Quality Driven Dynamic Low-power Reconfiguration of Handhelds

S.S. Verma, Hiren Joshi[1] and G.K. Sharma[2]

[1] Rajasthan University, India
Email: ssv445@rediffmail.com, hiren@mlvti.ac.in
[2] I.T. Department, ABV-IIITM,Gwalior, INDIA
Email: gksharma@iiitm.ac.in

Abstract. Run time reconfiguration of mobile devices to optimize power according to user experience is a significant research challenge. The paper investigates a novel technique of generating power-optimized bitstream to be used for on-the-fly remote-reconfiguration of mobile devices. The approach dynamically minimizes the bitwidths of variables of applications residing in mobile and allows the user to tradeoff between power and quality. Experimental results for MPEG2-Decoder show that the approach is able to reduce power consumption dynamically by 33.58% for 25% PSNR degradation. Much higher power savings can be achieved if user is willing to tolerate more quality degradation.

1 Introduction

To fulfill the user requirement of Quality of Services (QoS), designers reserve extra resources that lead to an inefficient design in terms of power. This penalty of power for getting better quality starts creating problem when the user is out of power.Therefore, the system needs to have real time properties and must adapt itself with respect to changes in user requirement of QoS. When user feels that a device is running out-of-power, he can save power by requesting low quality of service. The proposed well-structured framework presents a practical approach for run-time optimization for power in mobile devices. This framework is experimented on bitwidth optimization for MPEG-2 and results agreed with the practical importance of runtime and remote optimization of bitwidth for power saving.

For optimization of multimedia algorithms during high-level synthesis, several lossless static Bit-Width Optimization (BWO) techniques have already been proposed. But the overall saving can be increased if Quality driven lossy approach is adapted according to run-time requirements. Although the application of lossy approaches results in a moderately degraded quality but with this acceptance one can save significant amount of power. In lossy approach we reduce the bit width of different variables of algorithms beyond the permissible limit and achieve significant power saving.

As mobile devices provide a low computation capabilities and little power budget, above lengthy and compute intensive processes cannot be executed on

them. Present framework provides the support for runtime BWO at remote server resulting in power, time and resources saving of mobile device. The overall process goal can be stated as "To generate an optimum reconfiguration bitstream, in response to desired quality request from user, which can dynamically reconfigure the mobile device for improved power efficiency against quality compromise."

The contributions of this paper are therefore:

1. The novel technique of runtime RQoS driven bitwidth analysis.
2. Concept of user participation in the trade-off of Quality and Power.
3. An introduction to design using the KSHITIZ framework that implements the proposed optimization procedures.
4. An evaluation of proposed technique for IDCT module of MPEG2, showing power saving upto 33.58% for the penalty of 25% PSNR.

Section-2 deals with the related work, section-3 presents an overview of present framework where as Section-4 embodies Video Quality in presepective of PSNR.Section-5 explains two concepts of Static BWO and Quality driven Dynamic BWO. Section-5 and Section-6 the elobrates process of Quality Selection. The next section deals with the implementation of our framework and experimental results. Section-6 presents the related work and Section-7 concludes the work.

2 Related Work

Extensive research have been carried out on power optimization in high-level synthesis for ASIC designs [1, 2, 3, 4, 5]. It is customary to note that the dynamic reconfigurability charater of FPGA opens several new avenues for power optimization.Dynamically reconfigurable FPGAs are presented in [6, 7]. It is noteworthy to record that such type of reconfiguration involves cosiderably very small fraction of time.

A variety of work has been done on QoS with the application of reconfigurability for energy efficiency. Padmanbhan Pillai et. al.[9] developed the energy aware framework. It can manage real time tasks for given energy budget targeting energy saving. When experimented with DVS, the framework results into significant savings.

The technique described in [10], explores the acceptable degradation in PSNR of MPEG compressed image for power savings. The authors shown the results for MPEG decoder and overall approach was implemented using Dynamic Voltage Management(DVM). Clark N. Taylor and Sujit Dey [11] introduced bandwidth and energy consumption as major bottlenecks of wireless multimedia communication. To overcome these, authors approached for an adaptive image compression algorithm that can adapt itself for current communication conditions and constraints. The experiments were realized on JPEG and minimized the energy

in compression while respecting latency, bandwidth and quality constraints. In [12], Lodewijk T. Smit et. al. described the need of runtime reconfiguration in changing mobile environment and minimization of energy consumption in mobile systems for certain QoS. For the purpose of resource allocation, decision has been made effective by studying trade-off between energy saving and reconfiguration cost(in terms of time and power).

Lossless bitwidth optimization exposed greatly but lossy bitwidth optimization has not been exposed very well. The root cause behind this is that the lossy approach depends upon application. Emre Ozer et al. [13] applied dynamic bitwidth analysis and reduced 35% bit width of overall program. Richardson [14] investigated the relationship between precision and error for image compression and decompression. For different fixed precision the DCT algorithm was executed and PSNR v/s Bit-saving trade-off was recorded. Nikil Dutt et. al.[23] proposed an integrated power management approach for streaming video applications on handheld devices. The approach consists of low level architectural optimizations (CPU, memory, register), OS power-saving mechanisms (DVS) and adaptive middleware techniques (admission control, optimal transcoding, network traffic regulation). The approach resulted into energy gains of 57.5% for the CPU and memory while 70% of energy saving in the wireless network interface.

Our approach differs from what is discussed above in respect that we focus on QoS for reconfigurability of device rather than data streams. It empowers the user to select any desired quality, not restricting him to few quality levels and even the automatic quality selection by the device. The approach also evolves a smart user-interface to immediately select and transmit the quality parameters.

3 Framework

The framework presents a practical approach for run-time high-level optimization of power with user-desired compromise in quality. Framework supports series of static and dynamic optimizations like loop unrolling, dynamic voltage scaling etc. and generates bitstream for partial reconfiguration of mobile architecture. As quality requirement changes dynamically, quality related optimizations must be done dynamically while quality independent optimization must be applied statically. In this paper we focused on "Quality Independent Bitwidth Optimization" (QIBO) and "Quality Dependent Bitwidth Optimization" (QDBO) and results are verified for the practical importance of runtime and remote bitwidth optimization for power saving.

The network view of framework is shown in fig. 1 explains the overall approach. Client is the user of mobile device while remote server provides the reconfiguration support to user. Whenever user feels that the device is running out-of-power, and if one wishes to save power, initiates the process by selecting the required level of QoS. This is converted to technical specifications by client model of the framework and the parameters are sent to remote server through communication network. Server calculates the new optimized design and extracts the reconfiguration bit-stream, which when sent to the client, re-

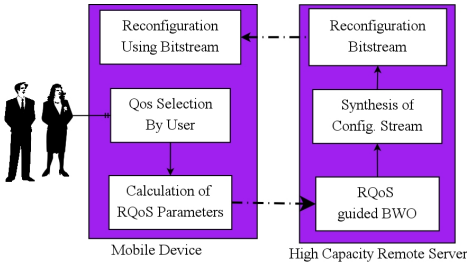


Fig. 1 Framework: Across Network

configures the device for reduced power consumption and compromised quality. For Virtex XCV1000, reconfiguration takes 58.658 microseconds, *idct* configuration bit-stream transmission requires 28.56 seconds on GSM with transfer rate 9.6 Kbps while synthesis process on a high capacity server consumes few seconds only. Thus user can achieve reconfigured device in less than 1 minute.

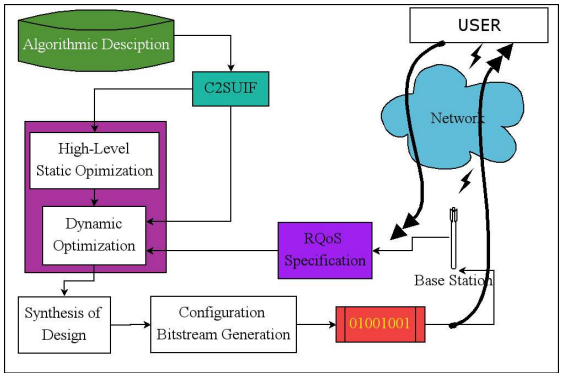


Fig. 2 Processing at Remote Server

Fig. 2 explains the remote processing at high capacity server. Server has the results of QIBO process, which is done only once for a particular program code, as these properties do not change at runtime while QDBO is performed for every user request. QDBO carefully reduces the accuracy of the variables and at the same time observes the impact on quality by executing the algorithm. The accuracy is reduced further till the limit of quality parameters is attained. Resultantly, the quality is gracefully degraded with decrease in accuracy of variables.

At the server, both QIBO and QDBO are done on equivalent SUIF-IR of program code, obtained using SUIF compiler. This design is synthesized into

reconfiguration bit-stream using SUIF2VHDL converters and standard synthesis tools.

4 Video Quality and PSNR

The Peak Signal to Noise Ratio (PSNR) of an image is a measure of distortion of an image related to a reference image. It can be used to measure the distortion of an image or frame due to one or the other errors as compared with the original error-free image. PSNR is defined as:

$$PSNR(dB) = 10log_{10}\frac{(2^n - 1)^2}{MSE} \tag{1}$$

where, n is the number of bits required to represent each pixel and MSE is the Mean Square Error between the distorted frame and the original frame.

PSNR is relatively easy to calculate(2) and provides a rough approximation to the visual quality of the video frame. In general, a high PSNR indicates a high-quality frame. Quality degradations due to high compression and/or transmission errors, lead to a drop in PSNR. It is possible to obtain an approximate comparison of the quality of two video sequences by comparing the PSNR of the frames in each sequence, relative to the original video sequence. Calculation of average PSNR of all the frames in a video sequence gives a quality measure in dB.

5 Bitwidth Analysis

It has been proved that often the most efficient implementation of an algorithm in the reconfigurable hardware is one in which the bitwidth of each internal variable is fine-tuned to the best precision [15, 16, 17, 22, 13]. The quality observable at the mobile device is a function of the accuracy and therefore depends upon bitwidths used in representation of all intermediate variables in the algorithm.

We have proposed a two fold scheme of saving in the bitwidths of variables. First we find the bits in each variable, which are not affecting the output quality. This normally includes the blank MSB bits (e.g. in an integer variable which is assigned a value 4, the 29 MSB bits out of total reserved 32 bits are blank) and some LSB bits.

This analysis can be done once, statically for an algorithm and is named as Quality Independent Bitwidth Optimization (QIBO). The QIBO is lossless i.e. bitwidth saving in this phase does not introduce error and thus does not affect the output quality. Secondly bitwidths of all variables are gradually decreased from LSB side till the degradation in quality up to acceptable level. For getting maximum saving with fewer penalties on quality we have proposed a greedy approach. This phase is named as Quality Dependent Bitwidth Optimization (QDBO).

Input: Min_Err
Output: msb_saving, lsb_saving

```
1: vector msb_saving (1 to n) ← 0
2: vector lsb_saving (1 to n) ← 0
3: vector variables(1 to n)
4: constant assigned_bit_width ← 32
5: for all i such that 0 ≤ i ≤ n do
6:    for k = 0 to assigned_bit_width do
7:       lsb_saving(i) ← lsb_saving(i) + 1
8:       Calculate MSE with lsb_saving in bitwidth
9:       if (MSE ≤ Min_Err) then
10:         continue
11:      else
12:         lsb_saving(i) ← lsb_saving(i) - 1
13:         break
14:      end if
15:   end for
16:   for k = 0 to assigned_bit_width do
17:      msb_saving(i) ← msb_saving(i) + 1
18:      Calculate MSE with msb_saving in bitwidth
19:      if (MSE ≤ Min_Err) then
20:         continue
21:      else
22:         msb_saving(i) ← msb_saving(i) - 1
23:         break
24:      end if
25:   end for
26: end for
```

Fig. 3 Algorithm of QIBO: No. of variable in function is n. msb_saving and lsb_saving are saving in variable from MSB side and LSB side respectively. Bitwidth initially assigned by programmer/compiler is $assigned\_bit\_width$. MSE is Mean Square Error and Min_Err is threshold error in an image. Line 6-15 identifies LSB bits having error impact on image less than threshold (Min_Err). Simillarly line 16-25 identifies MSB bits with the same property.

5.1 Quality Independent Bitwidth Optimization (QIBO):

Several systems with bitwidth optimization have been proposed in the recent past[15, 16, 17, 22, 13]. In most of them concept of forward and backward propagation, based on arithmetical operations are used. In a departure from previous work, for static analysis we have used a pseudo-simulation method, bitwidths are varied from MSB as well from LSB side till the MSE remains less than threshold (MIN_Err). MSE is calculated by adding difference i.e. error between all pixels

of original and new image.(cf.- 2).

$$MSE = \frac{1}{n^2} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \{f^{'}(i,j) - f(i,j)\}^2 \tag{2}$$

We have experimentally proved that much larger bit saving can be achieved by this method than the forward-backward propagation. The method is more time consuming but as it is done once statically therefore not a bottleneck for real-time requirements. The pseudo-code of the algorithm is shown in fig. 3. The QIBO phase generates two vectors *lsb_saving* and *msb_saving* as shown in fig.4. The vectors are n-bit long. The i^{th} bit, in *msb_saving* vector represents the bit saving from MSB side in the i^{th} variable. Similarly i^{th} bit in the *lsb_saving* vector represents the bit saving from LSB side in the i^{th} variable. The QIBO is done only once for any algorithm and results are made available for the QDBO at remote server.

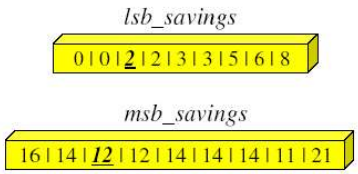


Fig. 4 Two vectors *msb_saving* and *lsb_saving* generated by QIBO. In *lsb_saving*, 3^{rd} bit from left represents that 2 bit can be saved from variable *x2*. While 3^{rd} bit in *msb_saving* represents that 12 bits are unused from MSB side.

5.2 Quality Dependent Bitwidth Optimization (QDBO):

The key idea over here is that once the user selects the required quality at which he is comfortable the tolerable error is calculated for this quality-level. Precision of all the variables are again fine-tuned and the narrowest bitwidth of each variable is determined that retains the requested quality level. The multimedia algorithm, which resides in the mobile as hardware is then reconfigured to provide the desired quality at lower power consumption. When the RQoS is high, less error can be tolerated but for low quality levels larger amount of errors are acceptable and therefore greater scope for bitwidth and power saving.

Quality parameter (PSNR) is calculated from the user request made through quality selection interface on mobile device. The QDBO phase (pseudo-code in fig. 5)initiated at server every time after receiving the required PSNR and the acceptable MSE for the desired PSNR is calculated. The QDBO uses *lsb_saving*

Input : Req_PSNR, lsb_saving
Output : lsb_saving, $Error$

1: $Req\_MSE \leftarrow 255 * 255/(10^{Req\_PSNR/10})$
2: **vector** $variables$ [1 to n]
3: $var\_sel \leftarrow 0$
4: **while** $var\_sel \neq -1$ **do**
5: $Error \leftarrow Req\_MSE$
6: **for all** i such that $0 \leq i \leq n$ **do**
7: lsb_saving(i) $\leftarrow$ lsb_saving(i) + 1
8: Calculate MSE with lsb_saving in bitwidth
9: **if** $(MSE \leq Error)$ **then**
10: $Error \leftarrow MSE$
11: $var\_sel \leftarrow i$
12: **end if**
13: $lsb\_saving(var\_sel) \leftarrow lsb\_saving(var\_sel)$ - 1
14: **end for**
15: Increment $lsb\_saving(var\_sel)$ by 1
16: $Error$ = Calculate MSE with new bitwidths specified by lsb_saving
17: **end while**

Fig. 5 Algorithm of QDBO: No. of variable in function is n. lsb_saving are saving in variable from LSB side. $PSNR$ is Peak Signal to Noise Ratio. MSE is New MSE each time bit-width changes. Line 6-14 identifies a variable whose LSB bits have least error impact on image(fig. 6). Line 15 increases the saving in selected variable by 1. At last, $Error$ stores introduced error by lsb_saving.

vector as starting point and further increase the bit saving at each position in the vector by one. The corresponding error is each case is calculated and finally bit saving in the bit of lsb_saving vector which introduces minimum error, is only considered. The lsb_saving vector is modified by making increment in the corresponding bit, all other bits remain unchanged. At the same time the bitwidth of the variable under consideration is reduced by one from LSB side, see fig. 6. The process is repeated till the error introduced remain lower then the MSE. The final bit saving in each variable is reflected by the two vectors msb_saving and lsb_saving.

6 Quality Selection Interface

The present framework suggests a user-friendly interface for quality selection as shown in fig. 7. The interface is based on random error generation. With the aid of inbuilt random error generator the quality of image is slightly perturbed, the degraded PSNR of resulting image is observed. Therefore, by doing this, an approximate quality variance is simulated at low capacity handheld.

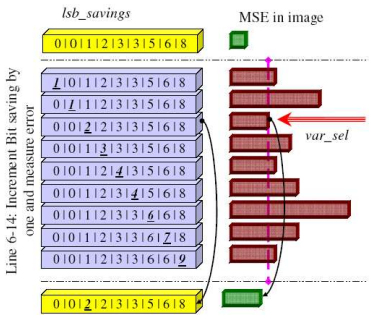


Fig. 6 Selection of Target Variable During QDBO

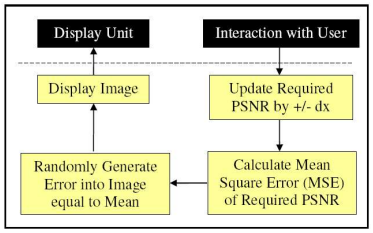


Fig. 7 Quality Selection Interface

The selection of quality parameters from user point of view are as follows: Firstly user changes the quality by using a fine-tuning variable command which internally changes the amount of error introduced by random error generator. The corresponding effect on quality would be displayed on the screen. The moment a user finds it suitable to his purpose, he would confirm this decision to device. As soon as the user confirms the quality, corresponding PSNR and algorithm specifications details are being forwarded to server for further optimization of Bit-Width.

Many researchers like [23], have suggested to store different reconfiguration bitstreams of discrete video quality levels for self-reconfiguration support. But some open questions are still remain unanswered with the approach such as, How can a small handheld accommodate significantly increased area due to storage of bitstreams? e.g. A small reconfigurable application takes 50,000 bits. so each copy stores almost 48 KB. Thus for each application with 8 quality levels, device will have to store approximately 48*8=384 KB, merely supports only a limited number of applications. Moreover, if the user wants to compromise with certain features of application then it would no be possible with discrete quality level approach.

To cop-up with future requirement, our framework is viable under the user

requirement of varied quality. This is possible due to proposed real-time remote reconfiguration support. The approach promises the benefits at a low cost of network transmission, which can be further reduced using difference based partial reconfiguration schemes. The biggest advantage is that network service provider can be a remote server and can guide user's device more precisely to save power.

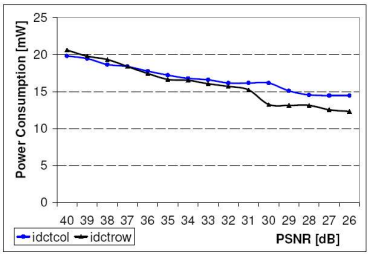
7 Experiments and Result

For the validation of proposed methodology, here we are presenting the experiments and results. The experiments were solely based upon functionality of MPEG2decoder. C source code of MPEG2decoder is downloaded from MPEG Software Simulation Group. The MPEG-2 core consists of several function blocks such as IDCT, motion estimation blocks, variable length encoding/decoding blocks and many more. IDCT is the kernel part of Mpeg2decode for computation, it consists of three functions, which are *idct()*, *idctrow()* and *idctcol()*. What follows is the experiment on *idctrow* and *idctcol*, the core and compute-intensive functions of MPEG2decoder algorithm.

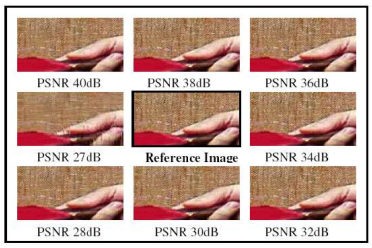
The functions contain 9 integer variables (x0 to x8). We refine bitwidth of each of 9 variables firstly by applying QIBO and then QDBO. Thus each variable is fine-tuned to contribute maximum in power saving. We calculated these bit-widths for 40 dB to 26 dB PSNR, which was calculated with reference to the image obtained using standard implementation of MPEG decoder. The framework supports every positive PSNR but our experiments showed that the quality of image less than 26dB is degraded beyond the acceptable range, thus presenting only the saving between 40dB to 26dB PSNR (Table 1).

Table 1 Power consumption and bitwidth saving with PSNR: Here bit saving due to dynamic analysis is shown. Power is reducing with decreasing PSNR. Average power saving for *idctrow*and *idctcol* is 28.77% and 23.26%.

PSNR	***idctrow***		***idctcol***	
	Dynamic Bit-Saving	**Power Consumption**	**Dynamic** Bit Saving	**Power** Consumption
40	16	20.62	23	19.84
38	27	19.33	36	18.66
36	43	17.46	46	17.77
34	51	16.53	57	16.78
32	58	15.71	64	16.16
30	79	13.26	64	16.16
28	80	13.14	82	14.55
26	87	12.32	83	14.46



(a) Power consumption with quality specification

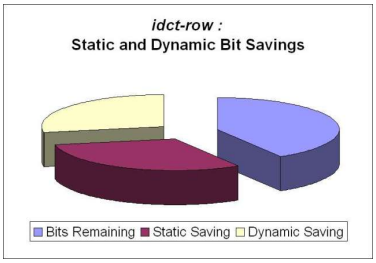


(b) Visual Image quality results with decreasing PSNR

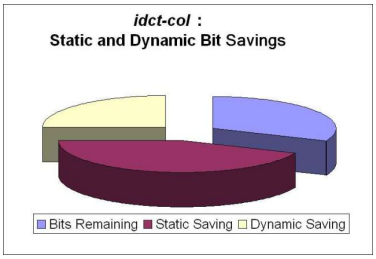
Fig. 8 Experimental Results Obtained at Mobile Device

The power saving results were extracted using the tool Xpower of Xilinx Inc. The code of *idctrow* and *idctcol* was firstly converted into equivalent VHDL code and then synthesized using Xilinx synthesis tools for VirtexII series FPGA. The synthesized design was used to estimate power consumption in the above functions using Xpower. The process was repeated for each PSNR ranging between 40dB to 26dB with the optimum bitwidth suggested by the present framework (fig. 8(a),fig. 8(b)).

Fig. 9 shows the static and dynamic bitwidth savings ratio. Dynamic saving is significant to contribute power saving. Bitwidth saving can be observed in fig. 10, 10(b).

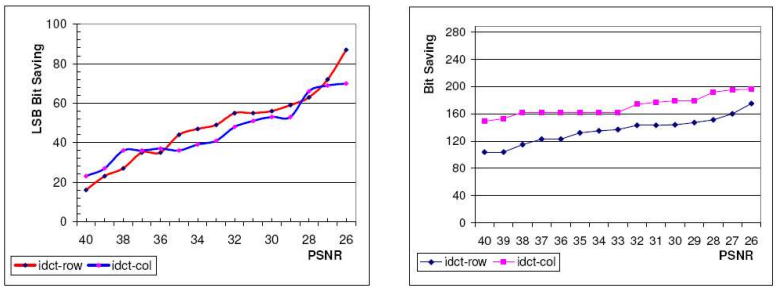


(a) *idctrow*



(b) *idctcol*

Fig. 9 Static and Dynamic Savings



(a) Dynamic saving (b) Total Saving

Fig. 10 Bitwidth Saving

8 Conclusion

Battery life constraints will limit the capabilities of a mobile device without significant energy reduction techniques and energy saving architectures. The methods presented here show the great potential of incorporating user experience of quality in power optimization in an inherent dynamic environment of mobile systems. The high level optimizations, when driven by RQoS, explore more wide scope for power saving at the expense of a small amount of quality. The framework presented here explores the power saving capability of run-time bitwidth optimization with acceptable compromise in quality of multimedia. We first introduced a new Quality Dependent Bitwidth Optimization (QDBO) algorithm. The experiments on *idct* show that it is saving 23.09% more bitwidths by QDBO. A new method for measurement of user perception of quality at client is also proposed. Other high level optimization mixing with RQoS may provide better and wider ranging quality/energy tradeoffs, and are a subject of our future work. Furthermore, we are going to use these techniques to optimize not only the MPEG but also the other modules.

References

1. A. Raghunathan and N.K. Jha, "Behavioral Synthesis for Low-power", in Proc. of *International Conference on Computer Design*, Oct 1994.
2. P. Kollig and B.M. Al-Hashimi,"A New Approach to Simultaneous Scheduling, Allocation and Binding in High-level Synthesis", in Proc. of *IEEE Electronics Letters*, vol. 33, Aug 1997.
3. A.P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey and R.W.Brodersen, "Optimizing Power Using Transformations", in Proc. of *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 1,pp. 12-31, Jan. 1995.

4. A. Raghunathan and N.K. Jha, "SCALP: An Iterative Improvementbased Low-power Data-path Synthesis System", in Proc. of *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16 11, pp 1260-1277, Nov. 1997.
5. M. Ercegovac, D. Kirovski and M. Potkonjak, "Low-power Behavioral Synthesis Optimization using Multiple Precision Arithmetic", in Proc. of 37^{th} *Design Automation Conference*, 1999.
6. M. Vasilko and D. Ait-Boudaoud, "Scheduling for Dynamically Reconfigurable FPGAs", in Proc. of *International workshop on logic and architecture synthesis*, 1995.
7. J. C. Alves and J. S. Matos, "A Simulated Annealing Approach for High-level Synthesis with Reconfigurable Functional Units", in Proc. 38^{th} *Midwest Symposium on Circuits and Systems*, 1996.
8. F. G. Wolff, M. J. Knieser, D. J. Weyer and C. A. Papachristou, "High-level Low Power FPGA Design Methodology",*IEEE National Aerospace Conference*, 2000.
9. Padmanbhan Pillai, Hai Huang and Kang G. Shin "Energy-Aware Quality of Service Adaptation", *PhD. Thesis*, University of Michigan
10. Krishnan Srinivasan, Vijay Ramamurthi and Karam S. Chatha. "A Technique for Energy versus Quality of Service Trade-off for MPEG-2 Decoder", *PhD. Thesis*, Arizona State University.
11. Clark N. Taylor and Sujit Dey. "Adaptive Image Compression for Mobile Multimedia Communication", *PhD. Thesis*, University of California.
12. Lodewijk T. Smit, Gerard J.M. Smit, Paul J.M. Havinga. "Run Time Energy Efficiency in Mobile", *PhD. Thesis*, University of Twente.
13. Emre Ozer, Andy P. Nisbet and David Gregg. "Stochastic Bit-Width Approximation using Extreme Value Theory for Customizable Processor", *PhD. Thesis*, Trinity College, Ireland.
14. Iain Edward Garden Richardson. "Video Coding for Reliable Communciations", *PhD. Thesis*, Robert Gordon University, 1999.
15. Mark Stephenson, Jonathan Babb and Saman Amarasinghe. "Bit Width Analysis with Application to Silicon Compilation", in Proc. of *Conference on Programming Language Design and Implementation*, pp108-120, 2000.
16. Mihai Budiu, Seth Copen Goldstein. "Bit-value Inference: Detecting and Exploiting Narrow Bit-width Computations", in Proc. of 6^{th} *International Euro-Par Confrence*, August 2000.
17. Scott Mahlke, Rajiv Ravindran, Michael Schlansker, Robert Schreiber, Timothy Sherwood. "Bit-width Sensetive Code Generation in A Custom Embedded Accelerator Design System", *Technical Report*, Hewlett-Packard Laboratories, Palo Alto, CA.
18. Paul J. M. Havinga,Lodewijk T. Smit,Gerard J. M. Smit, Martinus Bos, Paul M. Heysters. "Energy Management for Dynamically Reconfigurable Heterogeneous Mobile Systems" in Proc. of 10^{th} *Heterogeneous Computing Workshop HCW 2001*- Volume 2,pp. 20086.1, 2001.
19. M. Xu and F. J. Kurdahi, "Layout-driven High-level Synthesis for FPGA based Architectures", in Proc. *IEEE Symposium on FPGAs for Custom Computing Machines*, 1998.
20. A. A. Duncan, D. C. Hendry and P. Gray, "An Overview of the COBRA-ABS High-level Synthesis System for Multi-FPGA Systems", in Proc. of *IEEE Symposium on FPGAs for Custom Computing Machines*, 1998.

21. G. A. Constantinides, P. Y. K. Cheung, and W. Luk, *"The Multiple Wordlength Paradigm"*, in Proc. of *IEEE Symposium on Field Programmable Custom Computing Machines*, 2001.
22. Altaf Abdual Gaffer, Oskar Mencer, Wayne Luk, Peter Y.K. Cheung and Nabeer Shirazi. "Floating-point Bit-width Analysis via Automatic Differentiation", in Proc. *IEEE International Conference on Field-Programming Technology*, Hong-Kong, 2002.
23. Shivajit Mohapatra, Radu Cornea, Nikil Dutt, Alex Nicolau, Nalini Venkatasubramanian "Integrated Power Management for Video Streaming to Mobile Handheld Devices", in Proc. of 11^{th} *ACM International Conference on Multimedia*, pp 582-591, 2003.