IET Circuits, Devices & Systems

Research Article

IET Journals
The Institution of Engineering and Technology

Timing error detection and correction for power efficiency: an aggressive scaling approach

ISSN 1751-858X
Received on 30th April 2018
Revised 13th August 2018
Accepted on 6th September 2018
doi: 10.1049/iet-cds.2018.5143
www.ietdl.org

Prasanthi Rathnala[1], Tim Wilmshurst[1], Ahmad Kharaz[1] ✉

[1]*Department of Electronics, Computing and Mathematics, College of Engineering and Technology, University of Derby, Derby DE22 3AW, UK*
✉ *E-mail: a.kharaz@derby.ac.uk*

Abstract: Low-power consumption has become an important aspect of processors and systems design. Many techniques ranging from architectural to system level are available. Voltage scaling or frequency boosting methods are the most effective to achieve low-power consumption as the dynamic power is proportional to the frequency and to the square of the supply voltage. The basic principle of operation of aggressive voltage scaling is to adjust the supply voltage to the lowest level possible to achieve minimum power consumption while maintaining reliable operations. Similarly, aggressive frequency boosting is to alter the operating frequency to achieve optimum performance improvement. In this study, an aggressive technique which employs voltage or frequency varying hardware circuit with the time-borrowing feature is presented. The proposed technique double samples the data to detect any timing violations as the frequency/voltage is scaled. The detected violations are masked by phase delaying the flip-flop clock to capture the late arrival data. This makes the system timing error tolerant without incurring error correction timing penalty. The proposed technique is implemented in a field programmable gate array using a two-stage arithmetic pipeline. Results on various benchmarks clearly demonstrate the achieved power savings and performance improvement.

1 Introduction

Owing to the increased use of mobile and portable devices, power consumption has become a major constraint in the design of modern processors. Many techniques ranging from architectural to system level have been developed. Of all these, dynamic voltage and frequency scaling (DVFS) has received more attention over the last two decades. This technique dynamically adjusts voltage and frequency depending on the present task requirements, thereby reduces the power consumption of the system [1]. The technique has been proven ideal to trade-off performance and power consumption of a processor because supply voltage reduction is the ideal way of achieving low-power consumption. Owing to its benefits, DVFS has been used not only to reduce power and temperature but also to improve the security of the systems from power analysis attacks. The traditional DVFS approach scales the supply voltage based on either workload or timing margins available [2]. To guarantee safe operation, voltage and frequency pairs are pre-determined along with workload conditions by considering all process corners. The hardware realisation of this open-loop system uses look-up tables (LUTs) to store voltage and frequency pairs. Since the LUT is pre-loaded with values, the system is not able to adapt to process variations or environmental conditions. Owing to an increase in the process–voltage–temperature variations and advancements in the process technology, the basic DVFS technique has a reduced demand. On the other hand, timing margin-based DVFS is a closed-loop system, which adaptively changes the supply voltage to a minimum level to meet performance requirements by considering the process, voltage and temperature variations. The typical closed-loop system employs a monitoring algorithm to provide application requirements in order to configure a programmable DC–DC converter and a programmable clock generator. In recent years, this technique uses critical path monitors to monitor critical path delay in a process and adjusts the frequency/voltage so as to minimise timing margins during runtime.

According to the available literature, critical path monitoring techniques can be classified into three categories: direct monitoring, indirect monitoring and time-borrowing monitoring. The direct monitors track the actual critical path delay during runtime for detecting and correcting timing errors. This technique is more effective in reducing all the timing margins. However, this approach detects errors after their occurrence and incurs a large timing penalty for error correction. The indirect monitors use critical path replicas (CPRs) that reflect the delay behaviour of the actual critical paths in the design. The design of these CPRs is most critical and it must be as close to the actual delay of the circuit as possible because these are used to generate the feedback signal controlling voltage or frequency of the processor. This technique requires a large timing margin compared with the direct technique because of the difference between actual and predicted paths. The time-borrowing monitors mask timing error by borrowing time, either delaying the arrival time of the correct data to the next pipeline stage or delaying the clock to capture the late arrival data. This technique needs a large checking window to be more effective. The checking window is referred to as the period of time after the clock edge reserved for error detection and masking.

To summarise, the main challenges for the proposed technique are: (i) reducing large timing penalty for error correction, (ii) reducing large timing margins and (iii) having large checking window for error detection and correction. The proposed methodology presents a comprehensive solution to the above three issues in developing an appropriate critical path monitor leading to a power-efficient DVFS system.

The rest of this paper is organised as follows: the background related to the present work is presented in Section 2. The principle of operation of the proposed technique is presented in Section 3. The timing analysis and extension of the technique to the general pipeline architecture are presented in Sections 4 and 5, respectively. The simulation results are provided in Section 6 and the concluding remarks are presented in Section 7.

2 Background

In the recent past, aggressive scaling with critical path monitoring techniques has received considerable attention from industry [3]. Intel and advanced RISC machines (ARM) have initiated research efforts focused at evaluating these techniques for low-power microprocessor cores [4]. In [5], the ARM-based industrial prototype was developed, where razor dynamic adaptation led to 52% energy savings at 1 GHz operation. Still, several challenges

© The Institution of Engineering and Technology 2018

Table 1 Comparison of the proposed technique to other latest techniques

	Indirect (CPR)	Razor	TDTB	Timber	DSTB	Proposed technique
architecture modification	no	yes	yes	yes	yes	less, no need to replace all registers in the pipeline
datapath metastability	no	yes	no	yes	no	no
error correction overhead	no	yes	no	no	no	no
design complexity	high	low	high	high	low	low
timing margin	large	small	small	small	small	small
power overhead	CPM	CPM + additional pipes + recomputing + buffers	CPM	CPM	CPM	CPM + buffers

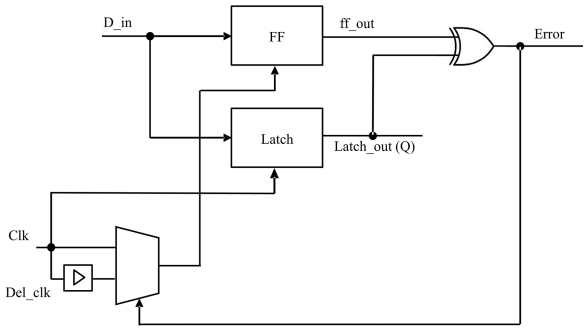

Fig. 1 *Gate-level circuit diagram of the proposed technique*

remain before these techniques are to be widely adopted as a mainstream technology. However, they deliver improved power savings or performance improvements suitable for high-performance processor designs. The first reported technique based on critical path monitors are direct monitors. The best example for this category is razor technique [6]. This technique detects timing error by double sampling the critical path output at different points in time and compares them with each other. Some of the benefits of this technique are highlighted in [6]; however, it poses significant challenges. The technique monitors critical path signals for late arrival data, detects error after the system is corrupted and stalls the current execution to recover from the error. Some of the limitations of razor circuits are listed below: (i) error correction overhead – takes multiple cycles to recover from error; (ii) dependency between checking window and hold time constraint; (iii) susceptibility to datapath metastability – require substantial design overhead [7]. The bubble razor [8] technique addressed these limitations and proposed a new solution using a latch-based design. The latch-based flows have been around for a long time; however, the majority of design flows in the industry rely heavily on the use of edge-triggered flip-flops [9]. In the case of error-free operation, the shadow latch is susceptible to glitches as long as it is open. Hence it is more prone to glitches than flip-flop-based designs. This causes unnecessary activity and consumes more power.

TEAtime [10] is the best example of the indirect critical path monitoring. This method uses a CPR to estimate the timing error before the occurrence of an error. This is achieved by replicating the most critical paths to estimate the behaviour of the path with respect to voltage or frequency variation. The advantages of this method are: (i) errors are detected before they occur and (ii) architecture modification of processor is not required. However, the major difficulty is in the design of replica that closely resembles the actual critical path.

The third classification is based on time-borrowing feature. Many techniques have been proposed related to this including transition detector with time borrowing (TDTB) [11], double sampling with time borrowing (DSTB) [11] and timber [9]. In DSTB, the positions of flip-flop and latch are interchanged to eliminate datapath metastability but the checking window for timing speculation is small compared with razor flip-flop. Timber proposed a new time-borrowing scheme, which detects the late arrival of data and masks them by reducing the arrival time of the correct data to the next pipeline stage [7]. The drawback of timber is the small checking window for timing speculation. In conventional systems, a closed loop is used to change the delay of the flip-flop clock based on timing error on the fly. This will consume more power as the phase of the clock is changing on the fly. Moreover, the clock signal and clock tree are the most power-consuming elements of a digital processor.

In this paper, we propose a new aggressive scaling technique to address the issues of the above-mentioned techniques. The features of the proposed technique are compared with the previously introduced techniques as shown in Table 1. As can be seen from this table, all the techniques mentioned require high area overhead because of the complexity involved in error detection and recovery, except the proposed technique.

The area overhead involved the proposed technique is small compared with the other techniques. In addition to this, it is quite attractive for most of the features listed in Table 1. The summary of our main contributions is: (i) a new technique to reduce power consumption or improve the performance of processor designs, (ii) error correction without timing penalty, (iii) use of phase-delayed clock to borrow time reduces the sequential clock energy overhead in traditional designs and (iv) experimental work by prototyping the design on to a field programmable gate array (FPGA) to validate the effectiveness of the proposed technique.

3 Principle of operation

The aim of the proposed technique is to operate systems beyond worst-case estimates with minimal error correction overhead and timing margins. To reduce excess timing penalty of error correction overhead, a timing monitor unit employing time-borrowing approach is used. Fig. 1 shows a gate-level circuit diagram of the proposed unit. The fundamental idea here is to delay the main clock in such a way that the late arrival data can be captured with time borrowing. The proposed circuit mainly consists of a flip-flop, a latch and a multiplexer. The combination of flip-flop and latch performs double sampling to identify when there is any timing violation as in razor circuit described earlier. However, in this circuit, as in DSTB, the position of flip-flop and latch are interchanged to avail the feature of time borrowing and to reduce sequence overhead. The drawback of DSTB is having a small checking window for timing speculation. The proposed technique overcomes this drawback by adding a multiplexed clock signal to further extend the checking window. Having a larger checking window helps to detect the timing error if the data arrives after the rising edge of the clock signal [12]. The proposed circuit detects timing error by double sampling the data using flip-flop and latch combination. When the data arrives late, the output of flip-flop and latch differ and thereby activates the error signal, which in turn changes the phase of the clock signal to the flip-flop. Now, the flip-flop clock is phase delayed to capture the late arrival data and eliminate the timing error. If the error continues beyond this point, further action needs to be taken to change voltage or frequency.

To reduce the sequential clock energy overhead, both main and delayed clock signals are generated beforehand and only the selection is done on the fly.

3.1 Timing waveform

Fig. 2 shows the timing waveforms of the proposed technique. The main clock and the input data signals are clk and D_in, respectively. The same input data D_in is fed to both the sequential elements flip-flop and latch.

© The Institution of Engineering and Technology 2018

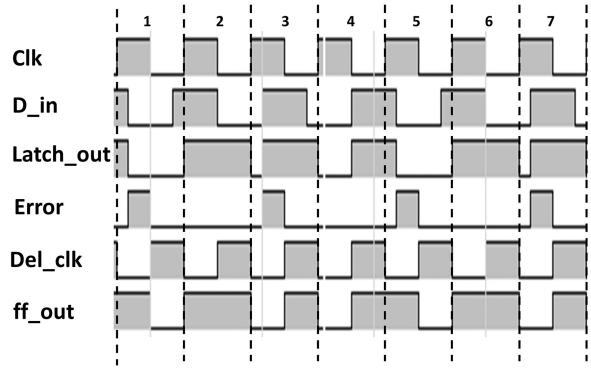

Fig. 2 *Timing diagram of the proposed technique*

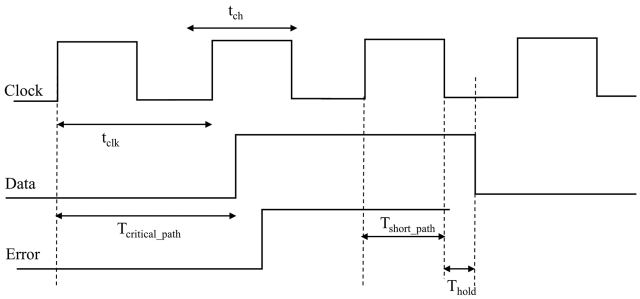

Fig. 3 *Conceptual timing diagram for maximum and minimum delay constraints*

During cycle 2, D_in arrives longer than the setup time and before the rising edge of the flip-flop. This is the ideal case of operation for any digital circuit. The input data is captured correctly by both flip-flop and latch. Hence, there is no error signal during this cycle.

During cycle 3, D_in arrives after the rising edge of the main clock signal clk. As frequency is scaled beyond the worst-case estimate, the input data arrives late [13]. This means that the clock frequency is not sufficient for the critical signal to travel along the path. So the flip-flop missed the data, causing the error signal to be activated. When this happens, the multiplexer selects the phase-delayed clock and flip-flop uses this to capture the late arrival data. The error signal is masked immediately as soon as both flip-flop and latch capture the same data. Since the error is masked, the proposed technique resumes the normal pipeline operation.

4 Timing constraints

To ensure correct functionality, the input data must arrive at a setup time prior to the rising edge of the clock. Sometimes data can arrive after the rising edge due to the worst-case dynamic variation. The proposed technique can be used to ensure the correct functionality. The two main elements in the technique are the flip-flop and latch; both elements are clocked by different clock signals ff_clk and latch_clk. Both the clocks have the same frequency but ff_clk is phase delayed by 180° with respect to the latch_clk. Timing constraints must be analysed thoroughly to safeguard the correct operation of any digital circuit. The main basis for the analysis is taken from [8] for the DSTB circuit and modified further to analyse the benefits of the proposed technique. The timing constraints for the proposed technique are given below.

$t(\text{su}, F)$ = setup time of flip-flop; $t(\text{su}, L)$ = setup time of latch; t_{clk} = clock period; t_{ch} = checking window; t_{pw} = pulse width; t_{pd} = propagation delay; t_{pcql} = latch clock-to-Q propagation delay; and t_{d} = delayed clock

$$t_{\text{ch}} = t_{\text{pw}} + (t(\text{su}, F) - t(\text{su}, L)) \tag{1}$$

Equation (1) ensures that the proposed circuit has a checking window, which detects late arriving data. In general, the input data to be captured properly by the flip-flop must stabilise at least t (su, F) before the rising edge of the clock. Because of the variations, there is a possibility for the late arrival of data. The circuit must have a large checking window to ensure the late arrival data can be sampled by the latch if flip-flop missed it. The checking window should be as wide as possible to eliminate large guard bands. In our case, the clock to the flip-flop is delayed by 180° to capture the late arrival data, which is indicated by an error signal. Therefore, (1) can be modified as shown below:

$$t_{\text{ch}} = \frac{t_{\text{clk}}}{2} + (t(\text{su}, F) - t(\text{su}, L)) \tag{2}$$

From (2), it is evident that the checking window is almost 50% of the clock period which is wide enough to eliminate the excess timing margins. In the case of error, the checking window reduces as in (3) but other times the checking window is large enough to detect timing violations

$$\begin{aligned} t_{\text{ch}} &= t_{\text{pw}} + (t(\text{su}, F) - t(\text{su}, L)) - t_{\text{d}} \\ &= \frac{t_{\text{clk}}}{2} + (t(\text{su}, F) - t(\text{su}, L)) - \frac{t_{\text{clk}}}{2} \\ &= (t(\text{su}, F) - t(\text{su}, L)) \end{aligned} \tag{3}$$

Another important metric in digital circuits is the maximum propagation delay, which is the maximum delay of the circuit under worst-case conditions. This delay helps to hide the flip-flop setup time from the critical path

$$\begin{aligned} t_{\text{pd}} &\le t_{\text{clk}} - t_{\text{pcql}} - (t(\text{su}, F) - t_{\text{d}} \\ &\le \frac{t_{\text{clk}}}{2} - t_{\text{pcql}} - t(\text{su}, F) \end{aligned} \tag{4}$$

In addition to this, two other timing conditions must be met on the circuit behaviour: max-delay and min-delay. The timing diagram for the delay constraints is illustrated in Fig. 3. The critical path exhibiting a maximum delay in the presence of worst-case dynamic conditions. The timing analysis of both delays is followed as in [11] and defined in (5) and (6). If the max-delay constraint is not satisfied, false detections can occur causing timing violations and capturing wrong data. The min-delay constraint depends on short path delay and holds time as defined in (6). The short paths in the design must be padded to have a delay greater than the sum of hold time and the checking period. The max- and min-delay constraints are common issues that all critical path monitoring (error detection) techniques and the proposed technique need to address. The potential hold time violation can be overcome by using additional buffers or clock phase tuned with a duty cycle control circuit [11]. Microprocessors with shallow pipelines greatly relax the min-delay requirements as compared with a deep pipeline design, enabling a more effective trade-off of max-delay improvement for min-delay penalty [11]

$$t_{\text{ch}} \ge t_{\text{critical\_path}} - t_{\text{clk}} + t(\text{su}, F) \tag{5}$$

$$t_{\text{short\_path}} \ge t_{\text{hold}} + t_{\text{ch}} \tag{6}$$

5 Extension to processor architecture

Fig. 4 shows the integration of the proposed technique into the general five-stage pipeline architecture. Unlike the techniques in [1, 6], all the pipeline registers need not be replaced with the new circuit. Only the critical paths that get negative slack, because of the variations, need to be replaced. However, the worst-case situation is shown in Fig. 3. Note that the paths that require time borrowing need to be replaced by the proposed circuit.

The pipeline executes instructions normally when there is no error at any stage of the pipeline. If the operating frequency is scaled beyond the worst-case estimate, timing errors start to occur. The error signal is activated to indicate the occurrence of timing error at each stage of the pipeline. All these error signals are combined to generate a global error signal. As explained in Section 2, timing errors are masked by phase delaying the clock to capture

© The Institution of Engineering and Technology 2018

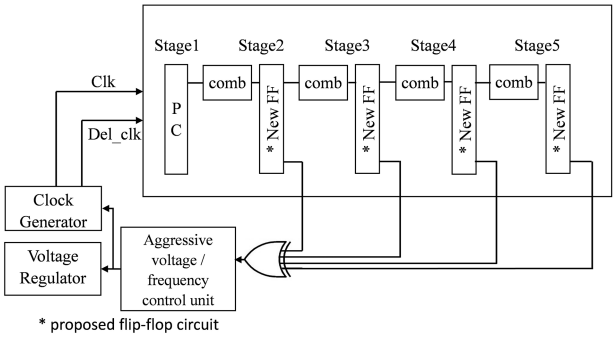

Fig. 4 *Extension to general processor architecture*

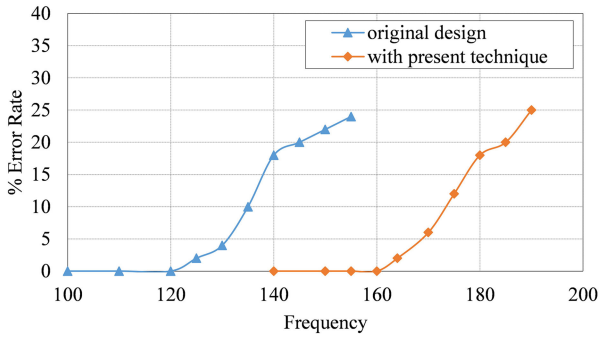

Fig. 5 *Error rate versus frequency for the proposed technique*

the late arrival data. The activation of the global error signal indicates the worst-case scenario of the circuit. This situation shows that the present operating frequency/voltage needs to be altered to perform normal operation. The control unit makes the decision and commands the clock/voltage generator to supply a new set of operating frequency/voltage to recover from the error.

6 Simulation and evaluation

The proposed technique is implemented and analysed on a 90 nm Spartan 3E FPGA development board. The reason for choosing FPGA environment is the flexibility of FPGA architecture to modify circuit post-placements to insert the proposed technique to monitor critical paths post-compilation. Such opportunities are not available in application-specific integrated circuit (ASIC) without significant incremental cost [14]. The technique is validated by implementing a simple application circuit, two-stage pipeline counter and targeted to FPGA. The Spartan 3E FPGA development board has chosen two reasons: (i) ease of frequency tuning and (ii) no need to alter the hardware for evaluating the proposed design. In general, timing error estimation, detection and correction circuits use either frequency or voltage to verify the effectiveness of their techniques. In this case, to validate the proposed technique, the frequency is varied by monitoring the timing error of critical paths. The whole idea here is to evaluate the circuit behaviour with respect to timing error and not the source of error.

The flip-flop clock is delayed by using digital clock manager (DCM) and the latch clock is set to the base clock frequency. Since the clocks have the same frequency, the base clock frequency is set as the main clock frequency and the DCM is used to provide the required phase difference to the flip-flop clock. We implemented the proposed technique in a two-stage arithmetic pipeline. The first stage of the pipeline consists of two counters with 16 bit output. The second stage of the pipeline uses 32 bit input exclusive-OR tree to output the synchronised counter output. The design is simulated, synthesised and targeted to Xilinx Spartan 3E FPGA. The potential critical paths are identified with the static timing analysis. For these critical paths, we replaced 9% of the original flip-flops in the netlist with our proposed technique.

6.1 Experimental methodology

To evaluate the effectiveness of the proposed unit, a control scheme presented in [15] is used to switch clock frequency between the worst-case clock frequency F_{min} and the possible over-clock frequency F_{max}. In this case, F_{min} is the estimated worst clock frequency by the timing analysis. The flip-flop clock is phase delayed by 180° with respect to the latch clock. Owing to this delay, the late arrival of data detection can be extended from the rising edge to the negative edge of the main clock signal. This allows the circuit to go beyond the maximum clock frequency. The maximum clock frequency can be extended to 1.5 times the original/estimated worst-case clock frequency. To evaluate the performance of the proposed circuit, three scenarios, as described below, are monitored:

(a) *Normal operation*: A clock frequency was chosen below the worst-case clock frequency. From the static timing analysis, the worst-case clock frequency, in this case, was 120 MHz. The above-mentioned two-stage arithmetic was run for 10,000 cycles to count the number of errors occurred. No errors were found in this case and the circuit executed the normal operation.
(b) *Over-clocking*: A clock frequency was chosen between worst-case clock frequency F_{min} and maximum frequency F_{max}. The same procedure as in the first case was repeated here, to monitor the number of errors that occurred during 10,000 cycles. Few errors occurred, which were masked by the time-borrowing feature of the circuit.
(c) *Aggressive over-clocking*: A clock frequency was chosen just below the maximum frequency F_{max} with the same as above procedure repeated for 10,000 cycles. These errors were masked by using this proposed circuit, without even error correction timing penalty. Thus this circuit permitted aggressive over-clocking beyond worst-case clock frequency without any performance penalties.

6.2 Results

The above procedure was used to evaluate the performance increase of the proposed technique as a function of error rate. Fig. 5 shows the relation between error rate and frequency of the two-stage design with and without the proposed technique. In general, the occurrence of timing errors below the worst-case clock frequency estimated by the computer-aided design timing analysis was very rare. The same can be seen in this figure. The blue line corresponds to the design without the proposed technique; no errors occurred up to about 124 MHz. As the operating frequency was increased beyond this point, the percentage of error rate started increasing.

The same procedure was repeated using two-stage arithmetic with the proposed technique. No errors occurred until the frequency has reached up to 160 MHz. The timing errors occurred as the frequency was tuned beyond worst-case frequency of 120 MHz. However, these errors were masked by the technique, causing performance improvement of around 33%. The percentage of error rate started increasing after 160 MHz. This indicated that critical path timing violations exceeded the maximum time borrowing allowed with the proposed technique.

We simulated the two-stage design with razor technique to evaluate the performance improvement. Fig. 6 shows the results of razor and proposed techniques. As shown in this figure, the performance of the proposed technique is comparable with the razor technique and the existing minor rise in error rate is due to the fixed phase delay of the clock signal. This limitation can be improved by using tunable phase delay element to capture late arrival data and to mask timing errors. However, this has an impact on area overhead and hence the power consumption.

We also simulated different benchmarks to evaluate the performance improvement of the proposed technique. The results of three benchmarks [advanced encryption standard (AES) and multiply and filter (MUL)] are presented in Fig. 7. From the results, it can be seen that the proposed technique provides ~20% performance improvement when compared with the design without

© The Institution of Engineering and Technology 2018

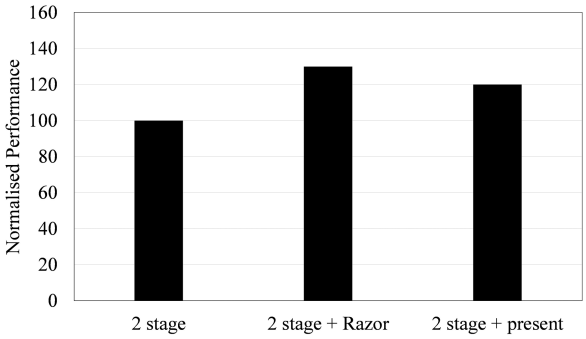

Fig. 6 *Comparison of the proposed technique performance improvement to the original design and razor technique*

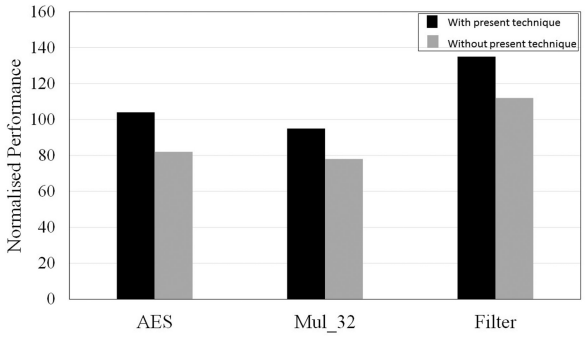

Fig. 7 *Performance improvement of various benchmarks using the proposed technique*

Table 2 Hardware resource utilisation report

	AES	MUL	FILTER
slices	1526	1043	1624
slice flip-flops	1334	63	416
LUTs	6245	229	117

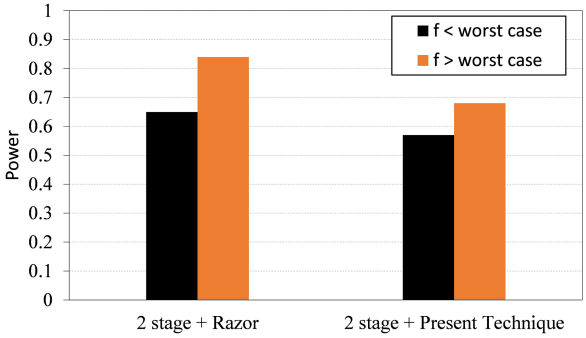

Fig. 8 *Power consumption of razor and the proposed technique*

the proposed technique. The main benefits of this are less area, lower-power consumptions and reduced design overheads compared with other reported techniques. The hardware resource utilisation of all the benchmarks circuits implemented is presented in Table 2.

We evaluated the proposed technique power overhead involved in dealing with error detection and correction. The original two-stage design was run with voltage fixed to 1.0 V and frequency was varied to measure the total power consumption of the design with razor and the proposed technique. One frequency was chosen below the worst-case frequency estimated by the timing analysis and the other frequency was selected to be beyond the worst-case clock frequency. The purpose was to evaluate the power consumption of the proposed technique when there were no timing errors and again in the presence of timing errors. Results of both techniques, power consumption versus frequency are presented in Fig. 8. In both cases, razor technique consumed more power compared with the proposed technique. Although razor technique

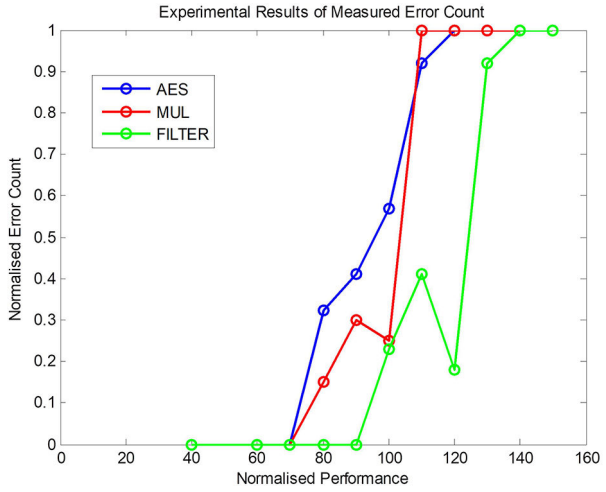

Fig. 9 *Measured error count*

provided a slightly better performance improvements compared with the proposed technique, razor consumes more power before and after timing error occurrences. Razor involves complex error correction mechanisms such as counterflow pipelining or architectural replay that consume more power in the event of errors. They involve stalling the pipeline and flushing the wrong data to recover from the error. These mechanisms are activated when there is an error. However, the complex circuitry still consumes power in the case of no errors.

Fig. 9 represents the measured error count versus frequency for all the benchmark circuits implemented. In this figure, the region up to 60 MHz is error-free zone, no errors occurred due to the positive time slack. Within the region from 60 to 90 MHz, slight errors occurred due to the clock jitter or the discrepancy between the rise and fall times of the critical paths. These errors were masked by the proposed circuit. The region after 90 MHz, the timing violations caused an increased number of errors.

7 Conclusion

In this paper, a novel technique that reduces the power consumption of embedded system has been presented. It reduces the excessive timing margins which result in the reduction of power consumption. This technique corrects the timing error caused by aggressive voltage scaling or frequency boosting by phase delaying the clock. The main benefits of this are less area, lower-power consumption and fewer design overheads compared with other reported techniques. The technique was validated by prototyping on to an FPGA in a Spartan 3E FPGA development board. Results are presented that show both performance and power consumption improvements. Our experiments are based on FPGA platform, an extension is planned to implement in ASIC technology. This work presents an initial exploration of the possibilities of using time borrowing to benefit power savings or performance improvement. The effect of varying supply voltages and the circuit behaviour at near threshold voltages will be carried in the future work.

8 References

[1] Rathnala, P., Kharaz, A., Wilmshurst, T.: 'An efficient adaptive voltage scaling using delay monitor unit'. Ph.D. Research in Microelectronics and Electronics (PRIME), Glasgow, UK, 2015, pp. 109–112

[2] Park, J.: 'Self-tuning dynamic voltage scaling techniques for processor design'. PhD dissertation, The University of Texas, Austin, May 2013

[3] Das, S.: Razor: a variability – tolerant design methodology for low-power and robust computing'. PhD dissertation, The University of Michigan, 2009

[4] Savage, N.: '*Intel and ARM are exploring self-correction schemes to boost processor performance and cut power*' (IEEE Spectrum, 2008). Available at http://spectrum.ieee.org/semiconductors/design/intel-and-arm-are-exploringselfcorrection-schemes-to-boost-processor-performance-and-cut-power, Accessed on July 30 2018

[5] Das, S., Bull, D.M., Whatmough, P.N.: 'Error-resilient design techniques for reliable and dependable computing', *IEEE Trans. Device Mater. Reliab.*, 2015, **15**, (1), pp. 24–34

© The Institution of Engineering and Technology 2018

[6] Ernst, D., Kim, N.S., Das, S., *et al.*: 'Razor: a low-power pipeline based on circuit-level timing speculation'. Proc. 36th Annual IEEE/ACM Int. Symp. Microarchitecture, December 2003, pp. 7–18
[7] Shin, I., Kim, J., Shin, Y.: 'Aggressive voltage scaling through fast correction of multiple errors with seamless pipeline operation', *IEEE Trans. Circuits Syst.*, 2015, **62**, (2), pp. 468–477
[8] Fojtik, M., Fick, D., Kim, Y., *et al.*: 'Bubble razor: eliminating timing margins in an ARM cortex-M3 processor in 45 nm CMOS using architecturally independent error detection and correction', *IEEE J. Solid-State Circuits*, 2013, **48**, (1)
[9] Chouhury, M.R., Chandra, V., Aitken, R.C., *et al.*: 'Time-borrowing circuit designs and hardware prototyping for timing error resilience', *IEEE Trans. Comput.*, 2014, **63**, (2), pp. 497–509
[10] Augustus, K.U.: 'Going beyond worst-case specs with TEAtime', *IEEE Computer Society*, 2004, **37**, (3), pp. 51–56
[11] Bowman, K.A., Tschanz, J.W., Kim, N.S., *et al.*: 'Energy-efficient and metastability – immune resilient circuits for dynamic variation tolerance', *IEEE J. Solid-State Circuits*, 2009, **44**, (1)
[12] Kwanyeob, C., Mukhopadhyay, S.: 'A dynamic timing error prevention technique in pipelines with time borrowing and clock stretching', *IEEE Trans. Circuits Syst.*, 2014, **61**, (1), pp. 74–83
[13] Prasad, A.N.D., Somani, A.: 'Countering power analysis attacks using reliable and aggressive designs', *IEEE Trans. Comput.*, 2013, **63**, (6), pp. 1408–1420
[14] Stott, E., Levine, J.M., Cheung, P.Y.K.: 'Timing fault detection in FPGA-based circuits'. IEEE 22nd Annual Int. Symp. Field-Programmable Custom Computing Machines (FCCM), Boston, May 2014
[15] Prasad, A.N.D., Somani, A.: 'Low overhead soft error mitigation techniques for high-performance and aggressive designs', *IEEE Trans. Comput.*, 2013, **61**, (4), pp. 488–501

© The Institution of Engineering and Technology 2018