Dynamic Voltage Scaling for Commercial FPGAs

[1]C.T. Chow, L.S.M. Tsui, P.H.W. Leong, [2]W. Luk, [3]S. Wilton
[1]Dept. of Computer Science and Engineering, The Chinese University of Hong Kong
[2]Dept. of Computing, Imperial College
[3]Dept. of Electrical and Computer Engineering, University of British Columbia
[1]{ctchow, lstsui2, phwl}@cse.cuhk.edu.hk, [2]wl@doc.ic.ac.uk, [3]stevew@ece.ubc.ca

Abstract

A methodology for supporting dynamic voltage scaling (DVS) on commercial FPGAs is described. A logic delay measurement circuit (LDMC) is used to determine the speed of an inverter chain for various operating conditions at run time. A desired LDMC value, intended to match the critical path of the operating circuit plus a safety margin, is then chosen; a closed loop control scheme is used to maintain the desired LDMC value as chip temperature changes, by automatically adjusting the voltage applied to the FPGA. We describe experiments using this technique on various circuits at different clock frequencies and temperatures to demonstrate its utility and robustness. Power savings between 4% and 54% for the V_{INT} supply are observed.

1. Introduction

Field Programmable Gate Array (FPGA) technology is gaining importance for embedded appliances since it is able to combine high performance with low cost and short design time. However, reconfigurable architectures have much higher parasitic capacitance compared with an ASIC. As a result, FPGAs can consume considerably more power than ASICs, in some cases, up to two orders of magnitude in the same technology [12]. This makes FPGAs less suitable for power sensitive applications such as handheld devices. Power reduction in FPGAs is hence important.

Previous work has been conducted to find ways to reduce power consumption in FPGAs. Some methodologies involve modifying the FPGA itself; these include dual V_{dd}/V_{INT}, gated clock routing tree and power aware FPGA architectures [7, 4]. Other methods work with standard FPGAs; these methods include pipelining, power aware CAD algorithms and power aware coding of finite state machines [4]. While the first set of methodologies can really only be exploited by FPGA vendors, the second set can be exploited by FPGA users.

Voltage scaling involves reducing the supply voltage of a circuit [1]. It can reduce both dynamic and leakage current, but at the expense of increasing circuit delay. For best results, the circuit should operate at the voltage that reduces power consumption as much as possible, while maintaining reliable operation. Finding this threshold is difficult, however, since the optimum operating voltage changes with time and between devices, and varies with die temperature.

This paper introduces a new methodology to support dynamic voltage scaling (DVS) on commercial FPGAs. Rather than powering the FPGA with a fixed voltage, we dynamically adjust the voltage supply of the FPGA; the voltage reduction leads to power savings. To avoid over reducing the voltage, we embed a novel logic delay measurement circuit (LDMC) to measure the on-chip delay of a dummy circuit. The LDMC readings are affected by the temperature and voltage of the FPGA's logic cells, allowing us to dynamically adjust the supply voltage of the FPGA in a closed loop fashion according to the sensor value. Fig. 1 shows the system architecture of our DVS implementation.

In this paper, we investigate the effectiveness of DVS on an FPGA. Our contributions include:

- dynamic voltage scaling for reducing FPGA power consumption: we believe this is the first reported methodology for applying DVS techniques to commercial FPGAs;
- a novel Logic Delay Measurement Circuit using FPGA resources: to the first order, the reading produced by the LDMC tracks the critical path delay of a circuit that we wish to operate under DVS; we also show experimentally that by using a closed loop DVS system which keeps the LDMC reading above a threshold, no errors occur;
- demonstration of the effectiveness of our approach: we achieve power reductions from 4% to 54% (typically 20–30%) in the V_{INT} supply while maintaining correct operation over a wide range of temperatures.

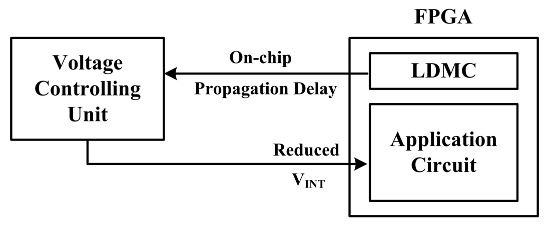

Figure 1. System architecture of our DVS implementation.

Experimental results using a Xilinx Virtex XCV300E FPGA are presented, which show that this approach achieves considerable power reduction and is robust to changes in die temperature. The technique can be combined with other approaches, and requires no modification to the FPGA itself.

The remainer of this paper is organised as follows. Section 2 provides a brief introduction to power reduction strategies. Our approach is described in Section 3 along with details of the LDMC circuit which is needed for our approach. Results regarding the calibration of the LDMC and the overall power reduction abilities of our technique are provided respectively in Sections 4 and 5. Finally, conclusions are given in Section 6.

2. Background

Power consumption of a CMOS technology FPGA has two major components, static power consumption and dynamic power consumption. Static power consumption is due to gate oxide tunneling current, subthreshold conduction of MOS transistors and leakage in the reverse biased junctions. As the fabrication process becomes more advanced and the feature size of transistors are decreased, the leakage current increases significantly and becomes a major component of the power consumption. We do not directly address the problem of reducing static power consumption in this work although we note that reducing the operating voltage reduces the static power.

The main source of dynamic power consumption is due to the charging and discharging of capacitances in the integrated circuit. The dynamic power consumption can be modeled by the following formula:

$$P = \sum (C \cdot Vdd^2 \cdot f) \quad (1)$$

where *C* is the parasitic capacitance of each part of the circuit, V_{dd} is the supply voltage and *f* is the switching frequency of the circuit. Since there is a quadratic relationship between V_{dd} and the dynamic power, reducing the voltage will reduce the dynamic power significantly.

Techniques for power reduction in FPGAs can be classified into two groups, those that require changes to the FPGA architecture or circuitry, and those that do not require such changes. The latter methods, which are the focus of this work, are applicable to existing devices and can be applied at the design or system level. Some of these proposed techniques include:

- Pipelining long combinational circuits. The difference in the arrival time of inputs is reduced. As a result, glitches are reduced and the dynamic power consumption is significantly reduced. It has been reported that this methodology can reduce the power used for every operation by 40-90% [8]. As register resources are abundant in FPGAs and pipelined circuit usually have better performance, pipelining is one of the best solutions for reducing power in FPGAs.
- Power-aware CAD algorithms have been studied and shown to be effective for reducing power consumption [4, 5]. These algorithms include retiming for power optimisation, reduction of gate-tunneling leakage, zippering etc.

Unlike the methods above, DVS is applied at a system level and can reduce both static and dynamic power consumption without changes to the design. Another advantage of DVS is that voltage has a square relation with the dynamic power consumption so a small decrease in voltage leads to significant power reduction.

3. Dynamic Voltage Scaling Architecture

This section describes our dynamic voltage scaling scheme, which includes a Logic Delay Measurement Circuit (LDMC).

Fig. 1 shows the system architecture. The power supply of the FPGA is controlled by a voltage controller, which dynamically adjusts the supply voltage of the FPGA. By lowering the voltage, the power dissipated by the FPGA can be reduced, at the expense of reduced performance of the FPGA circuit. In our implementation, the voltage controller is implemented in a personal computer (PC), however, it would also be possible to implement the controller using simple electronics.

Note that it is necessary to keep the IO voltage levels unchanged to maintain compatibility with other chips at the board level. Fortunately, modern FPGAs have a separate supply networks for the input output blocks (V_{IO}) and internal circuit (V_{INT}). Our strategy is to apply DVS to the internal circuits (logic cells, routing elements and storage cells) while keeping the input output block (IOB) voltage unchanged. Analogue components in the FPGA, such as delay locked loops (DLL) or phase lock loops (PLL) are oper-

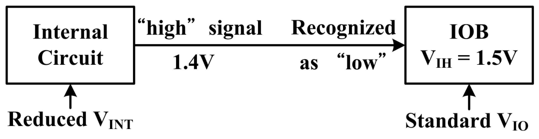

Figure 2. Example of IO error in DVS.

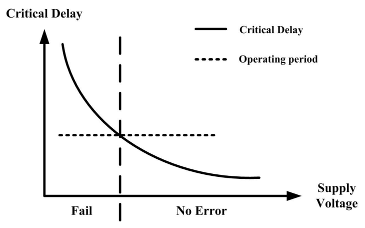

Figure 3. Delay as a function of supply voltage. Figure shows regions where the circuit will operate correctly and where it will fail.

ated from a third, independent supply in current Xilinx and Altera devices.

The voltage controller is responsible for ensuring the voltage supply to the FPGA is not lowered so much that the FPGA ceases to operate properly, or it does not meet the frequency requirements of the application. Clearly, FPGAs are designed to operate within a specified voltage range. When we operate the FPGA's V_{INT} at a voltage lower than this range, errors may occur. We find that two types of error can occur: *IO errors* and *delay errors*.

Fig. 2 illustrates an IO error. In this scenario, the core is operating at such a low voltage that a high output signal from the core is less than the threshold voltage of the IOB. In this case, the IOB may mistakenly interpret the high signal as a low value, leading to an incorrect FPGA output. Our experiments in Section 4 show that the pass/fail voltage for an IO error is a strong function of the chip temperature. Note that this is not a concern for FPGA inputs, since a high signal from an IOB to the core will still be interpreted as a high value, since $V_{INT} < V_{IO}$.

The second type of error that can occur is a *delay error*. When the voltage is lowered, the switching speed of the transistors is reduced. As the voltage is lowered, the critical path delay will increase, eventually becoming longer than the clock period of the system clock. When this occurs, the FPGA can no longer meet its timing requirement, and the system will fail. This is shown in Fig. 3.

To ensure that the FPGA does not experience IO or delay errors, the voltage controller uses a feedback signal from the FPGA to indicate the status of the circuit implemented on the FPGA. This signal is used as a "warning signal" to indicate when the FPGA is about to experience IO or delay errors. By adjusting the voltage based on this signal, the voltage controller can ensure that IO and delay errors do not occur.

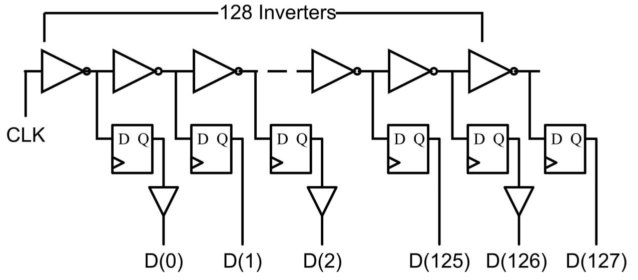

Figure 4. Schematic of the LDMC.

The feedback signal is obtained from a Logic Delay Measurement Circuit (LDMC) which, along with the user circuit, is implemented using normal FPGA logic resources. The LDMC consists of three major components: a delay line, registers, and a leading zero detector. The schematic for the delay line and registers is shown in Fig. 4. The delay line consists of 128 inverters connected in series. The same CLK signal is connected to the input of the delay line and used to clock all of the D flip flops (DFFs). As the skew of the FPGA's clock distribution network is very small, we assume that these signals arrive almost simultaneously.

On each falling clock edge, a wavefront begins passing through the inverter chain. Half a cycle later on the rising clock edge, the propagated signal is latched into the registers. At this rising clock edge, some of the register inputs will have switched, and some will not yet have switched. The number of inputs that have switched will depend on the delay of the inverters; the delay of the inverters depends on the temperature and supply voltage. A leading-zero detector is then used to estimate the circuit's propagation delay. In this way, the LDMC measures how many delay stages the falling edge propagates in half a clock period.

In our implementation, the placement of the inverter chain is constrained so that each delay line inverter and the associated D flip-flop are in the same logic cell. Adjacent cells are placed in adjacent sites on the FPGA as shown in Fig. 5. Such a placement ensures that each delay stage have an approximately equal propagation delay. In our implementation on an XCV300E, the LDMC uses 177 slices, a small fraction of the total FPGA resources.

Given the feedback signal, the voltage controller uses the following algorithm to control the voltage supplied to the FPGA. In this algorithm, `LDMC` denotes the reading from the LDMC register, `Voltage` denotes the voltage provided

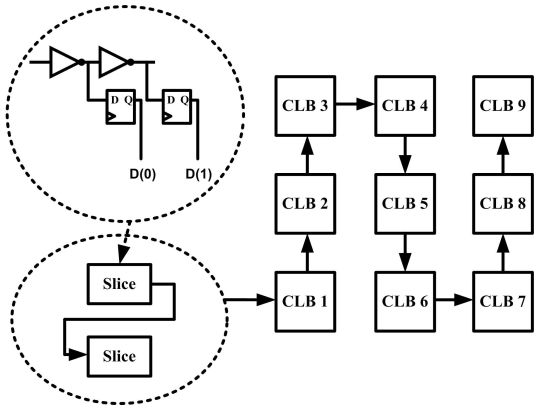

Figure 5. Constrained placement of LDMC.

to the FPGA, and `Threshold` denotes the LDMC threshold indicating the onset of failure modes.

```
Voltage = InitialVoltage;
while true
do
  if ((LDMC - Threshold) > 8)
    Voltage = Voltage - 0.05;
  elseif ((LDMC - Threshold) > 3)
    Voltage = Voltage - 0.01;
  elseif ((LDMC - Threshold) > 0)
    Voltage = Voltage - 0.005;
  elseif ((LDMC - Threshold) = 0)
    Voltage remain unchanged;
  elseif ((LDMC - Threshold) < 0)
     Voltage = Voltage + 0.01;
  wait for 200 ms;
done
```

As the voltage is reduced, the value read from the LDMC register goes down because the propagation delay of the inverter chain within the LDMC increases. As long as the LDMC is above a predefined threshold, the FPGA is deemed to be working properly, and the voltage is lowered further. As the LDMC approaches the threshold value, the voltage is no longer lowered, and if it goes below the threshold value, the voltage is increased. In this way, the voltage can be adjusted as the propagation delay of the inverter chain varies with the chip temperature. The rate of change of voltage should be large so that the applied voltage can approach the lowest possible voltage quickly. However, additional noise will be introduced if the voltage is changed too quickly. We found that the voltage step size and 200 ms interval used in the algorithm was suitable for our particular experimental setup.

Of critical importance is the selection of a proper threshold value. As described in Section 3, lowering the voltage too far can cause the chip to fail in two ways: the core voltage may become too low to drive the output blocks properly, or the chip may run slower than is required by the application. Experimentally, we can determine LDMC threshold values that indicate the onset of each of these failure modes; clearly, these values would be FPGA-dependent. We have found that the threshold value corresponding to the onset of both types of errors is different, depending on whether the voltage is changing quickly (we refer to this threshold as the *dynamic threshold*) or slowly (*static threshold*). Given these thresholds, we then calculate the `Threshold` variable in the above pseudo-code as follows: `Threshold` = $\max(TH_{ds}, TH_{dd}, TH_{is}, TH_{id}) + TH_{sm}$ where TH_{ds} is the static delay threshold, TH_{dd} is the dynamic delay threshold, TH_{is} is the static IO threshold value, TH_{id} is the dynamic IO threshold value, and TH_{sm} is a safety margin – we found $TH_{sm} = 2$ works from experiments.

4. Experiments

In this section, we experimentally evaluate our technique and show how the threshold values described in the previous section can be obtained.

We use the Pilchard card [6] in an 800 MHz Pentium III host personal computer (PC) as the hardware platform for the experiments. Pilchard is a reconfigurable computing platform that uses the SDRAM bus instead of the conventional PCI bus for the host interface. The board contains a Xilinx Virtex 300E-8 device, which contains a 32×48 CLB array implemented in $0.18\mu m$ with 6-layer metal CMOS technology.

To conduct dynamic voltage scaling experiments, we replace the 1.8 V regulator that supplies V_{INT} on the Pilchard board with the output of a Keithley sourcemeter 2400 [3]. The V_{IO} for the IOBs is kept at 3.3 V. The sourcemeter is used as a voltage source, its output being programmable with 0.02% accuracy via a RS-232 interface of the sourcemeter. The sourcemeter can also give current measurements with a basic accuracy of 0.22%. This feature is used to measure the current consumption of the FPGA. In a practical system, the sourcemeter can be replaced by a digital to analogue converter with sufficient current drive for the FPGA.

As the IO bandwidth of the Pilchard board is lower than that of circuit under test and since IO operations between the FPGA and other circuit components significantly affect the power consumption, test vector generation and error detection is done on-chip.

Next, we describe experiments to demonstrate correlation between LDMC readings and (i) IO errors, and (ii) delay errors.

First, we conduct experiments to demonstrate the relation between IO errors and the LDMC reading. A 64-bit register is implemented on the FPGA and preset to output 0xFFFFFFFFFFFFFFFF to the bus interface. The FPGA is also populated with dummy linear feedback shift register

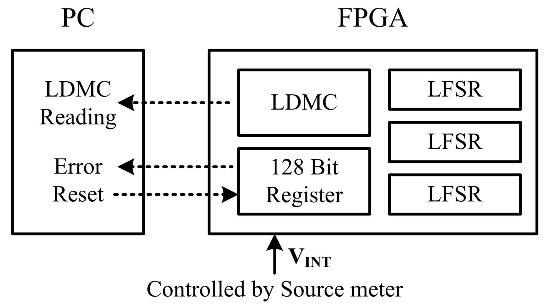

Figure 6. Block diagram of IO error experiment.

Circuit activity*	TH_{IO_STATIC}	LDMC value
Minimum	1.26 V	69
1/6	1.31 V	69
2/6	1.36 V	69
3/6	1.42 V	69
4/6	1.45 V	69
5/6	1.53 V	70
Maximum	1.57 V	68

* Circuit activity is reported as % of the logic resources on the chip.

Table 1. IO static threshold value and voltage as a function of circuit activity.

(LFSR) circuits which are used to simulate different chip activity, as shown in Fig.6. We reduce the V_{INT} supply voltage, and find that the LDMC output decreases accordingly. Eventually, an IO error occurs: the value read back from the Pilchard card is not 0xFFFFFFFFFFFFFFFF and the threshold voltage is recorded. Table 1 summarises the results of this experiment.

The table shows that IO errors occur at different voltage levels depending on the circuit activity. This is probably due to the LFSRs causing the temperature of the die to rise. IO errors occur only if the LDMC output decreases below a certain value (70 in our implementation). We use this value as our static IO threshold, TH_{is}.

We also investigate whether a fast changing V_{INT} voltage level changes the value of LDMC output at which IO errors occur. The same testing approach described earlier is used, but with the V_{INT} level being switched between the standard level (1.8 V) and another voltage at the maximum speed achievable by the sourcemeter (5 Hz with each transition taking several microseconds). The lowest LDMC value for which correct operation is observed (the *dynamic IO threshold*, or TH_{id}) is found to be 71. One would expect that this value is higher than the static threshold for the same circuit, since a fast changing supply voltage will introduce additional noise to the FPGA.

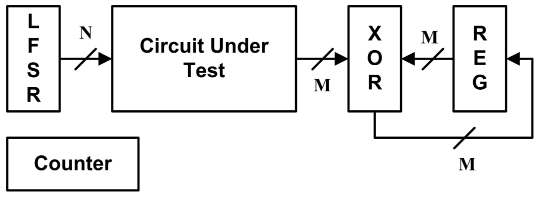

Figure 7. Architecture of test circuit.

Second, we use the LDMC as a reference for the DVS only if LDMC output readings track the delay errors. Since, to a first order, the delay of both are dominated by the delay of the logic cells and routing resources in the FPGA, we expect this to be true. We implement the test circuit shown in Fig. 7 to detect occurrence of delay error in our circuit under test.

Upon initialisation, the LFSR and the register is reset to a known value and the counter is reset to zero. The LFSR and circuit under test will then start to run, generating an output which is "XOR-ed" with the value of the register and stored back to the register. This process is repeated until the counter reaches a certain value (2^{28} in our implementation), after which the register checksum value is latched in another register to be read by the PC host. The test circuit is constrained so that its critical path is dominated by that of the circuit under test. We thus ensure delay errors in the circuit under test will occur before the rest of the test circuit fails. The entire circuit is replicated so as to fill up most of the area in the FPGA. Before we start the experiment, the circuit is operated at the standard voltage to obtain the correct checksum. After that, we decrease the voltage level of V_{INT} until it fails. The checksum will be correct only if 2^{28} computations are correct.

We test several different circuits including integer multipliers, dividers, and CORDIC cores generated by the Xilinx CORE Generator [10, 11, 9]. Floating-point multipliers from opencores.org [2] are also tested. For each circuit, we use different bit-widths, and two different clock frequencies (66 MHz and 100 MHz). Table 2 shows the results. For each benchmark, at each of the two clock speeds, we calculate the *tolerance* which is defined as the amount we can slow down the circuit, by reducing the supply voltage before the circuit fails to meet timing requirements. More precisely, the tolerance is defined as:

$$Tolerance = (P_1 - P_2)/P_2 \qquad (2)$$

where P_2 is the minimum operating period reported by the vendor tool, and P_1 is period of the clock used to test the circuit; in general, P_2 is less than P_1 for a circuit operating

Name	Speed	Tolerance (%)	TH_{STATIC}	TH_{DYN}
sqrt8	66MHz	26.36	83	89
sqrt8	100MHz	88.51	*	*
sqrt12	100MHz	17.43	94	97
sqrt12	66MHz	76.51	*	*
sqrt16	100MHz	10.19	95	94
sqrt16	66MHz	66.19	*	*
sqrt24	100MHz	-0.45	105	101
sqrt24	66MHz	51.19	*	*
mul5	100MHz	42.94	85	90
mul5	66MHz	114.93	*	*
mul7	100MHz	28.62	89	94
mul7	66MHz	92.68	*	*
mul11	100MHz	1.48	115	115
mul11	66MHz	51.04	81	85
fp8_4	100MHz	17.86	88	92
fp8_8	66MHz	16.80	92	92
div32	100MHz	25.72	71	76
div32	66MHz	88.56	*	*

* An asterisk indicates that no delay errors are detected before IO errors occur.

Table 2. Static and dynamic delay thresholds.

Circuit	LDMC threshold	Supplied Voltage (V_{INT})		
		35 °C	50 °C	65 °C
sqrt8a	89	1.41	1.42	1.45
sqrt16a	101	1.52	1.54	1.56
sqrt16b	75	1.28	1.29	1.30
sqrt24a	111	1.78	1.81	1.87
fp8_8a	98	1.57	1.60	1.63
mul12b	87	1.42	1.42	1.45
mul7a	95	1.46	1.48	1.50
reg64 [a]	72 [b]	1.65	1.68	1.71

[a] Circuit for IO error testing.
[b] IO error threshold.

Table 3. Impact of Chip Temperature on V_{INT}.

correctly. For each benchmark and each clock speed, we indicate the tolerance and the measured the static dynamic delay threshold (T_{ds} and T_{dd}).

From Table 2, we have the following findings:

- Circuits having larger tolerances tend to have a lower threshold value, but there is no direct relation between tolerance and threshold value. For example, circuits sqrt8a and div32a have a similar tolerance but they have different threshold values. A possible reason is that they may have a different ratio of logic delay and routing delay, one being less sensitive to voltage scaling than the other.
- For circuits having large tolerance (more than 60%), IO errors occur before delay errors. In this case, the TH_{is} and TH_{id} will determine the threshold value used by the voltage control circuit.

In this experiment, it is important that the input vectors exercise the critical path of the circuit. As an example, a 128-bit adder circuit has 2^{256} input combinations and random inputs are not likely to result in the critical path being tested. This is a limitation of our experimental method, and may result in a threshold value that is smaller than what it should be. Increasing TH_{sm} will compensate somewhat for these sorts of errors.

Finally, we show that our technique can maintain correct operation even as the temperature of the FPGA changes.

The FPGA chip surface temperature is increased using a hair dryer. During the experiment, we record the correctness of the circuit and the V_{INT} voltage at different temperatures. Results are summarised in Table 3. We find that the circuits under test do not have any error if the LDMC is kept at the LDMC threshold. When the chip surface temperature increases, the supply voltage is automatically increased to keep the LDMC reading at the threshold. If the supply voltage is not increased when temperature is increased, the LDMC reading will decrease and the circuit under test fails.

5. Power Savings and Trade-offs

In this section, we illustrate the effectiveness of the proposed DVS methodology for power reduction, and provide guidelines for applying this methodology.

We first apply the DVS methodology to some test circuits at room temperature and record the power consumption using DVS. A summary of the results is given in Table 4.

As shown in the table, the power reduction achieved varies from 4% to 54%. Typically we can achieve 20-30% power savings. Circuits having a LDMC threshold near the IO error LDMC threshold have the best power reduction. Circuits having larger tolerance usually have large power savings so a maximally pipelined version of a circuit results in the largest power savings. It should also be noted that a pipelined version of a circuit can reduce the power consumption even if voltage scaling technique is not applied because pipelining can reduce glitches.

Using this technique, an FPGA runs correctly even if the operating frequency is higher than the maximum frequency reported by the vendor tools (such as for circuit sqrt24a).

Name *	Area ** (Slices)	Tolerance (%)	LDMC threshold	Power at 1.8 V (mW)	Power at LDMC threshold (mW)	Power save (%)
sqrt8a	787	26.36	89	361.52	225.85	37.53
sqrt12a	907	17.43	100	481.89	348.57	27.67
sqrt16a	1043	10.19	101	576.30	418.99	27.30
sqrt20a	1203	4.18	105	759.24	617.54	18.66
sqrt24a	1379	-0.45	111	915.10	845.04	7.66
sqrt8b	787	88.51	73	322.52	148.46	53.97
sqrt12b	907	76.51	73	405.12	190.72	52.92
sqrt16b	1043	66.19	75	464.10	219.89	52.62
sqrt20b	1203	56.81	74	597.34	295.13	50.59
sqrt24b	1379	51.19	74	697.03	378.04	45.76
mul5a	743	42.94	91	307.43	203.09	33.94
mul7a	795	28.62	95	350.99	241.35	31.24
mul9a	879	3.38	117	453.61	397.79	12.31
mul11a	975	1.48	121	539.12	513.85	4.69
mul5b	743	114.93	75	273.54	125.20	54.23
mul7b	795	92.68	75	308.75	142.94	53.70
mul9b	879	54.64	88	371.14	223.91	39.67
mul11b	975	51.04	87	435.16	258.38	40.62
div16a	1166	69.03	76	190.66	88.72	53.47
div32a	2196	25.72	77	662.73	357.99	45.98
div16b	1166	151.55	75	215.76	98.84	54.19
div32b	2196	88.56	75	644.02	314.90	51.10
fp8_4a	1071	17.86	94	500.59	333.37	33.40
fp8_8a	1932	16.80	98	768.42	554.23	27.87

* Circuits with suffix 'a' have a clock frequency of 100 MHz;
circuits with suffix 'b' have a clock frequency of 66 MHz.

** Circuits are synthesized using Xilinx ISE6.2, optimized for speed
All experiments are conducted at room temperature.

Table 4. Power reduction achieved using DVS.

Applying DVS allows the voltage to be reduced even in such circuits, and a 7.66% power saving is achieved. This is because FPGA vendors use a fixed supply voltage with margins for chip and temperature variation. Our DVS methodology can reduce this margin because we monitor the FPGA's delay and adjust the supply voltage accordingly.

Next, we explore design tradeoffs between throughput, power consumption and area using the DVS technique. Each benchmark circuit can operate at two frequencies, 100 MHz and 66 MHz, and with the appropriate threshold settings. The throughput per joule (million operations / joule) and throughput per area (thousands operations / slice) of each implementation are recorded in Table 5.

It can be seen that running a circuit at a lower clock speed results in a lower LDMC threshold. The savings resulting from DVS can then be increased. From the table we find that if the original circuit, such as sqrt24, has a large LDMC threshold, reducing the operating frequency can increase the throughput per joule significantly. At the same time, the throughput per area will decrease, so there is a tradeoff. If the original circuit, such as sqrt8, has a LDMC threshold near the IO error threshold, reducing the operating frequency would decrease the throughput per area, while the throughput per joule does not increase.

The results of these experiments are important, because they suggest that we can decrease the operating frequency of some circuits to increase the savings due to DVS. To compensate for the loss in processing power due to the reduction in frequency, we can increase the number of parallel functional units. Clearly, one needs to be careful to en-

Name	Speed	Tolerance (%)	LDMC threshold	Power at threshold (mW)	Throughput / Energy (MOp / J)	Throughput / Area (KOp / Slice)
sqrt8a	100MHz	26.36	89	221.98	450.49	127.06
sqrt8b	66MHz	88.51	73	151.33	436.12	83.86
sqrt16a	100MHz	10.19	101	439.05	227.77	95.88
sqrt16b	66MHz	66.19	76	231.90	284.61	63.28
sqrt24a	100MHz	-0.45	111	904.83	110.52	72.52
sqrt24b	66MHz	51.19	74	365.86	180.40	47.86
mul5a	100MHz	42.94	91	193.17	517.68	134.59
mul5b	66MHz	114.93	75	132.71	497.32	88.83

* The area of each circuit can be obtained from Table 4.

Table 5. Table showing tradeoff between throughput, energy and area.

sure that the decrease in power consumption due to DVS outweighs the increase in power consumption due to the increased number of functional units.

6. Conclusion

This paper shows that an LDMC implementation for FPGA technology can be used as a reference for dynamic voltage scaling. We have also shown that the DVS methodology can provide up to 56% power reduction. One important advantage of our methodology is that it does not require additional design effort or changes to the FPGA itself. The methodology can be applied to the application after its development, and no changes to the circuit are required. Another important advantage of this methodology is that DVS reduces power consumption at physical level. It can be used with system-level power reduction techniques like pipelining to provide additional power reduction.

The main limitation of this approach is that it requires experimentation to find appropriate threshold values for each FPGA. For some applications, however, the reduction in power consumption may be sufficient to motivate this additional system-level effort. Our current and future work is focused on applying the proposed techniques to a wide range of applications, such that the experience gained will enable us to refine and generalise our approach.

References

[1] T. Austin, D. Blaauw, T. Mudge, and K. Flautner. Making typical silicon matter with Razor. *IEEE Computer*, pages 41–49, March 2004.

[2] T. Hawkins. *CF Floating Point Multiplier*. `http://www.opencores.org/projects.cgi/web/cf_fp_mul/overview`.

[3] Keithley Instruments Inc. *Keithley sourcemeter model 2400*. `http://www.opencores.org/projects.cgi/web/cf_fp_mul/overview`.

[4] T. Kuroda, T. Fujita, S. Mita, T. Nagamatsu, S. Yoshioka, K. Suzuki, F. Sano, M. Norishima, M. Murota, M. Kako, M. Kinugawa, M. Kakumu, and T. Sakurai. How to manage power consumption in advanced fpgas. *IEEE Journal of Solid-State Circuits*, 2002. `http://www.xilinx-china.com/publications/xcellonline/partners/xc_synpl%icity44.htm`.

[5] J. Lamoureux and S. Wilton. On the interaction between power-aware FPGA CAD algorithms. In *ICCAD*, pages 701–708, 2003.

[6] P. Leong, M. Leong, O. Cheung, T. Tung, C. Kwok, M. Wong, and K. Lee. Pilchard - a reconfigurable computing platform with memory slot interface. In *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 170–179, 2001.

[7] F. Li, Y. Lin, L. He, and J. Cong. Low-power fpga using pre-defined dual-vdd/dual-vt fabrics. In *FPGA '04: Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*, pages 42–50, New York, NY, USA, 2004. ACM Press.

[8] S. J. E. Wilton, S.-S. Ang, and W. Luk. The impact of pipelining on energy per operation in field-programmable gate arrays. In *Field-Programmable Logic and Applications. Proceedings of the 13th International Workshop, FPL 2004, Lecture Notes in Computer Science, LNCS 3203*, pages 719–728. Springer-Verlag, 2004.

[9] Xilinx. *Intellectual Property : CORDIC,*. `http://www.xilinx.com`.

[10] Xilinx. *Intellectual Property : Multiply Generator*. `http://www.xilinx.com`.

[11] Xilinx. *Intellectual Property : Pipelined Divider*. `http://www.xilinx.com`.

[12] P. S. Zuchowski, C. B. Reynolds, R. J. Grupp, S. G. Davis, B. Cremen, and B. Troxel. A hybrid asic and fpga architecture. In *ICCAD*, pages 187–194, 2002.