IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁂ Peer-reviewed & Refereed journal ⁂ Vol. 13, Issue 12, December 2024

DOI: 10.17148/IJARCCE.2024.131217

Design and Implementation of Loeffler Architecture for 2D DCT/IDCT

Prof. Sujatha S Ari[1], Abhishek H R[2], Chandana D A[3], Druthi M[4], Govind V[5]

Assistant professor, Electronics and communication Department, East West Institute of Technology, Bangalore, India[1]

Student, Electronics and communication Department, East West Institute of Technology, Bangalore, India[2]

Student, Electronics and communication Department, East West Institute of Technology, Bangalore, India[3]

Student, Electronics and communication Department, East West Institute of Technology, Bangalore, India[4]

Student, Electronics and communication Department, East West Institute of Technology, Bangalore, India[5]

Abstract: This brief presents an approach for a technique to systematically tradeoff accuracy in exchange for area, power, and delay savings in digital circuits is proposed: gate-level pruning (GLP). The methodology is first demonstrated on adders, achieving up to 78% energy-delay-area reduction for relative error. It is then detailed how this methodology can be applied on a more complex system composed of a multitude of arithmetic blocks and memory: the discrete cosine transform(DCT), which is a key building block for image and video processing applications. Even though arithmetic circuits represent less than the entire DCT area, it is shown that the GLP technique can lead to energy-delay-area savings over the entire system for a reasonable image quality loss. This GLP approach can be Implemented using Verilog HDL and Simulated by Modelsim 6.4 c. Finally it's synthesized by Xilinx tool.

Keywords: DCT, Verilog HDL, Xilinx tool.

I. INTRODUCTION

Probabilistic pruning is a design technique that consists of removing circuit blocks and their associated wires in order to trade exactness of computation against power, area, and delay savings without any overhead. The amount of pruning is dictated by the application's error tolerance. A formal definition of probabilistic pruning, as well as the proof of concept, has already been addressed where the nodes are components such as gates, and whose edges are wires. The decision to prune a node is generally based on two criteria: the significance, which is a structural parameter, and the activity or toggle count (TC). The nodes with the lowest significance-activity product (SAP) are pruned first. By doing so, the error magnitude grows with the amount of pruning. Alternatively, depending on the application's requirements, the designer may choose to prune nodes according to the activity only in order to minimize the error rate, or by significance only in order to shorten design time by skipping the gate-level simulation process. The activity of each wire is extracted from the .SAIF file (Switching Activity Interchange Format) obtained through gate-level hardware simulations. This file contains the TC of each wire, as well as the time spent at the logic levels 0 and 1 (T0 and T1), respectively. While TC is used to rank the nodes, T0 and T1 are used later in the pruning process to set unconnected gate inputs to a specific value. Note that to get accurate activity estimation; the system should be simulated with an input stimulus representative of the real operation of the circuit. Pruning once the nodes are ranked according to their SAP, significance only or activity only, the gate-level net list is modified in order to remove unessential nodes from the design. For the sake of simplicity, and in order to maximize the use of the existing EDA tools, the probabilistic pruner does not literally remove the gates form the net list, but it disconnects the corresponding wires. Gates whose outputs are unconnected will automatically be removed by the synthesis tool. However, leaving gate inputs unconnected would fail the re synthesis of the design. For this reason, and in order to minimize the error, those inputs are set to 0 if they statistically spend most of the time at 0 (i.e., $T0 \geq T1$). otherwise they are connected to 1 (i.e., $T0 < T1$). This should allow to statistically reducing the error magnitude.

II. PROPOSED SYSTEM

A technique to systematically tradeoff accuracy in exchange for area, power, and delay savings in digital circuits is proposed: gate-level pruning (GLP). pruning is a design technique that consists of removing circuit blocks and their associated wires in order to trade exactness of computation against power, area, and delay savings without any overhead. The amount of pruning is dictated by the application's error tolerance.

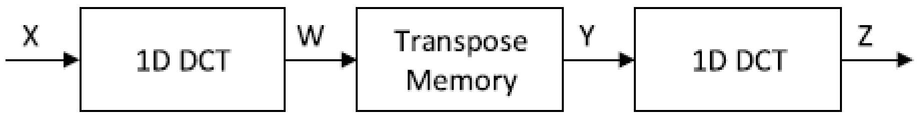


Fig:1-2 D DCT BLOCK DIAGRAM

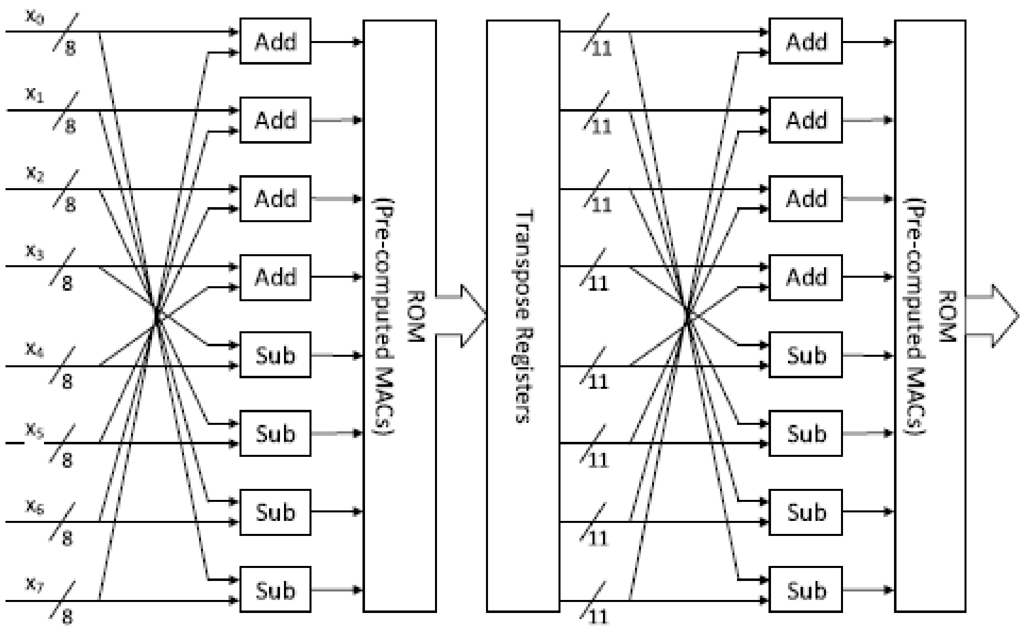


Fig:2 1 D DCT Block

The Two-Dimensional DCT

The objective of this document is to study the efficacy of DCT on images. This necessitates the extension of ideas presented in the last section to a 20space.

The 2-D DCT is a direct extension of the 1-D case and is given by

$$C(u,v)=\alpha(u)\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right]$$

For u,v=0,1,2,…,N −1 and $\alpha(u)$ and $\alpha(v)$ are define in above equation.
The inverse transform is defined as

$$f(x,y)=\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}\alpha(u)\alpha(v)C(u,v)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right]$$

for x,y=0,1,2,…,N −1.

The 2-D basis functions can be generated by multiplying the horizontally oriented 1-D basis functions. The basis functions for N = 8 are shown in. Again, it can be noted that the basis functions exhibit a progressive increase in frequency both in the vertical and horizontal direction. The top left basis function of results from multiplication of the DC component in with its transpose. Hence, this function assumes a constant value and is referred to as the DC coefficient.

DOI: 10.17148/IJARCCE.2024.131217

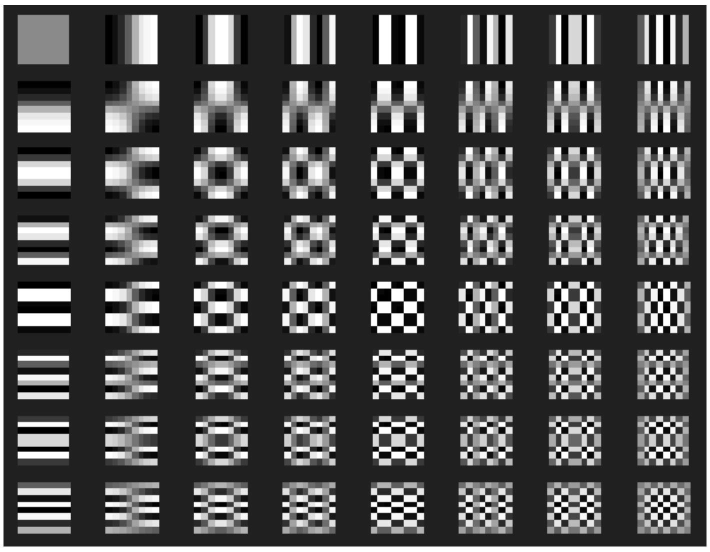


Fig: 3 Two dimensional DCT basis functions (N = 8). Neutral gray represents zero, white represents positive amplitudes, and black represent negative amplitude

III. SIMULATION IMPLEMENTATION

HARDWARE DESCRIPTION LANGUAGE (HDL):

The dramatic increase in the logic density of silicon chips has made it possible to implement digital systems with multimillion gates on a single chip. The complexity of such systems makes it impractical to use traditional design descriptions (e.g., logic schematics) to provide a complete and accurate description of a design. Currently, all complex digital designs are expressed using a hardware description language (HDL). An HDL, unlike traditional programming languages such as C or C++, can describe functions that are inherently parallel. A major advantage of an HDL is that it provides a better and more concise documentation of a design than gate-level schematics. Two very popular HDLs are VHDL and VERILOG. In this text we use VHDL. VHDL is an acronym for VHSIC hardware description language; VHSIC in turn is an acronym for very highspeed integrated circuit. The development of VHDL was funded by the U.S. Department of Defense (DoD) in the early 1980s. The syntax of VHDL is similar to that of programming language ADA; however, it has some significant differences from ADA. We present the important concepts of VHDL, especially the ones that are used in digital circuit design. VHDL can provide unambiguous representation of a design at different levels of abstraction as shown. Modern CAD (computer-aided design) tools can generate gate-level implementation of a design from its VHDL description. A behavioral VHDL description of a circuit describes the function of the circuit in terms of its inputs using the types of statements used in a high-level programming language. The objective is to describe the correct operation of a circuit to be designed without being concerned with redundant details. This description does not specify how the function is actually implemented; thus the same description may result in several implementations of a circuit. The register transfer level (RTL) description of a circuit specifies the flow of data from an input or a register to another register or the output of the circuit through combinational logic blocks. The RTL description is also known as data flow description. The structural level description specifies what components a circuit is composed of and how these components are interconnected. This is similar to the logic schematic diagram of a circuit. The purpose of this tutorial is to describe the modeling language VHDL. VHDL includes facilities for describing logical structure and function of digital systems at a number of levels of abstraction, from system level down to the gate level. It is intended, among other things, as a modeling language for specification and simulation. We can also use it for hardware synthesis if we restrict ourselves to a subset that can be automatically translated into hardware. VHDL arose out of the United States government's Very HighSpeed Integrated Circuits (VHSIC) program. In the course of this program, it became clear that there was a need for a standard language for describing the structure and function of integrated circuits (ICs). Hence the VHSIC Hardware Description Language (VHDL) was developed. It was subsequently developed further under the auspices of the Institute of Electrical and Electronic Engineers (IEEE) and adopted in the form of the IEEE Standard 1076, Standard VHDL Language Reference Manual, in 1987. This first standard version of the language is often referred to as VHDL-87. Like all IEEE standards, the VHDL standard is subject to review at least every five years. Comments and suggestions from users of the 1987 standard were analyzed by the IEEE working group responsible for VHDL, and in 1992 a revised version of the standard was proposed. This was eventually adopted in 1993, giving us VHDL-93. A further round of revision of the standard was started in 1998.

DOI: 10.17148/IJARCCE.2024.131217

That process was completed in 2001, giving us the current version of the language, VHDL-2002. This tutorial describes language features that are common to all versions of the language. They are expressed using the syntax of VHDL-93 and subsequent versions. There are some aspects of syntax that are incompatible with the original VHDL-87 version. However, most tools now support at least VHDL-93, so syntactic differences should not cause problems.

LEVELS OF REPRESENTATION AND ABSTRACTION

A digital system can be represented at different levels of abstraction . This keeps the description and design of complex systems manageable. Figure 1 shows different levels of abstraction.

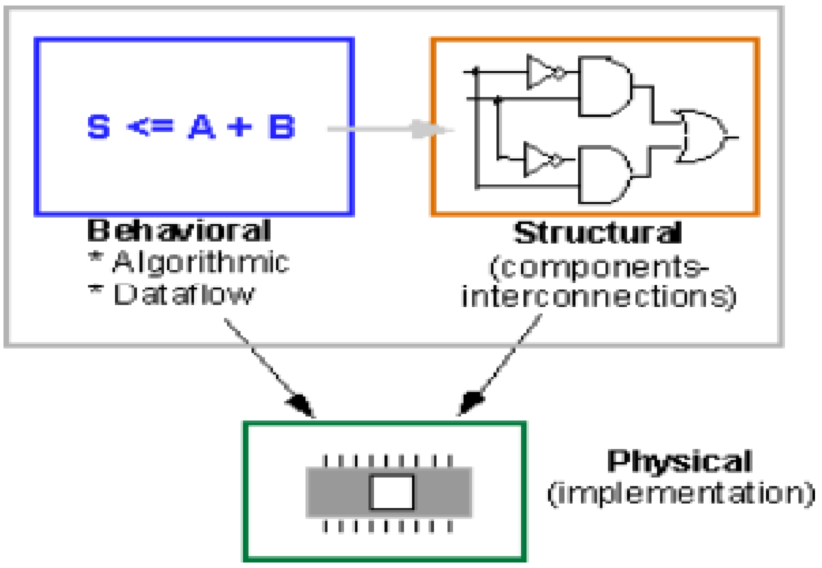


Fig:4 Behavioral, Structural and Physical

BEHAVIORAL LEVEL

The highest level of abstraction is the **behavioral** level that describes a system in terms of what it does (or how it behaves) rather than in terms of its components and interconnection between them. A behavioral description specifies the relationship between the input and output signals. This could be a Boolean expression or a more abstract description such as the Register Transfer or Algorithmic level.

STRUCTURAL LEVEL

The **structural** level, on the other hand, describes a system as a collection of gates and components that are interconnected to perform a desired function. A structural description could be compared to a schematic of interconnected logic gates. It is a representation that is usually closer to the physical realization of a system. For the example above, the structural representation is shown.

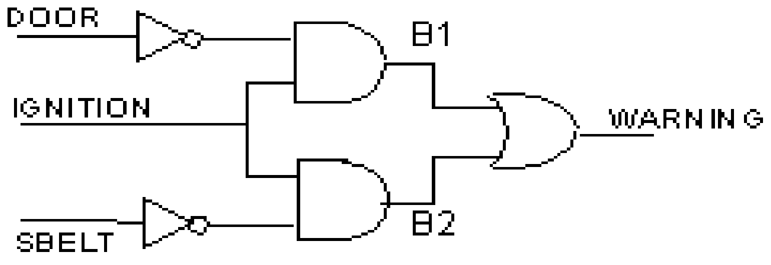


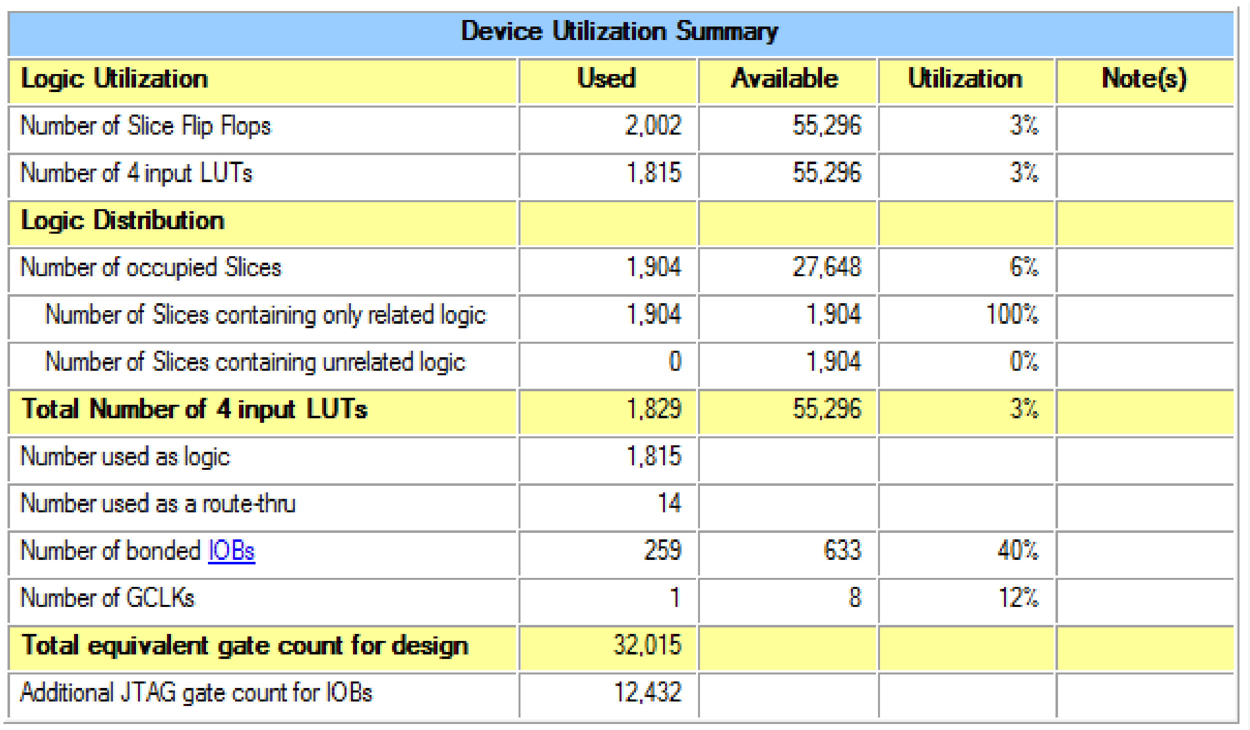
Fig:5 Structural representation of a "buzzer" circuit

VHDL allows one to describe a digital system at the structural or the behavioral level. The behavioral level can be further divided into two kinds of styles: **Data flow** and **Algorithmic**. The dataflow representation describes how data moves through the system. This is typically done in terms of data flow between registers (Register Transfer level). The data flow model makes use of concurrent statements that are executed in parallel as soon as data arrives at the input. On the other hand, sequential statements are executed in the sequence that they are specified. VHDL allows both concurrent and sequential signal assignments that will determine the manner in which they are executed.

VERILOG

Verilog is one of the two major Hardware Description Languages (HDL) used by hardware designers in industry and academia Verilog is very C-like and liked by electrical and computer engineers as most learn the C language in college. Verilog was introduced in 1985 by Gateway Design System Corporation, now a part of Cadence Design Systems, Inc.'s Systems Division. Until May, 1990, with the formation of Open Verilog International (OVI), Verilog HDL was a proprietary language of Cadence. Cadence was motivated to open the language to the Public Domain with the expectation that the market for Verilog HDL-related software products would grow more rapidly with broader acceptance of the language. Cadence realized that Verilog HDL users wanted other software and service companies to embrace the language and develop Verilog-supported design tools.

DEVICE UTILIZATION SUMMARY:

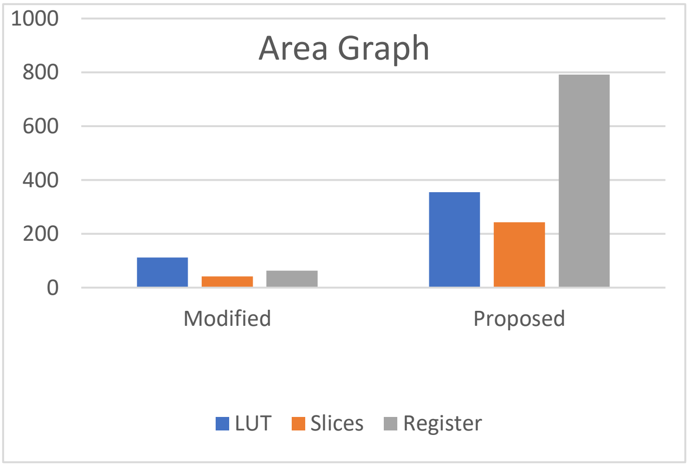


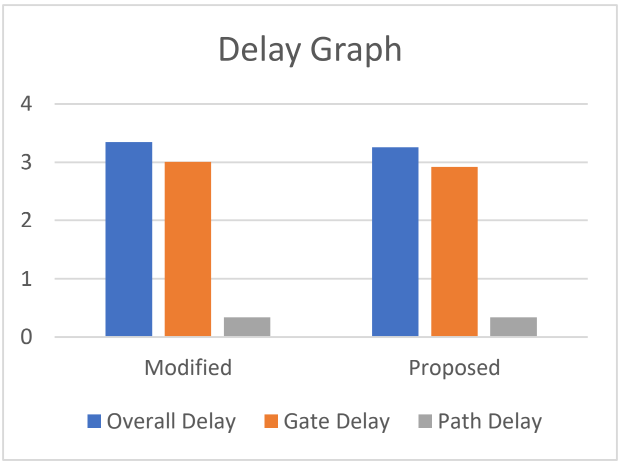
Device Utilization Summary

Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	2,002	55,296	3%	
Number of 4 input LUTs	1,815	55,296	3%	
Logic Distribution				
Number of occupied Slices	1,904	27,648	6%	
Number of Slices containing only related logic	1,904	1,904	100%	
Number of Slices containing unrelated logic	0	1,904	0%	
Total Number of 4 input LUTs	1,829	55,296	3%	
Number used as logic	1,815			
Number used as a route-thru	14			
Number of bonded IOBs	259	633	40%	
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	32,015			
Additional JTAG gate count for IOBs	12,432			

IV. RESULTS

Selected Device: Vertex 5vlx50tff665-1	Area			Delay		
	LUT	Slices	Register	Overall Delay	Gate Delay	Path Delay
Modified	112	42	64	3.344ns	3.008ns	0.336ns
Proposed	355	243	792	3.259ns	2.923ns	0.336ns





V. CONCLUSION

In this paper, we have proposed a recursive algorithm to obtain orthogonal approximation of DCT where approximate DCT of length could be derived from a pair of DCTs of length at the cost of additions for input preprocessing. The proposed approximated DCT has several advantages, such as of regularity, structural simplicity, lower-computational complexity, and scalability. Comparison with recently proposed competing methods shows the effectiveness of the proposed approximation in terms of error energy, hardware resources consumption, and compressed image quality. We have also proposed a fully scalable reconfigurable architecture for approximate DCT computation where the computation of 16-point DCT could be configured for parallel computation of two 16-point DCTs or four 8-point DCTs. This paper presented a methodology and a CAD tool integrated in a standard digital flow to automatically trade a determined amount of accuracy in exchange for area power and delay savings. While gains achieved on adder circuits are already interesting, reduction area for adders, those could be insignificant since an adder generally only represents a small fraction of the system it is placed in. It is therefore interesting to apply hardware accelerator such as the DCT with the state-of-the-art distributed arithmetic architecture, which is built out of multiple arithmetic circuits and memory.

REFERENCES

[1]. S. Jankowski, J. Covello, H. Bellini, J. Ritchie, and D. Costa, "The Internet of Things: Making sense of the next mega- trend," Goldman Sachs, 2014. [Online]. Available: http://www.goldmansachs.com/ourthinking/pages/internet-of-things/iot-report.pdf
[2]. K. V. Palem, "Energy aware computing through probabilistic switching: A study of limits," IEEE Trans. Comput., vol. 54, no. 9, pp. 1123–1137, Sep. 2005.
[3]. S. Cheemalavagu, P. Korkmaz, K. V. Palem, B. E. Akgul, and L. N. Chakrapani, "A probabilistic CMOS switch and its realization by exploiting noise," in Proc. IFIP Int. Conf. VLSI, Oct. 2005, pp. 535–541.
[4]. P. Korkmaz, B. E. S. Akgul, K. V. Palem, and L. N. Chakrapani, "Advocating noise as an agent for ultra-low energy computing: Probabilistic complementary metal–oxide–semiconductor devices and their characteristics," Jpn. J. Appl. Phys., vol. 45, no. 4B, p. 3307, 2006.
[5]. G. Karakonstantis and K. Roy, "Voltage over-scaling: A cross-layer design perspective for energy efficient systems," in Proc. 20th Eur. Conf. Circuit Theory Design (ECCTD), Aug. 2011, pp. 548–551.
[6]. J. George, B. Marr, B. E. S. Akgul, and K. V. Palem, "Probabilistic arithmetic and energy efficient embedded signal processing," in Proc. Int. Conf. Compil., Archit. Synth. Embedded Syst., Ser. (CASES), New York, NY, USA, Oct. 2006, pp. 158–168. [Online]. Available: http://doi.acm.org/10.1145/1176760.1176781
[7]. D. Ernst et al., "Razor: A low-power pipeline based on circuitlevel timing speculation," in Proc. 36th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO), Dec. 2003, pp. 7–18.
[8]. P. K. Krause and I. Polian, "Adaptive voltage over-scaling for resilient applications," in Proc. Design, Autom. Test Europe Conf. Exhibit. (DATE), Mar. 2011, pp. 1–6.
[9] S. Ghosh, S. Bhunia, and K. Roy, "CRISTA: A new paradigm for lowpower, variation-tolerant, and adaptive circuit synthesis using critical path isolation," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 26, no. 11, pp. 1947–1956, Nov. 2007.