THE MFIBVP REAL-TIME MULTIPLIER

Yusrila Y. Kerlooza[1], Sarwono Sutikno[2], Yudi S. Gondokaryono[3] and Agus Mulyana[4]

[1,4]*Universitas Komputer Indonesia (UNIKOM), Indonesia, kerlooza@gmail.com*
[2,3]*Institut Teknologi Bandung (ITB), Indonesia, ygondokaryono@stei.itb.ac.id*

ABSTRACT. This paper presents the architecture of the MFIBVP real-time multiplier which is ideal to be used in real-time system application. The MFIBVP technique is a combination of the *MSB–First* computation, the *Interval-Bounded Arithmetic* and the *Variable-Precision* computation techniques. The MFIBVP computation guarantees the computation carried out will produce high accuracy from the early computation time, self error estimation and time-optimal computation. This paper shows the performance of the MFIBVP real-time multiplier unit that can gives accuracy of it's intermediate-result more than 99% since the second phase of its process.

Keywords: real-time system, interval-bounded arithmetic, *MSB-First*, variable-precision, multiplier

INTRODUCTION

Real time system is a system built to support the success of the processes that is time-bounded, according to it's definition:

"*A Real-time System is one whose logical correctness is based on both the correctness of the outputs and their timeliness"*(Laplante, 2004).

Based on this definition, the main characteristics of real-time system are:

1. System have to produce a computation result correctly (*logical/functional correctness*), and
2. System have to produce a computation result before exceed the deadline (*timing correctness*).

From the point of view of the real-time computation in the hardware level, most present-day strategies are focused on increasing hardware computational performance by using parallelism, segmentation or multiprocessing design techniques in order to decrease the average response delay.

These strategies are not always the most suitable ones for solving certain problems and they give rise to a multitude of questions: in the demand for requirements of reduced size applications, is the incorporation of multiprocessor architectures embedded in the system acceptable? For minimum timing constraint applications, can a logically correct decision be made only on an imprecise numeric result? Does adaptation to changes in environmental requirements require the system architecture to be redesigned? The investigation described in this paper considers these questions in the current implementations of calculation techniques and proposes a real-time architecture for arithmetic calculations that adapts the processing delay to the required time of the task.

BASIC THEORY OF MFIBVP

MSB-First

Conventionally, computation process is carried out by a computer from the least significant bit first (LSB-First) just like we calculate, thus this technique gives slow numeric accuracy escalation throughout the process.

Nielsen and Kornerup (Nielsen & Kornerup, 1995) conducted research on MSB-First digit serial arithmetic and our previous research on MSB-First arithmetic architecture (Kerlooza, 2004a; 2004b; 2007a; 2007b; 2008) shows the potential advantage of this technique over conventional ones. Those previous researches also shows the need of the intermediate-result: a successive product of ongoing arithmetic process execution that can be accessed by other computation tasks or elements during process time. Figure 4 below shows difference between the calculation concept of the LSB-First (conventional) vs. the MSB-First computation.

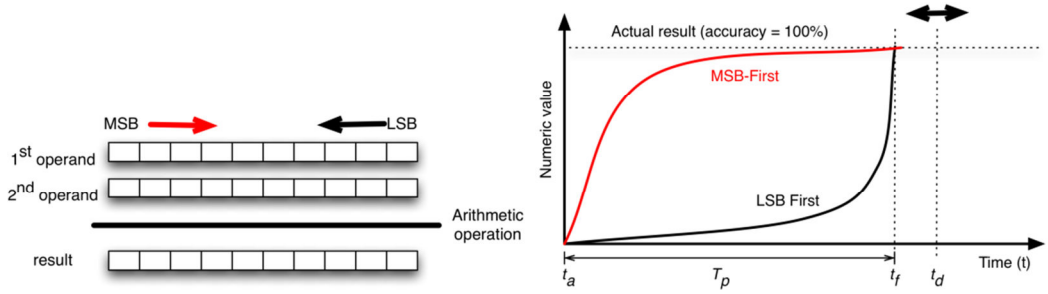

Figure 4. The calculation concept and the performance of LSB-First vs MSB-First

By starting the process of addition from number (can be bit or digit) with the highest value first will provide result with high accuracy since the beginning of the process. We can predict how the two technique (LSB-First and MSB-First) performance in gaining numeric value in arithmetic operation as depicted in Figure 4. To maximized the advantage of the MSB-First computation, the incomplete result (we call it the intermediate-result) should be able to be accessed during the computation time.

Interval Arithmetic

If we look again to Figure 4, both LSB-First and MSB-First techniques can not tell us its computation accuracy before t_f. We can add the ability to predict where the final computation value lies by using the same idea of the interval arithmetic methodology introduced by (Moore & Yang, 1959), (Moore, 1962), and (Boche, 1963). The interval arithmetic produces two values for each arithmetic operations. The two values correspond to the lower and upper endpoints (bounds) of an interval, such that the true result is guaranteed to lie on this interval. The width of the interval, i.e., the distance between the two endpoints, indicates the accuracy of the result. Interval arithmetic was originally proposed as a tool for bounding rounding-off errors in numerical computation (Moore, 1962). It is also used to determine the effects of approximation errors and errors that occur due to non exact inputs. Interval arithmetic is especially useful for scientific computations, in which data is uncertain or can take a range of values.

We can produce lower and upper bound for LSB-First and MSB-First by adopting several algorithms, one of the simplest thing to compute the upper bound is by subtracting the maximum value of the arithmetic operation with the lower bound (computed by the original algorithm) in parallel. In this way during computation time, there will be two intermediate-results which denote the lower and upper bound of the true value. Figure 5 depicts the basic idea of the interval bounded concept.

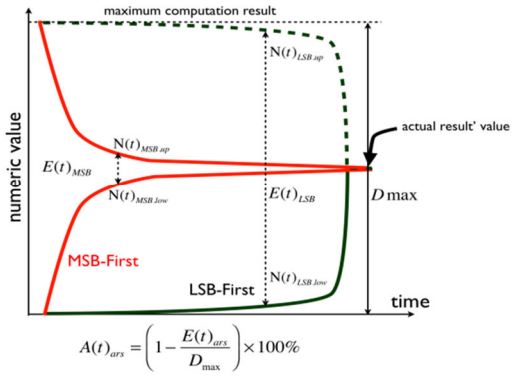

Figure 5. The concept of interval-bounded in calculating the accuracy of the intermediate-result before the final answer is produced

Variable Precision

The delay adjustment ability and the variable quality of the result of each function depends on the possibility of partially executing its implementation. In general, each operator has a part that must be executed obligatorily and another that can be partially calculated (Deng & J. W.-s Liu, 1997; J. W. S. Liu, SHIH, Lin, Bettati, & Chung, 1994). The execution control of this optional part will allow us to adjust the function performance according to the application requirements (accuracy needed or time available). In this aspect, the implemented partial execution technique (stages or iterations) must provide capabilities for successive refinement of the solution and thus, support real-time requirements. Response delay is related to the number of calculated stages or iterations of the operations. Normally, a shorter process time results in less accuracy in the results. If the computation accuracy is met or the time left for computation is up, the execution can be stop and intermediate-result can be accessed by other process. This is the basic concept of variable precision computation covered in this paper, as shown by Figure 6.

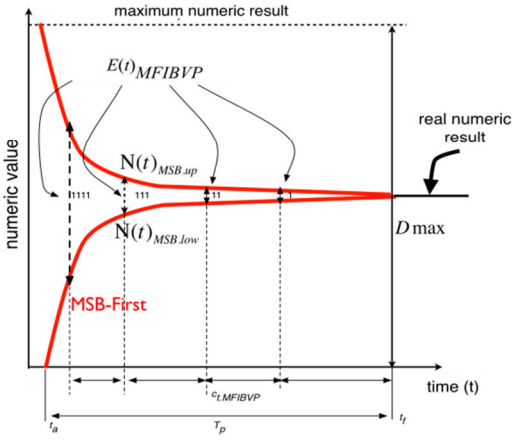

Figure 6. The Variable-Precision concept in the MFIBVP technique

THE MFIBVP REAL-TIME MULTIPLIER ARCHITECTURE

The first implementation of the multiplication operation depicted in Figure 11 is basically a well-known multiplier technique: the unsigned array multiplier (Mi Lu, 2004). It consists the following steps:

Generation of partial products: The partial products generation process is crucial to the operation's overall performance. Two aspects must be taken into account in its design: the

complexity of the generating circuit and the number of partial products generated. The first aspect is linked to the time taken in generating each partial product, whereas the second one affects the time taken in the second step below to reduce them into two operands that will be added in the last step.

Reduction in the number of partial products: The general way in which a high performance multiplier works consists of combining the partial products in order to reduce their number until a total of two is reached. We can use Wallace tree method (Wallace C.S., 1964) for the reduction of the partial products.

Final addition: It can be implemented by well-known addition methods; nevertheless, due to the MFIBVP features, we have used the previously proposed adder, the MFIBVP adder.

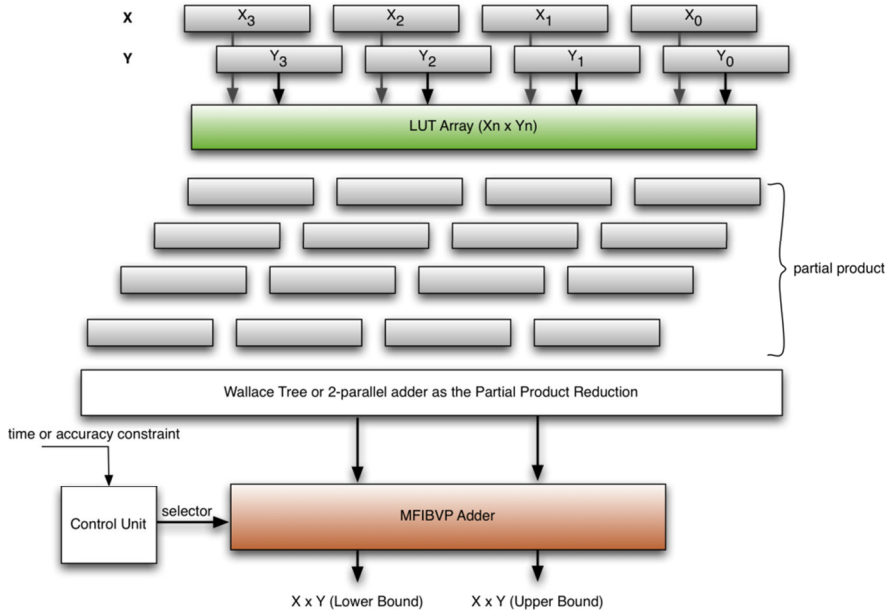

Figure 7. The architecture of the MFIBVP multiplier.

TESTING AND ANALYSIS

The tests presented in this paper are conducted to reveal:

Comparison of calculations accuracy and time required in each phase between the MFIBVP multiplier and the commonly used multiplier architecture.

The accuracy of calculations and time required by the MFIBVP multiplier in each phase.

The accuracy of intermediate-result is calculated using Eq.2 below

$$A(t)_{ars} = \left(1 - \frac{E(t)_{ars}}{D_{\max}}\right) \times 100\% \quad (2)$$

As performance comparison of the MFIBVP real-time multiplier, the performance of two commonly used multiplier architecture designs, the Array Multiplier using Carry Propagate Adder and the Carry Lookahead Adder will also be presented. Figure 8 below presents the performance comparison of these multipliers' architecture.

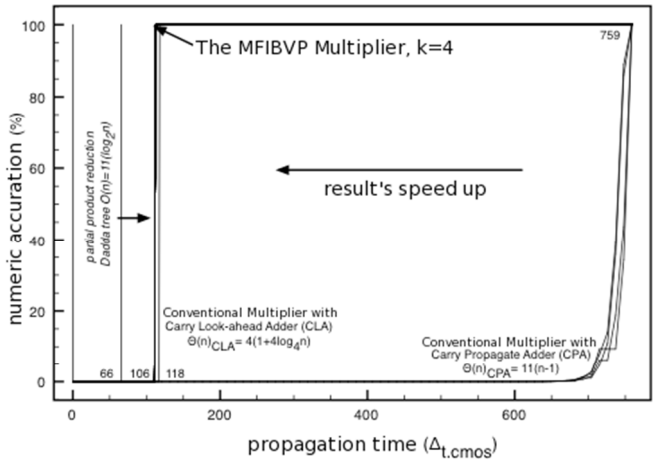

Figure 8. Performance comparison between the MFIBVP Real-Time Multiplier vs the Array Multiplier using Carry Loo*k*-ahead Adder and Carry Propagate Adder on 50 pairs of random numbers (64-bit,*k*=4)

The Figure 8 shows that since 2nd phase or $104\Delta_{t.cmos}$ propagation time, the MFIBVP real-time multiplier has produced an error calculation less than 1%. This deterministic execution time and accuracy certainty provided by the MFIBVP real-time multiplier can improve the performance of the Real-Time System due to the ability to make trade-off between the accuracy needed and time available that can be integrated into arithmetic instruction.

By using transistor propagation time, this architecture can be implemented on any microelectronic technology that eventually determine the actual processor clock.

CONCLUSION AND FUTURE WORKS

By the performances measurement we can conclude that the MFIBVP real-time multiplier gives better computation performance than conventional multiplier architecture by it's ability to:

1. produce intermediate-result during execution time,

give certainty in computation accuracy even before the process finish time by providing two intermediate-results which act as the lower and upper bound of the real and complete computation result.

gain high computation accuracy from the early time of the execution process.

This research is a part of greater research that tries to develop new paradigm in numeric calculation that incorporates time or accuracy as a parameter of calculation. By this paradigm, real-time computation will be easier to manage both automatically on-the-fly by new scheduling system in the operating system's level, or manually by a programmer in the software design phase. Other arithmetic units and logic units with the MFIVBP technique should be designed in order to develop new real-time processor as well as new programming language, compiler and real-time operating systems.

NOTATION

A*(*t*)*ars	the accuracy of the calculations produced by arithmetic unit with *ars* architecture
E*(*t*)*ars	the error or the difference between the upper bound and bound value
Dmax	the largest calculated value that may be produced from arithmetic operations of two operand with *n* bit wide

REFERENCES

Boche, R. E. (1963). An Operational Interval Arithmetic. IEEE-Illinois Inst. of Tech.-Northwestern Univ., Univ. of Illinois. Abstract of a paper given at National Electronics Conference.

Deng, Z., & Liu, J. W.-s. (1997). Scheduling real-time applications in an open environment. in Proceedings of the 18th IEEE Real-Time Systems Symposium, IEEE Computer (p. 308–319). Society Press.

Kerlooza, Y. Y., Kuspriyanto. (2007a). Towards Real-Time Processor: Multioperand MSB-First Minimax Addition. International Conference on Electrical Engineering and Informatics (ICEEI2007).

Kerlooza, Y. Y., Kuspriyanto. (2007b). Towards Real-Time Processor: The Implementation of Multioperand MSB-First Adder Arithmetic Unit on the Computation of $y = \Sigma$ ai bi. International Conference on Electrical Engineering and Informatics (ICEEI2007).

Kerlooza, Y. Y., Kuspriyanto. (2008). Real-Time dan Adjustable Computing. e-Indonesia Initiative 2008 (eII2008).

Kerlooza, Y. Y., with Kuspriyanto. (2004a). Keandalan Unit Multioperand MSB-First Real-Time Adder Pada Operasi Penjumlahan Data Acak. Proceeding: Seminar on Intelligent Technology and Its Applications 2004 (SITIA'2004).

Kerlooza, Y. Y., with Kuspriyanto. (2004b). Towards New Real-Time Processor: The Multioperand MSB-First Real-Time Adder. Proceedings of the EUROMICRO Systems on Digital System Design (DSD'04), 524 529.

Laplante, P. A. (2004). Real-Time Systems Design and Analysis. (S. V. Kartalopoulos, Ed.) (3rd ed.). IEEE Press Wiley Interscience.

Liu, J. W. S., SHIH, W.-K. S., Lin, K.-J., Bettati, R., & Chung, J.-Y. (1994). Imprecise Computations. Proceedings of the IEEE, 82(1).

Mi Lu. (2004). Arithmetic and Logic in Computer Systems. John Wiley & Sons Inc.

Moore, R. E. (1962, November). Interval Arithmetic and Automatic Error Analysis in Digital Computing. Department of Mathematics, Stanford University, Stanford, California.

Moore, R. E., & Yang, C. T. (1959). Interval analysis I (Space Div. Report No. LMSD703073). Lockheed Missiles and Space Co.

Mora-Mora, H., Mora-Pascual, J., Garcia-Chamizo, J. M., & Jimeno-Morenilla, A. (2006). Real-time arithmetic unit. Real-Time Systems, 34, 53-79.

Nielsen, A. M., & Kornerup, P. (1995). MSB-First Digit Serial Arithmetic. Journal of Universal Computer Science, 1(7).

Parhami, B. (1999). Computer Arithmetic: Algorithms and Hardware Designs (1st ed.). Oxford University Press, USA.