Comparative study of power reduction of various solutions of an image processing IP on modern FPGAs

Henryk BLASINSKI, Frederic AMIEL, Thomas EA

Institut Supérieur d'Electronique de Paris
21 rue d'Assas, 75006 Paris, France
firstname.lastname@isep.fr

Abstract

In this paper, we describe the applicability and efficiency of several power reduction techniques applied on a modern 65nm FPGA. For the given algorithm, image erosion and dilation, two major solutions were tested and compared with respect to power and energy consumption. Firstly the algorithm was run on a general purpose processor (gpp) NIOS and then hardware architecture of an Intellectual Property (IP) was designed. Furthermore IPs design was improved by applying a number of power optimization techniques. They involved RTL level clock gating, power driven synthesis with fitting and appropriate coding style. Results show that hardware implementation is much more energy efficient than a general purpose processor and power optimization schemes can reduce the overall power consumption on an FPGA.

Keywords

FPGA, Power Optimization, Dilation, Erosion, General Purpose Processor, SoPC.

1. INTRODUCTION

FPGAs have become as powerful as ASICs for current applications, mainly thanks to a reduction in the gate length. However because of the same reason the static power consumption of FPGAs has started to increase dramatically. Thus, when designing a digital circuit more and more often FPGA users can be provided with a number of techniques to reduce power consumption. Some of them come directly from the circuit manufacturers, who have started to realize the importance of low power designs. This is the case of the state of the art 65nm FPGAs that are now available on the market. Some of the techniques are technology dependent and thus not accessible to the designer and will not be discussed in detail. They include variable gate oxide thickness transistors, the use of low K dielectric materials or strained silicon, which improves carrier mobility. Other ones let the user decide about the compromise between the performance and power consumption. In the case of Stratix III device the designer can supply the core with 1.1 or 0.9 V which gives grounds for dynamic voltage scaling techniques. The others reduction techniques come from ASIC such as clock gating, coding style, etc.

The goal of this article is, for a given data flow algorithm to point out the efficiency of power reduction techniques in FPGA, including comparison between processing done by IP and general purpose processor (GPP) running in different modes. Quantitative data coming from experimental measurements demonstrates the contribution of each presented reduction techniques and allows us to induce the applicability of these techniques.

This paper is organized as follows: in section 2, we introduce the iris recognition algorithm and especially erosion and dilation computation. Section 3 presents the IP circuit design. The optimization techniques for reducing power consumption are presented in section 4. Finally in section 5, results and comparisons are presented before concluding on efficiency of lower power reduction techniques on FPGA.

2. IRIS RECOGNITION ALGORITHM

The iris recognition algorithm developed by the research team at ISEP [1,2] is composed of four major parts. Firstly the image has to be pre-processed and iris has to be segmented out from the image, then it is unwrapped to a rectangular shape, thirdly it undergoes the wavelet decomposition and finally features are extracted, compared with the database and a decision is taken. Program profiling has shown that one of the longest parts of the algorithm is image segmentation. Within that part, in order to determine the boundaries of the iris and pupil a number of operations based on morphological image processing is performed. In all of the cases they can be decomposed into a series of image erosions and dilations with various structuring elements (masks). These operations require processing large amounts of data, and therefore their execution on a general purpose processor NIOS is time consuming. Due to their frequent calls and dataflow character we believe that these two functions are good candidates for hardware implementation.

Erosion and dilation, in more general terms are used to extract properties of interest from the image while neglecting irrelevancies. Morphological operations can be easily explained via the set theory [3]. Say we have two sets A and B. Let the translation of B to the point x, denoted B_x be defined as:

$$B_x = \{e | e = b + x, b \in B\}$$

And the reflection of B about the origin, denoted $\hat{B}$:

$$\hat{B} = \{x | x = -b, b \in B\}$$

Then operations of erosion ($\odot$)and dilation ($\oplus$) are given by:

$$A \oplus B = \left\{x \middle| \widehat{B_x} \cap A \neq \emptyset\right\}$$
$$A \odot B = \{x | B_x \subseteq A\}$$

It is common to call the set B, which is generally smaller than the image A, the structuring element (SE) or the mask. Structuring elements are simply matrices of zeros and ones, usually of odd dimensions allowing to define the central pixel as the reference and symmetrical about this reference. In other words set B defines a certain neighbourhood around its central point. Now consider operations of erosion and dilation, applied to a greyscale set A, the image. Elements of A, pixels, can attain values from 0 to 255. Morphological operation on a pixel at the position (x,y) may be expressed as finding the minimum (erosion) or maximum (dilation) value in the neighbourhood of the pixel defined by the structuring element B and assigning that value to this pixel.

3. IP DESCRIPTION

Each operation of erosion or dilation requires processing large amounts of data, especially if the size of the structuring element becomes important. If we suppose that our image has the width *w*, length *l*, and the structuring element being a square of the size *a*, the total number of pixels needed to be processed is equal to:

$$Pixel\ count = l \cdot w \cdot s^2$$

For the test image used, that is 202x202 pixels and a square mask of 23 this gives us:

$$Pixel\ count = 202 \cdot 202 \cdot 23^2 \approx 21{,}5\ Mpixels$$

In the most efficient case each pixel is copied from the memory to the hardware block only once[5], and stored in the IP as long as the processing requires it. In such a circuit the calculation time is in general proportional to

$$Time \approx \frac{w \cdot l}{frequency}$$

For a 202x202 pixel image, and operating frequency of 100 MHz this gives us the best execution time of the order:

$$Time \approx \frac{202 \cdot 202}{100 \cdot 10^6} = 0{,}4ms$$

Such a solution, even though extremely fast, does require huge amount of logic elements. In our case we were aiming at a much slower and smaller circuit with the operating time of the order of a few tenths of a second.

The proposed hardware implementation has two interfaces, one is an Avalon slave, by writing to which we send image pixels, determine mask coefficients, image size and other process parameters. Ideally the configuration is done via the NIOS processor, which later on launches the DMA which initiates memory transfers of the image to the IP. The second interface is an Avalon Master, which writes the calculation results directly to memory at the address specified during configuration. Once the last pixel of the image is written, an interrupt is generated.

The proposed IP is composed of four main modules (Figure 1). The memory, in which all mask coefficients together with image parameters are saved, a form of a FIFO where pixels received via the Avalon bus are stored and if needed fed to the kernel. At the input of the FIFO pixels arrive one by one, however at the output they have to be shifted accordingly to the structuring element size. The kernel as input is accepts a line of the image, equal to the length of the currently used mask. These lines are red column wise in a quasi raster scan order starting from the leftmost upper pixel of the entire image. Such pixel arrangement corresponds to moving the mask downwards along the column (Figure 2). Once the end of one column is reached, the starting position is shifted by one pixel to the right and the whole process restarts. Such architecture in the worst case scenario requires retransmission of each pixel as many times as the size of the mask at the same time diminishing the speed by an identical factor. Inside the kernel the masking operation is performed, that is only the corresponding bits are taken into account during the calculation of the minimum (erosion) or maximum (dilation) which is then available at the output. The control logic is responsible for receiving slave transfers, initiates master transfers and interrupt requests and finally generates gated clocks for the kernel, FIFO and memory blocks.

4. OPTIMISATION TECHNIQUES

4.1. Structural optimisation

The manufacturer already provides some tools to

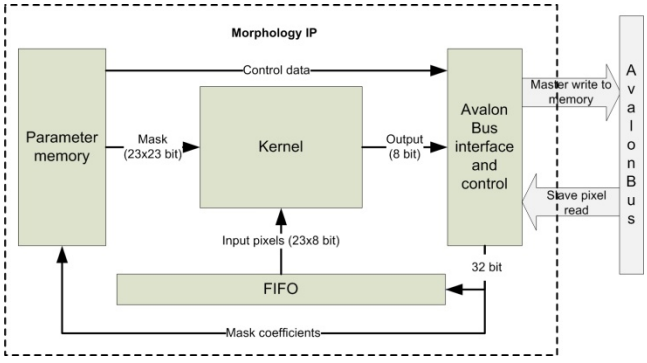

Figure 1. Morphology IP internal architecture.

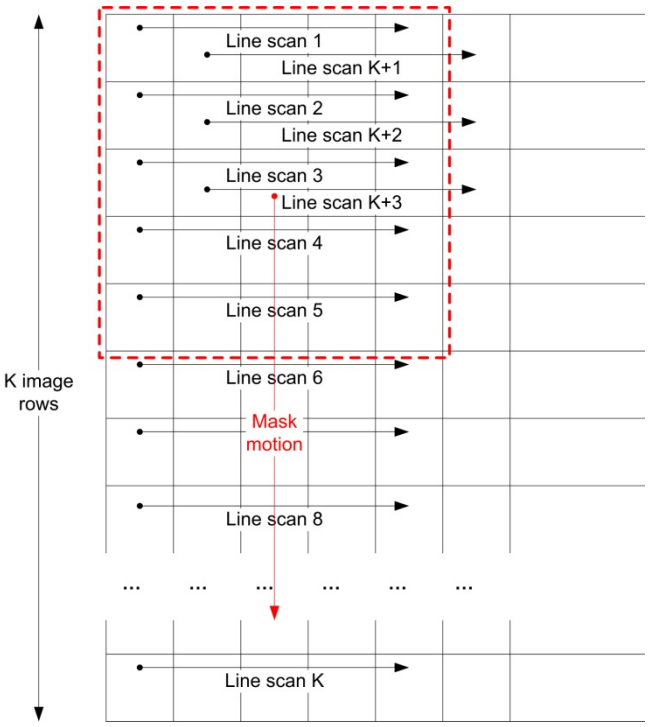

Figure 2. Pixel scanning order and corresponding mask (5x5) motion.

create power aware designs. Logic elements inside the core of Stratix III are separated into tiles, which can operate in two modes: performance or low power. This technique exploits the fact that only a small percentage of paths in the design are critical and requires high speed, while the constraints for others are much lower. This feature is not available directly but by means of the PowerPlay analyzer, in which user defines to what degree the design has to be optimized versus power consumption.

4.2. Architectural optimisation

Clock gating

The most popular technique is clock gating which consists in deactivating the clock signal to the parts of the circuit which are not in operation. This improvement realized at the HDL level of IP description was applied in our circuit. There were two clock domains, one corresponded to the mask memory sub-circuit, and the other one to the FIFO, kernel and control logic connected with master transfers. The slave transfers control logic remained active at all times. Clocks were activated and deactivated by means of commands issued by the NIOS processor. The second possibility of clock gating is done not at the level of HDL; but directly with the use of PLL's. Different clock domains are connected directly to the PLL which is being switched on and off by the NIOS.

Bus transfers reduction

In the initial version of the IP each master transfer was sending only one pixel (8 bits) to the memory. This system does not exploit the full width of the bus (32 bits), which could reduce the number of transfers by four. This improvement is interesting not only as it reduces the congestion but also reducing the number of transfers can diminish the overall consumption since in current circuits busses usually have large parasitic capacitances.

Glitch elimination

As described in literature [4] one of the potential sources of power losses are glitches, that is signal fluctuations arising during switching but not affecting the functionality and output of the given circuit. In our IP potential sources of such events are 23 input comparators. One could try to avoid this effect in a number of ways. The seemingly simplest solution would be to insert registers between levels of the comparator tree, however as a result we would have a sub-system with a considerably higher latency which would have to be accounted for later on. Since registers are not very power efficient the overall result, even though removing glitches, would possibly consume much more power. An interesting alternative however represents the proper coding style which becomes crucial for binary images. It is enough to code such an image as 0's and 1's instead of 0's and 255's reducing in this way the number of signal transitions during processing.

In our circuit we have tested the performance of manufacturer provided power optimization via PowerPlay, the HDL level clock gating, software coding style and finally the influence of the bus width.

5. RESULTS

5.1. Platform

In order to test the efficiency of these power optimization techniques in practice we have chosen the Stratix III FPGA, which is a state of the art 65nm circuit operating at up to 550MHz. In such high performance FPGAs the power consumption becomes an important issue and therefore power saving schemes is of particular interest. The model used, EP3SL150F1152, consists of about 113 000 logic elements and 736 pins. It is delivered on a development board with a range of supplementary components (memories, displays, I/Os etc.). The interesting features of the board are the built in resistances for measuring voltage and current drawn by different parts of the FPGA. In parallel a CPLD MAXII together with a 24bit ADC is used to measure and display these values.

5.2. Method

We have created a SoC which was composed of a NIOS II fast processor with 4kbytes of data and instruction caches, JTAG interface, 150 Kbytes of on chip memory, DDR2 interface, DMA, timer, button and LED I/Os and a performance counter. The last part is used to measure the execution time of parts of code with the clock period accuracy. All blocks were operating at 100MHz. In order to verify the efficiency of our power reduction schemes we

have created a number of systems on chip each with an IP optimized with a different technique. This allowed us to determine their efficiency independently from one another. As the static power consumption reference we used a simple AND gate connected to pushbuttons and a led. For all erosions and dilations tests the same 202 x 202 pixel 8 bit greyscale image was used on which a square mask 23 x 23 bit was applied. Table 1 summarizes the logic element utilization of proposed solutions.

5.3. Architectural optimisations

Table 2 summarizes the power consumption of various techniques. The IP power consumption is estimated to be about 200mW, which is the difference between systems with and without the IP in which NIOS performs the same operations (lines 2,4 or 3,6). Line 1 gives us the estimate of the static power consumption since we implement a simple AND gate only. Lines 2 and 3 correspond to the system without the IP in which the NIOS is either idle (line 2) or executing the software version of image erosion (line 3). We suspect that the reduction of power consumption between these two cases is due to unmeasured power dissipated by the DDR2 block, not used in the idle mode. Line 4 corresponds to the SoC with an implanted IP. In this particular case the power consumption difference is 173mW, if the NIOS is executing erosion or dilation, this value reaches 200mW. Lines 5,7,9 and 11 confirm that clock gating reduces power consumption of the IP between 70% (software algorithm, lines 6,7) and 10% (hardware algorithm, lines 8,9). Even though the system with IP consumes more power, the execution time is greatly reduced providing a decrease of total energy used. Lines 10 and 11 discuss the effect of diminishing the number of transfers. Globally the consumption remains rather similar, since the activity of on the bus does not change, and additional buffer had to be introduced to generate 32 bit transactions. Finally lines 12 and 13 present the influence of coding style, even though it proves to have some influence its applicability is limited to binary images only.

5.4. Structural optimisations

The following two tables (Table 3, Table 4) summarize the influence of a power driven synthesis. Even though the number of high performance tiles is smaller in the extreme effort case however as shown, not in all conditions does it diminish power consumption.

In any case it can clearly be seen, that the SoC hardware implementation of morphological algorithm, consumes about 20 times less energy than the corresponding software implementation.

Table 1 Logic utilisation of various SoC solutions.

SoC	Combinatorial ALUTs	Memory ALUTs	Registers	Memory [bits]
AND 2 input	1	0	0	0
NIOS	8 001	32	6 142	1 358 576
NIOS + IP	28 726	34	14 875	1 358 448

Table 2 Power and energy consumption of IPs with various power optimisation techniques applied.

No.	SoC	Algorithm	Power [mW]	Time [s]	Energy [J]
1	AND 2 input	None	901	-	-
2	NIOS	usleep()	1 717	-	-
3	NIOS	Software	1 675	5,038	8,439
4	NIOS+IP	usleep()	1 890	-	-
5		usleep() with clk gating	1 734	-	-
6		Software	1 875	5,832	10,935
7		Software with clk gating	1 734	5,833	10,114
8		Hardware	1 783	0,249	0,444
9		Hardware with clk gating	1 760	0,250	0,440
10		Hardware 32bit master	1 809	0,249	0,450
11		Hardware 32bit master with clk gating	1 770	0,249	0,441
12		Hardware, binary image (0,255)	1 803	0,250	0,451
13		Hardware, binary image (0,1)	1 794	0,250	0,449

Table 3 Power driven synthesis influence on power reduction.

SoC	PowerPlay	Power [mW]	Time [s]	Energy [J]
NIOS + IP	Off	1 783	0,250	0,446
	Normal	1 789	0,250	0,447
	Extra effort	1 787	0,236	0,422
NIOS + IP with clk gating	Off	1 761	0,249	0,438
	Normal	1 760	0,249	0,438
	Extra effort	1 743	0,235	0,410

Table 4 Power driven synthesis vs. tiles utilisation.

Power Play	Tiles			LAB Tiles		
	High speed	Low power		High speed	Low power	
		used	unused		used	unused
Off	381	1 485	1 393	340	1 485	1 015
Extra effort	348	1 420	1 491	304	1 420	1 116

6. CONCLUSIONS

First of all, our hardware IP has reduced the execution time and we have reached the limit we were aiming at. Obviously this implementation is much larger than the pure NIOS system, however once we start to consider energy consumption, the hardware solution is much more efficient. It can also be seen, that clock gating the HDL level does reduce power consumption in all cases where it was applied. The study has also revealed the importance of the appropriate coding style, which reduces transitions between signal states. Contrary to our expectations, reducing the number of master transfers does not diminish the power consumption even increasing it slightly. Nevertheless in complex systems this is a price worth paying for a four times smaller congestion on the bus. Presented results are insufficient to draw any conclusions as far as the manufacturer provided power optimization tools are concerned. In some cases they have proven to significantly diminish the power consumption, in others however its influence was even the opposite.

In future we are planning to further exploit power optimization techniques which were discussed in earlier sections, namely clock gating via a dedicated PLL. Realising the dataflow nature of erosion and dilation we are currently designing another type of IP, more control like. The applicability of all of these techniques will be tested for the other type of the circuit and another family of FPGAs, Cyclone III, which has been designed as a cost efficient device, and therefore is less optimized by the foundry with respect to power consumption. The relative reduction of power consumption via these techniques may prove to be different.

ACKNOWLEDGMENTS

This work was a part of a study realized for Thales group. The first author would like to thank prof. Andrzej Napieralski and Ph. D. Małgorzata Napieralska from Department of Microelectronics and Compter Science, Technical University of Lodz for their support and helpful advice.

REFERENCES

[1] E. Rydgren et al., 'Iris Features Extraction Using Wavelet Packets,' IEEE International Conference on Image Processing, October 2004, Singapore

[2] F.Rossant, M.Torres Eslava, T.Ea, F.Amiel, A.Amara, "Iris Identification And Robustness Evaluation Of A Wavelet Packets Based Algorithm", ICIP'05, Genoa, Italy

[3] R. Gonzales, R. Woods, 'Digital Image Processing,' Third edition, Prentice Hall 2008

[4] C. Piguet et al., 'Low-power Electronics Design,' CRC Press 2005

[5] A. Amara, F. Amiel, T. Ea, 'FPGA vs. ASIC for low power applications,' Microelectronics Journal, July 2006, Elsevier Ltd.

[6] S. Hun Jin, J. Uk Cho, J. Wook Jeon, 'Pipelined Virtual Camera Configuration for Real-time Image Processing based on FPGA,' Proceedings of the 2007 IEEE International conference on Robotics and Biomimetics