TIME-PRECISION FLEXIBLE ADDER

Juan Manuel García Chamizo, Jerónimo Mora Pascual, Higinio Mora Mora
Departamento de Tecnología Informática y Computación
Universidad de Alicante, Spain
{juanma, jeronimo, hmora }@dtic.ua.es

ABSTRACT*

A new conception of flexible calculation that allows us to adjust a sum depending on the available time computation is presented. More specifically, the objective is to obtain a calculation model that makes the processing time/precision more flexible. The addition method is based on carry-select scheme adder and the proposed design uses precalculated data stored in look-up tables, which provide, above all, quality results and systematization in the implementation of low level primitives that set parameters for the processing time. We report an evaluation of the architecture in area, delay and computation error, as well as a suitable implementation in FPGA to validate the design.

1. INTRODUCTION

There are a great number of applications that are difficult to fit into the rigid schemes of the calculation of conventional arithmetic architectures. For these applications it would be advantageous to have operators that provide control on the results and act on the even quality of the result and processing cost based on the specific computational requirements of each case [1], [2].

We can find several examples in which an intensive processing of data provided by peripheral takes place. In these cases, a strong coordination among sensors and the rest of system is necessarily produced. For example, in systems of mobile objects guidance, when the speed of the object is increased, the system has less time to process the information that is received from the sensors and to make decisions about its movement. In this application, a fast answer in appropiate time that allows decisions to be made at every moment may be advisable, although at the expense of less precision in the results.

In this paper, we propose a method of flexible arithmetic sum. Its main characteristic is the variable quality of the result based on the available time. The algorithm is based on the use of strategies that contribute determinism to the response time and, at the same time, allow for parallel designs.

2. DESIGN PRINCIPLES

The proposed method consists of the combination of two techniques: obtaining the result in a successive processing way and using precalculated data in look-up tables.

- Response quality is related to the number of calculated stages of the sum, and therefore, will be able to act on the time-quality-parallelism relationship. This approach forms a new architecture that will implicitly incorporate flexibility in order to adapt the duration of the calculation to time availability, which is the instrument for real-time management. This characteristic provides capabilities for successive refinement of the solution.
- The precalculated data memories (LUT —*Look Up Table*) have interesting characteristics relating to real-time processing: they work in a totally determinist way and they can incorporate error detection and correction mechanisms. The treatment of the operands in small blocks promotes segmentation and a high level of parallelism. These construction possibilities provide robustness and flexibility to the operations [3].

Their disadvantage is the high area complexity for overlarge operands [4], nevertheless, the progressive improvement in performance provided by electronic technology justifies the search for new proposals that would probably have been prohibitive some time ago.

To carry out the sum, we propose the use of a LUT with all the possible results of the sum of two numbers of k bits, called the *LUT-Adder*. The use of tables in the computation of functions is a well-known technique. In arithmetic literature, several implementations of elementary functions based on the use of look-up tables are analyzed [4], [5]. In this approach, we use look-up table to process directly an elemental operation. For performance reasons, it is assumed that the LUT memory is implemented in the circuitry of the Arithmetic Unit itself, thus reducing communication costs. Figure 1 illustrates the direct sum of k-bit numbers using this method.

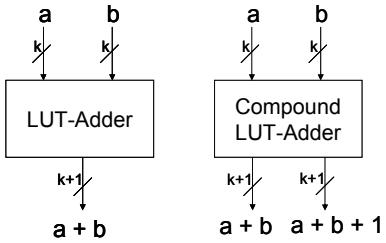

Figure 1: Adder based on Look-Up Table

Note that this operation is only possible for some k values, as will be explained below.

3. ALGORITHM

3.1. Addition Method

The proposed addition method is based on the carry-select adder scheme [6] and it is made up of the following steps:

1. *Fragmentation of operands into k-size blocks*: It is immediate from the original operands. For numbers of m bits (with m>k), we can divide the number into n blocks of k bits, so that $n \cdot k \geq m$.

2. *Addition of the corresponding pairs of blocks*. The partial additions are obtained directly from a look-up table containing the precalculated results: *LUT-Adder*. The processing of the carry is directly made by obtaining the sum and its successor from a *Compound LUT-Adder*. Figure 1. By designing multiple memory access routes, simultaneous access can be gained without the need for several memory chips.

3. Ordered concatenation of the partial additions taking the carry logics into account: The selection of each block is a function of the carry bit of the preceding block selected

* This work is being backed by grant DPI2002-04434-C04-01 from the *Ministerio de Ciencia y Tecnología* of the Spanish Government.

according to the algorithm carry-select adder [6]. The compound LUT-Adder is used to consider the carry in a direct way, since adding the carry to a block is the same as obtaining its successor in the LUT.

3.2. Flexible addition

The addition operation based on Look-Up Tables offers predictability of the response times. The basic idea consists of only performing the sum on blocks for which available time exists. Therefore, this design has real-time properties. Thus, depending on the time available, the system will adapt the quality of response. According to the increase in the number of iterations, the error rate will decrease.

The flexible adder design is based on the previous algorithm with the special feature that only part of the blocks obtained from the operands are combined, according to the time availability.

In a scheme of sequential concatenation of the partial additions, it is proposed that the combination of the blocks will begin with block i, depending on the availability of time, and move towards the left, figure 2. It is possible to obtain fast approximations of the result by selecting only most of the left blocks and selecting the rest at random or the upper ones.

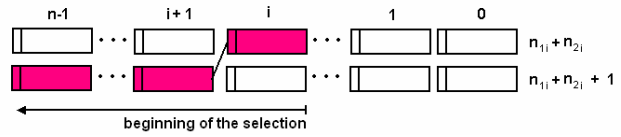

Figure 2: Sequential selection

For a tree concatenation scheme, figure 3, the process is carried out from bottom to top, acting on all the blocks of partial solutions concurrently per level. The total number of bits from the result increases exponentially with the number of steps.

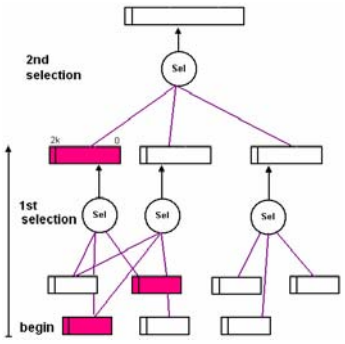

Figure 3: Tree selection

An arithmetic unit must have an operation control that translates the requirements into the number of processed stages. The operation control module consists of a combinational circuit that has problem conditions in its inputs and a number of operation stages in its outputs, for example, a coder, multiplexor or table circuit.

4. ARCHITECTURE

This architecture is suitable for specific purpose applications where time restrictions are present.

4.1 Design

Figure 4 shows the proposed architecture. We assume, for example, the numbers are fragmented into 4 blocks.
The main features of this architecture are:

- Access to the Compound LUT-Adder provides the result and its successor for all the pairs of blocks.
- The selection circuit has a simple design since the effective sum is carried out in the memory with precalculated results.
- The tree selection combines the partial results until the final result of the complete operation is obtained. Three results with different qualities of degree and delay are extracted from the tree selection circuit.
- The operation control circuit selects the partial result that best fits the conditions of the problem.

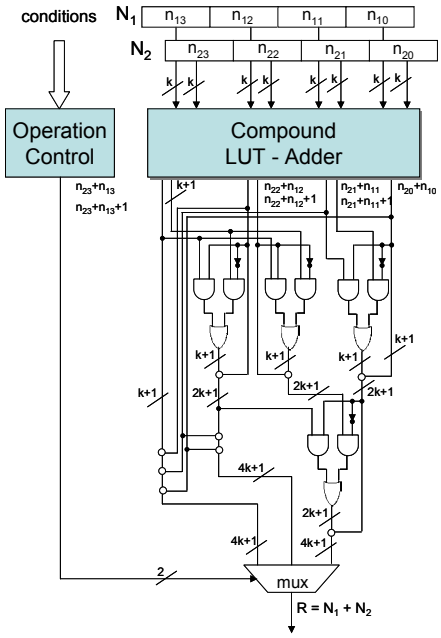

Figure 4: Block diagram of the proposed architecture

4.2. Path time

The path time of the proposed architecture is set by the delay of the slowest path in the circuit. Depending on the application conditions, it can be any of the main paths of the implementation. There are the same number of paths as possibilities of result selection. These paths, as shown in figure 4, are the following:

- Max(Op.Contr.,Comp.LUT-Adder) + mux
- Max(Op.Contr.,Comp.LUT-Adder) +and+or+mux
- Max(Op_Contr.,Comp.LUT-Adder)+2and+2or+mux

Apparently, the improvement in time of one incomplete sum is not important. Nevertheless, when the amount of the sum to be made is elevated, the architecture acquires a greater relevance.

5. EVALUATION OF THE ARCHITECTURE

In this section, we present estimates of the area costs, execution time and error computation of the architecture proposed in the previous section. The power consumption of the circuit is not important for this research and is therefore not dealt with in this paper. It will be studied in depth in the event that it can be implemented in a chip.

5.1. Area estimations

The main contributions to the area of the architecture come from the compound LUT-adder. The area of the selection circuit is small when compared to the area of the LUT. The model we use for the area estimations is taken from [4], [5] and [7].The unit used is the size of a complex gate τ_a, since the area of the compound LUT-adder, selection circuits, and multiplexor are easily expressed in this unit. The LUT-adder is the component of the architecture that occupies the greatest area. The others

have a marginal area in comparison and, therefore, the estimation is focused only on the LUT-adder.

Data storage imposes severe restrictions on k block size. As we can see in table 1, the area cost increases exponentially with the k value. Therefore, we have to achieve a balance between the memory required and the complexity of the circuit. Table 2 shows the cost in terms of τ_a for the most common sizes.

Table 1: Compound LUT-Adder size (bits)

k	Addition-LUT Size
4	1360 bits
6	3.56 KB
8	72.28 KB
16	≈ 26 MB

Table 2: Compound LUT-Adder size (τ_a)

k	Area estimations(τ_a)
4	≈ 47 τ_a
6	≈ 855 τ_a
8	≈ 11564 τ_a

As shown in the previous tables, the amount of area is much greater than in conventional adder designs based on simple combinational circuits, nevertheless, this architecture is still suitable for applications in which the size of the circuit is not a problem.

5.2. Delay estimations

Delays in the complete addition calculus is divided into:

- *Access time to the LUT-Addition in order to obtain the precalculated results*. This time will only be determined by memory access time T_{LUT}. Let τ_t be the delay of a complex gate, such as one full-adder. According to [5], [7] analysis† we assume a delay of about $T_{LUT} = 3.5\tau_t$ for 8 input bit tables, $T_{LUT} = 5\tau_t$ for 12-13 input tables and $T_{LUT} = 6.5\tau_t$ for 16 input bit tables.
- *Selection of the blocks that make up the result*. In the case of tree concatenation, total selection time is obtained by taking into account that all the selections at one tree level are carried out in parallel and that the total number of tree levels is $\lg_2 n$. Let T_{sel} be the time taken in the selection on one tree level, so the expression for the total selection time is $T_{sel} \cdot \lg_2 n$. A selection step consists of two single gates: (and, or), or one complex gate τ_t, so $T_{sel} = 1\ \tau_t$. For operands of $m = n \cdot k$ bits, the proposed algorithm calculates the addition in: $T_{add} = T_{LUT} + T_{sel} \cdot \lg_2(m/k)$ time units.

We perform a comparison of the proposed architecture with other known adder algorithms.

In the first place, in terms of the expression of the asymptotic temporal complexity, those adders have a growth in the delay equal to the proposed design. The table 3 shows the temporal complexity of adder designs [8], where m is the length in bits of the operands. The addition algorithms differ in the constants that modify the general cost expression. In the TLA algorithm, the compound LUT-Adder performance plays a fundamental role in the final calculation time.

In addition, the circuit delays depend on the technology used and on the implementation itself. In order to prove this, the TLA-4 and TLA-8 have been implemented in VHDL and tested on FPGA in comparison with the implementation provided by [9]. The LUT implementation corresponds to the design presented in [10], [11], and has been integrated into the selection circuit.

Table 4 shows the results obtained after the synthesis and simulation of each adder for some number wordlength, including k wordlength. COSA codification is not available in [9]. We do not implement it to get objectivity in the results.

† Implementation using a family of standard gates from the AMS 0.35 μm CMOS library

Table 3: Asymptotic Temporal complexity

Adder	Architecture	Asymptotic Temporal complexity	
CLA	Carry Lookahead Adder standard	$4\lg_2 m$	O(lg m)
PPA-SK	Parallel-Prefix Adder Sklansky	$2\lg_2 m + 4$	O(lg m)
PPA-BK	Parallel-Prefix Adder Brent-Kung	$4\lg_2 m$	O(lg m)
COSA	Conditional-Sum Adder	$2\lg_2 m + 2$	O(lg m)
TLA-4	k=4 LUT adder tree selection	$T_{TLU} + 2\lg_2 m - 4$	O(lg (m/4))
TLA-8	k=8 LUT adder tree selection	$T_{TLU} + 2\lg_2 m - 6$	O(lg (m/8))

Table 4: Delays in the calculation of the sum (ns)

Adder	bits (m)				
	8	16	32	64	128
CLA	11,1	25,5	54,0	110,9	224,8
PPA-SK	9,4	20,4	34,5	37,5	48,6
PPA-BK	9,4	13,7	19,8	24,3	32,7
TLA-4	3,3	4,8	7,3	13,1	23,2
TLA-8	3,1	3,7	5,2	7,7	13,5

The previous results demonstrate that the proposed adder design presents a delay similar or better to the conventional designs for this particular implementation, and they show the technology's high degree of dependency on performance.

5.3. Error computation analysis in flexible addition

With the objective of testing the error computation while ignoring the correct concatenation of all the sequence of partial results, an exhaustive set of experiments has been made to prove the method for all cases. The experiments are both in individual operations and in sequences of successive operations. The profile of the experiments is the following: A compound LUT-Adder with k = 8; number length m = 48 bits; operands are rational numbers at the interval [0, 1). The error evolution is shown in figure 5.

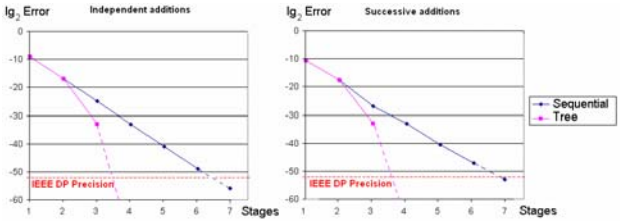

Figure 5: Evolution of the error average in the RT addition

- The *independent additions* test consists of calculating the average error rate in 10^7 additions of two random rational numbers.
- The *Successive additions* test is aimed at empirically analyzing error propagation while adding inaccurate values consecutively. In this case, the error average is calculated in 1,000 sets of 1,000 successive additions of random rational numbers within the interval [0,1) for each of the operation's loops, that is, the result of each of the additions acts as an addend of the following addition operation and so on. The numbers are generated at a positive interval, so they do not compensate positive errors with negative ones in the successive additions. However, the unselected partial results are selected alternatively by excess or default in order to provide compensation on a complete operation level.

6. APPLICATION EXAMPLE

The experiment is located in the *Specialized Processing Architectures* research group of the *University of Alicante, Spain*. One of the most interesting lines of research consists of the development of a *Real-Time processor* that considers the temporal restrictions in the low level of the architecture [12]. It is usefulness in many interest applications: calculation of trajectories for moving bodies, guidance and positioning systems, high frequency communications etc.…To process this successfully, the flexibility and determinism of the calculations plays a fundamental role in the correct working of the processor.

This section describes a simple example that illustrates the specific application of the proposed architecture. Let there be an object that moves according to combination of several 3D forces $\vec{f}_i = \langle x_i, y_i, z_i \rangle$; for example: magnetic field, electric field, gravity, etc …

The application consists of calculating the final force that is formed by the combination of individual forces in real-time way. This is the sum of all of them: $\vec{F} = \sum_i \vec{f}_i$ (1)

We proposed the following architecture to resolve the expression (1):

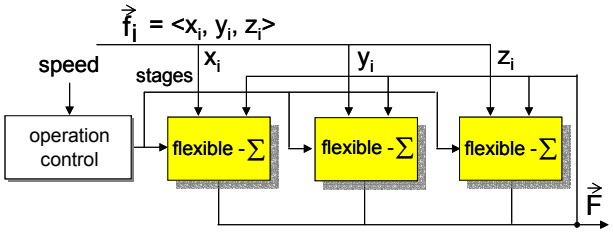

Figure 6. Architecture for the final force calculation

The sum of the individual components of each force is made independently in the flexible adders simultaneously. The operation control module decides the number of stages that will be made in the sum operations. The criterion is based on the speed of the moving object. At greater speed, fewer partial results are selected.

We assume: $x_i, y_i, z_i \in [0, 1]$, speed. $\in [0, 150]$, $n = 32$ bits, $k = 8$ and five forces: $F = f_1 + f_2 + f_3 + f_4 + f_5$.

Table 5 shows the number of stages in a tree selection, the time-saving that takes place in the five sums and the computation error. Simulation is made in FPGA for a set of 1000 series of five consecutive sums.

Table 5: Experimental results

speed	selection stages	delay (ns)	Saving of time (ns)	|Error|
[0, 32)	4	4 · 7,7	0,0	$2^{-64,91}$
[32, 64)	3	4 · 5,2	10,0 (32,4%)	$2^{-32,97}$
[64, 96)	2	4 · 3,7	16,0 (51,9%)	$2^{-16,82}$
[96, 150]	1	4 · 3,1	18,4 (59,7%)	$2^{-8,89}$

The simulation results demonstrate that this technique saves considerable time in cases in which a fast response is necessary. Error is maintained within the acceptable margins. Although the value of the final force is not obtained with absolute precision, the result can be sufficient to make a movement decision.

7. CONCLUSIONS

The following conclusions have been drawn from the research described in this paper:

- The use of precalculated results in stored logic permits the construction of fast operators comparable to existing methods and lays the foundations for the design of high performance architectures. Adder complexity turns out to be logarithmic with the number of blocks: $T_{suma} \in O(\log(n))$. The proposal equals the asymptotic temporal complexity of present-day high performance adders. Technological improvements in manufacture or in communication with the selection circuit will tend to reduce T_{LUT} access time and, therefore, total addition time.
- The adder behavior, which produces more and more precise results as the number of iterations increase, is suitable for the construction of systems with temporal/precision restrictions, in which result quality is exchanged for response determinism and speed. Developed methodology for the adder, due to its features of high performance and obtaining imprecise calculations with limited and decreasing error, can be used in the development of other arithmetic operations with temporal restrictions.
- Finally, the error analysis carried out shows that the algorithm provides limited results in addition operations, even in cases in which successive calculations are made with imprecise operands.

8. REFERENCES

[1] W. A. Halang, K. M. Sacha. *Real-Time Systems. Implementation of Industrial Computerised Process Automation,* World Scientific Publishing Co. Singapore, 1992.

[2] G. C. Butazzo. *Hard Real-Time Computing Systems. Predictable Scheduling Algorithms and Application,* Kluwer Academic Publishers. Netherlands, 1997.

[3] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, Oxford University Press, New York, 2000.

[4] W.F. Wong, E. Goto, "Fast Hardware-Based Algorithms for Elementary Function Computations Using Rectangular Multipliers", *IEEE Transaction on Computers*, vol. 43 (4), 278-294. 1994.

[5] M.D. Ercegovac, T. Lang, J-M. Muller, A Tisserand, "Reciprocation, Square Root, Inverse Square Root, and Some Elementary Functions Using Small Multipliers", *IEEE Transaction on Computers*, vol 49, 628-636, 2000.

[6] J. Sklansky. "Conditional-sum addition logic." *IRE Transactions on Electronic Computers*, vol. 9, pp. 226–231, 1960.

[7] J-A.. Piñeiro, J.D. Bruguera, J.M. Muller. "Faithful powering computation using table look-up and a fused accumulation tree". *Proceedings of the 15th International Symposium of Computer Arithmetic (ARITH'15)*, 2001.

[8] R. Zimmermann. *Binary Adder Architectures for Cell-Based VLSI and their Synthesis*, PhD Thesis, Swiss Federal Institute of Technology. Switzerland. 1997.

[9] R, Zimmermann. "VHDL Library of Arithmetic Units", *Forum of Design Languajes*, Lausanne, September 1998.

[10] S.J.E. Wilton, N.P. Jouppi. "An Enhanced Access and Cycle Time Model for On-Chip Caches". *Digital Western Research Laboratory*. 1994.

[11] H. Nambu, K. Kanetani, K. Higeta, M. Usami, T. Kusonoki, K. Yamaguchi, N. Homma. "A 1.8 ns Access, 5550 Mhz 4.5 Mb CMOS SRAM". *IEEE ISSCC*. 1998.

[12] J.M. Mora Pascual, *Real-Time Floating Point Arithmetic Unit*, PhD Thesis, University of Alicante, Spain. 2001.