the delay is reduced to $t_{buff}^{L=80}=5.2\ t_{nd2a}$. The delays of the two schemes are as follows:

- Reverse Add/Round module without combined HA and dual reverse adder ($t^{w/o}$, see equation (4))

For $L_4=0$

$$max\{(t_{HA}+t_{DRA}),(t_{RMSC}+t_{mx}+t_{bf}^{L=32})\}+t_{mx}= max\{(2+12.7),(8+2+2.5)\}+2=16.7\ t_{nd2a}$$

For $L_4=3$

$$max\{(t_{HA}+t_{DRA}),(t_{RMSC}+t_{mx}+t_{bf}^{L=80})\}+t_{mx}= max\{(2+16.0),(10.2+2+5.2)\}+2=20.0\ t_{nd2a}$$

- Reverse Add/Round module with combined HA and dual reverse adder (t^{w}, see equation (5))

For $L_4=0$

$$max\{(t_{comb}),(t_{RMSC}+t_{mx}+t_{bf}^{L=32})\}+t_{mx}= max\{(14.8),(8+2+2.5)\}+2=16.8\ t_{nd2a}$$

For $L_4=3$

$$max\{(t_{comb}),(t_{RMSC}+t_{mx}+t_{bf}^{L=80})\}+t_{mx}= max\{(16.0),(10.2+2+5.2)\}+2=19.4\ t_{nd2a}$$

The estimation results for other wire loads are shown in Table 4. The combined HA and dual adder contributes to reduce the delay of the reverse Add/Round module for intermediate wire loads ($L_4=1$ and 3). This improvement, with the technology considered, is about 6%. Note that the differences between both approaches is that when the combined HA and dual reverse adder is used, the HAs are eliminated from the critical path (carries calculation in the dual reverse adder). Then, the maximum (ideal) delay improvement is about 10% due to the elimination of the HAs delay (2 t_{nd2a}) of the critical path delay.

For very large wire loads ($L_4\geq 5$) the delay of both Add/Round implementations are equal, since the critical path is the calculation of the $sel(Wp1)$ signal (reverse MSC plus mux plus buffers) instead of the path through the dual reverse adder.

4.2. Comparison

We compare the reverse Add/Round FAR module implementation with the traditional implementation using a (non–reverse) MSC detector and conditional selection[8] (see Figure 1(b)). The delay is given by equation (3):

$$t=max\{(t_{HA}+t_{CA}),(t_{MSC}+t_{mux}+t_{buff})\}+t_{mux}$$

The last row in Table 4 shows the delay for different wire loads. In our estimations, the critical path for this implementation goes through the adder. This is not the case for the implementation of Figure 1(a) nor for that of Figure 1(b) without the conditional selection, which justifies the implementation we chose for comparison.

For all wire loads, the delay is reduced by using the reverse–carry approach. When wire loads are not considered the delay reduction is about 13%. But the delay reduction is larger when wire loads are considered, about 17% for intermediate loads and about 13% for large wire loads, $L_4\geq 5$. This shows that although

[8] Although the implementations we are aware of do not include these improvements

the delay is reduced, the actual reduction achieved is technology dependent.

With respect to the hardware complexity, the estimations done in [8] and [2] for the reverse–carry MSC detector and the reverse dual adder, respectively, show that the hardware requirements of both the reverse–carry MSC and the reverse dual adder are quite similar to their traditional (non–reverse) counterparts.

Therefore, the hardware complexity of the traditional and the reverse (without combined HA and dual reverse adder) Add/Round implementations are similar. However, in the combined reverse implementation the logic for the calculation of the the two sets of carries, C_i and C_i^{+1}, is separated from the carries calculation of the b–bits block dual adders, resulting in some hardware duplication. Moreover, the logic for the T_is calculation has been added.

As a consequence, the hardware complexity is somewhat larger than in the other implementations considered.

5. Conclusions

In this paper we have proposed an architecture of the Add/Round module for double-datapath floating-point adders. This proposal is based on the utilization of the multilevel reverse–carry algorithm in the implementation of the most-significant carry detector and of the dual adder.

The multilevel reverse–carry approach permits to reduce the delay of the MSC detector and the dual adder since they are split into levels and the operation of consecutive levels can be overlapped. This approach is used in both the *CLOSE* and the *FAR* datapaths. Moreover, in the *FAR* datapath we eliminate the row of half adders (HA) from the critical path by incorporating the HAs as part of the dual adder.

We have estimated the delay of the resulting architecture, the reverse Add/Round module for the FAR datapath, using the characteristics of a family of standard cells and considering the load introduced by long connections. The comparison with the traditional implementation of this module shows that the proposed architecture can reduce the delay by about 15%.

As mentioned by the reviewers, since the rounding stage of floating-point multiplication also includes a compound adder and a row of half adders [4, 13, 15], the improvements presented in this paper might also be beneficial in that situation.

References

[1] A. Beaumont-Smith, N. Burgess, S. Lefrere, and C. C. Lim. Reduced latency ieee floating–point adder architectures. In *Proc. IEEE 14th Int. Symp. Computer Arithmetic (ARITH14)*, pages 35–42, 1999.

[2] J. D. Bruguera and T. Lang. Multilevel reverse–carry adder. In *Proc. Int. Conf. Computer Desig (ICCD00)*, pages 155–162, 2000.

[3] J. D. Bruguera and T. Lang. Rounding in floating–point addition. Technical report, University of Santiago de Compostela (available at http://www.ac.usc.es), 2000.

[4] G. Even and P.-M. Seidel. A comparison of three rounding algorithms for ieee floating-point multiplication. *IEEE Trans. Computers*, 49(7):638–650, 2000.

[5] P. M. Farmwald. *On the Design of High Performance Digital Arithmetic Units.* PhD thesis, Stanford University, 1981.

[6] D. Greenley. Ultrasparc: The next generation superscalar 64–bit sparc. In *Proc. Digest of Papers COMPCON-95*, pages 442–451, 1995.

[7] S. Knowless. Family of adders. In *Proc. IEEE 14th Symp. Computer Arithmetic (ARITH14)*, pages 30–43, 1999.

[8] T. Lang and J. D. Bruguera. Multilevel reverse–carry computation for comparison and for sign and overflow detection in addition. In *Proc. Int. Conf. Computer Design (ICCD99)*, pages 73–79, 1999.

[9] LSI LOgic. *LCB500K Design Manual*, 1995.

[10] S. Oberman, G. Favor, and F. Weber. Amd 3dnow! technology: Architecture and implementations. *IEEE Micro*, 19(2):37–48, 1999.

[11] N. T. Quach and M. J. Flynn. An improved algorithm for high–speed floating–point addition. Technical report, Stanford University, 1990.

[12] H. A.-T. S. F. Oberman and M. J. Flynn. The snap project: Design of floating–point arithmetic units. In *Proc. IEEE 13th Symp. Computer Arithmetic (ARITH13)*, pages 156–165, 1997.

[13] M. R. Santoro, G. Bewick, and M. A. Horowitz. Rounding algorithms for ieee multipliers. In *Proc. IEEE 9th Symp. Computer Arithmetic (ARITH9)*, pages 176–183, 1989.

[14] P.-M. Seidel and G. Even. How many logic levels does floating–point addition require? In *Proc. IEEE Int. Conf. Computer Design (ICCD98)*, pages 142–149, 1998.

[15] R. K. Yu and G. B. Zyner. 167 mhz radix–4 floating point multiplier. In *Proc. IEEE 12th Symp. Computer Arithmetic (ARITH12)*, pages 149–154, 1995.