Energy Efficient Functional Unit For A Parallel Asynchronous DSP

W. Suntiamorntut, Nitin Gupta, L.E.M. Brackenbury, J. Garside
Department of Computer Science, The University of Manchester,
Oxford Road, Manchester, M13 9PL, UK Tel.+44-161-2753531
{wannarat, guptan, lbrackenbury, jdg}@cs.man.ac.uk

Abstract

With the explosive growth in portable applications, power efficient computing in a Digital Signal Processor (DSP) is becoming a significant and challenging area of research. Clock-less or asynchronous timing is applied to eliminate clock generation, buffering and distribution at the system level. This paper presents the energy efficient functional unit for an asynchronous DSP implemented on four-way parallelism architecture. In particular, as is well known, arithmetic operations such as multiply or multiply-accumulate are frequently performed and are power hungry. Reducing the power dissipation in multiplier and adder without sacrificing the performance will extend the limited source of energy used in portable applications. This has led to the incorporation of an unusual parallel datapath architecture, a balanced delay tree in the multiplier, a carry-look-ahead tree adder, a novel design of Hamming distance and normalization circuit, data dependent computing and pass transistor circuits to achieve better energy efficiency. The developing flexible FU for the user and system programmer is another highlight of our works. This has led to additional cost, particularly in the implementation of the associative configuration memory. The cost can be justified by the flexibility and performance gained by user. The full-custom datapath of the FU has been implemented on 0.18μm operating at 1.8 V. The preliminary results of the extracted full-custom FU layout netlist are represented which confirm that all energy efficient design techniques at the logic and circuit level demonstrate significantly improvement of the power, performance and area in functional unit.

1. Introduction

As is well-known, the capabilities of computation in portable applications has been increasing exponentially. However, the intensive and continuous computing of hand-held computers and other portable devices are restricted by the source of power. From [1], it is clear that the energy density of existing battery technologies are far from what is needed. Hence, an energy efficient design becomes vital.

Digital Signal Processors (DSPs) have been developed for wireless applications such as mobile handsets. Mostly, mobile phones are driven by cellular standards. In the first generation, many types of filters were expected to run on the DSPs. However, a modern mobile phone handles many more applications such as video decoding, data processing and speech recognition. Multiple standards are also needed in one device. Therefore, the trend for DSP architectures is parallelism; exploiting more than one processing unit gains high throughout. Whilst the area is increased by employing multiple computing units, the energy consumption is decreased by scaling down the supply voltage. For moderate voltage scaling, the logic depth, amount of switching and the workload factor of the circuit determine the energy efficiency. If the energy-delay product metric is used as the basis of comparison between designs, then the overall energy efficiency can be exponentially improved in a DSP with parallel FUs since each FU can be operated at lower throughput. This is because when the energy-delay product is taken into account, it means each component uses the smallest energy source per operation. When the speed has been slowed down, each component will dissipate less power. Meanwhile, the overall throughout of the system is still maintained because of the parallel architecture. Therefore, the system can gain an energy efficiency improvement merely by introducing parallelism.

An alternative approach to make an energy efficient system is to structure algorithms to fit the available hardware resource in a DSP. This usually has restrictions with the existing hardware architecture. Therefore, at this level, the system may not gain a big improvement. Therefore, several energy efficient design techniques should be combined to achieve larger energy improvement when running on a powerful architecture.

Another requirement of a DSP is flexibility. In recent years, many research works [2-5] propose reconfigurable digital signal processors with designs implemented on Field Programmable Gate Array (FPGA) technology. However, the power dissipation is still relative high. New generation FPGA includes specific arithmetic units, accelerators, IP cores that provide speed and flexibility, but still tend to be high power. The research challenge is to develop both an energy efficient and flexible design. Portable systems need a processing unit that gives them the lowest energy consumption, but at the same time must provide enough flexibility to the user and programmer. The functional unit presented here is an energy efficient and flexible component for an asynchronous parallel DSP performing computationally intense algorithms. A configuration memory has been attached to each energy efficient FU in our design for the flexibility. Internally, the design combines several energy efficient circuit/logic design techniques. The background of our previous DSP architecture will be described in the next section. Then, our energy efficient FU architecture including the arithmetic logical unit and the configuration memory are described in section 3. The low power circuits used in the datapath are described in section 4. Simulation results from the full-custom layout are presented in section 5 demonstrating the improvements in energy achieved, followed by some concluding remarks.

2. Background

Our FU has been designed and implemented for a parallel asynchronous DSP named CADRE [6]. CADRE was proposed to be a minimum power consumption DSP whilst meeting the performance requirements of next generation cellular phones. However, the simulation results show that the power dissipation is still on the high side. In [6], the power dissipation of CADRE was analyzed and approximately 50% of the overall power consumption was found to be dissipated in the FU. Thus, reducing the power consumption of the FU was the primary motivation behind this research. The original philosophy of the architecture is described to help the reader understand our FU easily.

A CADRE was implemented by exploiting four-way parallelism, as this appears to be optimal for power reduction [7]. This is based on the premise that area can be traded for increased speed because silicon area is rapidly becoming less expensive. Most of the DSP activity can be characterized by frequent repetition of fixed instruction sequences. So, the instruction encoding which determines the selection and passage of data for each operation can be predetermined and stored in advance in a configurable memory which is located locally to each FU. These encodings can then be recalled with a compressed instruction. Because the configuration memories are RAMs, this allows reconfiguration at any point in execution. In addition, these encodings could be expanded within the FUs. This dramatically reduces the size and amount of information that needs to be fetched from main memory. CADRE used a dual Harvard architecture that has one program memory and two separate data memories. A large on-chip register file of 256 16-bit words was included to avoid traffic and power dissipation in the main memories. In this way, the operands required by the FUs were provided directly from a register file. As with other DSPs, a 32-entry instruction buffer was also included to handle loop instructions and reduce traffic to or from the program memory. Finally, all standard hardware components in CADRE were operated using self-timed techniques.

The top-level architecture of the present work has been adapted for the current research work because time is too limited to fully implement the CADRE design. However, we still keep the major advanced feature such as four-way parallelism to give high throughput. Meanwhile, a new FU has been designed with its configuration memory. The new system consists of four FUs connected together with a global bus and a pair of FUs are connected locally. The input data of each FU will be directly provided from on-chip RAM blocks with the output data being kept in another RAM block. The encoded top-level instruction is stored in a program memory. It contains control bits to enable the FU and accumulator write-back plus a 5-bit address which accesses the configuration memory associated with each FU; the functional unit instructions are stored in advance into each configuration memory of the FUs.

The instruction set of the proposed FUs has two sets of instructions: computation and data movement and these can occur concurrently. 1) Computational instructions: These instructions contain arithmetic and logical operations; the arithmetic instructions include distance, normalization, shift, ADD, SUB, MPY and MAC (multiply and accumulate). The output destination of the operation can be 1 of 4 accumulator registers (AccA to AccD) inside the FU. 2) Data Movement Instructions: These instructions process the data movement in or out of the FU. These instructions include an accumulator register to accumulator register transfer within the FU itself, an accumulator register to an accumulator register in another FU, or a movement of data to the output RAM.

The FUs form part of an asynchronous pipelined design. To eliminate clock generation, buffering and distribution for power improvement, asynchronous timing has been used. This also gives a reduction of electromagnetic interference(EMI) as the switching of the logic is spread instead of being concentrated around the clock edge. These FUs are therefore based on the principle of micro-pipelines[8] as shown in Figure1, where the data transfer between blocks uses local handshake signals. The done signal allowing an output to propagate to the next micro-pipeline stage is generated either from the combinational logic for a data dependent operation or from a matched delay.

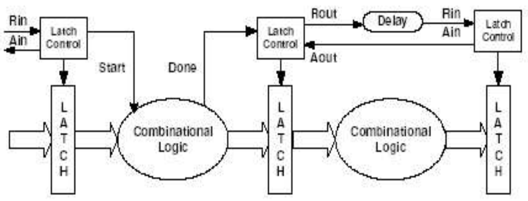

Figure1 Asynchronous pipeline

A novel approach has been adopted for delay matching by connecting a separate supply for the timing circuits, different from the supply for the datapath. This enables timing margins to be controlled from a single external pin without any hardware overheads. Then the delay can be increased (control voltage reduced) if timing is on the edge or the delay can be reduced(control voltage increased) to remove unwanted timing margins.

This simple technique offers a flexibility not apparent in other asynchronous timing approaches[9-12]. It therefore improves the likelihood of obtaining working asynchronous circuits at the first attempt and should lead to greater acceptability of asynchronous design techniques by easing the problem of designing timing and control.

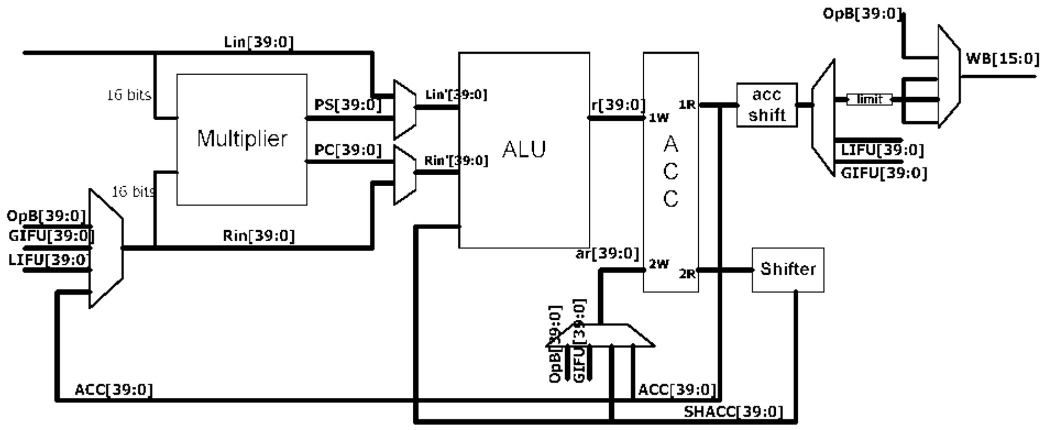

Figure 2 Functional unit datapath

3. Energy Efficient Functional Unit

The challenge here is to meet the requirements of future portable applications, such as mobile phones. These devices have a very small power budget but need high performance with a complexity approaching that of a desktop computer. The throughput has been achieved by adopting a parallel architecture allowing up to 4 instructions to be processed in parallel. In addition, parallel arithmetic logic is also used within a FU. Thus the next challenge is how to manage the hardware resources within each FU efficiently.

3.1 Configuration Memory

In our design, the VLIW instructions for a particular algorithm are cached by software and looked up using a short-form instruction. Up to 64 instructions are produced in a configuration memory, which is more than sufficient for an anticipated algorithm; for example, the DCT algorithm [17] requires only 6 encoded instructions in a configuration memory. Furthermore, the configuration memory instructions are completely user definable. The simple look-up avoids any tag overhead associated with a cache. The configuration memory makes the configuration of the FU flexible, enabling optimization by users. This flexibility has led to additional hardware costs, particularly in the implementation of the configuration memory. These costs can be justified by the flexibility and performance gained by users. In particular, the use of configurable memory embedded into the design is unique compared with the other DSPs mentioned above. This feature also allows each FU to operate on an independent instruction stream if required.

3.2 Energy Efficient Arithmetic Unit

The FU datapath is shown in Figure2. The number of available paths and input sources at different points mean that within the FU, many concurrent operations can occur within a single time-slot. For example, multiplication and addition with shifting can be performed in parallel with moving the contents of an accumulator result back to memory. Operations within the FU are termed major and minor. A major operation is usually performed in each time-slot and encompasses the arithmetic and logic operations. Data supplied to or from the FU is usually 16-bits. However, the FU datapath is 40-bits wide, as usual in a DSP. Multiplication is 16x16 bits and produces a 40-bit partial sum (PS) and partial carry (PC) which are then forwarded to the adder in the ALU block for completion. Here, up to four operands may need to be added i.e. PS (or Lin) and PC (or Rin) together with an accumulator value (SHACC) which comes via the Shifter and a rounding constant (not shown). The ALU output is written to one of four accumulators AccA to AccD in the ACC. Minor operations operate concurrently with a major operation. Here, in parallel with a major operation of addition and/or the multiplication, another Accumulator register can be written to either from another (shifted or unshifted) Accumulator register, or from the memory (OpB) or from the global bus GIFU. An Accumulator can also be written back to memory via the WB bus. The addition of the multiplier outputs PS and PC is performed by the adder in the ALU. This adder is therefore shared between the add/subtract and multiply/multiply-accumulate operations. This architecture is unusual in that most designs have a dedicated adder for the MAC operation and a separate adder provided for the addition. Having just one adder in the FU significantly reduces the amount of logic required in the datapath which in turn reduces power.

Finally, transparent latches are inserted in front of the multiplier, the ALU, the second write port (2W) of the ACC and the write back (WB) bus to prevent unnecessary switching in the datapath. This approach directly reduces the dynamic power dissipation. In addition, because pass gate transparent latches are used rather than normal D-latches or flip-flops, a further saving of power dissipation and better performance are achieved because less capacitance is switched and the latch delay is smaller.

Multiplier Implementation

The arithmetic unit has been implemented using two's complement rather than the sign and magnitude arithmetic in the initial design; this reduces design complexity, lowers dissipation and gives a better performance. The algorithms for Global System for Mobiles (GSM) operations indicate a high proportion of multiplication operations. Generally, the multiplier consists of two main components: a partial product generation (PPG) and the addition of those partial products (PPs). The current 16x16 low-power multiplier is shown in Figure3, has employed a modified Booth's algorithm[13] to produce the 8 PPs required in parallel. The addition of the eight PPs requires a carry propagate adder (CPA) which has a long latency. To avoid this, a Wallace Tree[14] structure of 4-2 compressors is used which not only improves performance by reducing the latency but also reduces power by significantly reducing the amount of overall logic required. As a result of using tree structure topologies, the number of stages traversed by each input is approximately the same for all inputs. This leads to a *balanced delay tree* and results in less switching activity due to input skew. In addition, the eight PPs have been divided into two groups, PP1-4 and PP5-8 to produce partial carry and partial sum in parallel. Therefore, this multiply structure can achieve both performance and balance the delay in the tree structure. As the multiplier generates 40-bit outputs, all PPs have to be sign-extended. In order to minimize the logic within the tree structure, a pre-calculated sign-extension is applied. Furthermore, eight signed bits from the modified Booth s logic are forwarded to the adder; this reduces the depth of the Wallace tree logic required by one stage.

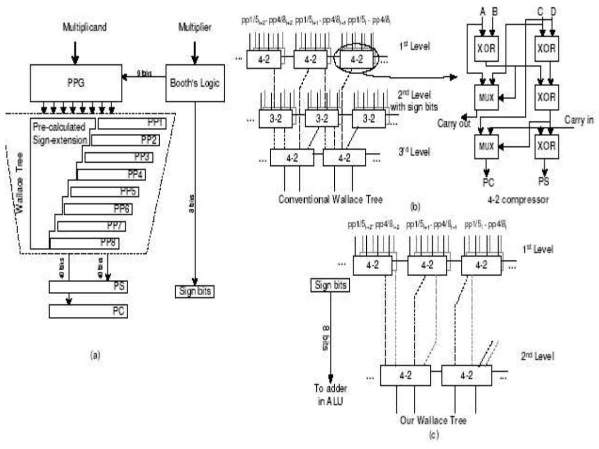

Figure3 Parallel multiplier structure

The post full-custom layout simulation on a 0.18μm process operating from 1.8V shows that our multiplier can produce a PS and PC every 1.5 nS in the worst case of random inputs and consumes an average power only 21.6 mW at this speed.

Adder Implementation

Operation analysis reveals that addition is the most frequent function performed in speech code. Thus, whilst the power dissipation of addition is approximately 1/5 that of multiplication, the very high proportion of additions means that it is essential to minimise the power required for this operation. In the FU, addition involves adding up to four variables as shown in Figure4. The four inputs are first compressed to two by using 4-2 compressors. The two compressor outputs are then added in a carry-look-ahead tree. The carry-look-ahead equation, given in [15], has been modified when used in our design to enable an easy efficient mapping onto pass-transistor circuits. The carry is generated across blocks of four bits because this offers the best speed-power product. The adder in [15] was implemented using VHDL, whilst a synthesis tool was then used to generate the circuit and layout; therefore, its energy efficiency was relatively poor. In our design, the logic has been optimised and organised to map efficiently onto low energy pass transistor circuits and the circuits have then been laid out by hand to give an area efficient implementation. Thus the design achieved here is power efficient.

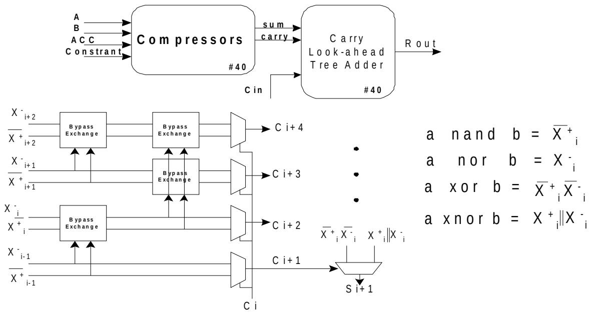
A
B
ACC
Constrant
Compressors
#40
sum
carry
Carry Look-ahead Tree Adder
#40
Rout
Cin
Bypass Exchange
Ci+4
Ci+3
Ci+2
Ci+1
Ci
Si+1
a nand b = $\overline{X^+_i}$
a nor b = X^-_i
a xor b = $\overline{X^+_i}\overline{X^-_i}$
a xnor b = $X^+_i \| X^-_i$

Figure4 Structure of 4 input 40 bit adder and carry-look-ahead tree.

All circuits in our adder are implemented by using only pass transmission gates. This leads to both high performance and low power. The preliminary results from post full-custom layout on a 0.18μm process operating from 1.8V show that our 4 input 40 bit adder consumes an average power of only 7.56mW with a delay of 4ns to complete the addition.

Hamming distance and Normalization

The Hamming distance and normalized instruction are instructions normally included in the instruction set since they are useful for most DSP algorithms. In our design, we combine these functions into only one logic circuit. This not only reduces the number of logic gates but makes its power dissipation low.

4. Energy Efficient Circuits

It is important to understand how the energy is consumed in a circuit. An energy efficient design can berealised by either minimizing the energy consumption subject to a throughput constrain, or by maximizing the amount of computation for a given amount of energy. The optimal design can be made if the trade-off between the energy and delay can be met since it is possible to determine the lowest energy for a given level of performance. One approach in the system is to incorporate parallelism or pipelining.

*0.18μm geometry operating at 1.8V. CASE	Load Capacitance = 10 ff (equal to 4 Inverters)		Load Capacitance = 20 ff (equal to 8 Inverters)		Load Capacitance = 30 ff (equal to 12 Inverters)	
	Et (pJ) (worse case)	Et (pJ) (average case)	Et (pJ) (worse case)	Et (pJ) (average case)	Et (pJ) (worse case)	Et (pJ) (average case)
1	6.049	5.691	18.298	14.220	33.778	23.542
2	6.909	6.831	19.989	16.112	37.726	26.848
3	8.302	8.118	22.879	18.265	41.784	29.696
4	8.422	6.258	14.409	11.119	21.769	17.390
5	8.802	6.563	14.895	11.530	22.160	17.795
6	7.791	5.669	14.029	10.585	21.698	16.924
7	6.054	6.054	17.680	11.676	29.061	19.178
8	9.719	6.638	18.110	12.256	29.280	19.703
9	8.460	6.421	16.467	12.282	27.310	20.079
10	7.319	5.159	16.624	11.267	28.157	18.946

Table1 Energy Delay product simulation results of pass transmission gate multiplexer using transistor sizing with various load capacitances.

CASE	Pass Gates (-e06 m)		Inv_S (-e06 m)		Inv_nQ (-e06 m)	
	NMOS	PMOS	NMOS	PMOS	NMOS	PMOS
1	0.28	0.28	0.28	0.28	0.28	0.28
2	0.28	0.28	0.70	1.24	0.28	0.28
3	0.80	0.80	0.70	1.24	0.28	0.28
4	0.80	0.80	0.70	1.24	0.70	1.24
5	1.00	1.00	0.70	1.24	0.70	1.24
6	0.28	0.28	0.70	1.24	0.70	1.24
7	0.28	0.28	0.70	1.24	0.36	1.46
8	0.80	0.80	0.70	1.24	0.36	1.46
9	0.80	0.80	0.28	0.28	0.36	1.46
10	0.28	0.28	0.28	0.28	0.36	1.46

Table 2: The size of transistors.

In addition, if energy efficient circuits are used to implement the sub-component in the system, good energy efficiency can be achieved. Therefore, this section will discuss and show the energy-delay related to the output loads and transistor sizing. The multiplexer cell as shown in Figure5 implemented by pass transmission gate is the main cell applied everywhere within the FU datapath. If S is low, the top CMOS pass gate is on passing B to the output, whilst if S is high, the bottom CMOS pass gate is on and A passes to the output.

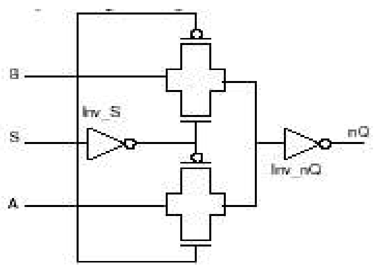

Figure5 A pass-transmission gate multiplexer

Gate sizing[16] can be used to trade-off energy and delay. The minimal energy delay point of a circuit is not only affected by its basic design but also by its output load capacitance. The logic gate capacitance and load capacitance including wiring both increase linearly with transistor size. Gate delay can be calculated as $t^d = \tau d$, where τ is a process-dependent constant, and d is a unitless delay of the gate. The unitless delay is determined as $d = h^{eff} + \rho$, where h^{eff} is the product of logical effort of the driving gate and electrical fan-out; a logical effort of 1 results from a minimum size inverter driving a similar inverter. The electrical fan out is the equivalent number of minimum size inputs being driven. The self-loading delay ρ is the product of logical effort (g) and the ratio of the equivalent driving gate width (W^{par}) to the equivalent driven gate width (W^{in}), $\rho = gW^{par}/W^{in}$. Because there are no publications relating to the logic effort of a pass-transistor logic as yet, we regard the multiplexer as a logic gate. In our work to find the minimal point of energy delay of multiplexer, the transistor size and loading capacitance for the gate in figure5 are varied as shown in Table1 and 2. The circuits were simulated on SPICE assuming a geometry of 0.18μm and an operating voltage of 1.8V.

Table1 shows the energy delay product resulting from the use of different transistor sizes in the multiplexer and different load capacitances. We have analyzed the energy delay product of the circuit with both average rise and fall times and worse case edge times. Clearly, transistor sizing can help the circuits to have a low energy delay product when load capacitance is increased. In practical, load capacitance is very important and has a large effect on both the performance and power of the system, especially in a large design. In addition, using the minimal transistor size cannot achieve optimum energy efficiency. It can be concluded from these results that for minimum energy-delay, the CMOS pass gate transistors should be minimum size, the inv_S should be a drive 1 gate (PMOS/NMOS width = 1.24e-06m/0.70e-06m) and be capable of driving the select signals on 2 or 3 multipliexer gates, and that the inv_nQ should be matched to the driven load (PMOS/NMOS width = 1.24e-06m/0.70e-06m). These figures do not include leakage power which is small for this process, provided the transistor sizing above is adopted.

5. Discussion and Conclusions

The results from remaining simulations on the full custom layout for the datapath shown in Figure2 indicate that our FU dissipates about 13.68mW@200MHz. Projecting this to four-way parallelism yields 800MHz with a power consumption of only 54.72mW. This indicates a factor of 10 improvement in the dissipation on the original FU design assuming a similar geometry and operating voltage.

Energy efficiency is a significant design constraint for battery powered DSPs requiring a coherent low energy technique applied at all levels and particularly at logic and circuit levels. The proposed FU architecture has been designed to be flexible, have low power and high throughput particularly in performing arithmetic operations involving the multiplier and adder. In addition, asynchronous data dependent operation and a tunable delay mechanism are incorporated to gain even better energy efficiency. At the circuit level, pass transistor logic is used which is low power without sacrificing performance. Transistor sizing has been applied to this pass transistor logic to identify the optimum trade-off between energy and delay.

Acknowledgement

This work was funded by EPSRC grant GR/S61270/01 and the authors are grateful for this support. The authors are also grateful to Dave Clark and Jeff Pepper for encouragement in this work.

References

[1] Reiner H., " Reconfigurable Computing: A New Business Model and its Impact on SoC Design." , Keynote speeches in *EUROMICRO Symposium on Digital System Design, Architectures, Methods and Tools*, September, 2001.

[2] Li-Hsun C., Chen O.T.-C. and Ruey-Ling M., "A high-efficiency reconfigurable digital signal processor for multimedia computing.", *Proceedings of the 2003 International Symposium on Circuits and Systems*, vol.2, pp. 768-771, May, 2003.

[3] Sangjin H., Shu-Shin C. and Connaway C., "Variable-rate pipelined multiplier design for reconfigurable DSP applications.", *45th Midwest Symposium on Circuits and Systems*, vol.1, pp. 587-590, August, 2002.

[4] Martina M., Masera G., Piccinini G., Vacca F. and Zamboni M., "Reconfigurable DSP IP for multimedia applications.", *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol.4, pp. 4179, May, 2002.

[5] I. Verbauwhede, P. Schaumont, C. Piguest and B. Kienhuis, "Architectures and design techniques for energy efficient embedded DSP and multimedia processing.", DATE'04, France, February, 2004.

[6] M.Lewis and L. Brackenbury, "CADRE: An Asynchronous Embedded DSP for Mobile Phone Applications. Design Automation for Embedded Systems.", Vol.6, No.4, pp.451-475, 2002.

[7]A. P. Chandrakasan and R. W. Brodersen, *"Low Power Digital CMOS Design."*, Kluwer Academic Publishers, 1995.

[8] I.E. Sutherland, "Micropipelines.", *Communications of the ACM*, vol.32, no.6, pp.720-738, June, 1980.

[9] M.E. Dean, T.E. Williams and D.L. Dill, "Efficient self-timing with level-encoded 2-phase dual-rail (LEDR). ", *MIT Conference on Advanced Research in VLSI*, March 1991.

[10] V. Varshavsky, V. Marakhovsy and M. Tsukisaka, "Data-controlled delays in the asynchronous design.", *Proceeding Of the 1996 IEEE International. Symposium Circuits and Systems (ISCAS 96)*, Atlanta (USA), vol. 4, pp. 153-155, May 1996.

[11] E. Grass, Viv Bartlett and Izzet Kale, "Completion-Detection Techniques for Asynchronous Circuits.", *IEICE Transaction Information and System*, vol.E80-D, no. 3, March 1997.

[12] S.M.Nowick, K.Y.Yun, P.A.Beerel and A.E.Dooply, "Speculative Completion for the Design of High-Performance Asynchronous Dynamic Adders.", *Proceedings of Async97*, pp. 210-223, April, 1997.

[13] A. D. Booth, "A Signed Binary Multiplication Technique.", *Quarter Journal Mech. Applicantion Math.*, vol. 4, pp. 236-240, 1951.

[14] C. S. Wallace, "A Suggestion for Fast Multipliers.", *IEEE Transactions Electronic Computer*, vol. EC-13. pp. 14-17, February, 1964.

[15] Keshab K. Parhi, "Low-Energy CSMT Carry Generators and Binary Adder.", *IEEE Transactions on Very Large Scale Integration(VLSI) Systems*, vol.7, no.4, pp.450-462, December, 1999.

[16] Ivan Sutherland, Robert Sproull and David Harris. *Logical Effort: Designing Fast CMOS Circuits*, Morgan Kaufmann, 1999.

[17] Keshab K. Parhi, "VLSI Digital Signal Processing Systems: Design and Implementation.", *John Wiley & Sons Inc.*, 1999.