To appear in ACM Journal on Emerging Technologies in Computing Systems, 2013.

Impact of Process-Variations on Speedup and Maximum Achievable Frequency of Extensible Processors

MEHDI KAMAL, ALI AFZALI-KUSHA, SAEED SAFARI
UNIVERSITY OF TEHRAN
AND
MASSOUD PEDRAM
UNIVERSITY OF SOUTHERN CALIFORNIA

In this paper, we investigate the impact of process variations on the speedup and maximum frequency of the extended ISA processor. First, without considering the process variations, a custom functional unit (CFU) is designed based on nominal timing parameters and then the timing variations of critical paths of the extensible processor including the baseline processor and the CFU are investigated by considering both systematic and random variations. Next, the maximum frequency of the extensible processor and the speed enhancement factor of the extended ISA for different benchmarks are investigated. Results show that the timing variation could reduce the speedup of the extensible processor. However, this reduction is highly dependent on the baseline processor and the CFU structures. Additionally, the impact of process variations in the case of worst-case design approach is studied. Results show that the speedup of the extensible processor is reduced more than the case when custom instructions (CIs) are selected without considering the process variations. To study the impact of each variation type, speedup variations due to the random and systematic variations are investigated separately. The study reveals that the random variation has a similar effect on the CFU and the baseline processor, while the impact of the systematic variation on the baseline processor is greater than the CFU.

Categories and Subject Descriptors: J.6 [COMPUTER-AIDED ENGINEERING]: Computer-aided design (CAD); [**SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEMS]:** Microprocessor/microcomputer applications; **B.2.2 [Performance Analysis and Design Aids]:** Worst-case analysis**.**

Additional Key Words and Phrases: Application-specific instruction-set processors, Custom instruction, Process Variation, Speedup, Clock Frequency.

1. INTRODUCTION

The use of embedded processors in a wide variety of platforms such as cell phones, health monitoring devices, and automotive applications is increasing. Similar to many other digital systems, the computational speed and power consumption are two critical design parameters [1]. One of the design approaches for these processors is application specific integrated circuit (ASIC) technique where both high speed and low power dissipation may be achieved with a penalty of higher design cost and lower flexibility. Another solution which has higher flexibility and lower design cost is the general purpose processor (GPP) where the speed is lower while the power consumption is higher than those of the ASIC approach. The application specific instruction set processor (ASIP) methodology is the third approach, which can improve the 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 using custom instructions (CIs) and the other parts of the application on the ALU of the processor. The CIs (which are executed using a hardware block in parallel with ALU of the GPP) improve processor speed (performance and speed are used interchangeably in this paper) by increasing the instruction level parallelism, and reducing the register file accesses. Similarly, by decreasing the accesses to the cache and register file, the CIs lower the power consumption.

Authors' addresses: Mehdi Kamal, Ali Afzali-Kusha, and Saeed Safari School of Electrical and Computer Engineering, University of Tehran, Tehran, IRAN, E-mail: {mehdikamal, afzali, saeed}@ut.ac.ir; Massoud Pedram, Electrical Engineering Department, 3740 McClintock Ave., Los Angeles CA 90089-2562, E-mail: { pedram@usc.edu}.

The ASIP design methodology starts by extracting the data flow graph (DFG) of the application [1][4]. Next, all subgraphs of the DFG that meet the constraints of the parallel hardware are enumerated as CIs. The I/O port count 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 function produces values of the parameters that the ASIP intends to improve. Normally, the main parameter is the clock cycle count saving, 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 the threshold voltage and effective channel length of transistors), which in turn has led to uncertainties in the performance and power consumption of the circuits [5]. There are two types of variations: (i) within die or intra die, and (ii) die to die or inter die variations. 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 is also some work in high level synthesis (HLS), where techniques have been proposed to improve the performance and reduce the hardware cost while considering process variations (see, e.g., [5]-[7]). The impact of the process variations in the pipelined processors have been considered in several works and some techniques have been presented to mitigate the impact (see, e.g., [8][9][10]). As has been stated in [8][10], due to the existence of many critical paths in the execution unit of in-order and out-of-order processors, it is crucial to consider process variations in their design. In the case of ASIP's, the designer should improve the speedup of the processor running an application by defining new instructions which are created by combining the basic primitives. Adding new instructions may increase the impact of the process variations on the extensible processor. In the area of ASIP design, a yield-aware ASIP design methodology to improve the performance yield of the extracted CIs has been proposed in [11].

In this paper, the impact of the process variations on the design of extensible processor is studied. We investigate timing variations of the critical paths of the extended ISA in the presence of the process variations. Based on this investigation, the speedup efficacies of extensible processors are assessed. Also, we explore process variations impact on the maximum frequency of the extensible processors. The remainder of this paper is organized as follows. Section 2 briefly reviews related works while the ISA extension methodology is described in Section 3. Process variations and their modeling are described in Section 4. Section 5 discusses the impact of the process variation on the extended ISA. The results are discussed in Section 6. Finally, the paper is concluded in Section 7.

2. RELATED WORK

In this section, we categorize some of the prior work in two groups of works on mitigating the process variation effects on processors and the research efforts about the design of the ISA extension.

A. Processors

Two compile time techniques for handling non-uniform latency of different integer functional units (IFUs) in VLIW processors were proposed in [12]. In the proposed approach, all the IFUs with high latency due to the process variation are turned off and the processor only uses the low latency IFUs. Whenever, the processor needs more IFUs, some of the highly latency IFUs are also turned on. Due to this policy the authors called this technique "on-demand turn-on". In [8], an architectural technique (Trifecta) was proposed to mitigate the timing variation in critical pipeline stages. In the proposed technique, the architecture completes its execution in one clock cycle when only the subcritical path operations are needed to generate the result. On the other hand, in the operations which involve the critical paths, the architecture makes it possible to complete the operations in two cycles.

In [13], a technique to tackle the performance reduction of out-of-order processors in the presence of the process variation was presented. In this technique, the instructions, based on their dependability on each other, were categorized in two groups. The instructions whose results are not (are) needed for other instructions are executed on a long-latency (short-latency) unit. In [14], two fine-grained post-fabrication techniques are proposed to mitigate the timing parameters fluctuation. The voltage interpolation and variable latency are the two methods proposed in this paper. In [9], the impact of the process variation on the propagation delay of the pipeline stages is investigated. In this work, to decrease the impact of the timing fluctuation on the performance of the pipelined processors, an architectural-level technique based on the cycle time stealing is proposed. In addition to the time borrowing technique discussed in [9], a method which controls the clock speed in multi-issue processors is suggested in [15]. The method categorized the runtime of the program in two different phases, Low-ILP (Instruction Level Parallelism) phases, and High-ILP phases. In each phase, the impact of the process variation is lowered by changing the clock speed.

B. ISA Extension

Now, we briefly discuss, the CI identification techniques. An algorithm that explores the search space exhaustively for a single feasible cut meeting the I/O constraint was presented in [3]. The algorithm which prunes the search space runs in linear time for most Directed Acyclic Graphs (DAGs) having practical numbers of nodes. In [1], the optimization problem was solved by using the genetic algorithm. In [16], the problem was formulated and solved by an Integer Linear Programming (ILP) approach. The works in [17][18][19] proposed methods to enumerating all sub-graphs of the DFG. In these methods, only the convexities of the subgraphs were important while the I/O constraint was not defined. The technique proposed in [20] was able to identify CIs with or without I/O constraint. In all of the proposed identification methods, the goal has been to increase the cycle saving of the selected CIs while satisfying the I/O constraint and convexity condition of the selected CI.

In addition to the identification phase, the other main phase of the ISA extension is selection phase. In [2], exact and approximate algorithms for solving the coverage and recurrence problems of candidate extensions are presented. The authors describe an optimal search technique that uses a branch-and-bound algorithm in conjunction with a greedy method. The objective of the technique was to improve the performance without considering the area budget. The technique proposed in [21] partitions the problem into several sub-problems and presents an optimal branch-and-bound algorithm to solve it. This algorithm uses subgraph size (number of nodes in the subgraph) as its merit function and relies on sorting the candidates in decreasing order of size. An automatic framework for designing a customized processor has been suggested in [4]. The proposed method handles both CI identification and CI selection. In the selection phase, the cycle savings and area budgets of the CIs were used as the merit function. In all of these methods, the merit function value corresponds to the cycle saving of the CI or combination of the cycle saving and area budget.

The aforementioned methods suggest techniques for improving the speed and output quality of the identification and selection phases of the instruction set architecture (ISA) extension. It should be noted that in these methods deterministic values of parameters are used in estimating the CI delays. As mentioned before, in nanoscale era, due to process variations, the speedup and maximum frequency of the extended ISA processor are not deterministic parameters. Consequently, the statistical variations of the parameters in the design of the ISA extension should be considered. In this work, we investigate the impact of the process variation on the speedup and maximum frequency of the extended ISA processor. The study performed in this work shows the importance of considering the statistical design versus the worst-case (deterministic) design. It includes the effects of both systematic and random variations, on the CFU and existing design of baseline processor. Also, it shows which part is the determining component (CFU or baseline processor) in the presence of the process variation. As will be concluded from this investigation, the statistical design of extensible processor should be used in coming technologies where the process variation becomes even more critical phenomenon.

3. ISA EXTENSION METHODOLOGY

The overall flow of the ISA extension is shown in Figure 1. The ISA extension starts by extracting the data flow graph (DFG) of a target application. The DFG captures the instruction flow of the application. Custom (extended) instructions are sub-graphs of this DFG that meet some predefined constraints, such as convexity, number of I/O ports, and propagation delay.

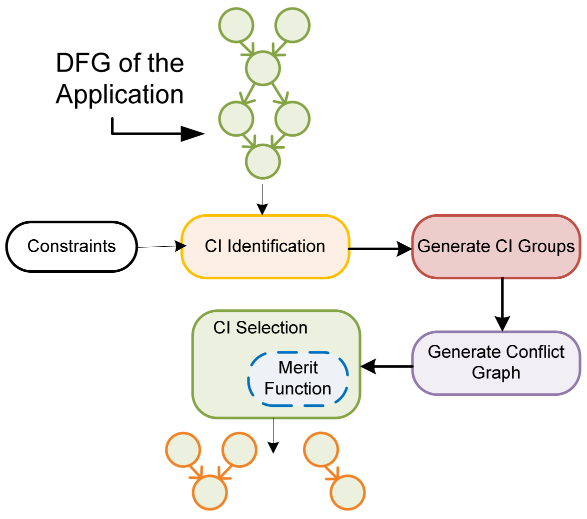


Figure 1 ISA extension flow

Note that a subgraph is architecturally feasible if its inputs are available at the time of executing that operation which is only possible if the subgraph is convex. A subgraph S is called convex when there does not exist any path from a node u ∈ S to another node v ∈ S while the node w ∉ S involves in the path. Examples of a DFG and two subgraphs are shown in Figure 2. The subgraph ABD is nonconvex due to the path A→C→D (which the node C does not exist in the subgraph) while the subgraph CD is convex. Satisfying the convexity guarantees that a CI may be executed independent of any other operations.

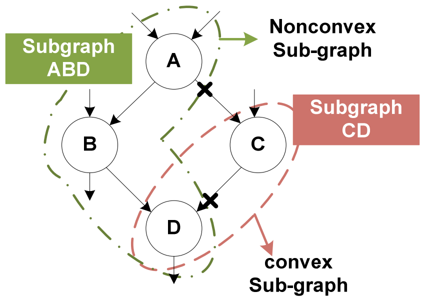


Figure 2 Examples of a DFG and two subgraphs.

Also, the number of inputs and outputs of the sub-graphs should be equal or less than the read and write ports of the baseline processor's register file. If the I/O number is larger than the register-file R/W ports, we should use a pipelining method to resolve this problem, which decreases the performance of the extensible processor. Finally, the CI delay is defined based on the clock period and the cycle latency of the execution unit of the processor. If the execution stage of the processor is single cycle, the sub-graphs whose propagation delay is less than one clock period cycle are acceptable. On the other hand, if the execution stage has multi-cycle latency, the CI delay could be more than one clock period. The enumeration of the feasible sub-graphs is done in a CI identification phase. Among the identified sub-graphs (CIs), there could be similarity based on functional and structural isomorphism. Similar CIs, which can be executed on a CFU, are identified as CI groups[2][21]. This results in less area overhead and thus better area utilization.

If a DFG node exists in two CIs, these two CIs are considered to have overlap with each other. Because no node in the DFG should be executed by more than one CI[2] , all CIs that have an overlap with a selected CI are removed from the list of the identified CIs in the first part of a *CI selection phase*. Based on the overlap between the CIs, a conflict graph is constructed. Nodes in the conflict graph are CIs and an edge between two nodes shows that these CIs have an overlap. Notice that removing CIs that conflict with the selected CI changes memberships of the CI groups, and consequently, results in changes in the overall cycle saving values of the CI groups.

In the last part of the CI selection phase, the best CI groups are selected from among all identified CIs. There are two general approaches for the CI selection i.e., branch-and-bound, and greedy [2][21][22][23]. The former is an exact method that tries to find the best CIs set by searching all the search space. Unfortunately, it is generally infeasible for real applications. The greedy approach is much more efficient, yet it does not guarantee to find the optimal set of CI's. In this paper we use the greedy approach to select the CIs.

In the selection phase, the CI candidates are selected based on their merit value. Normally, cycle saving is the main objective in the ISA extension. The cycle saving of the i^{th} CI (denoted as CI_i) is computed as

$$CS_i = \#Iterations \times \left(\#CI_i.SW - CI_i.IO\_Penalty - ceil\left(\frac{CI_i.CriticalPathDelay}{ClockPeriod}\right)\right) \quad (1)$$

where *#Iterations* denotes the execution frequency (count) of the basic block to which CI_i belongs, $\#CI_i.SW$ is the number of cycles that the CI needs to run on the baseline processor, and $CI_i.IO\_Penalty$ denotes the number of extra accesses to the register file for reading/writing data (when the number of CI I/O ports is more than the number of register file read/write ports). The last term calculates the number of cycles needed to execute CI_i on the CFU (CIs may be multi-cycle.) In this ratio, CI_i.*CriticalPathDelay* denotes the propagation delay of the CI critical path and *ClockPeriod* is the target clock period for the extensible processor.

Whether the design flow includes the selection and identification phases together or separate, when the area constraint is defined as a constraint in the ISA extension, we can formulate the instruction selection problem as

$$Maximize \sum_{i=1}^{|Selected\ CI\ Groups|} CS_{CI\ Group_i} \qquad (2)$$

while

$$\sum_{i=1}^{|Selected\ CI\ Groups|} Area_{CI\ Group_i} < Area_{Constarint} \qquad (3)$$

Equations (2) and (3) are the objective function and its constraint used in the CI selection process which is an optimization problem. To select CIs, one should use either these equations in an optimal (exact) optimization algorithm or a merit function in a non-optimal optimization algorithm (*e.g.*, greedy). In the non-optimal selection, when we wish to implement the CFU while considering the area overhead of selected CIs, the merit function, for example, may be specified as *Cycle Saving*/*Area* (see, *e.g.*, [24]). In this case, the merit function of the j^{th} CI group is modified as

$$Merit_j = \frac{\sum_{i=1}^{|CI\ group_j|} CS_i}{Area_j} \qquad (4)$$

where *Area* denotes the area of the j^{th} CI group. It is obvious that in the cases where the area budget constraint is not considered, the merit function may be defined as only the cycle saving. Hence, in this case, the merit function of the j^{th} CI group is formulated as

$$Merit_j = \sum_{i=1}^{|CI\ group_j|} CS_i \qquad (5)$$

Finally, note that the ISA extension is based on CI groups; however, for the sake of brevity, we shall use the term *CI* instead of the *CI group* henceforth.

4. PROCESS VARIATION MODELING

In this paper we will consider variations of the transistor effective channel length (L) and threshold voltage (V_{th}) as the two main sources of process variability in CMOS VLSI circuits (see, e.g., [8]). The delay of a path, in the presence of the process variations may be expressed as

$$D_{Path} = D_0 + \sum \Delta D_L + \sum \Delta D_{V_{th}} \qquad (6)$$

where the D_0 is the nominal delay of the path, and ΔD_L and $\Delta D_{V_{th}}$ are the delay variations caused by the channel length and threshold voltage fluctuations of the logic gates in the path, respectively. For both of the latter parameters, we use the first order approximation proposed in [10]

$$\Delta D_L = \alpha \times \Delta L, \qquad (7)$$

$$\Delta D_{V_{th}} = \beta \times \Delta V_{th}, \qquad (8)$$

$$L = L_0 + \Delta L = L_0 + \Delta L^{sys} + \Delta L^{rnd}$$

$$V_{th} = V_{th0} + \Delta V_{th} = V_{th0} + \Delta V_{th}{}^{rnd}$$

where α and β are coefficients extracted from curve fitting, ΔL denotes the channel length variation which consists of random and systematic variations (ΔL^{rnd} and ΔL^{sys}, respectively), ΔV_{th} is the threshold voltage variation which is due to random variation ($\Delta V_{th}{}^{rnd}$), L_0 denotes the nominal channel length, and V_{th0} is the nominal threshold voltage. As an example, Figure 3 shows the delay variation of a gate with FO4 versus the channel length variation in a 45nm technology [25]. Results, which were obtained by the HSPICE simulations, reveal that the characteristic may be modeled fairly accurately as a line.

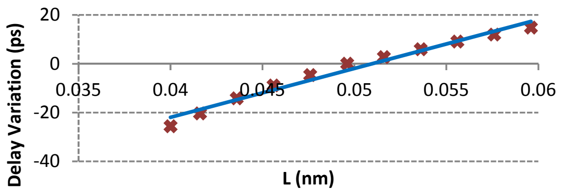


Figure 3 Impact of the *L* variation on the FO4 delay

The channel length fluctuation is modeled based on the systematic and random variations. We model the systematic variation of the channel length using multi-level quad-tree spatial correlation approach [26]. There are other methods such as spherical spatial correlation structure as was discussed in[27]. For this variation, which we consider a normal distribution with zero mean, the places of the gates in the critical paths are needed. Note that V_{th} is a function of L and channel doping density. The systematic variation of the V_{th} is due to the L variation while its random variation is a function of random dopant fluctuation (RDF). Since (7), implicitly includes the effect of the L variation on the V_{th}, similar to [10], the systematic variation of the V_{th} is not explicitly included in (8).

We assume that ΔL^{rnd} and $\Delta V_{th}{}^{rnd}$ are normally distributed with zero mean. In the case of random variations, the variation for each transistor is independent of those of the others. Since components of the random and systematic variations of L are normally distributed and independent of each other, the total variation of the L is normal with zero mean and standard deviation of

$$\sigma = \sqrt{\sigma_{sys}^2 + \sigma_{rnd}^2} \qquad (9)$$

Due to the delay variation of the design paths given by (6), the longest path delay of a given design may be represented by a delay probability distribution function (delay PDF). Hence, the maximum clock frequency of the design must be calculated based on the delay PDF. To model the maximum clock frequency, we can use the technique proposed in [28], which relies on identification and analysis of a set of timing critical paths in the circuit. The timing critical paths are those paths whose delay variations may violate a user-specified maximum propagation delay constraint. In a system with a set of timing critical paths, the delay PDF of the system is modeled by [28]

$$f_{\Delta T} = f_{D2D-T_{cp,nom}} * N_{cp} f_{WID-T_{cp,nom}}\left(t - T_{cp,nom}\right)\left(F_{WID-T_{cp,nom}}\left(t - T_{cp,nom}\right)\right)^{N_{cp}-1} \qquad (10)$$

where f_X denotes the normal PDF of the variations (X: D2D and WID stand for "Die to Die" and "With-In Die", respectively). For this distribution, the mean is zero. The corresponding Cumulative Distribution Function (CDF) for this distribution is denoted by F. Also, N_{cp} is the number of critical paths, $T_{cp,nom}$ is the nominal delay of the longest path delay, and * denotes the convolution operator. The variation is calculated by averaging the ratio of the standard deviation to the mean of all the critical paths. For more details on the calculation of (10), one may refer to [28]. Note that, in the presence of the process variation, the delay of a critical path may become larger than the predefined maximum propagation delay constraint. Therefore, a path is critical when its *timing yield* is less than 1 [28].

To model the spatial correlation needed for the calculation of ΔL^{sys} of CIs, we use the method proposed in [26]. In this method, the die area is divided into a multi-level quad-tree structure. Figure 4 shows the chip area partitioning in three levels. At each level of the partitioning hierarchy, the die area is divided into 4^{LN} squares where LN is the level number. Also, for each square (r) in each level (LN), a random variation is associated with the square ($\Delta L_{LN,r}$). Hence, the intra die variation of a gate is the summation of variations of the hierarchical squares to which the gate belongs. For example, if we assume gate A belongs to square 5 of level 2, the length variation of this gate, denoted by $\Delta L_{Gate\ A}$, will be

$$\Delta L_{Gate\ A} = \Delta L_{2,5} + \Delta L_{1,0} + \Delta L_{0,0} \qquad (11)$$

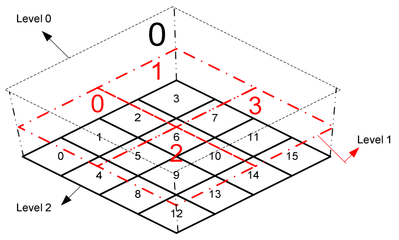


Figure 4 Multi-level quad-tree spatial correlation model

The total within die variation must be divided among the levels. The value considered for each level, is used to determine the variations of the squares in that level. Also, based on the variations defined for each level, one can control the amount of spatial correlation between two gates that are located in different squares. The smallest squares are the highly correlated areas with spatial correlation of 1. Additionally, because level 0 is shared by all gates, its variation is applied to all gates, and hence, may be considered as the inter-die variation. Hence, by decreasing the index of the levels, the spatial correlation of the corresponding level is decreased. In this paper, the spatial correlation of each level is obtained from

$$CS_i = \frac{1}{2^{MaxIndex-i}} \qquad (12)$$

where the CS_i is the spatial correlation in the i^{th} indexed level, and the *MaxIndex* is the index of the highest indexed level (*e.g.*, in Figure 4, *MaxIndex* is equal to 2).

To extract the spatial correlation based on the above method, we need the floorplan of the design. In the ASIP design, the ISA extension should be performed in early steps of the design flow when the floorplan of the extensible processor is not determined. In the ISA extension, only the execution unit is modified where a CFU part is added in parallel with the ALU.

Therefore, in designing an extensible processor, we need to add a CFU as a parallel unit to the existing ALU[*]. If we know the area usage of the extended ISA, we can model the CFU as a black box and add it as an layout object to be floor planned along with the other objects in the extensible processor floorplanning phase. Note that changing the area of the CFU changes the total area of the extensible processor, and hence, the optimum layout of the basic design (which contains all blocks of the extensible processor except for the CFU) typically changes. Figure 5 shows the floorplan of a MIPS processor, where a part of the die is set aside for the CFU usage. Because of the connection between the CFU and register-file and also the connection between the execution unit and CFU, CFU should be placed near the register file and the execution unit to meet the propagation delay constraint. Hence, in floorplanning of the MIPS processor, the CFU part was placed between the register-file and the execution unit. Assuming that the selected CIs are placed in this region we can then extract the systematic variation information. Using this approach, the impact of the process variation on the baseline processor and CFU parts of the extensible processor can be studied.

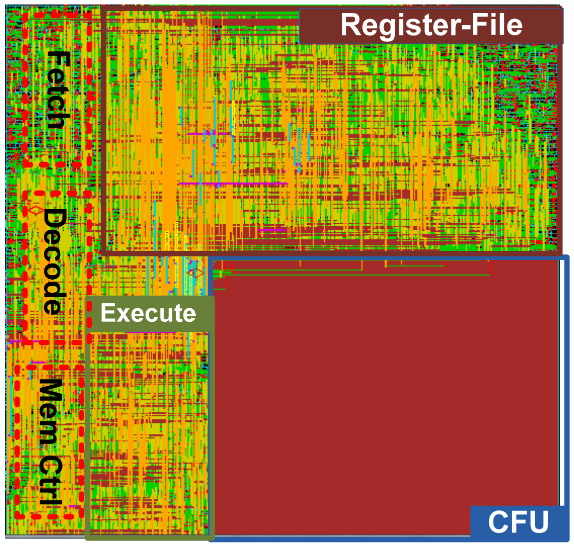


Figure 5 A layout design of a MIPS processor while a part of the area is reserved for CFU

In the selection phase, however, the locations of CIs have not been determined yet, and hence, their systematic variations cannot be calculated. To overcome this problem, we estimate the CI area and shape. In particular, the CI's shape is assumed to be rectangular with the rectangle area set to the CI's approximated area. The CI's approximated area is calculated as follows

$$Area_{apx} = \frac{1}{density\_factor} \times \sum_{i=1}^{|Gates|} Area_i \qquad (13)$$

where *Area*$_i$ is the area of the *i*th gate used in the implementation of the CI, and the *density_factor* is a coefficient capturing the component density in the gate placement step. This value of this coefficient may be set by the designer. In this work, the value of this coefficient was set to 0.83.

The strength of the systematic variation impact on the critical path depends on the distance between the gates on the path. To estimate the variation of a CI, we make use of variations of different points obtained from the multi-level quad-tree method.

[*] To add the new instructions, the decoder stage must be modified to accept them. In this work, we have assumed the processor contains some unused op-codes that are usable by the custom instructions. Hence, the decoder part is not changed.

The maximum amount of variation which exists in a component (e.g., ALU) in the case of the systematic variation depends on maximum distance between any two points in the layout of the component. As is illustrated in Figure 6 the maximum distance is the smallest in the case of square shaped layout compared to the case of rectangular shaped layout. Hence, we assume a rectangular shape for the realization of the CI and assess the impact of different aspect ratios of the rectangle ranging from 1 and 2 (see, Figure 6).

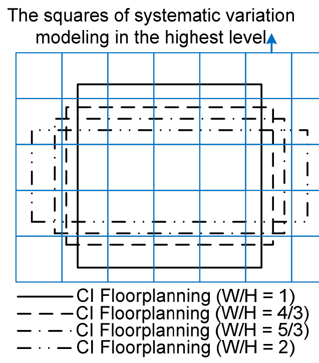

The squares of systematic variation modeling in the highest level
CI Floorplanning (W/H = 1)
CI Floorplanning (W/H = 4/3)
CI Floorplanning (W/H = 5/3)
CI Floorplanning (W/H = 2)

Figure 6 Floorplanning shape of a CI with different Width/Height ratios

For each pattern, the systematic variation modeling is done by partitioning the area of the CI and assigning a random value to each square based on [26]. The parameter variation of each square at the highest level (e.g., Level 2 in Figure 4) is calculated by summing the parameter variations of all the squares in the lower levels to which the square belongs plus its own parameter variation (see, e.g., (11)). After calculating the variation of the smallest squares (*i.e.*, squares in the highest indexed level), the variation of the pattern is calculated by finding the standard deviation of the variation of these squares. Finally, by averaging the standard deviations of different aspect ratios, the parameters of the systematic variation of CI are calculated.

5. PROCESS VARIATIONS IMPACT ON ASIP DESIGN

In the conventional approach, the delay of a CI is estimated as a scalar value. This value is calculated by summing up the node (primitive) worst-case delays in the critical timing path which are activated during the execution of the CI [5]. Due to the increase in the normalized standard deviation of parametric variability in the nanoscale technologies [29], using the worst-case approach is not appropriate and statistical approaches for the delay estimation should be utilized [5].

Figure 7 shows a CI in a given CFU, which contains four nodes, four inputs, and 2 outputs. The delay PDFs of the nodes are shown next to the nodes. If we wish to use the worst-case approach, the delay of the CFU is equal to the maximum critical path delay. For this approach, one should find the worst-case delay of each output (O1 and O2), and obtain their maximum which has a deterministic value. On the other hand, in the statistical approach, the propagation delay of each output has a PDF which might be computed using a statistical static timing analysis (SSTA) [26][28]. The CFU delay has also a PDF, which is obtained by taking the maximum of the PDFs of Path 1 and Path 2.

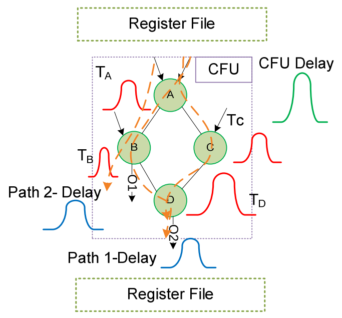
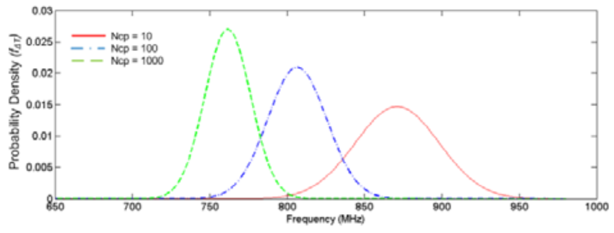


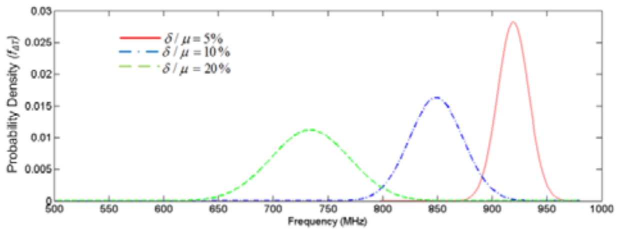
Figure 7 An example of the PDF delay model of different parts of a CFU.

The pipelined stages in a processor contain some critical paths whose delays are defined using PDFs. The processor maximum clock frequency or performance yield is extracted based on these PDFs. The results obtained from (10) reveal that the increase in the ratios of standard deviation to mean of the delays and the number of the critical paths may lead to the processor frequency decrease. As an example, we have plotted the PDFs of a circuit with a critical path delay of 1ns in Figure 8(a) and (b). In Figure 8(a), the standard deviation to mean ratio is constant and is equal to 15%, while in Figure 8(b), the N_{cp} value is constant equal to the 1000.

The dependence of the maximum operation frequency of the processor on the number of critical paths suggests that extending the ISA may result in reduced clock frequency for the extensible processor. This is due to the increase in the number of the critical paths in the system. This decrease in the maximum operating frequency may increase the execution time of the application, which is undesirable. This raises the question whether the speedup improvement will be achieved if we extend the processor. One of the main contributions of this paper is to find an answer to this question. We investigate the difference between the speedup improvements that we expect at the design time and the one that we would achieve after manufacturing the processor. In our study, both systematic and random variations are considered.



(a)



(b)

Figure 8 Impact of the N_{cp} and the standard deviation per mean ratio, on the PDF of the maximum frequency on a circuit while the mean of its critical paths dealy is 1ns (a) Standard deviation to mean ratio is constant equal to 15% (b) the N_{cp} value is constant equal to the 1000.

6. RESULTS AND DISCUSSION

A. Simulation Setup

To assess the performance of the proposed technique, we selected several applications from mibench [30], PacketBench [31], and SNR-RT benchmark suits [32]. The *IP-sec* and *MD5* from PacketBench, *lms* and *adpacm* from SNR-RT, and *G721encoder*, *G721decoder*, and *bitcounter* from mibench suit were chosen. The benchmarks' DFGs were extracted using GIMPL intermediate representation [33] generated by GCC. To evaluate the proposed design flow, we have implemented it with the programming language C#. The identification phase was performed based on the method presented in [1]. We modified the method in such a way that the clock period constraint could be included. Also, to implement the selection phase we used a greedy approach [2]. The area and delay of the primitives (e.g., ADD and SHIFT units) were obtained by Synopsys Design Compiler.

We used a 32-bit in-order MIPS processor as our baseline machine. To extract the critical paths, we implemented its layout using a 45nm technology [25]. For synthesis the baseline processor, we have used the Synopsys Design Compiler and defined 1ns as the timing constraint. The baseline processor was synthesized without any timing violation. The wire conditions (default conditions) and also the clock trees were selected automatically by the synthesis tool. Next, to obtain the layout from the gate-level netlist of the processor, we used the Cadence SOC Encounter. The only specific parameter that we set in the tool was the delay constraint of 1 ns. The options of placement and routing stages were set such that the tool fully attempted to meet the delay constraint.

By adding the CFU, the placement of the gates in the baseline processor was changed. In our study, the baseline processor with different CFU areas was placed and routed and the critical paths were extracted. In all the cases, we forced the place-and-route tool to map the design such that all the propagation delays of the paths were less than 1ns. Therefore, the maximum frequency of the processor was set to 1 GHz. For the same reason, in the identification phase, the defined propagation delay constraint was 1ns.

For the process variation analysis, we assumed $\sigma L^{sys}/L_0 = 5\%$, and $\sigma V_{th}^{rnd} \,/\, V_{th0} = \sigma L^{rnd} \,/\, L_0 = 10\%$ [34]. In the proposed approach, any other sources of variation affecting the delay of the CFU and baseline processor could be easily included. The reason that we did not include other sources is that they are not as important as RDF and channel length fluctuation. For example, oxide thickness variation is not of prime concern due to the use of high-κ dielectrics (higher thickness of the dielectric and hence considerably lower impact of its variations) in the MOSFET structure.

To model the systematic variation, we define the area of the squares in the highest indexed level (see, Figure 4) about 100μm^2 as the highly correlated area. To find the maximum frequency of the extensible processor, we determined the PDFs of frequency of the CFU and the basic part of the extensible processor separately using (10). Finally, the maximum value of the worst-case frequencies of these two PDFs was selected as the frequency of the extensible processor. Also, in this paper $\mu+3\sigma$ is used to calculate the worst-case frequency and delay.

B. Results for Impact of Process Variations on Speedup and Frequency

1) Impact of Process Variations on Speedup

First, we study the impact of process variations on the speedup of the extensible processor. The overall speedups of the extensible processor for different applications are reported in Figure 9. The CIs were selected using the merit functions given by (4) and (5) based on the nominal delay ("*Without PV*" and with "*PV*") and worst-case delay ("*PV Worst-Case*"). While in both cases of "*Without PV*" and "*PV*"**,** the CIs were selected based on the nominal delay of the primitives, in the latter case, the impact of process variations on the maximum frequency of the extensible processor was also considered. It means that in the case of "*PV*", after selecting the CIs, the impact of process variations on the extensible processor was modeled, and the frequency reduction was calculated. For the case "*PV Worst-Case*", instead of using the nominal delay, the worst-case delays of the primitives were used, and hence, we did not need to conduct a separate analysis for the impact of process variations on the extended ISA. The study includes the area constraints of 3%, 13%, and 51%. As the results presented in Figure 9 reveal, for the applications considered in this work, process variations may reduce the average (maximum) speedup of the extensible processor by about 1.44% (5%). If we select the CIs based on the worst-case delay, the achieved speedups in all cases (except

Bitcounter, *IP-sec* and *lms* benchmarks for the area budget of 3%) are lower than those based on the nominal delay ("*PV*"). In the "*PV Worst-Case*" approach, there is not any critical path in the CFU. When the selection is performed based on the nominal delay, it is possible that the CFU contains some critical paths decreasing the operating frequency. The comparison between the "*PV*" and "*PV Worst-Case*" cases shows that the speedup achieved through using the selected CIs has more effects than the maximum operating frequency reduction by increasing the number of the critical paths of the selected CIs. Also, the results show that the difference between the "*PV Worst-Case"* approach and "*PV"* is reduced by decreasing the area budget. In the *IP-sec* and *lms* benchmarks, when the area budget is 3% of the baseline processor area, the speedup of "*Worst-Case*" is equal to that of the "*PV*". The reason is that for these two benchmarks, under this area budget, the selected CIs in both "*PV*" and "*PV Worst-Case*" were similar. In the case of *Bitcounter* benchmark, when the area budget is 3%, the speed gain of the "*PV Worst-Case*" is better than the case of "*PV*". In this case, the speed gain of the "*PV Worst-Case*" was not reduced compared to the case of "*Without PV*". This shows that in this case, there were some CIs with high *CS* whose critical path degradations due to the process variation do not violate the timing constraint (timing yield of 1). On the other hand, in the "*PV*" case where the impact of the process variation on the selected CIs was considered, some of the CIs violated the timing constraint forcing us to reduce the operating frequency of the extensible processor. This caused the reduction in the speedup compared to the case of "*Without PV*" (which is about the same as the case of "*PV Worst-Case*").

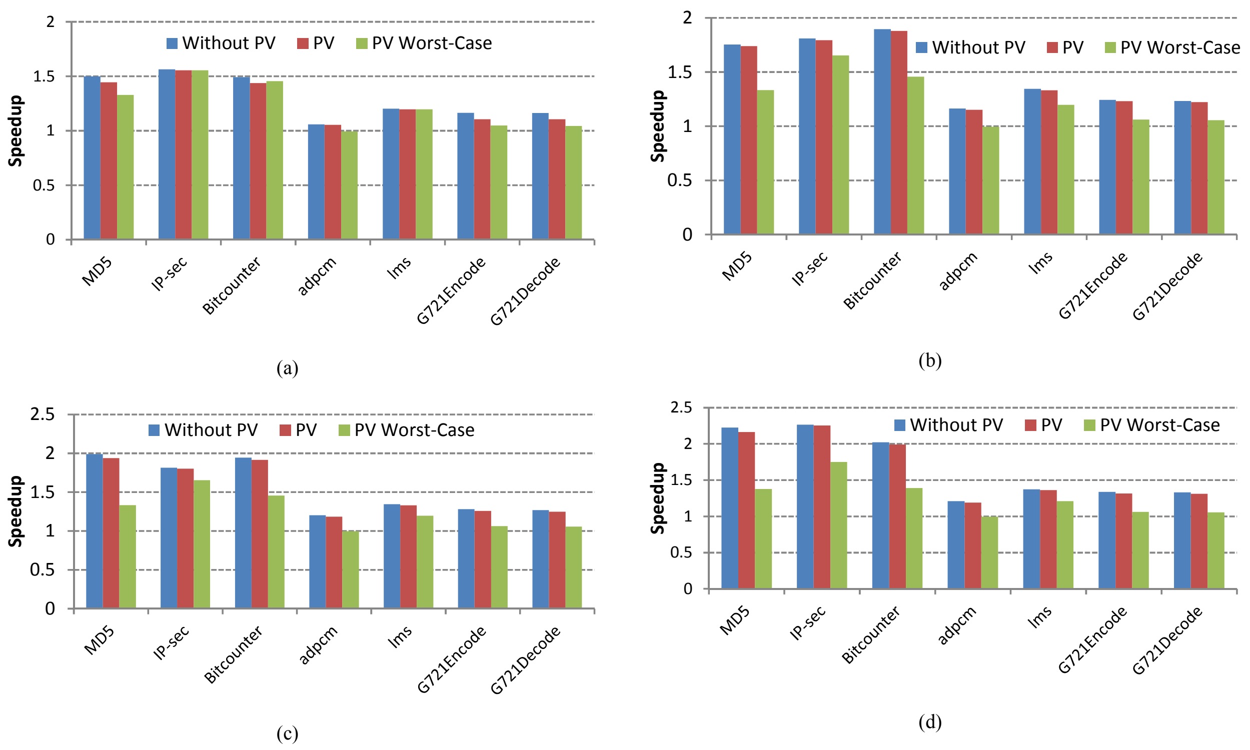


Figure 9 The impact of the process variations on the overall speedup of the extensible processor. The CIs were selected under a) area budget of 3%, b) area budget of 13%, c) area budget of 51% of the baseline processor, d) no area budget.

2) *Impact of Area Budget on Speedup Reduction*

Figure 10 shows the average of the speedup reduction versus the area budget. As may be inducted from the figure, enlarging the area budget increases the impact of the timing variation on the CFU, and hence, the speedup reduction of the extended ISA. For small area budgets, except in the case of 3% area budget (this will be discussed next), the PV impact on speedup reduction is lower.

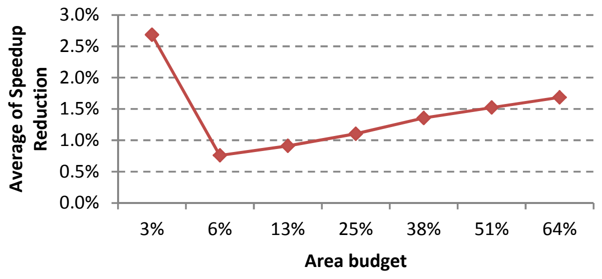


Figure 10 The average speedup reduction of the "PV" case compared to the "Without PV" case under different area budgets.

Speedup reduction originates from the increase in the number of critical paths in the CFU which leads to the decrease in the maximum frequency of the extensible processor. To study this further, as an example, the delays of the CFU and baseline processor, and also the number of critical paths of the selected CFU, N_{cp}, for the *MD5* benchmark under different area budgets are shown in Figure 11. Enlarging the area budget increases the number of the selected CIs which may lead to increase in the number of the critical paths (and the timing variation) which decreases the mean and standard deviation (σ) of the maximum frequency (see Figure 8(a)). On the other hand, by decreasing the N_{cp} the mean value of the maximum frequency is increased. Hence, it mitigates the impact of the CFU in decreasing the maximum frequency. However, the sigma of the delay variation also is increased by decreasing N_{cp} (see Figure 8(a)). The larger value of σ could decrease the worst-case of the maximum frequency more than the increase in the mean of the maximum frequency. This is exactly what happens in the case of where the area budget is 3%. In this case, the maximum value of N_{cp} for the proposed benchmarks was 1, and hence, the worst-case delay of the CFU was larger than that of the baseline processor. Therefore, in this case, for some benchmarks, the speedup reduction was increased. It should be mentioned that in Figure 11, the delays of the basic part for different area budgets are different. This originates from this fact that by changing the area of the extended ISA, the placement and routing of the critical paths in the basic part are changed.

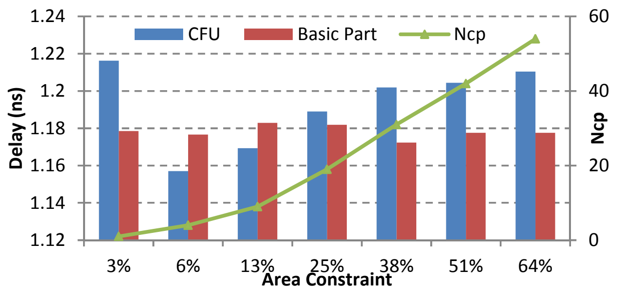


Figure 11 Left Axis) The worst-case delay of the CFU and the basic part for MD5 benchmark. Right Axis) The number of the critical paths (N_{cp}) of the selected CFU in *MD5* benchmarks.

Note that the speedup reduction highly depends on process variations impact on both the CFU and baseline processor. To show the impact of the number of critical path of the baseline processor on the speedup reduction of the extensible processor, we calculated the average and the maximum speedup reduction of the applications considered in this work under two different cases. In the first case, we assumed that the number of the critical paths of the baseline processor was decreased to half, while for the second case we assumed no critical path in the baseline processor. The results showed that, compared to the *"Without*

PV" case, in the first case the average(maximum) speedup reduction was 2.16% (6.1%), while for the second case process variations led to the increase of the average speedup reduction by about 18.35%.

3) Impact of Process Variations on Frequency

Figure 12 shows the frequency of the "*PV"* and "*PV Worst-case"* approaches when the ISA are selected without any area budget. Due to the impact of the critical paths on the CFU frequency, in the "*PV"* approach, the frequency of the extensible processor is less than that of the *"PV Worst-case"* approach (there are not any critical paths in the CFU). Finally, the speedup of the extended instructions in the *adpcm* benchmark in the worst-case approach is one. It shows that all the CI candidates of the *adpcm* benchmarks had a worst-case delay greater than the propagation delay constraint and hence no CI was selected.

To study the impact of the CFU in the presence of process variations on the frequency of the extensible processor, in our investigation we determined which part of the extensible processor (the basic part (*bP*) or the CFU (*C*)) defines the maximum frequency of the extensible processor. The results which are based on the worst-case delay are presented in Table I. Again, the results show that by increasing the area budget, the importance of the CFU in setting the maximum frequency increases. In the case of *lms*, by increasing the area budget, the number of critical paths for the selected CFU does not increase as much as other benchmarks. Hence, in the *lms* case, for all the area budgets, the critical paths in the basic part are more crucial than those in the CFU.

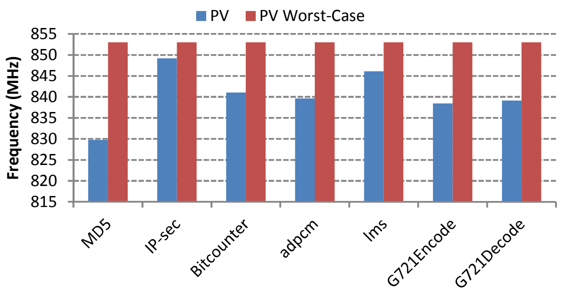


Figure 12 Maximum frequency of the CFU for the "*PV*" and "*PV Worst-Case*" approaches. The CIs are selected without any area constaint.

TABLE I WHICH PART OF THE EXTENSIBLE PROCESSOR DETERMINES THE MAXIMUM FREQUENCY OF THE EXTENSIBLE PROCESSOR. BASIC PART "BP" OR CFU "C".

	MD5	**IP-sec**	**Bitcounter**	**adpcm**	**lms**	**G721Encode**	**G721Decode**
WNAB*	*C*	*bP*	*C*	*C*	*bP*	*C*	*C*
3%	*C*	*bP*	*C*	*bP*	*bP*	*C*	*C*
6%	*bP*	*C*	*bP*	*C*	*bP*	*bP*	*bP*
13%	*bP*	*bP*	*bP*	*C*	*bP*	*bP*	*bP*
25%	*C*	*bP*	*C*	*C*	*bP*	*bP*	*C*
38%	*C*	*C*	*C*	*C*	*bP*	*C*	*C*
51%	*C*	*C*	*C*	*C*	*bP*	*C*	*C*
64%	*C*	*C*	*C*	*C*	*bP*	*C*	*C*

*WNAB stands of With No Area Budgets

C. Comparison of Random and Systematic Variation Impact

In this subsection, to determine the significance of the random and systematic variations, we will study the impact of each type separately on the propagation delay and speedup.

1) Impact of Systematic and Random Variation on Propagation Delay

Table II shows the impact of the systematic and random variation on the propagation delay of selected CFU in the *IP-sec* and *Bitcounter* benchmarks. Also, the effects of the variations on the baseline processor are given in this table. By increasing the systematic variation (σ/μ) from 0% to 10%, the worst-case delay of the CFU does not change considerably. On the other hand, for the baseline processor, by increasing the systematic variation from 0% to 5% (5% to 10%) the worst-case delay increases from 1.14ns to 1.17ns (1.17ns to 1.25ns). Because the primitives of the critical paths in the baseline processor (basic part) are scattered in the larger area in comparison with those in the CFU, the systematic variation has more effects on the baseline processor. Most of the critical paths in the baseline processor belong to the forwarding paths and register-file [8][10]. Figure 13 shows the example for an extensible processor for *lms* benchmark with 6 CIs (where the layout was designed by the Cadence SOC Encounter). Also, the placements of the primitives of the longest paths are shown with dashed boundaries. The primitives of the critical paths of the basic part are scattered in more areas in comparison with the primitives of CIs in the CFU.

TABLE II IMPACT OF THE RANDOM AND SYSTEMATIC VARIATIONS ON THE PROPAGATION DELAY OF SELECTED CFU IN 45NM TECHNOLOGY.

Delay (ns)

	Benchmark	Area Constraint	**Random Variation = 10%**			**Systematic Variation = 5%**		
			Systematic Variation			**Random Variation**		
			0%	5%	10%	0%	10%	15%
CFU	**IP-sec**	WNAB	1.18	1.18	1.18	1.01	1.18	1.25
		3%	1.00	1.00	1.00	1.00	1.00	1.00
		13%	1.21	1.21	1.21	1.00	1.21	1.30
		25%	1.15	1.16	1.16	1.00	1.16	1.22
		38%	1.18	1.18	1.18	1.00	1.18	1.25
		51%	1.18	1.18	1.18	1.00	1.18	1.25
		64%	1.18	1.18	1.18	1.00	1.18	1.25
	Bitcounter	WNAB	1.19	1.19	1.19	1.00	1.19	1.26
		3%	1.21	1.22	1.22	1.00	1.22	1.31
		13%	1.16	1.16	1.16	1.00	1.16	1.23
		25%	1.18	1.18	1.18	1.00	1.18	1.25
		38%	1.19	1.19	1.19	1.00	1.19	1.27
		51%	1.19	1.19	1.19	1.00	1.19	1.27
		64%	1.19	1.19	1.19	1.00	1.19	1.27
Baseline Processor			1.14	1.17	1.25	1.11	1.17	1.22

Also, the situation for the random variation is similar for both CFU and baseline processor where by increasing the random variation, the worst-case delay increases. Also, the results show ("*Random Variation* = 0%") the systematic variation impact are not as important as random variation on the CFU.

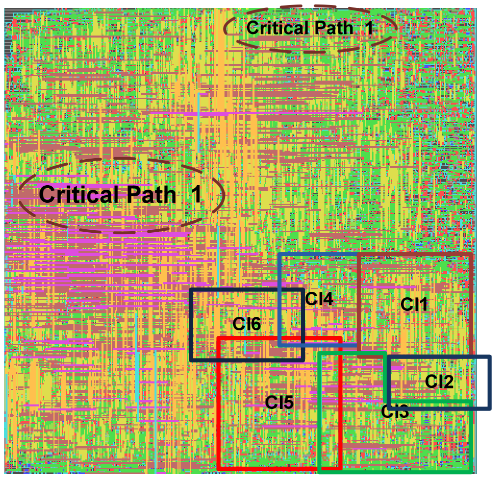


Figure 13 An extensible processor with 6 CIs (MIPS as the baseline processor). The placement of the longest critical path in the basic part is shown.

To see the effect of technology on the results reported in Table II, we performed the study for the 90 nm library for different amounts of parameter variation. The results are reported in Table III. Note that, in this case, due to the increase in the gate delay, 1.2ns was defined as the timing constraint.

As was expected, the behaviors regarding the impact of increasing or decreasing the process variation on the delay of CFU and baseline processor, were the same as those of the 45nm technology. Finally, it should be noted that the results show that the delay variations due to the parameter variation in the 90nm technology are smaller than those in the 45nm technology.

TABLE III IMPACT OF THE RANDOM AND SYSTEMATIC VARIATIONS ON THE PROPAGATION DELAY OF SELECTED CFU IN 90 NM TECHNOLOGY.

			Delay (ns)			
		Systematic Variation	0%	5%	2.5%	5%
		Random Variation	10%	10%	5%	0%
	Benchmark	Area Constraint				
CFU	**IP-sec**	WNAB	1.256	1.259	1.256	1.259
		3%	1.256	1.259	1.256	1.259
		13%	1.256	1.259	1.256	1.259
		38%	0.000	0.000	0.000	0.000
		64%	1.170	1.174	1.176	1.174
	Bitcounter	WNAB	1.166	1.169	1.171	1.169
		3%	1.166	1.169	1.171	1.169
		13%	1.166	1.169	1.171	1.169
		38%	0.000	0.000	0.000	0.000
		64%	1.251	1.260	1.257	1.260
	Baseline Processor		1.344	1.355	1.327	1.332

2) *Impact of Systematic and Random Variation on Speedup*

Table IV shows the impact of the systematic and random variations on the speedup of the extensible processor when running some benchmarks. When the random variation is 10% and the systematic variation varies, in most cases, the maximum values of speedup belong to the variation ratios of 5% and 10%. By increasing the systematic variation, the delay fluctuation of the CFU is not changed more (see, Table II), but the impact of the systematic variation on the baseline processor is increased, and consequently its delay is increased. Hence, the frequency of the extensible processor defined based on the delay of the critical paths in the basic part. Since the frequencies of the processors-with or without CFU- are similar, the CFU have more effects on runtime reduction. On the other hand, when the systematic variation is low, the CFU is the part whose worst-case delays of its critical paths are greater. Hence, in this case, the frequency of the extensible processor is less than the frequency of the baseline processor. For instance, Table V depicts the worst-case delays of the CFU, basic part, and the extensible processor for the *MD5* benchmark which has the trend that we just mentioned.

TABLE IV IMPACT OF THE SYSTEMATIC VARIATION ON THE SPEEDUP OF EXTENSIBLE PROCESSOR FOR SOME BENCHMARKS IN 45NM TECHNOLOGY.

	Area Constraint	**Random Variation = 10%**			**Systematic Variation = 5%**		
		Systematic Variation			**Random Variation**		
		0%	5%	10%	0%	10%	15%
MD5	WNAB	2.10	2.16	2.17	2.46	2.16	2.11
	3%	1.40	1.44	1.48	1.66	1.44	1.40
	13%	1.70	1.74	1.74	1.94	1.74	1.72
	38%	1.82	1.88	1.89	2.13	1.88	1.83
	64%	1.94	1.99	2.01	2.28	1.99	1.94
IP-sec	WNAB	2.18	2.25	2.21	2.49	2.25	2.21
	3%	1.56	1.56	1.55	1.73	1.56	1.55
	13%	1.78	1.79	1.79	1.99	1.79	1.80
	38%	1.74	1.80	1.79	2.00	1.80	1.76
	64%	1.74	1.80	1.79	2.00	1.80	1.76
Bitcounter	WNAB	1.93	1.99	1.99	2.23	1.99	1.95
	3%	1.40	1.44	1.47	1.65	1.44	1.39
	13%	1.83	1.88	1.88	2.10	1.88	1.85
	38%	1.86	1.92	1.92	2.15	1.92	1.90
	64%	1.86	1.92	1.92	2.15	1.92	1.87
adpcm	WNAB	1.15	1.19	1.19	1.33	1.19	1.16
	3%	1.03	1.05	1.05	1.17	1.05	1.04
	13%	1.11	1.15	1.15	1.28	1.15	1.13
	38%	1.15	1.18	1.19	1.33	1.18	1.15
	64%	1.15	1.18	1.19	1.32	1.18	1.15

TABLE V WORST-CASE DELAY OF THE EXTENSIBLE PROCESSOR, BASIC PART, AND CFU FOR THE *MD5* BENCHMARK.

Worst-Case Delay(ns)	0%	5%	10%
CFU	**1.20**	**1.21**	1.21
Basic Part	1.14	1.17	**1.25**
Extensible Processor	**1.20**	**1.21**	**1.25**

In the second case (the systematic variation is 5% while the random variation varies) the random variation has similar effects on both the CFU and baseline processor. Thus, by increasing the random variation, the impact of the CFU to increase the speedup of the extensible processor is reduced.

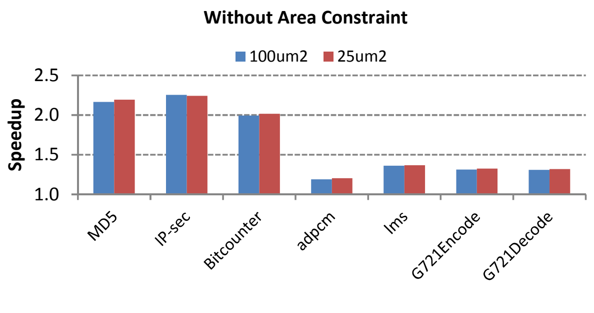
To support the results which are reported in Table IV, the results in 90nm technology are reported in Table VI. As was expected, the behaviors regarding the impact of increasing or decreasing the process variation on the speedup of the extensible processor were the same as that of the 45nm technology. Finally, it should be noted that the results show that the speedup variations due to the parameter variation in the 90nm technology are smaller than those in the 45nm technology.

TABLE VI IMPACT OF THE SYSTEMATIC VARIATION ON THE SPEEDUP OF EXTENSIBLE PROCESSOR FOR SOME BENCHMARKS IN 90NM TECHNOLOGY

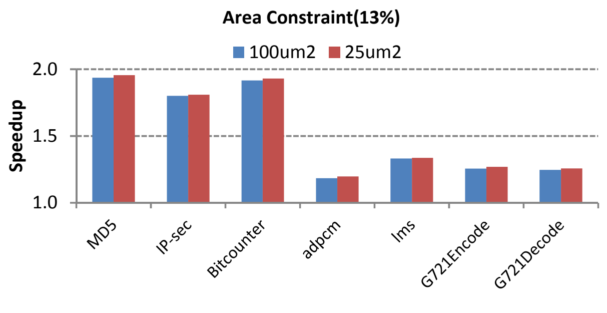
			Speedup			
		Systematic Variation	0%	5%	2.5%	5%
		Random Variation	10%	10%	5%	0%
	Benchmark	Area Constraint				
CFU	**MD5**	WNAB	1.537	1.550	1.551	1.555
		3%	1.356	1.357	1.359	1.361
		13%	1.438	1.442	1.439	1.442
		38%	1.470	1.483	1.482	1.486
		64%	1.501	1.514	1.516	1.519
	IP-sec	WNAB	1.507	1.515	1.511	1.516
		3%	1.391	1.392	1.394	1.396
		13%	1.464	1.467	1.479	1.482
		38%	1.460	1.468	1.480	1.485
		64%	1.460	1.468	1.480	1.485
	Bitcounter	WNAB	1.502	1.506	1.509	1.507
		3%	1.330	1.331	1.333	1.335
		13%	1.472	1.478	1.481	1.478
		38%	1.482	1.486	1.490	1.488
		64%	1.482	1.488	1.490	1.488
	adpcm	WNAB	1.271	1.273	1.275	1.273
		3%	1.169	1.170	1.188	1.190
		13%	1.256	1.259	1.256	1.259
		38%	1.256	1.259	1.256	1.259
		64%	1.256	1.259	1.256	1.259

3) *Impact of the Size of the Highly Correlated Area on Speedup*

By changing the systematic variation, we study the effects of the size of the highly correlated area on the speedup of the extensible processor. In this study, the size of the highly correlated area is shrunk from 100μm^2 to 25μm^2. The results which are reported in Figure 14 show, in most cases, the speedup is increased. This is due to the fact that, in these cases, the maximum delay belongs to the critical paths in the CFU due to the random variation. Also, as mentioned before, the impact of the systematic variation on the baseline processor is more than that of the CFU. By decreasing the highly correlated area, the worst-case delay of the baseline processor increases from 1.17ns to 1.19ns while in all benchmarks, the increase in delay of the CFU is negligible (< 1ps). Hence, in these cases, without changing the maximum frequency of the extensible processor, the maximum frequency of the baseline processor decreases leading to the speedup increase.



(a)



(b)

Figure 14 Impact of the size of the highly correlated area on the speedup of the extensible processor.

7. CONCLUSION

In this work, the performance of the extensible processor under process variation was investigated. First, the effects of random and the systematic variations on the extended ISA and the baseline processor were modeled. Then, the impact of the timing variation on the overall speedup and the maximum achievable frequency of the extensible processor was investigated. Additionally, the role of each of the CFU and the basic part of the processor in decreasing the speedup was determined. The results of the study showed that the process variation reduced the overall speedup of the extensible processor mainly due to the impact on CFU part. We considered the case of designing based on the worst-case timing. The study for this case revealed that the impact of the process variation was reduced. However, the speed enhancement of the extended ISA was also decreased which lowered the overall speedup of the design. In terms of the importance of variation type, our results indicated that only the random variation had dominant impact on the CFU part of the extensible processor while both random and systematic variations had major effects on critical paths of the baseline processor.

REFERENCE

[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 Transactions on Very Large Scale Intergrations (VLSI) Systems, Vol(16), pp. 1259- 1267, 2008.

[3] K. Atasa, L. Pozzi, and P. Ienne, ”Automatic application specific instruction-set extensions under microarchitectural constraints,” in Proceedings of the 40th Design Automation Conference(DAC), 2003, pp. 256-261.

[4] N.T. Clark, H. Zhong, and S.A. Mahlke, “Automated Custom Instruction Generation for Domain-Specific Processor Acceleration,” in IEEE Transactions on Computer, Vol(54), pp. 1258-1270, 2005.

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

[6] F. Wang, Y. Xie, and A. Takach, “Variation-Aware Resource Sharing and Binding in Behavioral Synthesis,” in Proceedings of the Asia and South Pacific Design Automation Conference(ASP-DAC), 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] P. Ndai, N. Rafique, M. Thottethodi, S. Ghosh, and S. Bhunia, “Trifecta: a nonspeculative scheme to exploit common, data-dependent subcritical paths,” in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, (Vol) 18, pp. 53-65, 2009.

[9] A. Tiwari, S.R. Sarangi, and J. Torrellas, “ReCycle: pipeline adaptation to tolerate process variation,” in Proceedings of the 34th annual International Symposium on Computer Architecture (ISCA), 2007, pp. 323-334.

[10] X. Liang and D. Brooks, ”Mitigating the impact of process variations on processor register files and execution units,” in Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture, 2006, pp. 504-514.

[11] M. Kamal, A. Afazli-Kusha, and M. Pedram, ”Timing Variation-Aware Custom Instruction Extension Technique,” in Proceedings of the Design, Automation and Test in Europe (DATE), 2011, pp. 1517-1520.

[12] N. V. Mujadiya, “Instruction scheduling for VLIW processors under variation scenario,” in Proceedings of the 9th international conference on Systems, architectures, modeling and simulation (SAMOS), 2009, pp. 33-40.

[13] T. Sato and S. Watanabe, ”Uncriticality-directed scheduling for tackling variation and power challenges,” in Proceeding of 10th international Symposium on Quality Electronic Design (ISQED), 2009, pp. 820-825.

[14] X. Liang, G.Y. Wei, and D. Brooks, ”Revival: A variation-tolerant architecture using voltage interpolation and variable latency,” in Proceeding of 35th International Symposium on Computer Architecture (ISCA-35), 2008, pp. 191-202.

[15] E. Chun, Z. chishti, and T.N. Vijaykumar, “Shapeshifter: Dynamically changing pipeline width and speed to address process variations,” in Proceedings of 41st IEEE/ACM International Symposium on Microarchitecture, 2008, pp. 411-422.

[16] C. Ozturan, G. Dunar, and K. Atasu, “An integer linear programming approach for identifying instruction-set extensions,” in Proceedings of the Third IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, 2005, pp. 172-177.

[17] K. Verma, P. Brisk, and P. Ienne, “Rethinking custom ISE identification: A new processor-agnostic method,” In Proceedings of the 2007 international conference on Compilers, architecture, and synthesis for embedded systems, 2007, pp. 125–134.

[18] K. Atasu, O. Mencer, W. Luk, C. Ozturan, and G. Dundar, “Fast custom instruction identification by convex subgraph enumeration,” In Proceedings of International Conference on Application-Specific Systems, Architectures and Processors, 2008, pp. 1-6.

[19] T. Li, Z. Sun, W. Jigang, and X. Lu, ”Fast Enumeration of Maximal Valid Subgraphs for Custom-instruction Identification,” in Proceedings of international conference on Compilers, architecture, and synthesis for embedded systems, 2009, pp. 29-36.

[20] M. Kamal, N. Kazemian Amiri, A. Kamran, S.A. Hoseini, M. Dehyadegari, and H. Noori, “Dual-Purpose Custom Instruction Identification Algorithm based on Particle Swarm Optimization,” in Proceeding of 21st IEEE International Conference on Application-specific Systems, Architectures and Processors(ASAP10), Rennes, France, 2010, pp. 159-166.

[21] N. Clark, A. Hormati, S. Mahlke, and S. Yehia, “Scalable subgraph mapping for acyclic computation accelerators,” in Proceedings of international conference on Compilers, architecture and synthesis for embedded systems, 2006, pp. 147-157.

[22] Y.S. Lu, L. Shen, L.B. Huang, Z.Y. Wang, and N. Xiao, “Optimal subgraph covering for customisable VLIW processors,” in IET Computer and Digital Techniques, Vol(3), pp. 14-23, 2009.

[23] L. Siew-Kei, T. Srikanthan, and C.T. Clarke, ”Selecting Profitable Custom Instructions for Area–Time-Efficient Realization on Reconfigurable Architectures, in IEEE Transactions on Industrial Electronics, Vol(56), pp. 3998- 4005, 2009.

[24] S. K. Lam, T. Srikanthan, and C. T. Clarke “Selecting Profitable Custom Instructions for Area-Time-Efficient Realization on Reconfigurable Architectures,” in *IEEE Trans. on Industrial Electronics*, vol. 56, issue 10, pp. 3998-4005, Oct. 2009.

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

[26] A. Agarwal, D. Blaauw, V. Zolotov, S. Sundareswaran, M. Zhao, K. Gala, and R. Panda, “Path-based statistical timing analysis considering inter-and intra-die correlations,” in Proceedings of the 8th ACM/IEEE international workshop on Timing issues in the specification and synthesis of digital systems, 2002, pp. 16–21.

[27] R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas, “ Mitigating Parameter Variation with Dynamic Fine-Grain Body Biasing,” in Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture, 2007, pp. 27-42.

[28] K. A. Bowman, S.G. Duvall, and J.D. Meindl, “Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration," in IEEE Journal of Solid-State Circuits,Vol(37), pp. 183-190, 2002.

[29] C. Kenyon et al., ‘‘Managing Process Variation in Intel’s 45nm CMOS Technology,’’ Intel Technology Jornal, Vol(12), No(2), pp. 93-110, 2008;

[30] M. R. Guthaus, J.S. Ringenberg, D. Ernst, T.M. Austin, T. Mudge, and R.B. Brown, “MiBench: A free, commercially representative embedded benchmark suite,” in Proceedings of International workshop on workload characterization, 2001, pp. 3-14.

[31] 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.

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

[33] The GNU operating system.[Online]. Available: http://gcc.gnu.org/.

[34] International Technology Roadmap for Semiconductors, 2007. [online]. Available: http://public.itrs.net/