AppAdapt: Opportunistic Application Adaptation in Presence of Hardware Variation

Aashish Pant, Puneet Gupta and Mihaela van der Schaar
apant@ee.ucla.edu, puneet@ee.ucla.edu, mihaela@ee.ucla.edu
Department of Electrical Engineering, University of California Los Angeles

***Abstract*—In this work, we propose a method to reduce the impact of process variations by adapting the application's algorithm at the software layer. We introduce the concept of hardware signatures as the measured *post manufacturing hardware characteristics* that can be used to drive software adaptation across different die. Using H.264 encoding as an example, we demonstrate significant yield improvements (as much as 30% points at 0% hardware overdesign), a reduction in overdesign (by as much as 8% points at 80% yield) as well as application quality improvements (about 2.0dB increase in average peak-signal-to-noise ratio at 70% yield). Further, we investigate implications of limited information exchange (i.e. signature quantization) on yield and quality. We conclude that hardware-signature based application adaptation is an easy and inexpensive (to implement), better informed (by actual application requirements) and effective way to manage yield-cost-quality tradeoffs in application-implementation design flows.**

I. Introduction

Variations in manufacturing process are increasingly affecting the performance (speed, power) of systems, both across multiple instances of a design and in time over its usage life. With technology scaling to finer geometry devices, the impact of manufacturing variations is getting worse [2, 3]. For high performance microprocessors in 180nm technology, measured variation is found to be as high as 30% in performance and 20 times in chip level leakage within a single wafer [4]. According to the International Technology Roadmap for Semiconductors (ITRS) [5], this trend is expected to get worse (see Figure 1).

A number of approaches have been proposed to handle the variability associated with the manufacturing process. Most of these approaches statistically model and forecast the effect of variations early in the design flow in an attempt to maximize the expected manufacturing yield, under the constraint that a certain minimum performance level is satisfied [3]. These methods often result in the creation of designs that are high on resources and designer effort. Other techniques like [6, 7] rely on post manufacturing tuning of the hardware. For example, threshold voltage of the gates on the critical path can be lowered after manufacturing in order to make the design run faster (forward body biasing) for slower chips. For extra leaky chips, the threshold voltage can be raised to reduce leakage (reverse body biasing). Often, these techniques require that the designs support requisite tuning knobs, thus making them complex. Moreover, tuning needs to be done on a chip by chip basis and this results in an increased test time. Performance-power optimization techniques like Dynamic Voltage Scaling (DVS) have been used to take process variations into account as in Razor [8].

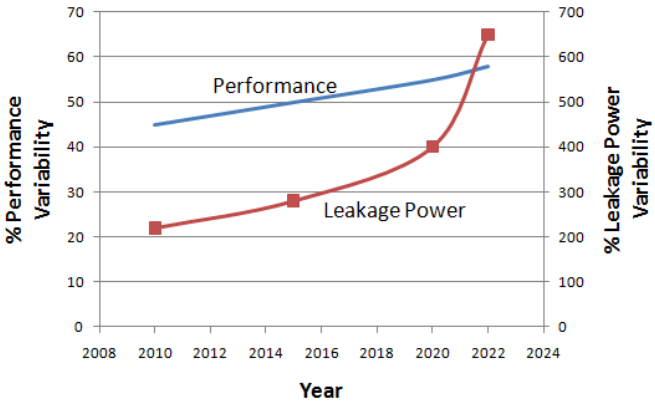

Fig. 1: ITRS Projection of Variability.

While process variability is increasing, the basic approach to designing and operating complex systems has remained unchanged. Software has always assumed the hardware to deliver a certain minimum level of performance, which the hardware designers try hard to meet without leveraging software's flexibility. This rigid hardware-software paradigm coupled with the objective to achieve a good manufacturing yield often leads to systems being overdesigned relative to their specification by addition of certain guardbands. Getting the last bit of performance incurs serious power and area overheads, thus increasing overall design costs. It also leaves enormous performance and energy potential untapped as the rigid software has to assume lower hardware performance than what a majority of the instances of that system deliver. Therefore, there is motivation to think of systems that have a flexible hardware-software interface.

In this paper, we seek to build a flexible hardware-software interface paradigm by proposing the notion of hardware instance guided software adaptation for performance constrained applications. The broad idea is indicated in Figure 2

Preliminary version of this work appeared in [1] and this work is a substantial revision and extension

where the actual hardware state guides application adaptation on a die specific basis. We show that, by adapting the application to the post manufacturing hardware characteristics (hardware signatures) across different die, it is possible to compensate for application quality losses that might otherwise be significant in presence of process variations. This in turn results in improved manufacturing yield, relaxed requirement for hardware overdesign and better application quality.

Our work is motivated by the following two observations:

1) A plethora of modern applications are reconfigurable and adaptive, e.g. video encoding and decoding, multimedia stream mining, gaming, embedded sensing [9] etc. They are capable of operating in various configurations by adapting to certain input or environmental conditions in turn producing similar or different quality of service. This notion can be extended to let variation-affected hardware drive application adaptation.
2) Process variation is increasing and hence, the conventional methods of incorporating variation-resistant design techniques, post manufacturing hardware tuning or hardware overdesign have become expensive [10] and may benefit from being complemented by alternate software-level strategies.

Communication and wireless systems provide an excellent analogy [11]. Communication systems adapt based on the underlying physical communication fabric which is dynamic (for instance [12–14]). Therefore, instead of designing protocols with rigid speed and power constraints, an approach that is flexible and allows for trade-offs is used and it has been proven to be far more effective. In the same way, a system can also adapt to the underlying variation-affected hardware layer.

The idea of modifying the non-hardware layer to suit the underlying hardware (for process variations or otherwise) is not entirely new. In a recent work [15], the authors propose a method to optimize the power management policy of a System-On-Chip (SOC) statistically across all chips taking process variations into account and its effect on leakage power. Further, they suggest approaches to adapt the policy on a chip by chip basis. Software fault tolerance schemes [16] detect hardware faults using methods like Error Correcting Codes (ECC) and correct them on the fly in the software layer. In a recent work [17], a new low power motion estimation framework is proposed in which the supply voltage is purposely lowered, occasionally triggering some timing faults which are then corrected using software fault tolerance techniques. To handle supply voltage variations, some authors [18] have proposed the use of a voltage sensor, error recovery hardware and runtime modification of the compiled software to prevent such voltage variations to get triggered again. Software thermal management techniques [19] perform scheduling in a multitasking scenario to ensure that thermal constraints are met. **Error resilience inherent in applications has also been leveraged to improve defect-limited hardware yield (see [20, 21])**.

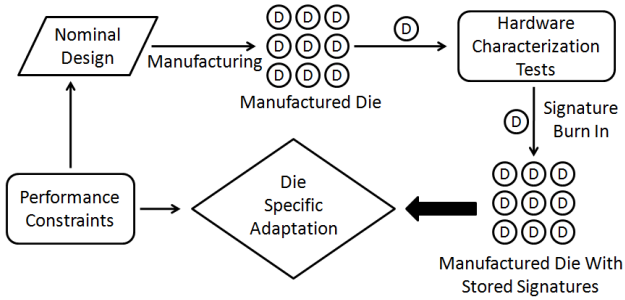

Fig. 2: Proposed Application Adaptation Model.

Most approaches either treat hardware inadequacy or malfunctioning as emergencies by modeling them as transient faults or rely on the inherent error tolerance of specific applications. Moreover, these techniques are employed when the hardware faults happen and some of them require special hardware for correction. **For process variations, software adaptation can utilize the application algorithm's quality or performance tradeoffs to achieve error free operation in the functional sense in presence of permanent manufacturing variations.**

Designing a robust and dependable hardware is indispensable in the presence of manufacturing variations. Statistical design and post silicon tuning significantly help to improve overall manufacturing yield. We believe that incorporating die-specific adaptation at the software layer can ease the burden off expensive robust-hardware design methodologies. This is because adaptation is much better informed of application quality trade-offs at the software layer. In this context, the main contribution of our work is the following

- To the best of our knowledge, this is the first work to discuss application adaptation based on process-variation affected manufactured hardware.
- Using an H.264 encoder, we show that implementing die-specific software adaptation increases manufacturing yield, improves overall application quality and thereby allows for under-design of hardware.
- We consider the implications of limited hardware-software information exchange and die test time by presenting methods to compute optimal signature quantization points.

This paper is organized as follows. In section II, we introduce the concept of hardware signature based adaptation in the context of applications that are performance constrained. In section III, we apply this methodology to an H.264 encoder and demonstrate its benefits. In section IV, we discuss the effects of signature quantization and present an algorithm to compute optimal signature measurement points. We conclude in section V.

II. Hardware Signature Based Adaptation

In this section, we describe the use of hardware signatures for software adaptation in performance constrained applications.

A. Hardware Guided Adaptation: Formulation for Performance Constrained Applications

Consider a system that comprises of an application running on a generic or dedicated hardware. The application can be tuned to run in different software configurations denoted by set $\boldsymbol{S}$. These configurations are associated with varying performance and quality trends. Note that, if the application is not adaptive, $\boldsymbol{S} = \phi$. Also, in this discussion, we assume the hardware to be static but the idea can easily be extended to reconfigurable hardware.

At this point, it might be worth noting that most applications can be made to support software tuning knobs. Even something as simple as sorting can have a variety of implementations to choose from. As an example, while one implementation requires a lower runtime, another one might have a less memory footprint etc. Our focus in this paper is on multimedia applications which inherently provide ample knobs for adaptation.

Adaptation attempts to find the optimal software operating configuration $c^{opt} \in \boldsymbol{S}$. Note that, the definition of optimality strictly depends on context and will differ from application to application. In our discussion, optimal software operating configuration is one that maximizes output application quality **Q** while satisfying application execution time constraints, ET_{max}. Examples of such systems include but are not limited to audio/video compression applications, gaming, stream mining, graphics processing etc. The notion of quality and configurations depends on the application. For audio/video compressions applications, quality can be the Peak-Signal-to-Noise Ratio (PSNR) of the encoded bitstream and the configurations can be different modes of operation of some block, say motion estimation.

Note that application execution time strongly depends on the underlying hardware characteristics (maximum operating frequency, memory etc.). Conventionally, this dependence is assumed to be implicit and worst-case (or expected post manufacturing) hardware characteristics are used to solve for c^{opt}. Therefore, regardless of the true post manufacturing hardware characteristics, the same c^{opt} is chosen as being optimal for all die. Because of the impact of process variations on hardware performance, post manufacturing hardware characteristics may differ significantly from idealized expectations and also from one die to another (because of die-to-die variations). Hence, the choice of c^{opt} may not be truly optimal for all die. We propose the inclusion of these hardware characteristics into the optimization problem as

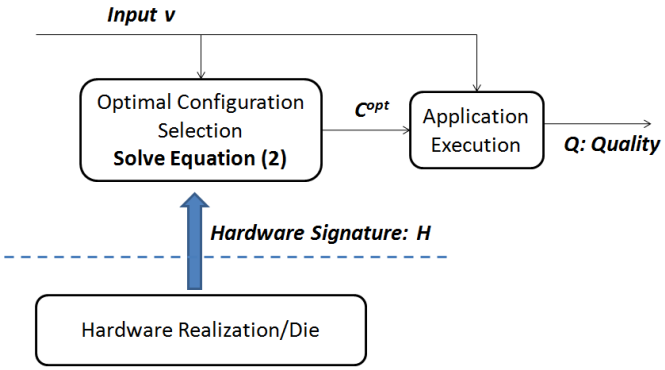

Fig. 3: Hardware Signature Guided Adaptation for Performance Constrained Applications.

$$c^{opt}(v) = \arg\max_{c \in S} Q(c, v) \quad \text{under the constraint that} \quad ET(c, v, h) \leq ET_{max} \tag{1}$$

In this equation, **v** is the input to the application, **c** is an operating configuration, **Q** is the quality and ET is the execution time that depends on the configuration **c** and the input **v**. **h** represents the hardware characteristics.

If the underlying hardware consists of more than one functional blocks or more than one independently fabricated components, each block can be affected by process variations in different ways (because of within-die process variations). The hardware characteristic **h** should therefore include the state of every functional block. Consequently, an application can knowledgeably adapt and redistribute the effort of computation among the hardware functional blocks to achieve the same desired performance given the manufactured hardware. A die that does not currently satisfy the performance constraint can be made usable by adapting the operating configuration to give the same performance at a small tolerable loss in output quality using Equation 1.

B. Hardware Signatures: Representing True Hardware Characteristics

Equation 1 assumes that the application is aware of the exact hardware characteristics on a die specific basis. We call these die specific hardware characteristics made known to the software application as $hardware\ signatures$. Hardware signatures are potentially different for different die and different functional blocks within the same die.

Choice of signature content depends on the particular system objectives. For systems that pose strict constraints on timing (real time applications), signature could comprise of the maximum operating frequency of individual functional blocks of hardware. System memory along with speed of the CPU-memory interface can be an important metric to include if memory intensive and computation intensive techniques are choices for application configuration. Indeed, exploiting

space-time tradeoff has been a major focus of research in algorithms. By knowing the exact frequency and memory characteristics of the hardware at hand, these algorithms can make decisions optimal for that particular hardware. For systems where low power operation is a concern, the exact value of leakage power and maximum switching current are valid signatures contents. Knowing the exact values of leakage and switching power can aid power management policies like $Dynamic\ Voltage\ and\ Freqeuency\ Scaling$ $(DVFS)$ to make optimal die specific power-performance trade-offs. High leakage variability [4] indicates tremendous potential for power savings through adaptation.

Hardware signatures can be measured once post-fabrication and written into a non-volatile software readable[1] memory element on-chip or on-package. Signature characterization can be done in software as well with some hardware support (e.g., mechanisms to detect errors and control frequency). This is likely more expensive though with a benefit of runtime characterization. Well-known parametric tests such as FMAX (performance) and IDDQ (leakage power) can yield such signature values. Signatures can also be measured at regular intervals during system operation to account for ambient voltage/temperature fluctuations and wearout mechanisms such as Time Dependent Dielectric Breakdown (TDDB) and Negative Bias Temperature Instability (NBTI). At-speed logic and memory built-in self test (BIST) techniques can be employed for faster and any time computation of such signatures. Approximations using on-chip monitors (e.g., ring oscillators or monitors such as [23]) can work as well. Since signature measurement involves using test techniques with well understood overheads, in this work we do not discuss these methods in more detail.

C. Q-C Plot and Modeling Hardware Signatures

The behavior of a performance constrained application can be represented by a *Quality-Complexity (Q-C) plot* [24–26] (see Figure 4). Every valid operating configuration **(for a fixed input, process and environmental condition)** c is represented by a point (x_c, y_c) on the Q-C plot, where the Y-coordinate (y_c) represents Quality $(Quality(c))$ and X-coordinate (x_c) represents execution time of the application $(ET(c))$ in that configuration. Operating configurations with larger execution times **(under constant input, process and environmental condition assumption)** are usually associated with higher quality as the application gets more time to process its input and therefore, can do a better job.

For illustrative purposes, we do not show the dependence on input **v**. Various recent works deal with the problem of capturing input dependence. Classification and machine learning [27, 28] is a recent example in which a set of training data is first computed by executing the application on various kinds of inputs and operating environments. Subsequently,

[1]Most modern chips already contain several such EEPROM or NVRAM components for storing hardware IDs, time, etc (e.g., see [22])

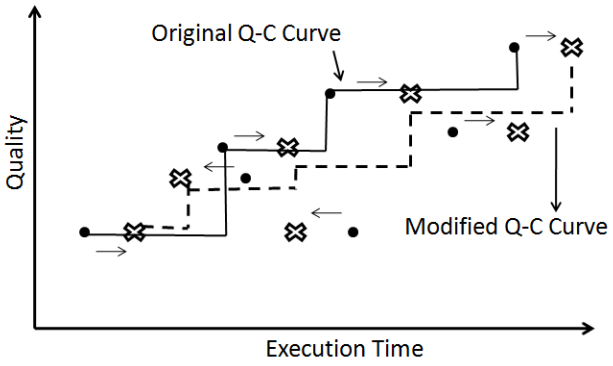

Fig. 4: Q-C plot Changes with Hardware.

relevant and easy to compute features are extracted from this training data. At runtime, input features are matched to the features computed offline. Various other ad-hoc solutions [29, 30] have been proposed in the same area. This is a well researched topic and is not the focus of this work. We urge the interested reader to refer to [27, 28] for details. **We also assume constant environmental conditions for our analysis.**

The behavior of the system as formulated in Equation 1 can be translated to the problem of finding the configuration with maximum quality that lies to the left of the vertical line $x = ET_{max}$, where ET_{max} is the application tolerated execution time constraint for the system. Therefore, over the range of such execution time constraints, the optimal operating points (the points of quality upperbound) form an envelope or a Q-C curve. Note that these operating configurations are discrete and not continuous. Therefore, a particular operating configuration will be optimal for a range of execution time constraints. A Q-C curve will therefore typically look like a staircase function.

The Q-C plot implicitly depends on hardware state. By making this dependence explicit as formulated in Equation 1, every die will have its own Q-C plot. Specifically, an application configuration will have different execution time $(ET(c))$ for different die depending on the process variation scenario, i.e. the configuration undergoes a horizontal shift in position on the Q-C plot. Therefore, the envelope or the operational Q-C curve also changes. The magnitude of the configuration point shift on the Q-C plot depends on the relative contribution of various constituent functional blocks in that application configuration and the magnitude of process variations for each of these functional blocks. Figure 4 demonstrates this Q-C curve change.

Hardware signatures make the application aware of such die specific Q-C plot perturbations. By knowing the exact die specific Q-C curve, the application is better equipped to make optimal c^{opt} selections. This results in improved manufacturing yield as systems may now successfully operate in die specific optimal configurations instead of being simply discarded for not satisfying the specified performance

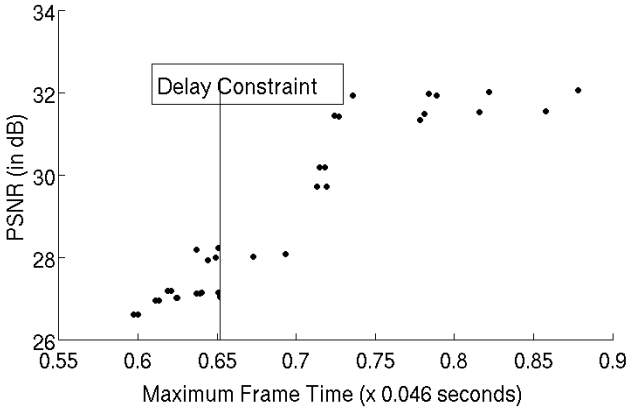

Fig. 5: Operating Configurations for the H.264 Encoder.

TABLE I: Experiment Specifications

Number of Frames	192
ET_{max}	0.03 seconds
Bitrate	800 kbps
Frames per second	33
Frequency Variation	I.I.D Gaussian Distributed Mean=0, 3σ=30% of Nominal Frequency

TABLE II: Encoder Configurations used in Experiments

1	Enable/Disable sub-pixel motion estimation
2	Enable/Disable bi-prediction sub-pel motion estimation
3	FFT transform window size: 8x8, 4x4 or combination of both
4	Run length encoding: CABAC [35] or CAVLC
5	Enable/Disable 8x8, 8x16, 16x8, 8x4, 4x8 motion estimation search window
6	Enable/Disable 8x8, 8x16, 16x8, 8x4, 4x8 bi-prediction motion estimation search window

constraints or minimum quality levels. This also translates to a smaller performance guardband requirement to achieve the same manufacturing yield.

Note that the presence of a quality-performance trade-off is essential for the above methodology to work. There is a large class of modern day applications that fall under this category. For example, video encoding, multimedia stream mining, gaming, embedded sensing [9] are examples of such applications. The class of RMS applications proposed in [31] are all conducive to this kind of trade-off. As shown in the next section, incorporating power into the optimization framework opens up yet another broad class of applications that can benefit from this strategy. Evaluating all such applications is out of scope of this paper. We concentrate on H.264 encoding for our analysis.

III. Proof of Concept: H.264 Encoding

In this section, we apply the proposed adaptation to an H.264 encoding scheme [32, 33]. We assume that motion estimation (M.E), DCT transform (T.X) and entropy coding (E.C) modules are the three independent functional blocks of the encoder. Quality $\mathbf{Q}$ is given by the PSNR of the encoded video. The encoder is required to maximize the output $PSNR$ subject to bitrate (R_{max}) and frame processing delay (ET_{max}) constraints. Please refer to Table I for details. The optimization in Equation 1 can be rewritten as

$$\begin{aligned} & c^{opt}(v) = \arg\max_{c \in S} PSNR(c, v) \\ & \text{under the constraint that} \\ & R(c, v) \leq R_{max} \\ & ET(c, v, h) \leq ET_{max} \end{aligned} \tag{2}$$

In Equation 2, $ET(c, v, h)$ is the sum of the execution times of the three functional blocks and $\boldsymbol{S}$ is the set of all available encoder configurations. Table II shows the various representative H.264 encoder tuning knobs used in our experiments. Please refer to [34] for detail description of these tuning parameters. Operating configurations are obtained by permuting the values of the knobs. These configurations range from algorithmically simple to more complex ones with varying levels of performance and quality trade-offs. Figure 5 shows the Q-C plot for the encoder using the above configuration set[2]. This plot is constructed from the data obtained from profiling the encoder running on a representative video sequence. In our experiments, we use the mobile video sequence because of its strong texture and complex motion content. Note that, from our discussion on input classification, in practical systems, every input type will have its own associated Q-C curve and some online learning technique may be employed to map to the correct input type at runtime. Hardware signatures are taken to be the independent frequency deviations of the three functional blocks (refer Table I).

We also consider and demonstrate improvements for overdesigned hardware in our experiments, where the percentage of overdesign is varied from -20% to +20%. Overdesign provides for a guardband/margin in hardware performance. In other words, the hardware is intentionally designed to achieve a higher performance than is required. This is done to overcome potential degradation in performance due to process variations to regulate manufacturing yield[3]. However, this overdesign has significant penalties in terms of area, power, cost and turnaround time [10]. In our experiments, overdesign (i.e. faster hardware) is handled by relaxing the input processing time constraint ET_{max}.

A. Results and Discussion

In Figure 6, we show the change in encoder PSNR as the operating frequency varies[4]. When encoding is done at nominal frequency (0% frequency variation), both the non-adaptive (red dashed line) and adaptive (blue solid line) cases have the same PSNR. This is because, they are operating in the same base configuration with no frame loss.

[2]In this context, it should be noted that a PSNR difference of 0.5 to 1 dB is significant and is perceivable to the human eye

[3]In this context, a negative value of overdesign simply means an under-designed hardware.

[4]For this analysis, all three hardware components are assumed to have the same variation so that the results can be shown on a 2-D plot.

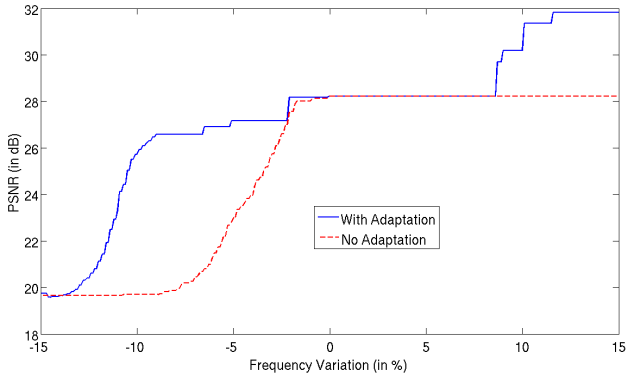

Fig. 6: Hardware Guided Adaptation Improves PSNR (For samples of video sequences encoded using adaptive and non adaptive methods, please see http://nanocad.ee.ucla.edu/Main/Codesign).

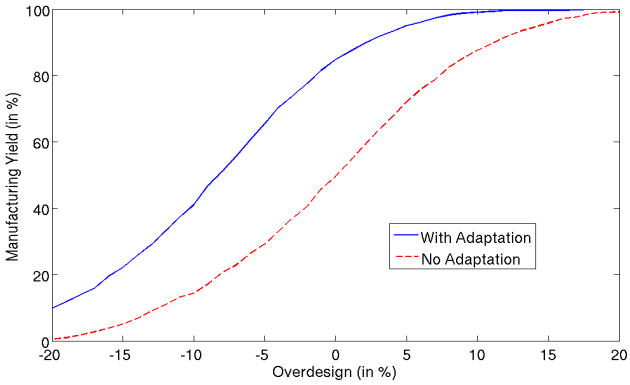

Fig. 7: Hardware Guided Adaptation Improves Manufacturing Yield.

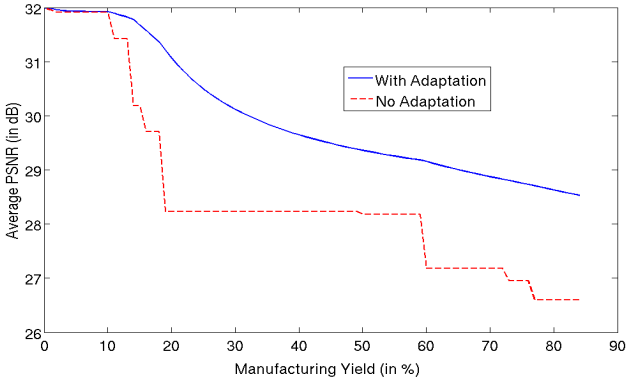

Fig. 8: Hardware Guided Adaptation Improves Overall Application Quality.

As frequency reduces, the PSNR of the non-adaptive encoder differs from the adaptive one. This is because, the non adaptive encoder operates in the same base configuration and starts dropping frames rapidly. Consequently, its PSNR falls sharply[5]. On the other hand, the adaptive encoder tries to adapt to a configuration that ensures maximum quality with no frame loss. Consequently, it is able to achieve a higher PSNR than the non-adaptive case. For example, when frequency decreases from nominal, the adaptive encoder adapts from a configuration with PSNR of 28.28dB to a less complex configuration with PSNR of 28.18dB, thus avoiding frame loss and achieving a better overall quality.

When frequency increases, the adaptive encoder shifts to a more complex configuration (still with zero frame loss) and achieves a PSNR higher than nominal, while the non-adaptive encoder is not able to utilize the faster hardware. Consequently, its PSNR stays flat over the higher frequency range, i.e., hardware-aware adaptation achieves the same desired PSNR with a lower frequency of operation, in turn implying that such a system can tolerate process variations to a greater extent.

We perform Monte-Carlo simulation on 1000 die samples (Table I). The Q-C curve perturbation for every die sample is estimated and optimal operating configuration c^{opt} is found using Equation 2. We plot the results by varying hardware overdesign. Overdesign provides a guardband in performance to counter the effect of process variations after manufacturing. We define manufacturing yield as the percentage of die that undergo no frame loss (i.e., a jitter constraint).

Figure 7 demonstrates significant yield improvements with adaptation. At 0% overdesign, yield of the non-adaptive encoder is 50% (intuitively, half of the manufactured die lie on either side of the nominal hardware under normal frequency distribution). When the encoder adapts according to the manufactured hardware, it operates in a configuration with minimal frame loss and yield increases to 80%. This trend is seen over the entire span of positive or negative overdesign. An important point to observe is that, given enough available configurations (scalable encoding), application adaptation can ensure almost constant quality by trading off work needed for different components. From Figure 7, we can also conclude that hardware-aware adaptation relaxes the requirement of overdesign to achieve the same manufacturing yield. For example, to ensure 80% yield, adaptation relaxes the overdesign requirement by 8%.

Figure 8 shows the variation of average $PSNR$ across all passing die with manufacturing yield for both hardware adaptive and non-adaptive cases. We only show the plot for 0% overdesign, i.e., nominal design as the data for other overdesign values follows the same trend. From the figure, it is observed that adaptation results in a higher average $PSNR$ over the entire range of manufacturing yield[6]. At 70% yield, average $PSNR$ for hardware adaptive case is higher by 2.0dB. For the non-adaptive encoder, increase in yield comes at significant $PSNR$ penalty because the encoder has to ensure a low enough complex configuration (for all die) that satisfies the required yield and hence a staircase $PSNR$ waveform is observed. However, adaptation

[5]We handle lost frames by replacing them with the previously known good frame and computing the output $PSNR$ as is usually done in real time multi-media decoders.

[6]For the adaptive case, the highest quality die are used to match the non adaptive case for the same yield

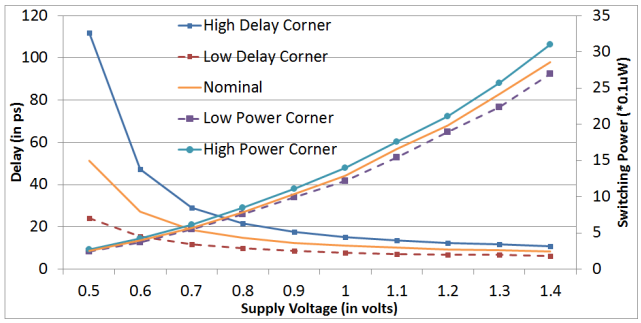

Fig. 9: Variation of Frequency and Power with Supply Voltage Under Process Variations

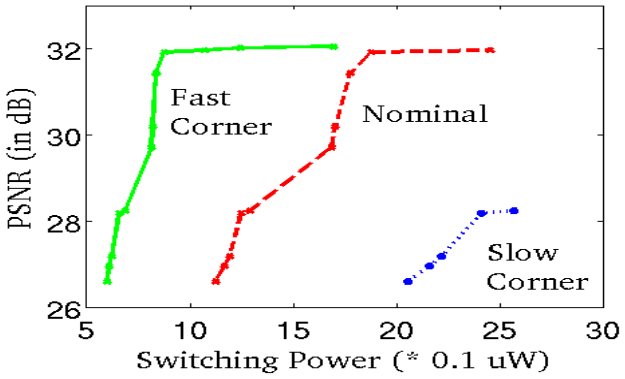

Fig. 10: Variation Space of the PSNR vs. Power Curves for Nominal/Slow/Fast Corners Under Process Variations for H.264 Encoder.

allows for graceful degradation in $PSNR$ when improving yield, as operating configurations can change on a die-by-die basis.

B. DVS: Power and Voltage as Hardware Signatures

In the above discussion, we considered a system where quality (PSNR) was maximized under the constraint that the input was processed within the alloted time. Frequency deviations from the nominal values were the hardware signatures in this case. For energy constrained systems, power dissipation is an important quality metric to include in the adaptation process. Consider Figure 9 which shows the dependence of frequency and switching power[7] on supply voltage for a simple 4 stage FO-4 inverter chain[8] under process variations (varying transistor length and threshold voltage by +-10%) using HSPICE. The curves indicate the nominal and the fast/slow delay/power envelopes. It can be seen that the supply voltage required to achieve the same frequency for different die is significantly different and so is power dissipation, resulting in a wide power-performance band. For example, at supply voltage of 1V, there is a variation of 64% in delay and 16% in switching power across the nominal. By knowing the exact power-performance tradeoff specific to a die, adaptation algorithms like DVS that try to optimize on a combined performance-power-quality metric can do a much better job by adapting in a manner specific to the die. This motivates the inclusion of power as a possible signature metric for such systems.

To estimate the returns that one can expect, we scale supply voltage to achieve the same performance for various sample die affected by process variations. As a result, power consumption of these sample die changes according to Figure 9. Using the Q-C curve of Figure 5, we construct the PSNR vs. power curves of the H.264 encoder for these sample die in Figure 10. Specifically, we show results for the fast corner. slow corner and the nominal. Intuitively, the power consumption of the faster corner is lower because it can operate at a lower supply voltage to achieve same performance. These curves show that different die have different power requirement levels to achieve the same performance (and quality) and this gives us a potential scope of improvement using signature based adaptation.

Hardware signature for such a system will consist of a look-up table that specifies the operational voltage (e.g., a look-up table based method is proposed in [36, 37] to store and track frequency-voltage relationships across process and temperature variations) and power dissipation as well for each frequency of operation. This information will let algorithms like DVS know of the exact operational PSNR-Power curve specific to that die.

IV. Hardware Signature Measurement Tradeoffs

Size (i.e., how many functional blocks and how many parameters per block) and quantization (e.g., discretization of performance into frequency bins) of the signature affects the potential benefit that can be derived from signature-based adaptation. Signature quantization influences storage and more importantly, post manufacturing test complexity. In this section, we focus on determining optimal signature quantization scheme. The problem is very similar to the concept of data compression using quantization in signal theory. Quantization results in an associated distortion during signal reconstruction. The choice of signal quantization levels is therefore very important to minimize distortion. For this analysis, we assume operating frequency as the hardware signature. Therefore, we focus on the problem of determining what frequencies to test (and store as signatures) to ensure minimum quality loss, given the maximum permitted number of such frequency tests.

Consider a hardware system with N independent components. When the system operates in software configuration c, a certain number of average execution cycles are spent in each component. Thus, every configuration c can be

[7]In this analysis, switching power is estimated at constant frequency of operation i.e. the variation in switching power is essentially the same as that of switching energy. This variation is mainly due to change in gate capacitance.

[8]45nm PTM models have been used for these simulations

represented by the load distribution row vector LD^c.

$$LD^c = \left[\begin{array}{cccc} m_1^c & m_2^c & .... & m_N^c \end{array} \right]$$

where m_k^c are the number of execution cycles spent in component *k* when executing in configuration *c*. Let t_k^c be the time spent in component *k* and t^c be the total input processing time when the system is operating in configuration *c*. We have

$$\sum_{k=1}^{N} t_k^c = t^c$$

Because of frequency quantization, let X_k^i be the i^{th} frequency quantization point ($i \in 1..s_k$) for component *k* ($k \in 1..N$) which need to be determined. Further, assume that

$$X_k^i < X_k^j \text{for } i < j \text{ and } k \in 1 \text{ to } N \tag{3}$$

For some die, let the maximum frequency of component *k* for $k \in 1..N$ be quantized to $X_k^{i_k}$. When the component operates at this frequency, then $t_k^c = \frac{m_k^c}{X_k^{i_k}}$, and therefore

$$\begin{aligned} t^c = & \sum_{k=1}^{N} \frac{m_k^c}{X_k^{i_k}} \\ = & LD^c \times p \end{aligned}$$

where ***p*** is the quantized hardware signature of the die and is given by,

$$p^T = \left[\begin{array}{cccc} \frac{1}{X_1^{i_1}} & \frac{1}{X_2^{i_2}} & .... & \frac{1}{X_N^{i_N}} \end{array} \right]$$

The optimal configuration is the one which meets the input processing time constraint and has the maximum output quality (refer Eq 1). The input time constraint is met when,

$$t^c = LD^c \times p \leq ET_{max}$$

The output quality of the application executing on the die is therefore,

$$Q_{die} = \max_{c \in S}(Q^c \times u(ET_{max} - (LD^c \times p)))$$

where Q^c is the output quality when the application operates in configuration *c* and $u()$ is the standard unit step function. Note that there are a total of $\prod_{k=1}^{N} s_k$ possible quantized signatures. Let these quantized signatures be denoted by set ***P***. At this point, it is helpful to visualize the sampling process as an *N*-dimensional space where an axis corresponds to the cycle time of a unique component. The signature set ***P*** is therefore represented by a set of points which divide the *N*-dimensional space into *N*-dimensional hyper-rectangles. The hyper-volume of such an *N*-dimensional hyper-rectangle encompasses all those die that will be quantized to one signature. This is shown in Figure 11 for two components. Note that the frequencies of component 1 and 2 are quantized at 3 points giving a total of 9 signature quantization points. Every configuration (c_1, c_2 and c_3 in the figure) is represented as a slanted line, $x^c + y^c = ET_{max}$. Therefore, all die that

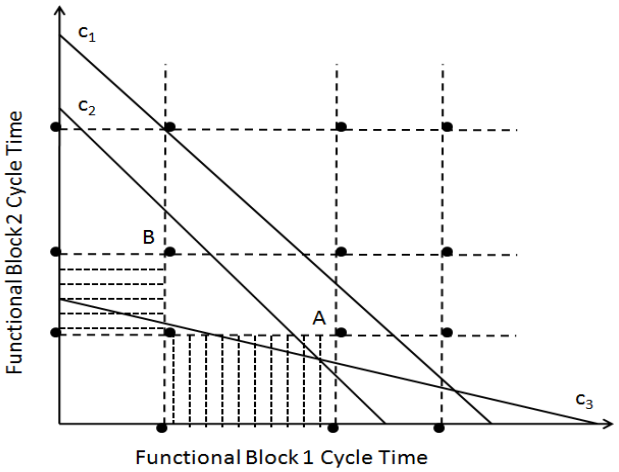

Fig. 11: 2 Component Signature Quantization.

lie to the bottom left of a configuration line can operate at that configuration to meet performance constraints. The vertically (horizontally) shaded hardware space in the figure will be quantized to the signature on the upper right corner. i.e. point A (B). Hence, while the hardware enclosed in the vertically shaded region can only operate at c_1, those that belong to the horizontally enclosed region will operate at $\max(Q(c_1), Q(c_2))$.

Let $f(x_1, x_2, ..., x_N)$ be the joint probability distribution function of the frequency variations of the hardware components. For signature $\boldsymbol{p} \in \boldsymbol{P}$, let V_p denote the $probability\ weighted$ hyper-volume of the *N*-dimensional hyper-rectangle that is quantized to ***p***.

$$V_p = \int_{X_1^{i_1-1}}^{X_1^{i_1}} ... \int_{X_N^{i_N-1}}^{X_N^{i_N}} f(x_1, ..., x_N) dx_1 ... dx_N$$

If the frequency variations of the hardware components are independent, then

$$V_p = \prod_{k=1}^{N} \int_{X_k^{i_k-1}}^{X_k^{i_k}} f(x_k) dx_k$$

For a given signature quantization scheme, the total quality benefit of having a signature at point ***p*** is given by $V_p \times Q_p$. For the most optimal signature quantization, this quality needs to be maximized. Therefore, the signature quantization problem can be formulated as an optimization problem.

$$\text{Maximize QuantGain} = \sum_{p \in P} V_p \times Q_p$$

where

$$\begin{aligned} & V_p = \int_{X_1^{i_1-1}}^{X_1^{i_1}} ... \int_{X_N^{i_N-1}}^{X_N^{i_N}} f(x_1, ..., x_N) dx_1 ... dx_N \\ & Q_p = \max_{c \in C}(Q^c \times u(ET_{max} - (LD^c \times p))) \\ & X_k^i < X_k^j \text{for } i < j \text{ and } k \in 1 \text{ to } N \end{aligned} \tag{4}$$

The above formulation is very similar to that of scalar quantization in multiple dimensions commonly encountered in signal compression theory. The problem is to determine the signature quantization values, given the maximum number of such values so that expected quality is maximized. However, the expression for quantization error in Equation 4 is significantly different from the standard minimum mean square (MMSE) quantization error formulation in signal compression theory.

Equation 4 is a generic optimization problem and therefore, various ad-hoc techniques can be employed. For our experiments, we use the cyclic coordinate descent approach to solve the optimization problem in Equation 4. Specifically, we employ an iterative strategy, where at each iteration step, we determine the best location of one signature quantization point, given the location of all other signature quantization points. We perform this analysis for all quantization points. We iterate until the quality benefit of performing another iteration is less than a certain threshold. The global range of quantization point variation is -3σ to 3σ of frequency variation. Note that, in this process, we arrive at a locally optimal solution that depends on the initial starting point. By repeating the process for different starting points, a sufficient amount of solution space can be covered[9]. In the next section, we will show how the optimal solution can be obtained for the special case of one-component hardware ($N = 1$). At this point, it is worth noting that computational complexity of solving Equation 4 is not critical as deciding on a signature quantization scheme needs to be done just once for a product.

For the Q-C plot of the H.264 encoder (Figure 5), we compare our proposed signature quantization scheme with a uniform quantization scheme[10]. Figure 12 and Figure 13 shows QuantGain and yield loss respectively as the number of signature quantization points per component is varied. Note that our proposed signature quantization scheme results in a higher QuantGain (and therefore a higher PSNR) as well as improved yield over the uniform quantization scheme. Uniform quantization scheme is not able to capture the sensitivity of expected quality (and yield) to the location of the quantization points and hence a rippling behavior is observed as the number of signature quantization points increases. For the proposed scheme, expected quality monotonically increases with the number of signature quantization points.

[9]We believe that for practical problems, this strategy is manageable. Moreover, specific ad-hoc methods can always be employed to perform this iteration efficiently. For example, in our H.264 encoder, we had 3 components and 34 configurations. We carried out the iteration procedure by initially starting with big iterator steps and then reducing the step size gradually. This helps in two ways. It ensures quick convergence near the optimal through big movements in signature quantization parameters when we are searching far away from the optimal solution in the solution space. On getting to the near optimal space, we reduce the iterator step size to fine tune the location of the signature quantization parameters. On a 2.5 Ghz Xeon processor, this analysis took 18 seconds for 5 quantization points per component in MATLAB.

[10]In the uniform quantization scheme, component signature quantization is done at equal frequency intervals lying between -3σ to 3σ of frequency variation

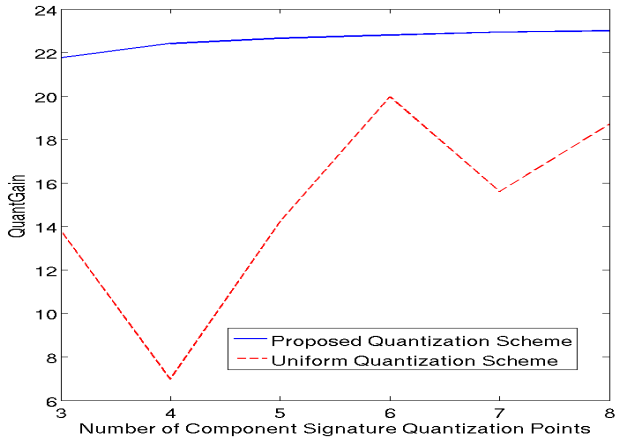

Fig. 12: QuantGain Vs. Total Number of Signature Quantization Points Per Component for 0% Overdesign.

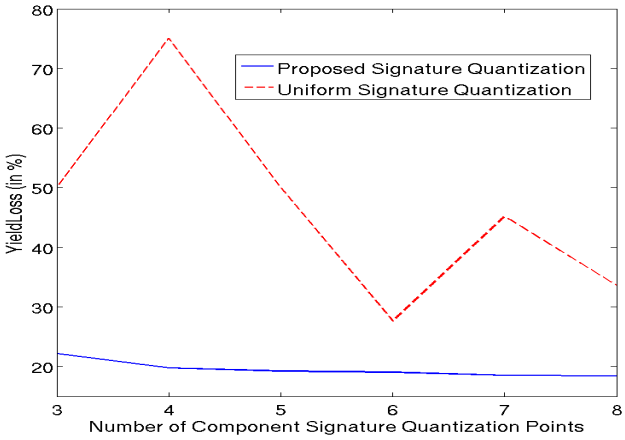

Fig. 13: Yield Loss Vs. Total Number of Signature Quantization Points Per Component for 0% Overdesign.

Note that, if the probability distribution $f(x_1, x_2, ..., x_N)$ is not known in closed form, Lloyd's algorithm [38] for vector quantization can be employed to solve the problem using representative hardware samples.

A. *Special Case: One Component Hardware*

It is interesting to think of the signature quantization problem of the previous section in the special case of one component hardware. The objective is to find optimum signature quantization points X^j ($j \in 1..s$) for maximum quality, where s is the maximum number of available quantization points and $|C|$ is the number of configurations. We have dropped the subscript k from the notation because there exists just one component. Note that, $N = 1$ can be plugged into Eq 4 and similar techniques as in the previous section can always be employed. Here, we will analytically solve the signature quantization problem for one component hardware using Q-C curves. This is important as there exists a definite solution to the optimization problem in the one-component case.

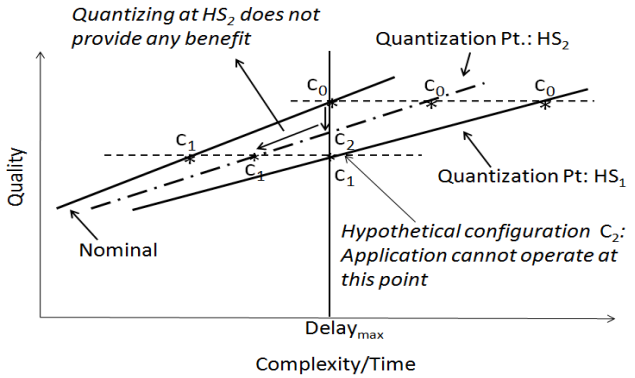

Fig. 14: Signature measurement for One Component Hardware.

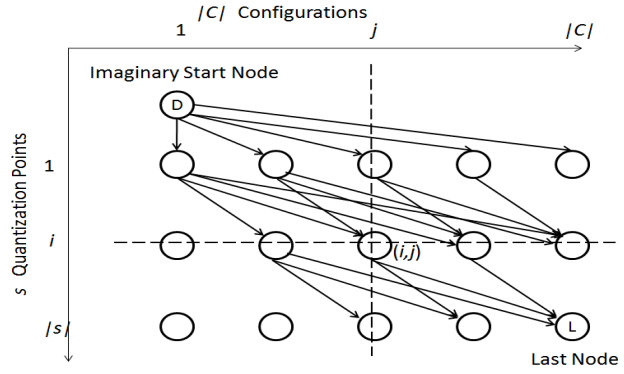

Fig. 15: Shortest Path Approach to Find Optimal Signature Quantization.

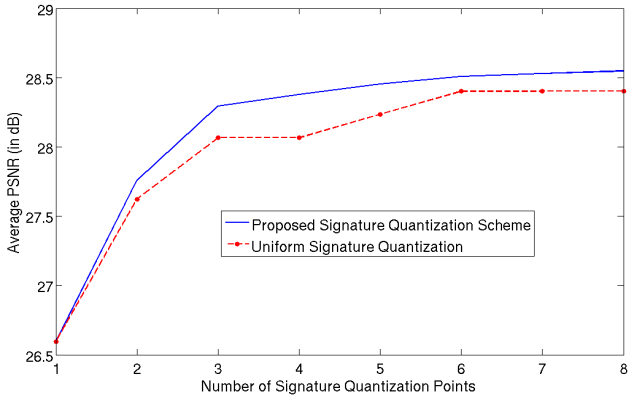

Fig. 16: Improvement in PSNR with finer signature granularity.

We will start by developing an intuition into the solution. Consider Figure 14. c_0 and c_1 are two operating configurations. The Q-C curve for nominal hardware and also for two slower hardware, HS_1 and HS_2 is shown, where hardware HS_1 is slower than hardware HS_2. For HS_2, c_2 (that lies on the ET_{max} line) is not a valid physically existing operating configuration. So, the application has to operate at c_1 for HS_2. For HS_1, c_1 lies on the ET_{max} line and the application operates at c_1. Therefore, HS_2 and HS_1 are equivalent from this perspective. Every die slower than the nominal but faster than HS_1 will operate on c_1. From the above, it makes sense to quantize signature at HS_1, but no additional benefit is obtained by having a quantization point between HS_1 and nominal. This result is important as it limits the potential solution space of the problem.

Therefore, when $s >= |C|$, the optimum location of signature quantization points correspond to those hardware which have their Q-C curves intersecting the ET_{max} line at valid operating configuration points on the Q-C plot. Any additional number of signature quantization points over the number of configurations are redundant and will not improve quality.

When $s < |C|$, a brute-force search technique would require $\binom{|C|}{|s|}$ operations to get to the optimal quantization set. As previously mentioned, computational complexity is not an issue. However, a clever technique that uses graph shortest path algorithm[39] can be used to solve this without brute-force.

Consider Figure 15. Let Q^{c_j} denote the quality corresponding to configuration c_j and let X^j be the corresponding signature quantization location. The number of nodes in the graph is $|C| \times s$ (arranged as a matrix) and the cost of an edge from node $(i1, j1)$ to $(i2, j2)$ ($cost^{(i2,j2)}_{(i1,j1)}$) is the quality loss incurred by having signature quantization point at configurations $j1$ and $j2$ and no quantization point between them (note that all nodes in column j have same quality Q^{c_j} and $Q^{c_{j1}} > Q^{c_{j2}}$ for $j1 < j2$). If *f(x)* is the probability distribution of the frequency variations of the hardware, then

$$cost^{(i2,j2)}_{(i1,j1)} = \begin{cases} \infty & \text{if } j2 \leq j1 \\ \infty & \text{if } i2 \neq i1 + 1 \\ \sum_{l=j1+1}^{j2} (Q^{c_l} - Q^{c_{j2}}) \int_{X^{l-1}}^{X^l} f(x)\, dx, & \text{otherwise} \end{cases}$$

Every path from node *D* (imaginary node corresponding to having a quantization point at ∞) to node *L* (last signature quantization location corresponding to the maximum tolerable variation) will consist of s nodes. The quality loss minimization problem maps to finding the shortest path from *D* to *L*. Nodes in the path correspond to the quantization points.

We perform this analysis for the Q-C curve of the H.264 encoder shown in Figure 5 for different values of s and the results are compared to a naive signature quantization approach, where quantization is done at uniform intervals. From Figure 16, it can be observed that the proposed signature quantization results in higher PSNR than the uniform quantization approach.

V. Conclusion

In this work, we have built on the notion of a flexible hardware-software interface by proposing the use of hardware instance guided software adaptation for performance constrained applications. With increasing variability, there is a need to shift from the basic approach of designing and

operating complex system with a rigid hardware-software interface. With more and more applications being adaptive by nature, we show that variation-aware software adaptation can ease the burden of strict power-performance constraints in design. Hardware signatures (once measured and stored) can be used to guide software adaptation to handle variability on a die by die basis. For an H.264 encoder, we illustrate that this approach can lead to an improvement in manufacturing yield, relaxed requirement for overdesign and an overall better application quality. Specifically, for the H.264 encoder

- Manufacturing yield improves by 30% points at 0% overdesign.
- For an objective yield of 80%, adaptation relaxes the need for overdesign by 8%.
- Encoding quality is better by 2.0dB over the non adaptive case with an objective yield of 70%.

We discuss the implications and cost of signature test and present methods to quantize signatures in an optimized way. We do this analysis for a generic multi-component hardware and then discuss the special case of one-component hardware. Overall, we believe that adaptation is better informed of application quality tradeoffs at the application layer rather than the hardware layer. Therefore, it is easier and cheaper to implement adaptation at the software layer as compared to designing a robust and dependable hardware.

We plan to extend, implement and show the improvements of this methodology for various other application scenarios. Dynamic voltage scaling is a potential application as was briefly hinted in this paper. Further, we will investigate signature based adaptation policy perturbations in already adaptive applications. **In future, it would also be interesting to compare hardware-level variation mitigation approaches with proposed opportunistic software approaches.**

REFERENCES

[1] A. Pant, P. Gupta, and M. van der Schaar, "Software adaptation in quality sensitive applications to deal with hardware variability," in *Proceedings of the 20th symposium on Great lakes symposium on VLSI*, GLSVLSI '10, (New York, NY, USA), pp. 85–90, ACM, 2010.

[2] Y. Cao, P. Gupta, A. Kahng, D. Sylvester, and J. Yang, "Design sensitivities to variability: extrapolations and assessments in nanometer vlsi," in *ASIC/SOC Conference, 2002. 15th Annual IEEE International*, pp. 411 – 415, 2002.

[3] S. Nassif, "Modeling and forecasting of manufacturing variations," in *Design Automation Conference, 2001. Proceedings of the ASP-DAC 2001. Asia and South Pacific*, 2001.

[4] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture," in *Proceedings of the 40th annual Design Automation Conference*, DAC '03, (New York, NY, USA), pp. 338–342, ACM, 2003.

[5] "Process Integration, Devices and Structures," Technical Report, International Technology Roadmap for Semiconductors, 2007.

[6] J. W. Tschanz, J. T. Kao, S. G. Narendra, R. Nair, D. A. Antoniadis, A. P. Ch, S. Member, and V. De, "Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage," *IEEE Journal Of Solid-State Circuits*, vol. 37, pp. 1396–1402, 2002.

[7] S. Sen, V. Natarajan, R. Senguttuvan, and A. Chatterjee, "Pro-vizor: process tunable virtually zero margin low power adaptive rf for wireless systems," in *Proceedings of the 45th annual Design Automation Conference*, DAC '08, (New York, NY, USA), pp. 492–497, ACM, 2008.

[8] D. Ernst, N. S. Kim, S. Das, S. Pant, T. Pham, R. Rao, C. Ziesler, D. Blaauw, T. Austin, T. Mudge, and K. Flautner, "Razor: A low-power pipeline based on circuit-level timing speculation," in *Proc. 36th Intl Symp. Microarchitecture, IEEE CS Press, 2003*, pp. 7–18, Computer, 2003.

[9] L. Wanner, C. Apte, R. Balani, P. Gupta, and M. B. Srivastava, "A case for opportunistic embedded sensing in presence of hardware power variability," in *Proceedings of the 2010 Workshop on Power Aware Computing and Systems*, HotPower '10, 2010.

[10] K. Jeong, A. B. Kahng, and K. Samadi, "Quantified impacts of guardband reduction on design process outcomes," in *Proceedings of the 9th international symposium on Quality Electronic Design*, (Washington, DC, USA), pp. 790–797, IEEE Computer Society, 2008.

[11] N. Shanbhag, "A mathematical basis for power-reduction in digital vlsi systems," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 44, pp. 935 –951, Nov. 1997.

[12] P. Bhat, V. Prasanna, and C. Raghavendra, "Adaptive communication algorithms for distributed heterogeneous systems," in *High Performance Distributed Computing, 1998. Proceedings. The Seventh International Symposium on*, pp. 310 –321, July 1998.

[13] S. Sampei, S. Komaki, and N. Morinaga, "Adaptive modulation/tdma scheme for personal multimedia communication systems," in *Global Telecommunications Conference, 1994. GLOBECOM '94. Communications: The Global Bridge., IEEE*, 1994.

[14] X. Qiu and K. Chawla, "On the performance of adaptive modulation in cellular systems," *Communications, IEEE Transactions on*, vol. 47, pp. 884 –895, June 1999.

[15] S. Chandra, K. Lahiri, A. Raghunathan, and S. Dey, "System-on-chip power management considering leakage power variations," in *Proceedings of the 44th annual Design Automation Conference*, DAC '07, (New York, NY, USA), pp. 877–882, ACM, 2007.

[16] G. A. Reis, J. Chang, N. Vachharajani, R. Rangan, and D. I. August, "Swift: Software implemented fault tolerance," in *In Proceedings of the 3rd International Symposium on Code Generation and Optimization*, pp. 243–254, 2005.

[17] G. V. Varatkar and N. R. Shanbhag, "Energy-efficient motion estimation using error-tolerance," in *Proceedings of the 2006 international symposium on Low power electronics and design*, ISLPED '06, (New York, NY, USA), pp. 113–118, ACM, 2006.

[18] V. Reddi, M. Gupta, M. Smith, G. yeon Wei, D. Brooks, and S. Campanoni, "Software-assisted hardware reliability: Abstracting circuit-level challenges to the software stack," in *Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE*, pp. 788 –793, 2009.

[19] J. Choi, C.-Y. Cher, H. Franke, H. Hamann, A. Weger, and P. Bose, "Thermal-aware task scheduling at the system software level," in *Proceedings of the 2007 international symposium on Low power electronics and design*, ISLPED '07, (New York, NY, USA), pp. 213–218, ACM, 2007.

[20] M. Breuer and H. Zhu, "An illustrated methodology for analysis of error tolerance," *Design Test of Computers, IEEE*, vol. 25, pp. 168 –177, march-april 2008.

[21] I. S. Chong and A. Ortega, "Hardware testing for error tolerant multimedia compression based on linear transforms," in *Defect and Fault Tolerance in VLSI Systems, 2005. DFT 2005. 20th IEEE International Symposium on*, pp. 523 – 531, oct. 2005.

[22] http://docs.sun.com/source/816-5772-11/funct.html.

[23] S. Mukhopadhyay, K. Kang, H. Mahmoodi, and K. Roy, "Reliable and self-repairing sram in nano-scale technologies using leakage and delay monitoring," in *Test Conference, 2005. Proceedings. ITC 2005. IEEE International*, pp. 10 pp. –1135, 2005.

[24] B. Foo, Y. Andreopoulos, and M. van der Schaar, "Analytical rate-distortion-complexity modeling of wavelet-based video coders," *Signal Processing, IEEE Transactions on*, vol. 56, no. 2, pp. 797 –815, 2008.

[25] Z. He, Y. Liang, S. Member, L. Chen, I. Ahmad, S. Member, S. Member, and D. Wu, "Power-rate-distortion analysis for wireless video communication under energy constraints," *IEEE Trans. Circuits Syst. Video Technol*, vol. 15, pp. 645–658, 2005.

[26] L. Su, Y. Lu, F. Wu, S. Li, and W. Gao, "Complexity-constrained h.264 video encoding," *IEEE Trans. Cir. and Sys. for Video Technol.*, vol. 19, pp. 477–490, April 2009.

[27] M. van der Schaar and Y. Andreopoulos, "Rate-distortion-complexity modeling for network and receiver aware adaptation," *Multimedia, IEEE Transactions on*, vol. 7, no. 3, pp. 471 – 479, 2005.

[28] M. van der Schaar, D. Turaga, and R. Wong, "Classification-based system for cross-layer optimized wireless video transmission," *Multimedia, IEEE Transactions on*, vol. 8, no. 5, pp. 1082 –1095, 2006.

[29] A. Majumda, D. Sachs, I. Kozintsev, K. Ramchandran, and M. Yeung, "Multicast and unicast real-time video streaming over wireless lans," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 12, pp. 524 –534, June 2002.

[30] Q. Li and M. van der Schaar, "Providing adaptive qos to layered video over wireless local area networks through real-time retry limit adaptation," *Multimedia, IEEE Transactions on*, vol. 6, no. 2, pp. 278 – 290, 2004.

[31] P. Dubey, "A platform 2015 workload model recognition, mining and synthesis moves computers to the era of tera," 2005.

[32] "Joint Video Team Reference Software JM 15.0." http://iphome.hhi.de/suehring/tml/.

[33] G. J. Sullivan, S. Member, Ieee, and T. Wiegand, "Video compressionfrom

concepts to the h.264/avc standard," in *Proceedings of the IEEE*, pp. 18–31, 2005.

[34] "H.264/MPEG-4 AVC Reference Software Manual." http://iphome.hhi.de/suehring/tml/JM%20Reference%20Software%20Manual%20(JVT-X072).pdf/.

[35] D. Marpe, H. Schwarz, G. Blttermann, G. Heising, and T. Wieg, "Context-based adaptive binary arithmetic coding in the h.264/avc video compression standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 620–636, 2003.

[36] M. Elgebaly, A. Fahim, I. Kang, and M. Sachdev, "Robust and Efficient Dynamic Voltage Scaling Architecture," *SOC Conference*, 2003.

[37] M. Elgebaly and M. Sachdev, "Variation-aware adaptive voltage scaling system," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 15, pp. 560 –571, May 2007.

[38] S. P. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, pp. 129–137, 1982.

[39] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959. 10.1007/BF01386390.

VI. Response to Reviewers' Comments

We would like to thank the editor, the associate editor and all the reviewers for their constructive comments. We have revised the paper to clarify and address the various issues that the reviewers pointed out.

A. *Response to Reviewer 1's comments*

Thank you.

B. Response to Reviewer 2's comments

1) **Comment:** The additional material from your GLSVLSI paper is less significant than you claimed. The main part is the formulation of multi-component signature quantization, but its importance is not supported by experiments. With multi-component signature quantization, within-die variation can be mostly eliminated and its benefit must be very interesting for readers.

   Response: We thank the reviewer for a careful study of the paper. As the reviewer has pointed out, multi component signature quantization can indeed be very beneficial particularly when the process has a high percentage of within die variations. In our analysis, we have mathematically formulated how multi component signature quantization can be performed by reducing the problem to a generic optimization problem. We have used the proposed quantization method to our H.264 encoder and demonstrated the benefits in Figure 12 and Figure 13. Indeed, the effects of within die variation are reduced. This is easily seen from comparing Figure 12 with Figure 16, where the difference between the red and blue curves is significantly larger for Figure 12 than Figure 16.

   Multi component signature quantization is the main but not the only contribution of this paper. Most of the experiments have been redone to make them more meaningful and realistic. For example, we have incorporated many more configurations of the H.264 encoder in our experiments in Section III-A to make the results more realistic (see Figure 5). We have re-drawn Figure 10 to make it more meaningful and intuitive. In Section IV, we present single component signature quantization as a special case of multi-component case and highlight the differences. Almost all sections have been rewritten to present the idea better. New figures have been added (see Figure 2, Figure 3, Figure 4) for ease of understanding. We hope this helps to address the reviewer's concern.

C. Response to Reviewer 3's comments

1) **Comment:** The paper proposes the concept of software adaptation under process variation with H.264 example. As the author mentioned in the paper, body biasing and voltage scaling have been used to compensate hardware variation. Therefore, it will be better to compare the different schemes as a future work if the author works on the subject, further.
   Response: We thank the reviewer for this suggestion. The reviewer is correct in pointing out that adaptive body biasing and voltage scaling have been used to compensate performance variation, albeit at cost of power increase. Our work tries to address variation problems at the application layer (as opposed to hardware layer) due to its proximity to the end user and it being more flexible. We agree that in future, it would be interesting to compare hardware-level variation mitigation approaches with opportunistic software approaches. We have added this to the future work section.

2) **Comment:** In equation (1), there is no constraint on power. It has only delay constraint. So, in figure 6, PSNR increases for +15% frequency variation case. However, +15% frequency variation can cause more power consumption. In that case, software adaption with delay and power co-constraints will guarantee more robust operation.
   Response: We thank the reviewer for this comment. We agree with the reviewer that future extensions of our work can include power and performance both as objectives or constraints. In this paper, we have taken execution time constrained multimedia as a first prototypical application to illustrate the hardware-signature based software adaptation. In Section III-B, we have provided a brief explanation and estimated the gains that can be obtained in a performance-power-quality tradeoff scenario. In fact, one of the co-authors of this work has another recent publication which deals with power hardware-signature based operating system adaptation (see L. Wanner, R. Balani, S. Zahedi, C. Apte, P. Gupta, and M. Srivastava, Variability Aware Duty Cycle Scheduling in Long Running Embedded Sensing Systems, in Design, Automation, and Test in Europe (DATE), March 2011).

D. Response to Reviewer 4's comments

1) **Comment:** This Q-C curve still does not make much sense to me. This is in part because delay and execution time is not totally the same thing. Delay is affected by process variations and environmental variations (temperature), etc. Execution time depends on clock frequency, the task/data size. Clock frequency can be affected by process variation. So execution time does not have a simple and clear dependency on process. Therefore, it is not a correct statement by saying Operating configurations with larger execution time are usually associated with higher quality. (page 5 , section C). Also, $x = Delay_{max}$ for vertical lines has the same issue. Long delays means more timing errors usually. It is very skeptical to think that the quality of the application would increase. I believe the authors need to rethink this curve to deliver their idea.
Response: We thank the reviewer for providing important constructive comments on the paper to make it more useful.
Delay/Execution time comparison: As defined in Section II-C, Delay refers to the application execution time, i.e. it is defined as being synonymous to execution time of the application. It is not the same as critical path delay. Consequently, similar to execution time, Delay depends on task/data size as well as the clock frequency which in turn depends on process variations and environmental variations. However, from the reviewer's comment, the idea of defining Delay as application execution time is not appropriate for our consideration. To address this problem, we have changed all references to the term Delay by execution time and $Delay_{max}$ to ET_{max} throughout the paper as appropriate. In Section II-C, we have also added a couple of additional references [25, 26] that can help to explain the concept of Q-C plots, as used in other related works. We hope this helps to clarify the reviewer's concern and makes the presentation of the concept simpler.
Execution time vs. Quality : As correctly pointed out by the reviewer, execution time of the application depends on 4 factors - the task/data size, operating configuration, process variation and environmental variations. As described in paragraph 2 of Section II-C, we do not show the dependence on the task/data size. Therefore, our definition of execution time assumes constant workload. We also assume nominal environmental conditions. Therefore, in our analysis, execution time changes with operating configuration and process variation. Application quality depends on operating configuration. A Q-C curve plots application quality vs. application execution time for a given process variation. When the process variation is different, we get a different (shifted) Q-C curve. The statement, operating configurations with larger execution times are usually associated with higher quality, is comparing Q-C points for the same Q-C curve (i.e. the same task/data, process variation and environmental variation scenario), which makes sense because, the more the time allowed for an application to process its input, the better will be its quality under the above assumption. To make this point explicit, we have amended our explanation in bold in Section II-C.

2) **Comment:** I believe it is important to show the hardware signature (certain features) has direct relationship with yield. For example, Figure 5 in reference 1. Here, yield depends on fault density. At software level, your task is to reconfigure the software so that you can lower down i.e. fault density (maybe by long execution time i.e.). So I think it makes a lot more sense to put figure 8 in your paper to earlier sections (without the with adaptation result). And then you show in the figure (maybe just a couple of points) that use different configurations (page 7, highlighted area in left column) in details about you can do to improve the PSNR.
Response: We thank the reviewer for creating the context for this interesting discussion. The question of errors/fault density is orthogonal. At the software level, we are not attempting to lower down the fault density by reconfiguring the software (by long execution time etc.). Our objective is to reconfigure the software to achieve correct functional operation (within the maximum permitted time constraint), by possibly compromising a little (tolerable) on the output quality. This improves the yield, as the number of correctly functioning manufactured units increases, i.e. yield in our case is defined in parametric fashion at the application layer. The fault density really does not change. Using hardware signatures, we are ensuring hardware always operates correctly, i.e. error-free (or as correctly as in conventional methodologies) in the functional sense. We understand that this may require some different mechanisms for test and thats why we also show PSNR improvements irrespective of the conventional yield.
To address the reviewer's concern, we have added a brief description of the same in Section I in bold and we have also provided references to the papers mentioned by the reviewer.

3) **Comment:** I also dont think you can achieve error free operations as stated in the paper. This is a very bold statement. You can achieve error tolerant. That is, we still have error in the system and hardware, however, by soft adjustment, the applications can run fine. Here, you may still have error in your application i.e. error rate is not zero, but it is lowered down to a good level.

Response: We thank the reviewer for providing valuable comments. From the previous discussion, using hardware signatures, we are reconfiguring the software to ensure that the system always operates correctly, i.e. error-free (or as correctly as in conventional methodologies) in the functional sense. We are able to do this by utilizing the inherent performance-quality tradeoff of the application. The term error-free in this context does not imply zero fault density. It symbolizes correct funtional behavior (possibly at cost of slightly lower, but tolerable, output quality). To make this point explicit, we have amended the appropriate statement in bold in Section I.

4) **Comment:** Figure 7 and figure 8 have yield related results. But how yield was introduced into the idea was missed in the explanation.
Response: We thank the reviewer for a careful study of the paper. As described above, yield in our case is defined in parametric fashion at the application layer, i.e., it is defined as the percentage of die that satisfy performance constraints, possibly at the cost of slightly lower output quality. In our H.264 proof of concept, this is also stated in Section III-A, "We define manufacturing yield as the percentage of die that ensure no frame loss (i.e. a jitter constraint)". We hope this helps to address the reviewer's concern.