Improving Efficiency of Extensible Processors by Using Approximate Custom Instructions

Mehdi Kamal[1], Amin Ghasemazar[1], Ali Afzali-Kusha[1], Massoud Pedram[2]
[1]School of Electrical and Computer Engineering, University of Tehran
[2]Electrical Engineering Department, University of Southern California
{mehdikamal, a.ghasemazar, afzali}@ut.ac.ir, pedram@usc.edu

***Abstract*—In this paper, we propose to move the conventional extensible processor design flow to the approximate computing domain to gain more speedup. In this domain, the instruction set architecture (ISA) design flow selects both exact and approximate custom instructions (CIs). The proposed approach could be used for the applications where imprecise results may be tolerated. In the CI identification phase of the flow, the CIs which do not satisfy the maximum propagation delay but can provide approximate results also may be included in the CI candidate set. Next, in the selection phase, we propose a merit function which selects CIs with higher cycle savings and small error rates. The efficacy of the proposed approximate design flow is investigated using the case studies of the discrete cosine transform (DCT) and inverse DCT (iDCT) of the MPEG2 application. Also, the impact of the process variation on the impreciseness of the results is investigated.**

I. INTRODUCTION

In recent years, the request for using embedded processors in different application domains has been considerably increased. Achieving high performance yet low power embedded processors is not normally obtainable through general-purpose processors. This, however, may be attained via the use of extensible processors which have emerged in the field of embedded computing as a promising approach [1]. In this approach, by extending the instruction set architecture (ISA) of the baseline embedded processor, the critical portions of the running application on the processor is accelerated. The extension includes adding some extra custom instructions (CIs) to the instruction set of the baseline processor. These CIs are executed through a hardware which is called custom functional unit (CFU).

The successful use of the extensible processor approach strongly depends on providing an efficient automated design flow for customizing the ISA of the processors. There are two main phases in the design flow including custom instruction identification and selection, which are computationally intensive and time-consuming. The ways that these phases are performed directly impact the effectiveness of the extensible processors. Many approaches have been proposed to automate these phases (see, *e.g.*, [1]). In the identification phase, all the convex subgraphs of the data flow graph (DFG) of the application which satisfy the micro-architectural constraints such as delay and I/O limitations, are enumerated and placed in a CI candidate set. Next, the isomorphic CIs are grouped to each other and generate a CI group set which can be executed using a piece of hardware in the CFU. Also, based on overlapping between the nodes of the identified CIs a conflict graph indicating these overlaps are generated. Finally, in the selection phase, the best CIs from the CI candidate set are chosen.

The constraints play a major role in the speed enhancement of the extensible processors. The maximum propagation delay (*MPD*) of the CIs is one of the main constraints which is defined based on the clock period (single or multi clock) of the processor. This constraint guarantees that the identified CI completes its operation before its results are fetched by the next stage of the processor. Satisfying this constraint eliminates CIs with higher cycle savings. If we select CIs which do not meet this delay requirement, either the operation will not be correctly performed or the function is partially performed providing imprecise results. If imprecise results can be tolerable, the use of CIs with higher cycle saving may provide us with more speed gain in extensible processors. This has been our motivation for proposing to move from conventional ISA extension design flow to an ISA extension design flow based on approximate computing. The idea of approximate computing is based on tolerability of result impreciseness in some applications which are resilient to errors during execution [2]-[4]. Since obtaining imprecise results is computation less expensive, the use of approximate computing may provide some speed and/or energy gains. The accuracy decrease may affect the functional correctness and reliability. Two targets may be considered for using the approximate computing. The first target is the reducing power consumption which is attained by using approximate functions, which consume lower power compared to the exact operations, or reducing the supply voltage. The second one is the increasing the performance which may be achieved by using approximate functions, which have smaller propagation delay compared to the accurate functions, or decreasing the period of the input clock [5].

In this paper, we suggest using the approximate computing approach for designing the extensible processors for the applications where imprecise operations are tolerable. The flow may improve the speedup of the extensible processor. In the design flow, the identification phase is modified such that the approximate CIs which can be identified. In the selection phase, a merit function is proposed to select CIs with higher cycle savings and smaller error rates. We also study the impact of the process variation on the selected CIs in the approximate computing approach. The rest of the paper is organized as fallows. In Section II, first, we define the approximate CI and then describe our proposed design flow based on approximate computing. The

978-3-9815370-2-4/DATE14/©2014 EDAA

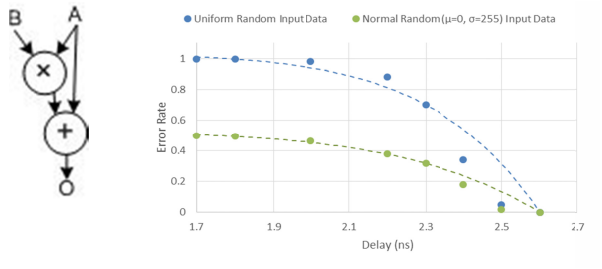
results are discussed in Section III while the conclusion is provided in Section IV.

II. Approximate Custom Instruction

As mentioned before, the extended ISA is used to reduce the runtime of the application. However, it is possible that a candidate CI meets all constraint instead of the propagation delay. In this case, in the conventional approach this CI is not enumerated to be added to the CI candidate set in the identification phase. This CI is omitted because its output results are not accurate. However, by accepting the inaccurate results from this CI, it could be added to the candidate set, which may result in more speed gain. Additionally, in the conventional approach, the process variation decreases the yield of the manufactured chips. Some methods to increase the yield of the extensible processors have been presented [6] while all of them results in decreasing the speedup. However, in approximate computing approach, the error in computations is tolerable. Hence, when the CIs are identified from the parts of the application where the inaccuracy is acceptable, these CIs could be selected without concern about process variation. Therefore, in the presents of the process variation, approximate computing approach may help to improve the speedup of the extensible processors.

An approximate CI does not guarantee to result a precise value, however, in some cases it may generate accurate results. Note that if the maximum propagation delay constraint is increased to a value equal to the propagation delay of approximate CIs, the operation of these CIs will be precise. However, since in this case, the clock period is increased, the speedup will be reduced. For the arithmetic operations such as adder and multiplier, the critical paths are belongs to the most significant bits. Hence, the accuracy of these types of operations depends on the range of the input values. Figure 1(a) shows an example of a CI which contains two 32-bit inputs and one 32-bit output while its *MPD* is 2.66 ns. Figure 1(b) shows the error rates of the generated results under different clock periods. To find the error rates for this CI, two different input patterns have been used. For the first input pattern, 100K uniform random data have been generated while for the second input pattern 100K normal random ($\mu = 0$, $\sigma = 255$) data have been used. As we expected, by decreasing the range of the data variation (around zero), the slope of increasing the error rate is decreased. In the case of the uniform random input, by decreasing the clock period to 1.7 ns, almost all output results are wrong, while in the case of the normal random input, by reducing the clock period to 1.7 ns, only half of the results are wrong. It shows, in approximate computing ISA extension design flow, beside cycle saving of each CI, the error rate of them should be considered to reduce the output error rate. Therefore, in the proposed approximate computing ISA extension design flow (see Figure 2), the identified CI candidate set contains accurate and also, inaccurate CIs, which results in larger candidate set that may lead to selecting a CFU with higher speedup. In CI selection phase of the proposed flow, both speed gain and also, the inaccuracy of the CIs are considered to reach higher speedup with smaller error rate. Now, the details of the proposed flow will be explained in the next paragraphs.

In the identification phase, the modified single-cut identification algorithm proposed in [7] have been used while the details of the algorithm are presented in Figure 3. This algorithm is modified to check the maximum propagation delay constraint for each identified CI (lines 10-15). In the conventional approach, if the propagation delay of an identified sub-graph is smaller than the *MPD* (it is checked by function *violate_MPD()*), the subgraph is added to the CIs candidate set, and the algorithm continues its search in the search tree. In despite, if the propagation delay of the sub-graph is larger than the *MPD*, the algorithm stops its search on the current branch of the search tree, and continues its search from one other branch. However, when inaccuracy of the CIs is acceptable, in the case where the propagation delay of the sub-graph is larger than the *MPD*, first, the identification algorithm adds this subgraph to the candidate set, and next leaves this branch of the search tree, and selects one other branch to search. Note that this added CI is an approximate CI, which its results may be imprecise. The algorithm is not continue its search in a branch that its propagation delay is more than *MPD* due to the increasing the error rate of the CIs which would be selected in this branch.



(a) (b)

Figure 1. a) An example of a CI, and b) its output error rate under different input patterns and propagation delay constraints.

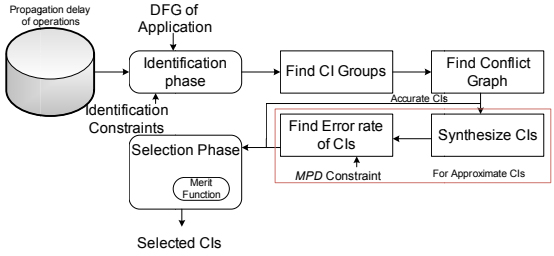


Figure 2. Proposed approximate computing ISA extension design flow

```
1:  identification() {
2:      topological_sort() ;
3:      search(1, 0) ;
4:      search(0, 0) ;
5:  }
6:  search (current_choice, current_index) {
7:      if (current_choice == 1)
8:          if (forbidden()) || (!output_port_check()) ||
              (!permanent_input_port_check()) || (!convexity_check()) return ;
9:          if ( input_port_check()) {
10:             if (!violate_MPD())
11:                 add_to_CI_candidate_set();
12:             else {
13:                 add_to_CI_candidate_set();
14:                 return;
15:             }
16:         }
17:     if ((current_index + 1 ) == NODES) return;
18:     current_index = current_index + 1;
19:     search(1, current_index);
20:     search(0, current_index);
21: }
```

Figure 3. The pseudo code of the identification algorithm

After the identification phase, similar CIs are grouped and the conflict graph of the CIs is generated. Next, the error rate of the CIs which their propagation delays are more than the *MPD*, should be calculated. Hence, to find the error rate of the approximate CIs, by using a synthesis tool, the gate-level netlist of them with their corresponding standard delay files are extracted. Now, by using a gate-level simulator, the error rates of the approximate CIs are calculated. The input of the CIs during the simulation is the profiled data which are collected during the running of the input application on the baseline processor. For each input data, the output result at the clock period of the processor is checked, and if the output is different from the expected output, we consider it as error. Now, based on the error counts, the error rate (*ER*) of each CI is calculated. Note that the *ER* value is bounded by [0,1], and for the accurate CIs this value is equal to zero. It should be noted that the error rate of a CI may be changed due to the process variation. In the existence of the process variation, the error rate may be decrease or increase.

Finally, in the selection phase, the CI groups with the highest cycle saving should be selected. In the approximation computing case, the selection phase should consider the inaccuracy of the CIs along with their cycle saving. Hence, in this case, we have a multi objective selection, where one of them is maximizing the cycle saving while the other one is minimizing the error rate. In this work, we use the greedy selection algorithm to select the CIs. Therefore, we modify the conventional merit function which is based on the cycle saving (*CS*) to a merit function which assigns higher merit value to the CIs with higher *CS* and smaller *ER* values. Therefore, the proposed merit function is formulated as

$$CIG_i.merit = \frac{\sum_{j=1}^{\#\ of\ CIs\ in\ CI\ Group\ i} CI_j.CS}{1 + \sum_{j=1}^{\#\ of\ CIs\ in\ CI\ Group\ i} CI_j.ER} \quad (1)$$

where $CIG_i.merit$ is the merit value of the ith CI group, while, $CI_j.CS$ ($CI_j.ER$) shows the cycle saving (error rate) of the jth CI in the ith CI group. During the greedy selection algorithm, the merit value of each CI is calculated, and the CI group with the highest merit value is selected. Next, the CI groups based on the conflict with the selected CI group are pruned. This process is continued until the CI candidate set becomes empty.

III. Experimental Results

To study the efficacy of the proposed design flow, we have extracted the custom instructions of the Discrete Cosine Transform (DCT), and inverse DCT (iDCT) which are used in image processing applications especially in lossy video and image compressing algorithms. To determine the impact of the imprecise DCT and iDCT components on the output quality of the algorithms, we have applied the imprecise DCT in the MPEG2 encoder while the approximate iDCT have been used inside the MPEG2 decoder. Three video streams have been used as the input of the MPEG2 encoder/decoder video compression algorithms which were bus (150 frames), football (260 frames), and miss America (150 frames). Note that only the custom instructions of the DCT and iDCT parts of the MPEG2 encoder and MPEG2 decoder have been extracted. Also, it should be mentioned that the DCT and iDCT in MPEG2 encoder/decoder take more than half of the runtime of these compression algorithms. The quality of the output videos are reported in Peak Signal to Noise Ratio (PSNR) while the video streams were in YUV color space. To find the approximate CFUs, we have used the proposed design flow in Figure 2. The delay of the CIs has been extracted in 45 nm technology [8]. The baseline processor was a 5-stages in-order pipeline MIPS processor, where its frequency was 666 MHz. All the operations, except of the multiplying, took one cycle, while the multiplying took 2 clock cycles. In this work, only the single cycle CIs have been extracted under the I/O constraint of 2/1. The CIs were identified under 1.5ns *MDP* constraint. TABLE I reports the number of the identified CIs, CI groups, and selected CIs for DCT and iDCT components in the two cases of the conventional (denoted by CONV) and approximate computing (denoted by APXC) ISA extension approaches. As the results show, the number of the identified CIs in the approximate computing approach is more than the conventional approach which leads to selecting CIs with higher cycle savings. The speedups are reported in two different clock periods, 1.5 ns and 1.75 ns. Note that in the case where the clock period is 1.75 ns, the *MPD* constraint during the identification phase was 1.5 ns.

The results show, in the case of the DCT (iDCT), moving the ISA extension design flow, from conventional approach to approximate computing leads to about 460%(70%) speedup improvement. In the case of the DCT, in conventional approach there is only one CI group which satisfied the constraint, while in the approximate computing approach 23 CI groups exist that cycle saving of most of them are higher than the identified CI group in conventional approach. Also, in the selection phase, eight of the selected CI groups where among the approximate CIs which lead to higher speed gain in the approximate computing approach. As an example, one of the approximate CI is $(125 \times In_1) + (106 \times In_2)$, where its propagation delay is about 2.1 ns. This CI was not identified in the conventional ISA extension design flow, while its error rate under all video streams was almost zero. Note that in the case where one of the inputs of a primitive is a constant value, the propagation delay of that primitive after the synthesis to gate level netlist, due to the optimization, is much smaller than the propagation delay of the primitive when both of its inputs are not constant. In the case of the DCT (iDCT), in the conventional approach, by extending the ISA, the number of the instruction fetch was reduced about 12% (12%), while in the case of the approximate computing, it was decreased about 57% (31%).

In the case of approximate computing, the cost of the speed gain is imprecise results, which provides to lower PSNR. TABLE II represents the average of the PSNRs of the video streams under different clock periods, and also under the impact of the process variation. To study the impact of the process variation we have considered channel length (L) and voltage threshold (V_{th}) variations. For both of them, we have assumed a random variation where the σ/μ of it was 10%. Also, the variation was modeled for one hundred CFU sample for each case. Note that the expected PSNR of the bus, football, and miss America video streams was 28.5 dB, 32.4 dB, 43.6 dB, respectively. In the case where the impact of the process variation is not considered, when the clock period is 1.5 ns, the PSNR of the bus, football and miss America output video streams of MPEG2 encoder (decoder) have been decreased about 10% (14%), 13% (10%), and 37% (46%), respectively. While, by increasing the clock period, which is about 16%, the PSNR reduction has been decreased significant.

TABLE I THE NUMBER OF THE IDENTIFIED CIS, CI GROUPS, AND SELECTED CIS FOR DCT AND IDCT COMPONENTS IN THE CONVENTIONAL AND APPROXIMATE COMPUTING APPROACHES. THE SPEEDUP OF THE SELECTED CIS WHERE THE CLOCK PERIOD IS 1.5 AND 1.75 NS.

	# of Nodes	# of identified CIs		# of identified CI Groups		# of Selected CI Groups		Speedup			Speedup Improvement	
		CONV Approach	APXC Approach	CONV Approach	APXC Approach	CONV Approach	APXC Approach	CONV Approach	APXC Approach (1.5ns)	APXC Approach (1.75ns)	APXC Approach (1.5ns)	APXC Approach (1.75ns)
DCT	132	16	62	1	23	1	9	1.05	5.92	5.07	463.8%	383.3%
iDCT	131	26	68	4	19	3	13	1.16	1.8	1.54	71.4%	46.9%

In this case, the PSNR reduction of the bus, football and miss America output video streams of MPEG2 encoder (decoder) have been reduced about 0.1% (0%), 0.2% (0%), 1.7%(0%), respectively. The results show, small increment of the clock period may results in eliminating the PSNR reduction due to the clock period violation. Note that, as aforementioned, increasing the clock period leads to decreasing the speedup that in this study the speedup reduction was about 14.28% (see TABLE I), which may be this reduction acceptable compared to the PSNR improvement.

One of the main challenges in the nano-scale design is process variation which affects the reliability of the circuits. Hence, in this wok, we have studied the impact of the process variation on the extracted approximate CFU. 0shows the impact of the process variation on the PSNR of the MPEG2 encoder and decoder. The results show, in the present of the process variation, the PSNR values of the video streams are not deterministic (see Figure 4). The results depict, in the most cases, the average of the PSNRs was decreased compared to the case when the process variation impact was not considered. Also, by increasing the clock period, the difference between the average and the maximum PSNR was reduced, which shows increasing the clock period may help to reduce the impact of the process variation. As an example, Figure 4 shows the distributions of the PSNRs of the MPEG2 decoder output in the case of the football video stream under two clock periods. The plots show by increasing the clock period, the distributions of the PSNRs are reduced, and they are localized around the maximum PSNR value. This behavior is observable in the other video streams, and also in the case of the MPEG2 encoder. The results show in the present of the process variation, when the clock period is 1.5 ns (1.75 ns), the PSNR may be reduced about 19.7% (28.78%). The PSNR variation in the case when the clock period is 1.5 ns is smaller compared to the case when the clock period is 1.75 ns. This behavior originates from the fact that by decreasing the clock period, the rate of increasing and decreasing the error rate, when the clock period is changed, is reduced (see Figure 1).

IV. CONCLUSION

In this paper, we have suggested to move the conventional extensible processor design flow to the approximate computing domain to gain more speedup. In this work, for the identification phase, we have proposed to identify CIs which do not satisfy the maximum propagation delay but can provide approximate results. Also, in the selection phase, we have proposed a merit function which selects CIs with higher cycle savings and small error rates. To evaluate the proposed design flow, we have extended the ISA of a baseline processor for the DCT and iDCT components of the MPEG2 encoder and decoder, respectively. The results revealed that the approximate computing approach may results in huge speed gain. In the case of the DCT, the speedup improvement was about 460% while the PSNR was reduced about 20%. Also, the impact of the process variation on the impreciseness of the results has been investigated. The results showed that the process variation leads to losing the quality, while its impact may degrade by increasing the clock period.

TABLE II PSNRS (IN DB) OF THE OUTPUT STREAMS OF THE MPEG2 ENCODER AND DECODER WITH AND WITHOUT CONSIDERING THE IMPACT OF THE PROCESS VARIATION

Video Stream	MPD	Without Process Variation		by Considering Process Variation					
		MEPG2 Encoder	MPEG2 Decoder	MEPG2 Encoder			MPEG2 Decoder		
				MIN	AVG	MAX	MIN	AVG	MAX
bus	1.5	25.6	24.4	17.5	25.3	27.1	23.7	24.8	28.4
	1.75	28.5	28.5	18.2	27.2	28.5	25.2	28.2	28.5
football	1.5	28.1	29.1	22.8	24.7	30.0	28.4	29.2	31.0
	1.75	32.3	32.4	29.3	32.2	30.9	29.7	32.1	32.4
miss America	1.5	27.3	23.6	15.4	25.9	40.3	19.1	23.8	43.3
	1.75	42.9	43.6	16.7	40.0	42.9	23.2	41.7	43.6

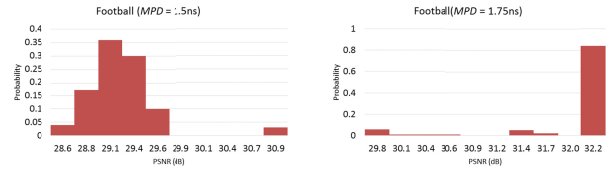


Figure 4. The distributions of the PSNR variation in the case of MPEG2 decoder (in the case of football stream) under process variation

REFERENCES

[1] C. Galuzzi and K. Bertels, "The Instruction-Set Extension Problem: A Survey," *ACM Transactions on Reconfigurable Technology and Systems*, Vol. 4, No. 18, pp.1-28, 2011.

[2] H. Esmaeilzadeh, *et al.,* "Architecture Support for Disciplined Approximate Programming," In *Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2012, pp.301-312.

[3] L. Leem, *et al.*, "ERSA: Error resilient system architecture for probabilistic applications," In *Proceedings of Design, Automation and Test in Europe (DATE)*, 2010, pp. 1560 –1565.

[4] V. Gupta, *et al.*, "IMPACT: Imprecise adders for low-power Approximate Computing," In *Proceedings of IEEE/ACM international symposium on Low-power electronics and design (ISLPED)*, 2011, pp. 409-414.

[5] R. Hegde and N. R. Shanbhag, "Energy-efficient signal processing via algorithmic noise-tolerance," In *Proceedings of IEEE/ACM international symposium on Low-power electronics and design (ISLPED)*, 1999, pp. 30–35.

[6] M. Kamal, *et al.*, "An Architecture-Level Approach for Mitigating the Impact of Process Variations on Extensible Processors," In *Proceedings of Design, Automation and Test in Europe (DATE)*, 2012, pp. 467-472.

[7] L. Pozzi, *et al.*, "Exact and Approximate Algorithms for the Extension of Embedded Processor Instruction Sets", *IEEE Transactions of Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, Vol. 25, No. 7, July 2006.

[8] Nangate 45nm Open Cell Library. http://www.nangate.com/.