Computing to the Limit with Heterogeneous CPU-FPGA Devices in a Video Fusion Application

Jose Nunez-Yanez, Department of Electrical and Electronic Engineering,
University of Bristol, MVB Building, Woodland Road,
BS8 1UB, Bristol, UK

j.l.nunez-yanez@bristol.ac.uk

Abstract. This paper presents a complete video fusion system with hardware acceleration and investigates its performance and energy optimization. The video fusion application is based on the Dual-Tree Complex Wavelet Transforms (DT-CWT). Video fusion combines information from different spectral bands into a single representation and advanced algorithms based on wavelet transforms are compute and energy intensive. In this work the transforms are mapped to a hardware accelerator using high-level synthesis tools for the FPGA resulting in an increase of performance of the system by a factor of 3. In a second stage the hardware engine is transformed with a set of tools and detector blocks that allow the CPU device to monitor and pre-detect timing failures in the FPGA fabric. The CPU device can control the voltage and frequency of the FPGA and obtain a new working point with reducing power requirements by approximately 70% with an equivalent performance level. This results in an energy proportional computing system in which only the energy required to maintain a required level of performance is used.

1 Introduction

Multi-sensor video data with visible and infrared images is increasingly being utilized in applications such as medical imaging, remote sensing and security applications. Multi-sensor data presents complementary information about the region surveyed and fusion provides an efficient method to combine the complementary information for better data analysis. Video fusion is just a special case of image fusion when two or more frames of different video sources are fused together continuously into a single fused video [1]. Image fusion can be performed based on wavelet transform techniques [2]. Compare to other schemes [3], wavelet transform achieves better signal to noise ratios and improved perception with no blocking artefacts. Moreover, among all the wavelet transform that applied to multifocal, remote sensing and medical image fusion, the use of the Dual-Tree Complex Wavelet Transform (DT-CWT) has been shown to produce significant fusion quality improvement [4]. The algorithm described in [4] is used in this paper which consists in applying DT-CWT to infrared and visible frames, combining the obtaining coefficients using a fusion rule and then proceeding to perform the inverse DT-CWT for reconstruction.

The proposed system is based on the ZYNQ System-on-Chip and the CPU and the FPGA work together to run the algorithm. The whole system runs under the Linux OS with a customized kernel level Linux driver. The main contributions of this paper are:

1. We create an open-source complete fusion system including processing engine, drivers, hardware interfaces and cameras. The most compute intensive parts of the algorithm are accelerated based on HLS tools using the FPGA.
2. We demonstrate the performance and energy advantages of using a heterogeneous platform for video fusion comparing to a software-only solution.
3. We extend this system with energy proportional techniques that maintain the same level of performance but they reduce power and energy significantly.

The remaining of this paper is organized as follows. Section II lists related work in this research area. Section III provides some fundamental knowledge of the DT-CWT based fusion algorithms and Section IV discusses the energy proportional implementation flow. Section V introduces our hardware architecture to implement the DT-CWT with a customized kernel level Linux driver, followed by Section VI, which presents our system architecture to capture and fuse multi-sensor data. Section VII compares the performance, power and energy consumption of the fusion application running in the ARM CPU and FPGA configurations including the use of the energy proportional approach. Finally, section VIII concludes the paper.

2 Related Work

Previous research on FPGA-based fusion systems is available in recent literature. Jasiunas et al. [5] presented a wavelet based image fusion system for unmanned airborne vehicles. This is a very early attempt to develop image fusion systems on reconfigurable platform alone that achieved latency of 3.81 ms/frame for visible and infrared 8-bit images of 512x512 pixel resolution. Sims and Irvine [6] presented an FPGA implementation of pyramidal decomposition based video stream fusion. This framework can achieve a 30 frame/s, real-time fuse of video streams in grayscale video graphic arrays (VGA). Yunsheng et al. [7] presents a real-time image processing system to combine the video outputs of an uncooled infrared imaging system and a low-level-light TV system. Song et al. [8] proposed an image fusion implementation based on Laplacian pyramid decomposition of two-channel VGA video for a better fusion quality and reasonable frame rate of 25 frame/s. Mohamed and El-Den [9] applied five different measures to evaluate the performance of several different fusion techniques and the hardware implementation of DCT, DWT and PCNN-based fusion algorithms are studied. However, although these designs achieves performance enhancement to do image fusion on FPGA, the fusion algorithms they used are not state-of-the-art.

Tao et al. [10] proposed an image enhancement and fusion system to improve visibility. In this paper, two videos are captured by CCD and LWIR cameras and fused by implementing DT-CWT fusion algorithms in Xilinx Virtex-II environment. Gudis et al. [11] built an embedded vision service framework on ZYNQ SoC with a “plug-and-

play" capability to allow the service-based software to take advantage of the hardware acceleration blocks available and perform the remainder of the processing in software. These designs share some similarities with our system but focus on the fusion quality more than the performance and energy efficiency.

3 The DT-CWT Based Fusion Algorithm

The aim of the wavelet transformation is to represent signals using a superposition of wavelets. The Discrete Wavelet Transform (DWT) is a spatial-frequency decomposition of a signal, which ensures the signal being decomposed into normalized wavelets at octave scales [12]. When applied to two-dimensions, signals are separately filtered and down-sampled in the horizontal and vertical directions. This creates four sub-bands at each scale, namely high-high (HH), high-low (HL), low-high (LH) and low-low (LL), as shown in Fig. 1. The name of each sub-band denotes the horizontal frequency first and then the vertical frequency. A multi-resolution decomposition of image can then be achieved by recursively applying filtering to the low-low sub-band.

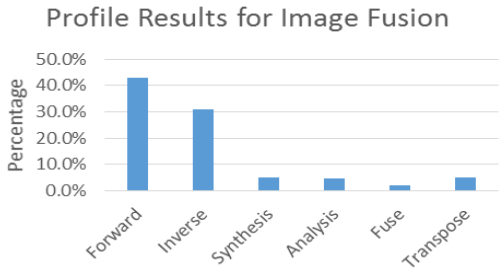

Fig. 1 Profiling results of fusing two input images

The DT-CWT transforms signals use two separate DWTs and apply spatial filters recursively to create frequency sub-bands. The application of DT-CWT to 2-D image is achieved by separable complex filtering in two dimensions. The DT-CWT is able to distinguish between positive and negative orientations and divides the horizontal and vertical sub-bands into six distinct sub-bands at each scale with the orientations of ±15°, ±45° and ±75°. Moreover, the DT-CWT gives perfect reconstruction due to the biorthogonal nature of the filters and also delivers approximate shift-invariance.
In this paper, the whole fusion algorithm with the forward and inverse DT-CWTs is written in C++ and executed by the ARM Cortex A9 Processor. The profiling results of the fusion process, as shown in Fig. 1, indicate that the forward and inverse DT-CWT are the most compute- and energy intensive tasks. Therefore, these parts of the algorithm are the ones selected for acceleration in the FPGA. The FPGA fabric available in the ZYNQ device can be made coherent with processor caches using the (Acceleration Coherence Port) ACP and this is the option that has been selected in this paper. To achieve the FPGA acceleration, the forward and inverse DT-CWT were mapped to the PL (FPGA) side of ZYNQ to create a hardware wavelet engine con-

trolled by the PS (CPU) side. This means that the input images are decomposed and reconstructed in hardware. The hardware accelerator has been created using the VIVADO_HLS high-level synthesis tools increasing productivity compared with a traditional RTL design. This also enables a fast exploration of the design space. The original C++ code makes use of float data types and operators. Fig. 2 shows the results of the design space exploration phase in which the floats have been replaced with fixed point data types with different number of fractional bits maintaining the integer bits constant at 12 to avoid overflows. These exploration can be done in a few hours and shows that there are important benefits in terms of performance and complexity by using the fixed point data types as expected. Fig. 2 also shows how PSNR reduces as accuracy reduces but this happens linearly In this research the results based around <24,12> accuracy (36 bits accuracy) have been selected since PSNR remains about 40 which is considered high quality while the number of DSP blocks required reduces significantly.

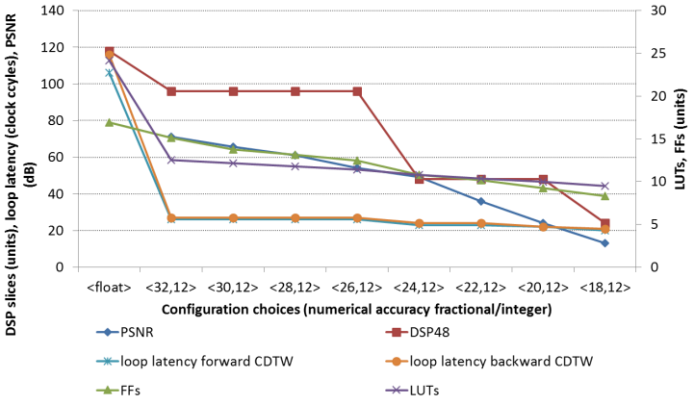

Fig. 2 Design space exploration as a function of accuracy

4 Energy Proportional Implementation Flow

The energy proportional implementation flow introduces the in-situ detectors in the design netlist guided by post place&route timing information. The core of the flow is the Elongate tool [13] that transforms the original design netlist into a new netlist with identical functionality and additional power management IP and in-situ detectors. Fig.3 shows the overall flow that can be decomposed into three distinct phases. During the first phase the original netlist goes through a full implementation run to obtain post place&route timing data in the form of a text file. In the second stage the Elongate tool takes as input the obtained timing data, the original netlist and Elongate component library that describes the power management core and in-situ detectors and produces the new power adaptive netlist. The third stage consists of a final implementation run of the power adaptive netlist to obtain the device bitstream ready to be downloaded in the device.

The input into the flow in Fig.3 is a C++ description of the forward and inverse DT-CWT that are initially compiled with Vivado HLS tools to obtain a netlist in either VHDL or Verilog format. This netlist is then synthesized by the Synplify synthesis tool to obtain a new VHDL netlist based on the implementation primitives available in the target technology. These initial synthesis steps are required to obtain the netlist that will be processed by Elongate. The need for this initial pre-processing is because the Elongate transformation does not take place at source level directly. The reason is that slight changes in the source can have a large effect on timing and also because it is possible to annotate the critical paths found after static timing analysis with the physical flip-flops in the netlist. The timing information is critical to allow Elongate to replace the end-point flip-flops in the critical paths with new soft-macro flip-flops that incorporate the in-situ detection logic. Each primitive flip-flop component in the technology library has a corresponding soft-macro flip-flop stored in the Elongate component library with identical functionality. Part of the user constraints input to Elongate indicate the level of coverage requested for the critical paths in the design. The coverage must be sufficient so that the critical paths of the final design have as endpoints the newly inserted soft macro flip-flops.

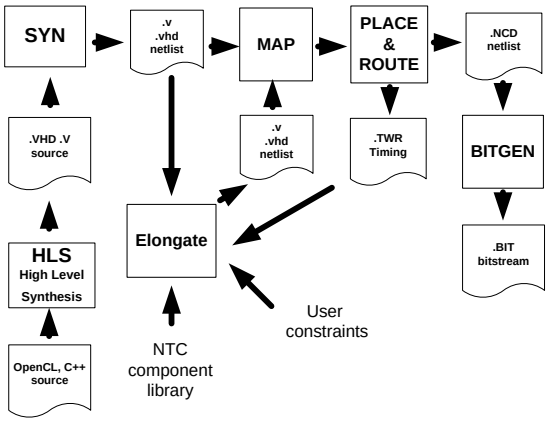

Fig. 3 Energy Proportional Implementation Flow

If there is not enough coverage then the final implementation netlist could have critical paths not protected by the soft macros and the design could not operate reliably across the range of frequencies and voltages considered. To detect this situation the tool analyzes the final timing data to verify that the critical paths end in soft-macro flip-flops and that the slowest main flip-flop is located inside a soft-macro. If these constraints are not met the designer is informed so that a new run can be launched using a different path coverage value. As a rule of thumb our experiments have indicated that a coverage level of 10% of the total number of flip-flops is sufficient but this is ultimately dependent on how balanced the signal paths in the design are. The addition of the detectors results in a largely unaffected critical path while complexity increases by around 5%.

5 FPGA Acceleration

The ZYNQ Processing System and Programmable Logic (PS-PL) interface is created to transfer commands, filtered coefficients, transformed coefficients and pixel data between the PS and the PL. The general purpose 32-bit ports available in Zynq do not obtain the require performance and every transfer requires around 25 clock cycles with the CPU moving the data itself. For this reason we created a custom DMA engine using the synthesis support of memcpy by VIVADO_HLS. The code for VIVADO_HLS is configured to generate two interfaces. An AXI4Lite slave interface is used to load filter coefficients and send commands to the engine to enable the execution of the forward and inverse transform. An AXI4M interface is used to load and store pixel and transformed data using the hardware implemented memcpy function through the ACP port. Fig.4 shows a section of the code corresponding to the forward wavelet transform synthesized into FPGA logic and memory by the VIVADO_HLS tools.

```
//read data
memcpy(buff_in, (float *)(memory + in_offset), (outwidth * 2 + 12)*sizeof(float));

wav_engine_master_label0:for (int i = 0; i<(outwidth + 6); i++)
{

	input_a = (data_t)buff_in[i * 2];
	input_b = (data_t)buff_in[i * 2 + 1];

	hpMult = coeff_register_hp[0] * shift_register[0];
	lpMult = coeff_register_lp[0] * shift_register[0];
	hpAcc = hpMult;
	lpAcc = lpMult;

	wav_engine_master_label1:for (int j = 1; j < 11; j++)
	{
		lpMult = coeff_register_lp[j] * shift_register[j];
		hpMult = coeff_register_hp[j] * shift_register[j];
		hpAcc += hpMult;
		lpAcc += lpMult;
		shift_register[j - 1] = shift_register[j + 1];
	}
	lpMult = coeff_register_lp[11] * shift_register[11];
	hpMult = coeff_register_hp[11] * shift_register[11];
	hpAcc += hpMult;
	lpAcc += lpMult;
	shift_register[10] = input_a;
	shift_register[11] = input_b;
	if (i > 5)
	{
		buff_out[i * 2 - 12] = (float)hpAcc;
		buff_out[i * 2 + 1 - 12] = (float)lpAcc;
	}
}
//write data
memcpy((float *)(memory + out_offset), buff_out, (outwidth * 2)*sizeof(float));
```

Fig. 4 Sample code Extraction for FPGA synthesis

The memcpy's move data between the external DDR memories and internal BRAMs and the for loops create the filters with the help of an internal shift register. The final if makes sure that only the correct outputs are written to the output buffers. Additional pragmas are used to ensure that the tool adds the require AXI interfaces and pipeline registers to obtain an initialization interval of one clock cycle so a new input en-

ters the pipeline in each clock cycle. Notice that the current VIVADO_HLS tools do not pipeline the memcpy's that need to complete before the loop processing can start. It is important to note that all the logic required to implement these functions is created on the PL side by VIVADO_HLS. Control variables not shown in this sample code activate one of three possible modes that correspond to 1) filter coefficient loading, 2) forward transforms and 3) inverse transform. With this setup, we wrote a kernel level Linux driver to allocate memory that can be accessed by the accelerator with physical addresses and by the processor with virtual addresses. The driver uses the standard "memcpy" function, implemented in this case in software at the user level, for data transfer. For this to work, it is necessary to obtain the physical addresses at which the memory is created by the "kmalloc" calls in the kernel driver, and then use the memory-map calls "mmap" to obtain remapped virtual addresses in user space that can be used by standard "memcpy". Additionally, the Linux driver implements the "ioctl" function, which can be used to control how the data movements take place. In our case, we used this to create different read and write offsets to the kernel allocated memory. To increase the performance of the system we divided the kernel memory into two areas or buffers. This double buffering mechanism is used to parallelize the transfer and processing of data from user space to kernel space as illustrated in Fig. 5. This approach reduces latency and hardware complexity compared with buffering the whole image in the FPGA memory. The input and output buffers have a size of 4096 32-bit, divided into two areas of 2048 32-bit, which is suitable for an image width up to 2048 pixels. Table I shows the implementation complexity of this hardware wavelet engine.

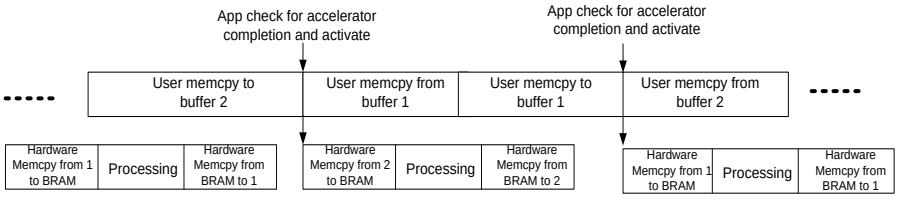

Fig.5 Design of the Kernel Level Linux Driver

TABLE I. IMPLEMENTATION COMPLEXITY OF WAVELET ENGINE

Wavelet Engine	**Implementation Complexity Part: *xc7z020clg484-1***		
	Unitization	***Available***	***Percentage***
Registers	23412	106400	22%
LUTs	17405	53200	32%
Slices	7890	13300	59%
BUFG	3	32	9%

6 The System Architecture

This section describes the overall system architecture we implemented to capture and fuse the multi-sensor data. In this paper, we have used the ZYNQ-based ZC702 Evaluation Board running UBUNTU Linux OS. The the overall architecture are shown in Fig. 6.

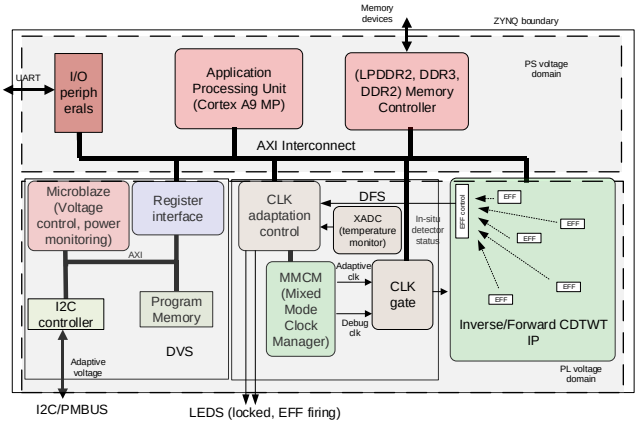

Fig. 6 Overview of the system Design

The data transfer between the PS and the PL is done through the AXI interface. Both input videos are decoded into continuous pixel frames and sent to the wavelet hardware on the PL side for DT-CWT decomposition. The transformed coefficients are sent to the PS for fusion and then sent back to the wavelet hardware for inverse DT-CWT reconstruction. Since the whole system is running under Linux OS, the decoded and the fused videos are shown on screen using OpenCV funtions, with no external video connectors or cables required. Fig. 7 demonstrates the video frame captured by the web-camera and the thermal-camera and the fused frame of the two. The original video captured by the web-camera was gray-scaled before fusing. A video demonstration is available as video 2[1] and a demo package available for the ZC702 board at [2].

7(a) Web-cam frame 7(b) Thermal-cam frame 7(c) fused frame

Fig. 7 Demonstration of the Designed Fusion System

[1] http://www.bris.ac.uk/engineering/research/microelectronics/enpower/demonstrations.html
[2] https://drive.google.com/file/d/0B-8fEeeC7UJvR1FTRy1mUUVuS2c/view?usp=sharing

7 Results Analysis and Comparison

This section compares the fusion performance and power consumption when the forward and inverse DT-CWTs are executed by the ARM processor and the FPGA including configurations that are voltage and frequency scale. Fig. 8 shows the result of profiling the application with and without hardware acceleration. The accelerator is run at two frequencies of 200 Mhz and 100 Mhz and the bars represent the time used in seconds by the corresponding functions. The CDTW functions are the most time consuming in the ARM only version while they do not show up once the hardware is accelerated. This analysis has been done with GNU GPROF function and indicates that the hardware functions after acceleration cannot be detected by GPROF. Since these functions represent 70% of total computation time this means that a total computation time of 1.5 seconds per frame is reduced to 0.4 seconds with means that the whole fusion application is accelerated by a factor of 3 when the FPGA performs the CDTW functions in parallel with the ARM processor. It is important to note that while the accelerator is running the software needs to prepare the next group of pixels and this task is done by the ARM that must move data to an area that can be accessed by the hardware engine as shown in the upper part of Fig 5. This data movement is the performance limiting factor and not the hardware computation and this means that the hardware configurations at 100 and 200 MHz obtain the same performance level. The conclusion is that if we run the hardware slower we can maintain the same level of performance but we can save power and energy by decreasing the frequency and voltage levels. Fig. 9 shows how reducing the FPGA frequency from 200 Mhz to 100 Mhz and its voltage from 1v to 0.8v reduces power and energy and increases the amount of time required to complete the DT-CWT transforms but this time still remains below the amount of time required by the processor to prepare the next group of pixels so there is no negative effect. Fig. 10 includes both the PS and PL side of the Zynq device. Once the accelerators start working power consumption increases compared with the software only solution however processing time reduces by a factor of 3 and there is a similar effect on energy.

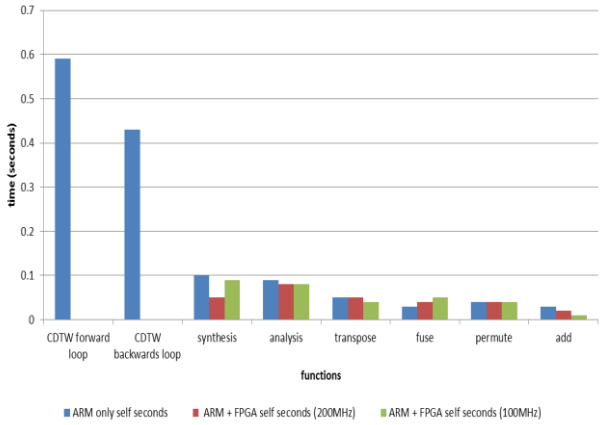

Fig. 8. Performance advantage of the hardware accelerated fusion algorithm for one frame

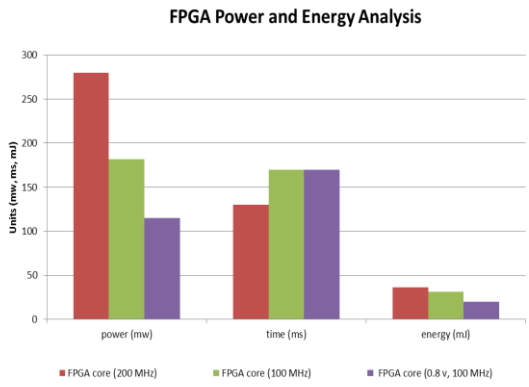

Fig. 9. Power and Energy used by FPGA PL side (excluding processor PS side)

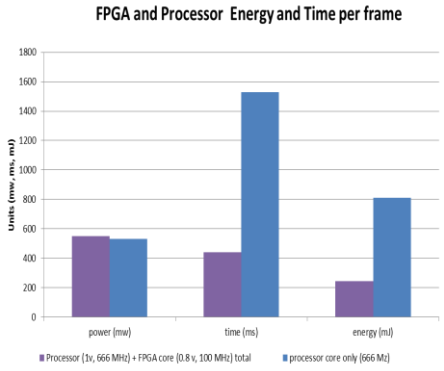

Fig. 10. Overall comparison of hardware accelerated and software only fusion

The final step consists in exploring the further gains possible by computing to the limit and letting the PL to find its optimal frequency and voltage levels with the PS as the monitoring unit. To do this we enable the fully adaptive mode so that once the software tasks of moving and preparing the next data block are completed the PL reads a register in the wavelet engine to check for completion. The idea is that if the hardware engine has already completed it is running to fast. Then the voltage is reduced by the DVS and the DFS unit shown in Fig. 6 proceeds to find the maximum frequency that can be supported at that voltage level using the status of the detector flip-flops to guide this search. The process repeats until the software finds that the hardware wavelet is still working when it is checked. This indicates the point in which the software and hardware are running at approximately the same speed and not waiting for each other. Fig. 11 shows the voltage and frequency points detected during this search from 1v to a minimum of 0.72v and the corresponding optimal frequencies and power at each point. The voltage drops are done in 2 mV intervals by the DVS unit and the DFS unit determines the optimal frequency for each voltage level. The PL moves from operating at 1V/168 MHz and 315 mW to 0.72V/69 MHz and 79 mW which repre-

sents a reduction of power of 75% without a negative performance effect. If there is a change in the operating conditions or workload size (e.g. frame size) then a new adaptation cycle can be started and a new voltage/frequency point will be found by the Elongate system that remains part of the bitstream.

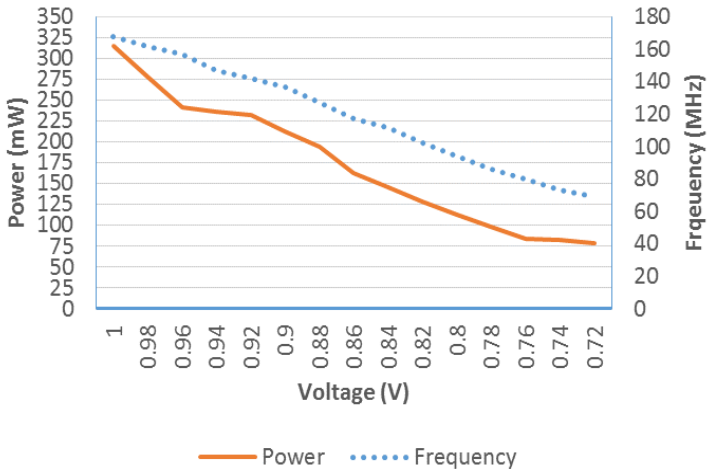

Fig. 11. Adaptive frequency and voltage search

8 Conclusions and Future Work

This paper has presented an energy efficient video fusion system based on the DT-CWT designed using high-level synthesis tools. In our design, the most compute intensive tasks, namely the forward and inverse DT-CWT were mapped to a closely coupled FPGA with a customized Linux kernel level driver to release the processor load. The performance and energy consumption of fusing input frames was compared considering configurations when the fusion process was executed by only the ARM processor and the ARM with FPGA accelerators scaled to different levels of voltage and frequency. Comparing to the execution using the ARM processor only, using the FPGA can save 70% of the execution time of the forward and inverse DT-CWT execution respectively. Once the whole application is taking into account a speed factor of 3 is obtained compared with the ARM only solution. The experiments also show that the FPGA is generally too fast and the limiting factor is the preparation of data in memory for the wavelet engines that must be executed by the ARM processor. Exploiting this characteristic and the Elongate adaptive voltage scaling technology it is possible to reduce power and energy without affecting overall performance. The results with this additional option show a reduction in power by up to 75% compared with running the FPGA at full throttle.

Acknowledgements: We would like to thank the support received from EPSRC for this work part of the ENPOWER project: http://www.bris.ac.uk/engineering/research/microelectronics/enpower/

References

1. D.K. Sahu and M.P. Parsai, “Different Image Fusion Techniques – A Critical Review,” Int’l Journal, Modern Engineering Research (IJMER), vol. 2, Issue. 5, Sept. 2012, pp. 4298-4301.
2. S. Nikolov, P. Hill, D. R. Bull, and C. N. Canagarajah, "Wavelets for image fusion," in Wavelets in Signal and Image Analysis: From Theory To Practice, Kluwer Academic Publishers, 2001.
3. A. Toet, “Hierarchical Image Fusion,” Machine Vision and Applications, March 1990, pp. 1-11
4. Hill, P, Bull, D & Canagarajah, C 2005, ‘Image fusion using a new framework for complex wavelet transforms’. in: IEEE International Conference on Image Processing 2005 (ICIP 2005) Genova, Italy. Institute of Electrical and Electronics Engineers (IEEE), pp. II-1338 - II-1341
5. D. Jasiunas et al., “Image fusion for uninhabited airborne vehicles,” Proc. Int’l. Conf. FPT, Dec 2002, pp. 348-351.
6. O. Sim and J. Ivine, “An FPGA implementation of pattern-selective pyramidal image fusion,” in Proc. Int. Conf. FPL, 2006, pp. 1-4.
7. Q. Yunsheng et al., “The real-time processing system of infrared and LLL image fusion,” Proc. Int’l. Symp. Photoelectron Detection Image process, 2008, pp. 66231Y-1- 66231Y-9.
8. Y. Song et al., Implementation of Real-time Laplacian Pyramid Image Fusion Processing based on FPGA,” Proc. SPIE, vol. 6833, 2007, pp. 16-18.
9. M.A. Mohamed and B.M. El-Den, “Implementation of Image Fusion Techniques Using FPGA,” Int’l. Journal of Computer Science and Network Security, vol.10, No.5, 2010, pp. 95-102.
10. L. Tao et.al., “A Multi-sensor Image Fusion and Enhancement System for Assisting Drivers in Poor Lighting Conditions,” Proc. Applied Imagery and Pattern Recognition Workshop, 2005, pp. 1-6.
11. E. Gudis et al., “An Embedded Vision Services framework for Heterogeneous Accelerators,” Proc. Computer Vision and Pattern Recognition Workshops 2013, pp. 598-603.
12. R.P. Singh, R.D. Dwivedi, and S. Negi, “Comparative Evaluation of DWT and DT-CWT for Image Fusion and De-noising,” Int’l. Journal, Applied Information Systems (IJAIS), vol. 4, Sept. 2012, pp. 40-45.
13. Nunez-Yanez, J.L., "Adaptive Voltage Scaling with In-Situ Detectors in Commercial FPGAs," Computers, IEEE Transactions on , vol.64, no.1, pp.45,53, Jan. 1 2015.