Timing Variation-Aware Custom Instruction Extension Technique

Mehdi Kamal[1], Ali Afzali-Kusha[1], Massoud Pedram[2]
[1]School of Electrical and Computer Engineering, University of Tehran
[2]EE Department, University of Southern California
m.kamal@ece.ut.ac.ir, afzali@ut.ac.ir, pedram@usc.edu

***Abstract*—In this paper, we propose a technique for custom instruction (CI) extension considering process variations. It bridges the gap between the high level custom instruction extension and chip fabrication in nanotechnologies. In the proposed method, instead of using the conventional static timing analysis (STA), statistical static timing analysis (SSTA) which in turn results in a probabilistic approach to identifying and selecting different parts of the CI extension is utilized. More precisely, we use the delay Probability Density Function (PDF) of the CIs in identification and selection phases of the CI extension. In the identification phase, the delay of each CI is modeled by PDF whereas the performance yield is added as a constraint. Additionally, in the selection phase, the merit function of the conventional approaches is modified to increase the performance gain of the selected CIs at the price of slightly sacrificing the design yield. Also, to make the approach computationally more efficient, we propose a method for reducing the modeling time of the PDF of the CIs by reducing the number of candidate CIs before extracting the PDF.**

Keywords-component; ASIP, Custom Instruction, Process Variation, PDF

I. INTRODUCTION

The use of embedded processors in a variety of platforms such as cell phones, health monitoring devices, automotive applications, and many others is increasing. Similar to many other digital systems, the computational speed and power consumption are two critical design parameters [1]. A solution for these issues is the application specific instruction set processor (ASIP) methodology which can improve speed and power consumption of the GPP technique [2][3]. In the ASIP approach, the instruction set of a GPP is extended through ASIC design based on the features of the specific application. The augmented instructions are determined such that the desired speed, power, and cost requirements are fulfilled. The main idea behind using ASIP is to run the hotspot parts of an application on the custom instructions (CIs) and the other parts of the application on the ALU of the processor.

The ASIP design methodology starts by extracting the data flow graph (DFG) of the application [1][4]. Next, in identification phase, all the subgraphs that meet the constraints of the parallel hardware are enumerated as CIs. The I/O ports number and propagation delays of the subgraphs are two common constraints in the CI enumeration. Finally, between the candidates, the best CIs based on their merit value will be selected. For each CI, the merit value shows the quantity of the parameters that the ASIP intends to improve. Normally, the main parameter is speedup, which shows how much performance is obtained by using CIs in the ASIP [1][2].

In the conventional ASIP approach, the worst-case delays of the primitives (e.g., AND, ADD, SHIFT, etc.) are used as the reference to extract the latency of the CIs. In sub-100nm nanotechnologies, however, complexities in the manufacturing of the transistors with small sizes have caused significant variations in nominal transistor parameters (such as threshold voltage and effective channel length), which in turn has led to uncertainties in the performance and power consumption of the circuits [5]. As the process variation impact increases, the gap between the high level design and fabrication may increase if proper statistical techniques are not invoked. Designing based on the process corners to meet the latency constraint is inadequate [5]. The design flow of the embedded system also is not an exception and should shift from deterministic to probabilistic approaches. There are many published results on modeling and mitigating the process variability at the device and circuit levels of design abstraction. There are also some work in high level synthesis (HLS) where techniques have been proposed to improve the performance and reduce the hardware cost considering process variations (see, e.g., [5]-[7]). To the best of our knowledge, there is no report of considering the process variability in the CI extension in the ASIP design.

In this paper, we propose to modify the design flow of ASIP by adding the statistical approach in identification and selection phases of the CI extension. In the identification phase, the delay of each CI is modeled by a Probability Density Function (PDF), and performance yield [5] is added to the constraints set to guarantee the yield of the design. Also, the merit function which is used in the CI selection phase is modified based on the probability of increasing the speedup and improving the design yield using the CIs.

The remainder of this paper is organized as follow. The problem statement is described in Section II with the overall approach is given in Section III. The experimental setup and the results are described in Section IV. Finally, the paper is concluded in Section V.

II. PROBLEM STATEMENT

In the selection phase of the ISA extension, the best CIs whose speedup is more than the other ones will be selected. Also, the selected CIs typically have to meet other predefined constraints such as the number of I/O ports and layout area [1][8][9]. While the I/O ports are usually considered in the identification phase, the area constraint is considered in the selection phase. To consider process variations, we specify a performance yield which shows the probability of the

978-3-9810801-7-9/DATE11/©2011 EDAA

manufactured chip meet the clock period constraint (CPC). This constraint means that the performance yield of the i^{th} selected CI (PY_i) given by

$$PY_i = CDF(CI_i) = \int_0^{CPC} PDF(CI_i) \quad (2)$$

must be large than a predefined value. The $PDF(CI_i)$ and $CDF(CI_i)$ show the Probability and Cumulative Density Function of the i^{th} CI, respectively. To simplify the analysis, we assume that the PDF of the CIs have a normal distribution

Therefore, the problem is formulized as

$$Maximize \sum_{i=1}^{\#Selected\ CIs} Speedup_i \quad (3)$$

$$\forall i: C_i \in Selected_{CI}, PY_i > PY_{const} \quad (4)$$

where the PY_{Const} is the predefined value for the performance yield constraint, and PY_i is the performance yield of the i[th] CI Also, the $Selected_{CI}$ is the set of the selected CIs.

III. Proposed Approach

The proposed probabilistic consists of the identification and selection phases of CIs. The identification part obtains the DFGs of the application and generates all the CIs that meet the predefined constraints. The output of this phase is a set of identified CIs. In the deterministic approach, the critical path delay of the CIs is compared with CPC where CIs with longer critical path are removed from the CIs set. In the probabilistic approach, based on the performance yield, some CIs are eliminated. For pruning the CI set, the PDFs of the CIs should be modeled.

There are two methods for extracting the PDFs which are Monte Carlo and SSTA [10]. While the accuracy of the Monte Carlo is better than SSTA, its runtime is much higher. In the identification phase, there are a large number of enumerated CIs, and hence, finding the PDF of all the CIs is not feasible. To reduce the runtime, one can use less accurate techniques to guess the CIs that are not able to meet the performance yield constraint without finding their PDFs. Here, we propose an expression for this when the delay variation can be modeled by a normal distribution. Note that for the cases where the delay distributions of the CIs may not be described by a normal distribution, the framework used for calculating the expression in the normal distribution case is still valid. In these cases, only the expressions which will be modified based on the distributions. For the normal distribution case, we only need the nominal and worst-case delay of the CIs. The CIs whose $\mu + k_1\sigma$ values are smaller than the CPC may be removed from the set. To determine μ and σ, we need to find the PDF for the CI which we wish to avoid in this stage. To overcome the problem, we approximate μ by the nominal value of the delay (T_{nom}). Also, we suggest using another constraint (independent of σ) instead of $\mu + k_1\sigma < CPC$. For this purpose, we need the worst-case delay which may is given by

$$Delay_{wc} = \mu + 3\sigma \quad (5)\text{-a}$$

Also, our original constraint is

$$\mu + k_1\sigma \le CPC \quad (5)\text{-b}$$

Combining (5)-a and (5)-b, one obtains

$$\sigma \ge \frac{Delay_{wc} - CPC}{3 - k_1}$$

$$\mu \le CPC - 3\left(\frac{Delay_{wc} - CPC}{3 - k_1}\right) \quad (5)\text{-c}$$

The coefficient k_1 can be extracted from a normal distribution table or calculated using

$$\text{erf}\left(\frac{k_1}{\sqrt{2}}\right) \ge 2PY_{const} - 1 \quad (6)$$

where PY_{Const} is the minimum performance yield constraint. Substituting μ with the nominal delay of the CIs one may write

$$T_{nom} \le \left(CPC - 3\left(\frac{Delay_{wc} - CPC}{3 - K_1}\right)\right) \times err \quad (7)$$

The coefficient *err* which has a value larger than one has been included to minimize the error of using T_{nom} instead of μ. This error can be determined using experimental or analytical methods. If the constraint in (7) is not satisfied, it is assured that the performance yield of the CI is smaller than the PY_{Const}. If the constraint is satisfied, then using the PDF of the CI, one should determine if the performance yield is greater than the PY_{Const}. Using (7), the set of the enumerated CIs may be pruned. Having reduced the number of CIs, the PDF and CDF of the remained CIs are calculated and the CIs with the performance yield smaller than PY_{Const} are removed. The extracted PDFs will also be used in the selection phase. The last part of the ISA extension is the CI selection phase in which the best CIs are selected based on the merit value.

A. PDF Determination

For finding the PDFs of CIs, we need the PDF of primitives such as adder, multiplier. When the primitive PDFs are available, the CI PDFs could be calculated using Monte Carlo or SSTA approaches. We developed a library that contains the PDFs of the primitives using the method explained here. First, we need the PDFs of the gates used in the design synthesis. These PDFs were extracted using their HSPICE model using its Monte Carlo analysis. In HSPICE, different types of variation sources are modeled. The delay PDFs obtained using HSPICE are put in a library. They are used to find the PDFs of complex circuits such as a 32-bits Adder. For finding the delay PDF of complex circuit, the HDL model of the circuit, which is synthesized based on the technology file, is needed. Using the model and either SSTA or Monte Carlo method the circuit PDF is determined.

B. Merit Function

Without considering the process variation, the value function is typically defined to be the number of clock cycle reduction (speedup) in the application runtime due to the CI extension. This function captures the speedup of the CI. The merit function may be expressed as

$$M_i = \#Iteration * \left(CI_i.SW - IO_i.Penalty - ceil\left(\frac{CI_i.CriticalPathDelay}{Clock\ Period}\right)\right) \quad (8)$$

where M_i is the merit value of the i^{th} CI, $\#Iteration$ is the number of times that the basic block to which the i^{th} CI belongs

is repeated, the $CI_i.SW$ shows the runtime of the i[th] CI on the base processor. The *IO.Penalty* is the number of extra accesses to the register file for reading data to/writing data from CI, and the last part of this equation is the number of clocks that the CI needs to find the result. If the I/O port number of the CI is equal to the read and write port number of the register file, $IO_{Penalty}$ is equal to 0.

By considering the process variation, we expect that the performance yield of the selected CIs will be larger than the predefined minimum performance yield and also the selected CIs reduce the application runtime more than other CIs. Also, we want the performance yield of the selected CIs are near the 100%. In the presence of process variation, we can modify the merit function such that it includes the performance yield as well. Hence, the merit function may be modified as

$$CY_i = \alpha f(PY_i) + Speedup_i \quad (9)$$

where CY_i shows the merit function of the i[th] CI when the performance yield is also considered, α is a coefficient that determines the weight of the performance yield in the merit function of the CI, and $f(PY_i)$ is a function obtained by a linear mapping of PY_i, which maps the smallest PY (equal to PY_{Const}) to zero and the largest PY (100%) to one, given by

$$f(PY_i) = \frac{PY_i}{1 - PY_{Const}} - \frac{PY_{Const}}{1 - PY_{Const}} \quad (10)$$

The $Speedup_i$ is equal to M_i given by (8).

IV. Evaluation Results

A. Experimental Setup

To assess the performance of the proposed technique, we used applications from *mibench* [11], *PacketBench* [12], and SNR-RT benchmark suits [13]. The *IPSec* and *MD5* were selected from *PacketBench*, *lms* from *SNR-RT*, and *G721encoder*, *G721decoder*, and *bitcounter* from *mibench* suit. The identification phase was performed based on the work described in [1]. We modified the method such that the clock period constraint could be included. Also, to implement the selection phase we used a greedy approach [2].

In this work, we considered the timing variation only due to the threshold voltage variation. To extract the threshold variability induced by process variations, the HSPICE model of the 45nm PDK technology gates [14] was simulated assuming 10% of variation for the threshold voltages of PMOS (V_{thp}) and NMOS (V_{thn}) For this analysis, we used Monte Carlo analysis implemented in HSPICE. To extract the PDF of the primitives, we developed a tool which takes the gate PDFs and the gate level implementation of the primitives as the input and use Monte Carlo method to find the primitive PDFs.

B. Experimental Results

First, the impact of the process variation in the identification phase was studied. As mentioned before, by estimating the performance yield of a CI before finding its PDF, the time for extracting the PDF of the CIs is reduced. Figure 1 shows the percentage of the enumerated CIs which are removed from the CIs set before finding the PDFs. The results are for three benchmarks under two different performance yield constraints and four different clock period constraints. As observed from Figure 1, by increasing the CPC, the number of removed CIs is decreased and more CIs may satisfy the timing constraint. For example, in *lms* the removing percentage is decreased from 97% to 36%. Since in *lms*, the average delay of the identified CIs is 4.2ns compared to 3.6ns in *G721encode*, the number of the removed CIs in *lms* is more than that of *G721encode*. The removed items in *lms* are about 36% while this value for *G721encode* is about 4% for a CPC of 5ns.

To show the accuracy of (7), we extract the number of CIs that are kept for the PDF calculation stage phase but their performance yield does not meet the PY_{Const}. Figure 2 depicts the percentage of number of CIs which are passed to the selection phase (after their PDF and performance yield were calculated) to the number of CIs which were initially predicted to remain for the selection phase. As the charts show, although the proposed prediction model lowers the number of CIs that their PDFs should be calculated, the number of CIs that are eliminated when the PDFs are obtained increases as the CPC decreases, especially when the CPC has a value smaller than the delay average of the application. For example, the number of removed CIs in *IP-Sec* is not decreased as much as the other benchmarks when the CPC is decreased to 2ns. Increasing the number of removed CIs in the PDF check stage originates from decreasing the difference between the nominal and worst-case delay. Thereby, the *err* value must be reduced for the CIs with smaller delays.

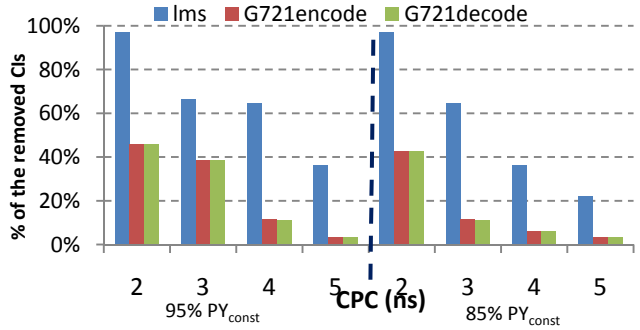

Figure 1. Percentage of the removed CIs in the identifcation phase due to the performance yield befor extarcting the CI PDFs.

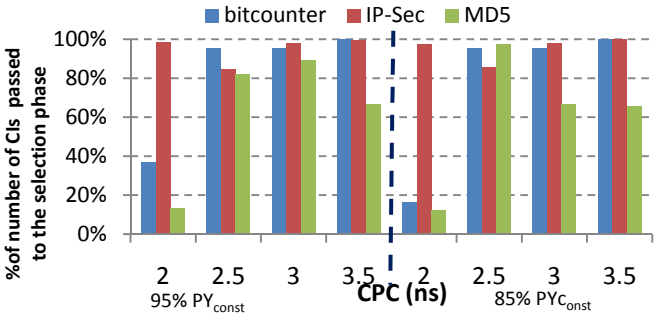

Figure 2. The percentage of number of CIs which are passed to the selection phase (after their PDF and performance yield were calculated) to the number of CIs which were initially predicted to remain for the selection phase.

Because of the process variation, a number of CIs are removed in the identification phase. This has some impacts on the overall performance gain of the ISA extension. On the other hand, because of the probabilistic nature of performance yield, some of the CIs that their worst-case delays do not meet the CPC, have a chance to remain and be selected as a CI. Hence, we expect that by reducing PY_{Const} the performance

improvement of the selected CIs increases. In the other word, this behavior is due to the fact that while the CIs with more nodes usually have more speedup, the probability that their delays are smaller than CPC is low. Figure 3 depicts the performance improvement of the benchmarks for different performance yields. In this figure the worst-case delay is based on equation (5)-a. The CPC for *lms*, *G721encode*, and *G21decode* was 4ns and for other benchmarks was 2.5ns. Also, the α value in merit function was 0.5. As the chart shows, by decreasing the performance yield the performance gain increases. In *IP-Sec*, however, the performance gains for all PY_{Const} from 95% to 80% are constant. This originates from the small delay values of the selected CIs. When considering a PY_{Const} of 95%, the performance gain is improved about 5% in comparison with the worst-case approach. Also, note that in *lms* and *bitcounter* benchmarks changing the PY_{Const} from 85% to 80% does not change the performance gain. Finally, in *G721encoder* and *G721decoder*, the performance gain between the worst-case approach and statistical approach with 95% PY_{Const} are about the same with a difference of about 1.5%.

Lastly, to investigate the impact of the α coefficient on the performance gain and average of the performance yield of the selected CIs in *lms* benchmark is presented in Figure 4. The average of the performance yield is decreased for a given PY_{Const} while it becomes larger as α increases. Also, the performance gain decreases as α increases. This shows that the performance yield average (performance gain) is directly (inversely) proportional to α coefficient.

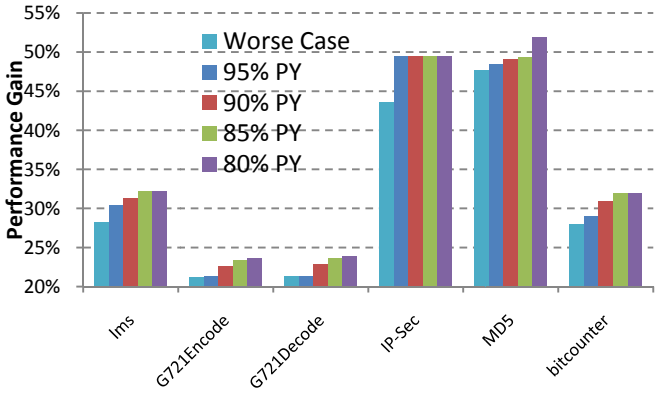

Figure 3. The performance gain of ISA extension under different PY_{Const}.

V. CONCLUSION

In this work, we proposed a probabilistic approach for the design flow of the ISA extension in the presence of the process variation. The technique, which included both identification and selection phase, modeled the timing constraint by the performance yield as a probabilistic timing approach. This converted the conventional deterministic approach to a probabilistic approach where instead of using a constant delay value, the timing variation of the primitives were used to model the delays of the custom instructions. Also, to improve the runtime speed of the CI identification, an approximate method was proposed. In the identification phase, the delay and performance yield were used as parameters for pruning the CIs set while the final CIs were chosen in the selection phase based on the merit function. To study the effect of the selected CIs on the speedup of the processor, the proposed technique was applied to different applications. The study included the relation between the speedup and performance yield as well as the comparison between the worst-case (deterministic) and the performance yield (probabilistic) approaches.

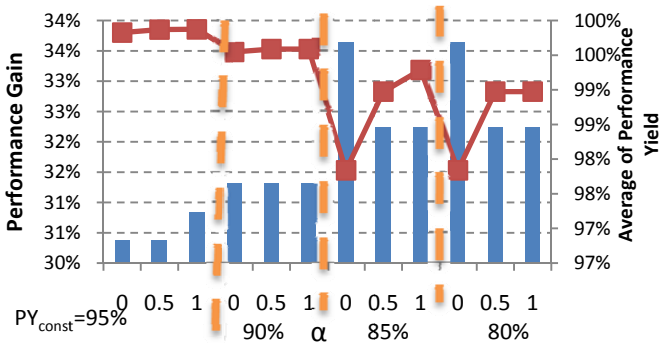

Figure 4. The impacts of the α coefficient on performance gain and average of the performance yield in *lms* benchmarks. The linear chart shows the average of the performance yield.

REFERENCES

[1] L. Pozzi, K. Atasu, and P. Ienne,"Exact and Approximate Algorithms for the Extension of Embedded Processor Instruction Sets", IEEE Trans. CAD, Vol. 25, No. 7, July 2006.

[2] P. Bozini, and L. Pozzi,"Recurrence-Aware Instruction Set Selection for Extensible Embedded Processors," in IEEE TVLSI, Vol. 16, pp. 1259-1267, 2008.

[3] K. Atasu, L. Pozzi, and P. Ienne, "Automatic application specific instruction-set extensions under microarchitectural constraints," International Journal of Parallel Programming, Vol. 31, pp. 411-428, 2003.

[4] N.T. Clark, H. Zhong, and S. Mahlke ,"Automated Custom Instruction Generation for Domain-Specific Processor Acceleration," in IEEE Trans. on Computers, Vol. 54, pp. 1258-1270, 2005.

[5] Y. Xie, and Y. Chen, "Statistical High-Level Synthesis under Process Variability, " in IEEE Transaction Design and Test Computers, Vol. 26, 2009, pp.78-87.

[6] F. Wang, Y. Xie, and A. Takach,"Variation-Aware Resource Sharing and Binding in Behavioral Synthesis," in Proceedings of the 2009 Asia and South Pacific Design Automation Conference, 2009, pp. 79-84.

[7] F. Wang, G. Sun, and Y. Xie, "A Variation Aware High Level Synthesis Framework," in Proceedings of the conference on Design, automation and test in Europe, 2008, pp. 1063-1068.

[8] K. Atasu, G. Dundar, and C. Ozturan."An integer linear programming approach for identifying instruction-set extensions," in Proc. of *CODES+ISSS*, 2005, pp. 172-177.

[9] N. Clark, A. Hormati, S. Mahlke, and S. Yehia,"Scalable subgraph mapping for acyclic computation accelerators," in Proc. of CASES, 2006, pp. 147-157.

[10] V. Veetil, Y. Chang, D. Sylvester, and D. Blaauw, "Efficient smart monte carlo based SSTA on graphics processing units with improved resource utilization," in Proceedings of the 47th Design Automation Conference, 2010, pp. 793-798.

[11] M. R. Guthaus et al. ,"MiBench: A free, commercially representative embedded benchmark suite," in *Proc. of Int. workshop on workload characterization*, 2001, pp. 3-14.

[12] R. Ramaswamy and T. Wolf, "PacketBench: A tool for workload characterization of network processing," *in Proc. of IEEE International Workshop on Workload Characterization*, October 2003, pp. 42-50.

[13] SNU-RT Real Time Benchmarks.[Online]. Available: http://archi.snu.ac.kr/realtime/benchmark/.

[14] FreePDK, AFree OpenAccess 45nm PDK and Cell Library for university, http:// www.eda.ncsu.edu