Direct Measurement of Power Dissipated by Monte Carlo Simulations on CPU and FPGA Platforms

Albicocco, Pietro; Papini, Davide; Nannarelli, Alberto

Publication date:
2012

Document Version
Publisher's PDF, also known as Version of record

Direct Measurement of Power Dissipated by Monte Carlo Simulations on CPU and FPGA Platforms

IMM-Technical Report-2012-18

Pietro Albicocco, Davide Papini and Alberto Nannarelli

Dept. Informatics and Mathematical Modelling
Technical University of Denmark
Kongens Lyngby, Denmark
Email: an@imm.dtu.dk

Abstract

In this technical report, we describe how power dissipation measurements on different computing platforms (a desktop computer and an FPGA board) are performed by using a Hall effect-based current sensor. The chosen application is a Monte Carlo simulation for European option pricing which is a popular algorithm used in financial computations. The Hall effect probe measurements complement the measurements performed on the core of the FPGA by a built-in Xilinx power monitoring system.

1 Introduction

In [1], we implemented an FPGA-based specific processor for European option pricing using Monte Carlo simulations, and we compared its performance and power dissipation to the execution on a CPU. The experimental results of [1] show that impressive results, in terms of speed-up and energy savings, can be obtained by using FPGA-based accelerators at expenses of a longer development time.

In [1] the power measurements are limited to the implementation on the FPGA platform. Here, we extend the measurements to simulations performed on desktop computers to further validate the findings of [1].

The measurements are performed by connecting a Hall effect-based current sensor [2] between the power supply and the motherboard's power socket, as done in [3].

The experimental results confirm that using FPGA based accelerators can significantly reduce both execution time and energy consumption.

Algorithm 1 Monte Carlo European option pricing.

`VsqrtT` $= \sigma\sqrt{T}$
`drift` $= (r - \frac{\sigma^2}{2})T$
`expRT` $= e^{rT}$
`sum` $= 0$
for $i = 1$ to n **do**
  `St` $= S_0 \cdot e^{(\texttt{drift + VsqrtT} \cdot \texttt{Vrnd})}$
  if (`St` $- K >$ `0`) **then**
    `sum = sum + (St` $-K$`) ·` `expRT`
  end if
end for
`S` $=$ `sum`$/n$

2 Monte Carlo Simulation

The value of an European option can be computed by using a Monte Carlo simulation by evaluating

$$S(T) = S(0) \cdot e^{\left[\left(\mu - \frac{\sigma^2}{2}\right)T + \sigma\epsilon\sqrt{T}\right]} \tag{1}$$

for several samples and then by computing the mean value.
This approach is shown in Algorithm 1 for a *risk-neutral world* [4].
In the algorithm, the inputs are:

- initial security price S_0
- strike price K
- risk-free interest rate r
- security volatility σ
- time to expiration T (in years)
- number of simulations n.

The parameters r and σ are constant for each simulation, and, therefore variables `VsqrtT`, `drift` and `expRT` are constant as well and can be precomputed.
The computations in the body of the loop originate from a floating-point random variable `Vrnd` with Gaussian distribution in $[-1.0, 1.0]$, zero mean and variance=1.
The algorithm is particularly suitable for a hardware implementation on an Application Specific Processor (ASP), because it does not require frequent memory access, and the communication with the input/output is limited as well.
The algorithm can be executed on a computer, or, as shown in [1], implemented in hardware on a FPGA based ASP.

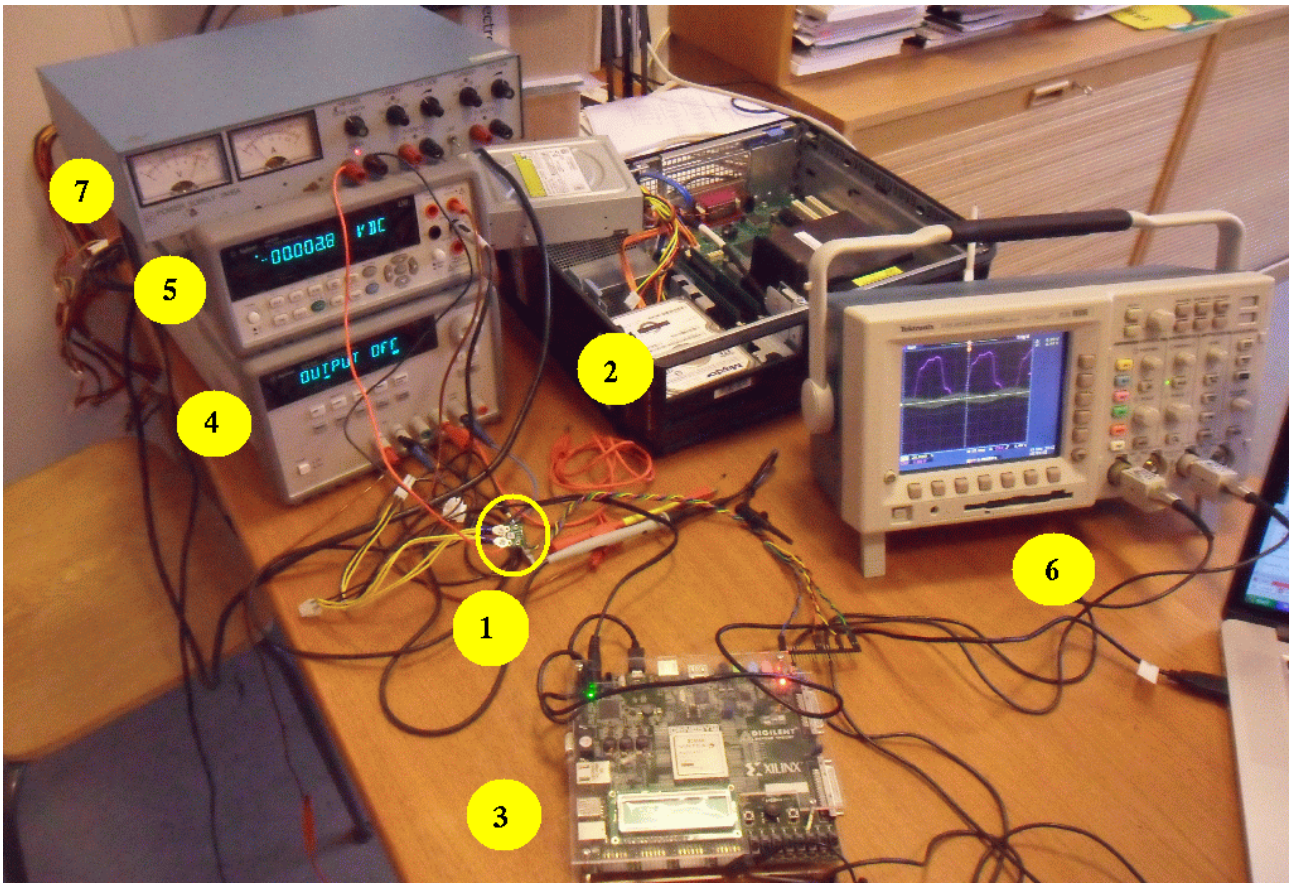
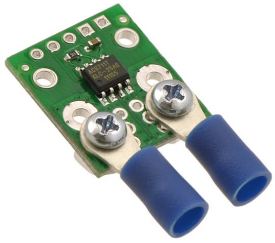


Figure 1: Measurements Set Up:

3 Measurements Set Up

This section presents the set up used in the experiments. We used the following equipment, listed below with the same numbers as in Figure 1.

1. An Allegro MicroSystem Hall effect current sensor, the ACS711 mounted on a Pololu board [2] (Figure 2 at left).
2. Desktop PC (Dell Optiplex 755) used to run C simulations equipped with an Intel® Core™2 Duo E4600 CPU running at a maximum frequency of 2.40 GHz.
3. Digilent Genesys board equipped with a Virtex-5 FPGA (XC5VLX50T).
4. A dual output power supply to bias the Hall effect probe and give the reference voltage (zero current in the probe) for the measurements.
5. A digital multimeter to measure the voltage drop across the probe.
6. A digital oscilloscope used to record the voltage measurements across the probe.
7. A second power supply for the FPGA board.



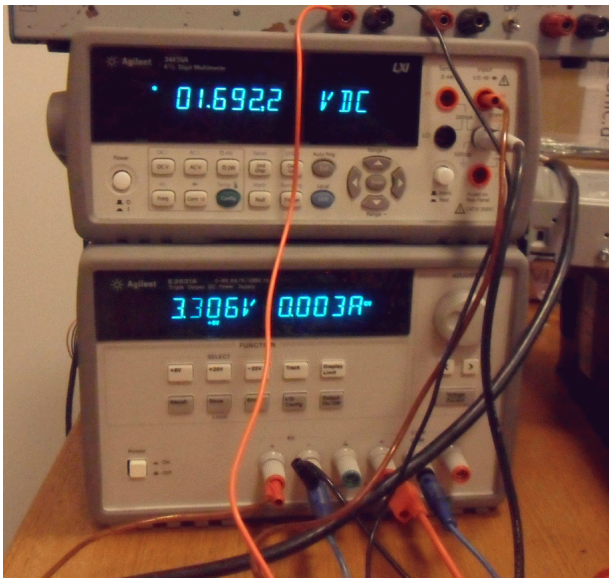


Figure 2: Hall effect probe [2] (left). Power and bias set up (right).

We call, in the following Device-Under-Test, or DUT, the desktop's CPU or the FPGA, depending on the experiment.

To measure the power dissipated by the DUT, the Hall effect current sensor is placed between the power supply and the DUT. The oscilloscope and the multimeter are used to read the voltage, proportional to the current that flows toward the DUT. The DC power supply, (4) in Figure 1, is used both to power the sensor, and to provide an offset voltage as a reference for the oscilloscope. This reference voltage is needed to bias the voltage reading up to 0 V when no current flows toward the DUT. The multimeter's probes are placed to measure the unbiased output voltage of the sensor. The power and bias set up of the probe is shown in Figure 2 at right.

The power consumption is computed through the following formula:

$$P = V_{supply} \cdot V_{meas}/S \quad [W] \tag{2}$$

where P is the power dissipated by the DUT in $[W]$, V_{meas} is the biased reading of the output of the sensor in $[mV]$, V_{supply} is the supply voltage in $[V]$, S is the sensitivity of the sensor $S = 110\ mV/A$, when powered at $3.3\ V$.

3.1 CPU Power Monitoring

The PC, running the Linux CentOS 6.3 (64-bit) operative system, is remote controlled through a `ssh` connection.

To measure the power dissipated by the CPU, the Hall effect current sensor is placed between the four-pin ATX cable of the power supply of the computer and the four pin ATX connector on the motherboard (Figure 3).

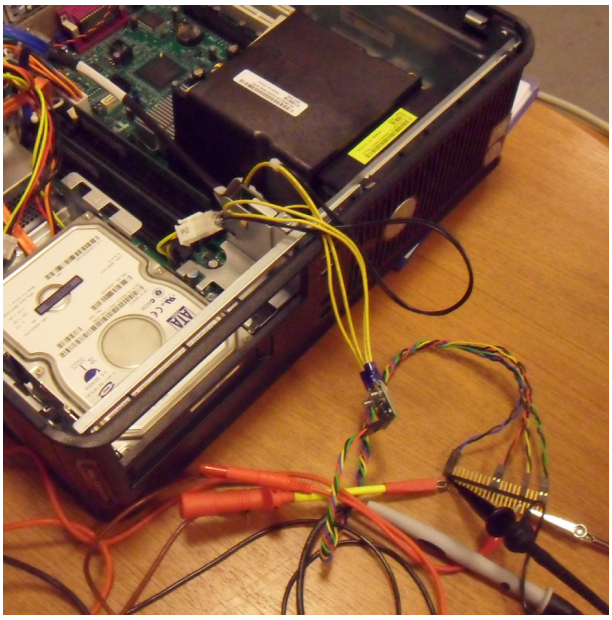


Figure 3: Detail of current sensor connection between the PC power supply and motherboard.

3.2 FPGA Power Monitoring

To measure the power dissipated by the whole Genesys board, a second DC power supply, (7) in Figure 1, is used to supply the board. The sensor is placed between the DC power supply and the board.

To measure the power dissipated by the FPGA chip only, the power monitoring system integrated on the board is used. The board is equipped with Kelvin resistors connected to the voltage regulators to monitor the power dissipation of the different parts of the FPGA chip. The voltage drop across these resistors is fed to a analog-to-digital converter, which output can be read by Digilient's Adept software through a PC (not depicted in Figure 1) connected to the FPGA board by a USB cable.

The Digilient's Adept power monitor can measure the power dissipated by the FPGA core and the other parts of the chip working at different voltages.

4 Experiments

4.1 CPU Measurements

The Monte Carlo simulation of Algorithm 1 is coded in C and compiled (gcc) for the 64-bit ISA of the Intel® Core™2 CPU. The simulation is run for $n = 10^6$ and the variations in the probe voltage drop are captured by the oscilloscope. The waveforms, under different operating frequencies are displayed in Figure 4.

Because CPU are energy optimized by having clock frequency and supply voltage scaling, depending on the required performance, the simulation is run with different clock frequencies f_C.

For the CPU-motherboard used two frequency/voltage scaling governors are available:

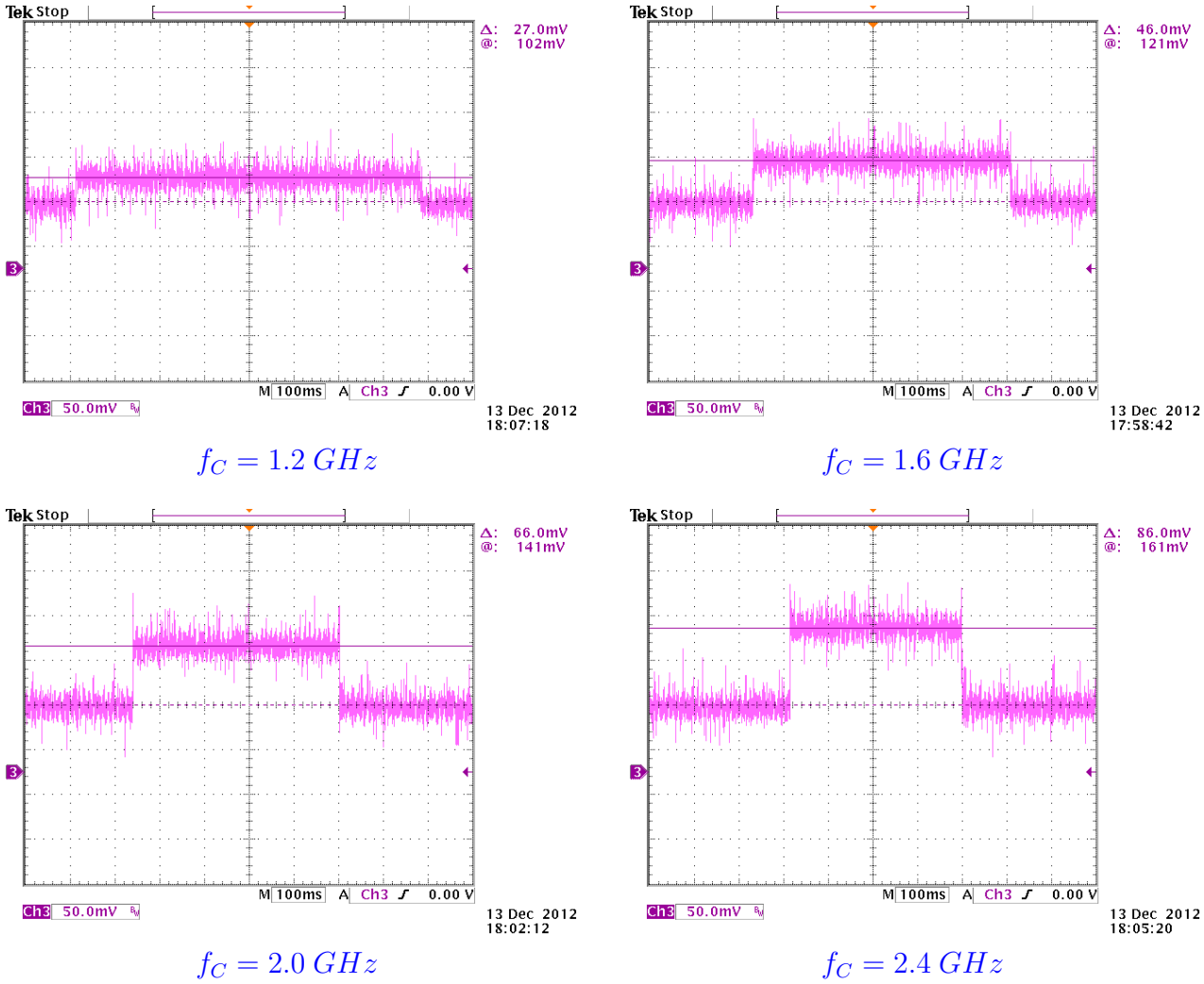


Figure 4: Oscilloscope snapshots for different clock frequencies (SpeedStep governor) and $n = 10^6$.

- the *"Legacy governor"*;
- the *"Enhanced SpeedStep governor"*.

Those governors can be set in the PC BIOS, while the frequency can be scaled by the Linux command `cpufreq-selector`.

The PC is running several processes (OS and user space processes) and we tried and keep those to a minimum by not running any graphic interface and by connecting to the PC by `ssh`. We call *idle* the state in which the PC is running only those basic processes and no user applications.

The measurements for the simulations under different governors/frequencies are reported in Table 1.

In Table 1, the columns (left to right) report the values:

c1 The clock frequency f_C set for the simulation.

c2 The simulation execution time, determined by a timer in the C program and measured on the waveforms of Figure 4.

Legacy governor

clock freq.	t(exec)	Readings			Power dissipation		
		run	idle	diff	run	idle	diff
$[GHz]$	[ms]	[mV]	[mV]	[mV]	[W]	[W]	[W]
0.3	2,170	125.00	88.00	37.00	13.64	9.60	4.04
0.6	1,390	130.00	88.00	42.00	14.18	9.60	4.58
1.2	798	138.00	83.00	55.00	15.05	9.05	6.00
2.4	384	163.00	75.00	88.00	17.78	8.18	9.60

SpeedStep

clock freq.	t(exec)	Readings			Power dissipation			CPU
		run	idle	diff	run	idle	diff	Vdd
$[GHz]$	[ms]	[mV]	[mV]	[mV]	[W]	[W]	[W]	[%]
1.2	768	105.00	75.00	30.00	11.45	8.18	3.27	0.83
1.6	575	123.00	75.00	48.00	13.42	8.18	5.24	0.90
2.0	461	143.00	75.00	68.00	15.60	8.18	7.42	0.96
2.4	384	163.00	75.00	88.00	17.78	8.18	9.60	1.00

Table 1: PC measurements for $n = 10^6$.

c3 The first of the readings is the probe reading (oscilloscope, Figure 4) of the high voltage level, when the simulation is running.

c4 The probe reading, low voltage level in Figure 4, when the simulation is not running: *idle*.

c5 The difference between the high and low level readings: the actual voltage drop due to the probe.

c6 The power dissipation corresponding to the reading in **c4** obtained by Eq. (2), with $V_{supply} = 12\ V$ and $S = 110\ mV/A$.

c7 The power dissipation by Eq. (2) for **c5**.

c8 The power dissipation by Eq. (2) for **c6**.

c9 The percent the CPU's supply voltage V_{DD} is scaled from the voltage V_{DD} at maximum frequency, when frequency is scaled. It only applies to the SpeedStep governor and it is measured indirectly by the measures at $f_{max} = 2.4\ GHz$ and at frequency $f_Y = Y\ GHz$:

$$\begin{cases} P_{Y\ GHz} &= (xV_{DD})^2 \cdot f_Y \cdot k \\ P_{2.4\ GHz} &= V_{DD}^2 \cdot 2.4 \cdot k \end{cases}$$

where the CPU switching capacitance k is assumed to be constant. By solving the above system, for the percent x we obtain the expression

$$x = \sqrt{\frac{P_{Y\ GHz} \cdot 2.4}{P_{2.4\ GHz} \cdot f_Y}} \tag{3}$$

used to compute the values in Table 1 column **c9** for $f_C = Y\ GHz$.

freq. $[GHz]$	P_{diff} $[W]$	t_{exec} $[ms]$	E_{sim} $[J]$
1.2	3.27	768	2.51
1.6	5.24	575	3.01
2.0	7.42	461	3.42
2.4	9.60	384	3.69

Table 2: Summary of PC-CPU measurements for $n = 10^6$.

The experimental results show that the fastest simulation (at f_{max}) results in a latency of $384\ ms$. The most power efficient simulation is the one run under the SpeedStep governor at $f_C = 1.2\ GHz$ with a power dissipation of $3.27\ W$ for the effective application power.

We introduce the energy-per-simulation, E_{sim} defined as

$$E_{sim} = P_{diff} \times t_{exec} \quad [J]$$

or the area of the box between the *run* and *idle* power levels in Figure 4.

Based on the E_{sim} definition, the most energy efficient simulation is the one run under the SpeedStep governor at $f_C = 1.2\ GHz$ as shown in Table 2.

4.2 FPGA Measurements

For the FPGA implementation of the accelerator to run the Monte Carlo simulation, we used the two implementations of [1]:

ASP-1P the single path implementation of the ASP, as shown in Figure 5.

ASP-4P the four path implementation of the ASP.

For the FPGA experiments, we measure the power in several parts of the system. By using Digilent's Adept software we can monitor the current drawn by different parts of the FPGA chip. Namely:

- DC-DC 3.3 V output, used to power the I/O of the FPGA chip.
- DC-DC 1.8 V output, used to power the DDR memory and the FPGA DDR I/O.
- DC-DC 1.0 V output, used to power the FPGA core: the actual processor.

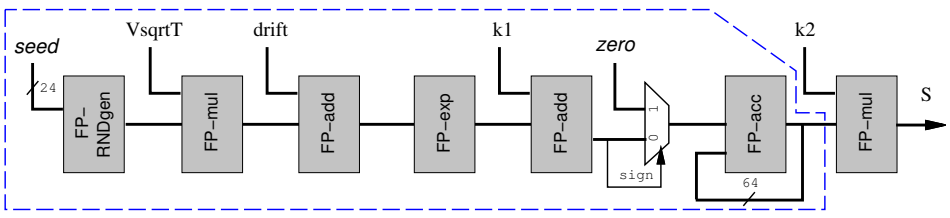


Figure 5: ASP: single path implementation.

	clock freq. [MHz]	status	Readings Mult. [V]	Readings Osc. [mV]	POWER DISSIPATION TOTAL (probe) [W]	Adept (FPGA chip) 3.3 V [mW]	1.8 V [mW]	1.0 V [mW]	2.5 V [mW]	Total [W]	Board [W]
No progr.	N.A.	N.A.	1.6930	43.2	1.96	-	-	-	-	-	-
ASP-1P	50	RUN	1.6925	43.3	1.97	841	25	426	248	1.540	0.428
		IDLE	1.6920	42.5	1.93	847	25	386	245	1.503	0.429
		RESET	1.6920	42.5	1.93	847	25	386	245	1.503	0.429
ASP-1P	100	RUN	1.6941	44.8	2.04	840	25	484	248	1.597	0.439
		IDLE	1.6297	43.0	1.95	847	25	408	248	1.528	0.427
		RESET	1.6925	43.1	1.96	854	25	402	248	1.529	0.430
ASP-4P	50	RUN	1.6968	48.1	2.19	847	25	582	248	1.702	0.484
		IDLE	1.6934	43.9	2.00	861	25	434	248	1.568	0.427
		RESET	1.6935	42.2	1.92	841	25	387	222	1.475	0.443
ASP-4P	100	RUN	1.7020	52.4	2.38	854	25	787	248	1.914	0.468
		IDLE	1.6944	45.2	2.05	867	25	483	248	1.623	0.432
		RESET	1.6914	42.2	1.92	834	25	390	222	1.471	0.447
										average	0.441
										std. dev.	0.019

Table 3: FPGA measurements.

- DC-DC 2.5 V output, used to power the FPGA auxiliary circuits.

Additionally, we monitor the whole current drawn by the FPGA board, by placing the Hall effect probe between the power supply and the on-board power connector, as shown in Figure 1: power supply (7), probe (1) and FPGA board (3).

The readings and power dissipation measurements in the experiments are reported in Table 3.

For each of the two processors (ASPs), we performed measurements with the circuit working at the maximum operating frequency of 100 MHz and at a frequency of 50 MHz to determine the components of dynamic and static power dissipation, as explained later in this section.

For each combination of ASP/frequency, we considered three processor states:

RUN : the simulation is running.

IDLE : the processor is idle. No inputs (random numbers) are generated.

RESET : the processor is in reset state. All registers are reset (state equivalent to $f_C = 0$).

Because the throughput is 1 processed random number per clock cycle ($1 \times f_C\ [elements/s]$) for ASP-1P, and $4 \times f_C\ [elements/s]$ for ASP-4P, we run significantly longer simulations ($n = 10^8$) than the CPU case to have stable readings on the instruments.

The first row in Table 3 refers to measurements done when the FPGA is not programmed (no circuit mapped). The last column in Table 3 (marked "Board") shows the difference between the total power dissipation measured by the probe and the power dissipated in the FPGA chip. That is, the power dissipated by the other components on the board mounting the FPGA. The results of the experiments show that the average power dissipated in the board (except the FPGA chip) is about 441 mW.

The total power dissipation in the whole system is in the range 1.9–2.4 W, and it is much less than the power required to run the simulation in the PC.

	freq. $[MHz]$	exec. time $[ms]$	P_{core} $[mW]$	P_f $[mW]$	E_{pc} $[10^{-11}J]$	P_{stat} $[mW]$	P_{dyn} $[mW]$	E_{sim} $[mJ]$
ASP-1P	50	20.0	426	214		156	58	-
ASP-1P	100	10.0	484	272	1.16	156	116	2.72
ASP-4P	50	5.0	582	370		165	205	-
ASP-4P	100	2.5	787	575	4.10	165	410	1.44

Table 4: Power dissipation components of FPGA core for ASP-1P and ASP-4P ($n = 10^6$).

The most interesting column of Table 3, marked in red, is the power dissipation measured in the FPGA core (1.0 V), called P_{core} in the following. From these P_{core} values, we can derive the static and dynamic portions of the power dissipated by the ASPs.

Each FPGA chip needs some energy to bias the interconnect fabric, the configuration memory, etc.. This required power does not depend on what circuit is mapped on the FPGA, but it depends on the size and features of the specific chip. We call this amount of power P_{bias}, that is the power dissipated in the chip core when the power supply is connected to the chip.

It is not possible to use Adept's software to perform the P_{bias} reading. However, the same FPGA chip, XC5VLX50T, is also mounted on a Xilinx ML550 board [5] which provides physical access to the Kelvin resistor connected to the core's DC-DC converter. By performing the measurement by a multimeter on the ML550 board, we found a $P_{bias} = 212\ mW$ for the XC5VLX50T chip [6].

Once determined the P_{bias}, the dynamic and static portions of the power dissipated in the ASP can be computed by two readings a different operating frequencies. By calling $P_f = P_{core}(@f) - P_{bias}$, the dynamic P_{dyn} and static P_{stat} power dissipation can be obtaining by solving:

$$\begin{cases} P_{50} &= P_{stat} + E_{pc} \cdot 50 \\ P_{100} &= P_{stat} + E_{pc} \cdot 100 \end{cases} \tag{4}$$

for the two measurements at f =50 and 100 MHz, and where the energy-per-cycle is defined as:

$$E_{pc} = \frac{P_{dyn} \cdot t_{exec}}{\textit{(n. clock cycles)}} \quad [J] \tag{5}$$

By applying the Eq. (4) and Eq. (5) to the P_{core} values (RUN state) of Table 3, we obtain, for ASP-1P and ASP-4P, the power dissipation data of Table 4.

A graphic display of power dissipation as frequency increases is shown in Figure 6. The measured values of P_{core} in RESET state are close to the values of $P_f(@\ 0\ Hz)$: 386 mW vs. 368 mW at 50 MHz, and 387 mW vs. 377 mW at 100 MHz.

Finally, by considering the energy-per-simulation ($E_{sim} = P_f \times t_{exec}$), last column in Table 4, the ASP-4P is almost twice more energy efficient than the ASP-1P ($272/144 = 1.88$).

4.3 Summary of Experiments

By comparing the experimental results of Section 4.1 and Section 4.2, we derive that the Monte Carlo simulation on the FPGA ASPs is much faster and power efficient than its execution on the PC.

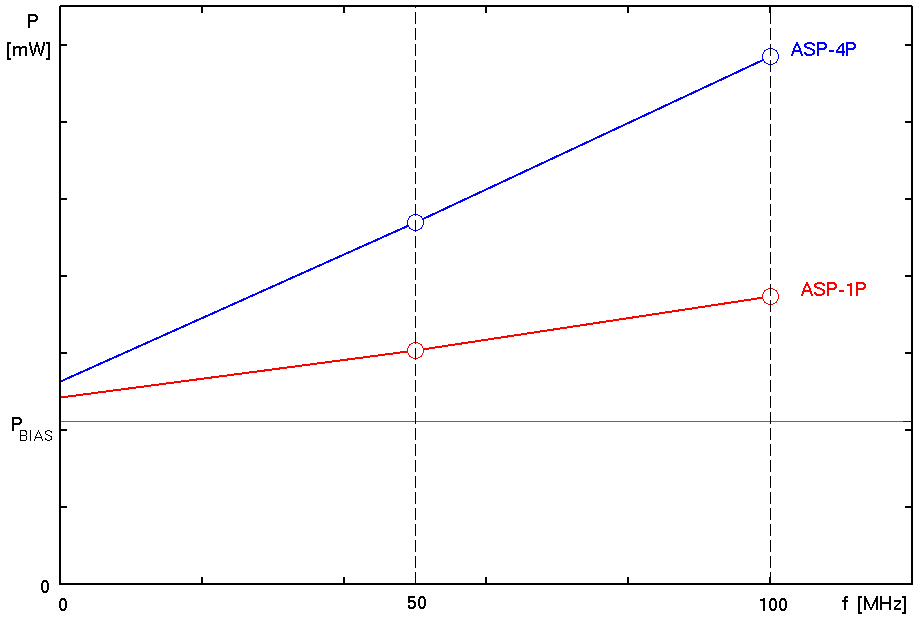


Figure 6: Power dissipation components of FPGA core, P_{core}, for ASP-1P and ASP-4P.

In this section, we finalize the comparison by indicating a realistic acceleration scenario and by evaluating the advantages of having FPGA based acceleration.

We assume to run simulations on 100 different option prices for $n = 10^6$ random values per simulation.

PC only simulation

By running the 100 simulations on the PC at f_{MAX} it requires (according to Table 2)

$$t_{exec}^{Fmax} = 100 \times 0.384 = 38.4 \quad [s] \tag{6}$$

The power dissipation in the CPU during the simulations (Table 1) is $P_{CPU}^{Fmax} = 17.78\ W$ and $9.6\ W$ are necessary for the simulation.

The total energy required by the CPU to run the 100 simulations is then:

$$E_{CPU}^{Fmax} = 17.78 \times 38.4 = 682.75 \quad [J] \tag{7}$$

By running the simulations at the lower frequency (1.2 GHz) the total energy is not reduced (lower power dissipation, but double execution time):

$$E_{CPU}^{Fmin} = 11.45 \times 76.8 = 899.97 \quad [J].$$

PC+ASP simulation

We use the PC to transfer the input parameters to the FPGA based accelerator and to receive back the estimated values of the options. This operation can be done by the PC in batch mode. By assuming the PC is consuming $P_{idle} = 8.18\ W$ during the simulations (executed on the FPGA) and assuming to implement to the ASP-4P processor (running at 100 MHz), we have:

$$t_{exec}^{ASP-4P} = 100 \times 2.5 \cdot 10^{-3} + 10\% = 0.275 \quad [s] \tag{8}$$

by assuming a data-transfer overhead of 10%.

Because we need to supply power to the whole board, according to Table 3, the total power dissipated (measured at the probe) during the simulations is $P_{FPGA}^{ASP-4P} = 2.38\,W$.

Consequently, the energy dissipated by the whole system on the board is:

$$E_{FPGA}^{ASP-4P} = 2.38 \times 0.275 = 0.65 \quad [J].$$

By putting together the energy necessary for the PC and the FPGA during the ASP accelerated simulations, we have:

$$E_{TOT} = (P_{CPU}^{idle} + P_{FPGA}^{ASP-4P}) \times 0.275 = (8.18 + 2.38) \times 0.275 = 2.90 \quad [J] \tag{9}$$

In conclusion, the results of the comparison on latency, Eq. (6) and Eq. (8), and energy consumption, Eq. (7) and Eq. (9), between the PC based simulations and the combined PC plus accelerator approach, show that not only, the system CPU+ASP is about 140 times faster, but also that the simulations can be run by consuming less than 1/100 of the energy.

5 Conclusions and Future Work

In this technical report, we explained how to measure the power dissipated by a CPU running some application program and by an FPGA accelerator implementing an Application Specific Processor executing a given algorithm.

In future work, we plan to integrate the data transfer CPU-FPGA in the computation flow and to precisely benchmark the energy consumption of an application running on the FPGA based ASP.

References

[1] J. S. Hegner, J. Sindholt, and A. Nannarelli, "Design of Power Efficient FPGA based Hardware Accelerators for Financial Applications," in *Proc. of the 30th NORCHIP Conference*, Nov. 2012.

[2] Pololu Robotics & Electronics. ACS711 Current Sensor Carrier. [Online]. Available: http://www.pololu.com/catalog/product/2197

[3] H. Esmaeilzadeh, T. Cao, X. Yang, S. M. Blackburn, and K. S. McKinley, "What is Happening to Power, Performance, and Software?" *IEEE Micro*, vol. 32, no. 3, pp. 110–121, May-June 2012.

[4] J. C. Hull, *Options, Futures and other Derivatives*, 8th ed. Prentice Hall, 2012.

[5] Xilinx User Guides. "ML550 Networking Interfaces Platform". UG202 (v1.4) April 18, 2008. [Online]. Available: http://www.xilinx.com/support/documentation/boards_and_kits/ug202.pdf

[6] J. S. Hegner, "Design of Power Efficient FPGA based Hardware Accelerators," M. Sc. thesis, Technical University of Denmark, Kongens Lyngby, 2012. [Online]. Available: http://www.imm.dtu.dk/pubdb/views/publication_details.php?id=6485