A comparison between DSP and FPGA platforms for real-time imaging applications

Article *in* Proceedings of SPIE - The International Society for Optical Engineering · February 2009

2 authors, including:

Mukul Shirvaikar
University of Texas at Tyler
49 PUBLICATIONS **360** CITATIONS

SEE PROFILE

A Comparison between DSP and FPGA Platforms for Real-Time Imaging Applications

Mukul Shirvaikar and Tariq Bushnaq
Electrical Engineering Department
The University of Texas at Tyler
Tyler, TX 75799
e-mail: mshirvaikar@uttyler.edu

ABSTRACT

Real-time applications impose serious demands on hardware size, time deadlines, power dissipation, and cost of the solution. A typical system may also require modification of parameters during operation. Digital Signal Processors (DSPs) are a special class of microprocessors designed to specifically address real time implementation issues. As the complexity of real-time systems increases the need to introduce more efficient hardware platforms grows. In recent years Field Programmable Gate Arrays (FPGAs) have gained a lot of traction in the real-time community, as a replacement for the traditional DSP solutions. FPGAs are indeed revolutionizing image and signal processing due to their advanced capabilities such as reconfigurability. The Discrete Wavelet Transform is a classic real-time imaging algorithm that is drawing the attention of engineers in recent years. In this paper, we compare the FPGA implementation of 2-D lifting-based wavelet transform using optimized hand written VHDL code with a DSP implementation of the same algorithm using the C language. The goal of this paper is to compare the development effort and the performance of a traditional DSP processor to a FPGA based implementation of an image real-time application. The results of the experiment proves the superiority of FPGAs over traditional DSP processors in terms of time execution, power dissipation, and hardware utilization, nevertheless this advantage comes at the cost of a higher development effort. The hardware platform used is an Altera DE2 board with a 50MHz Cyclone II FPGA chip and a TI TMS320C6416 DSP Starter Kit (DSK).

Keywords: Wavelet Transform, FPGA, DSP, TI 6416 DSK, JPEG 2000.

1. INTRODUCTION

Advancement of the capabilities of Field Programmable Gate Arrays (FPGAs) have generated interest in using them as a replacement for traditional DSP solutions. FPGA is a highly customized chip that consists of a large array of simple Logic Elements (LEs). The FPGA is programmed using Hardware Descriptive Language (HDL), which programs the connections between the individual elements to create logical functions such as: multipliers, registers, or adders. The main bottlenecks in FPGA development are the number of LEs and signal propagation. The new advances in FPGA development have increased the number of LEs on chip, allowing FPGAs to tackle more complex designs. FPGAs are also limited by the propagation delay, which is the time it takes to travel from one logic element to another as well as the time it takes to pass through a single logic element[1]. Advancements in FPGA technologies have increased the number of available gates while decreasing the propagation delay.

Unlike FGPAs, DSPs have a fixed hardware configuration. The DSP processor is a specialized microprocessor, which processes signals efficiently. Some of the advantages of DSPs are low power consumption, flexibility, and bundled signal processing peripherals such as Analog-to-Digital/Digital-to-Analog converters. Therefore DSPs are more capable than regular microcontrollers in implementing many standard DSP operations. DSPs are very cost efficient, since they can be mass produced so that the hardware is constant for all chips and the functionality is defined through software[2]. DSPs have the advantage of using well known programming languages such as C/C++ and assembly language.

The recent advances in FPGA technology are allowing them to perform the same signal processing applications as the DSP[3]. FPGAs started to include dedicated DSP functionalities to appeal more to signal engineers. Since the year 2000 FPGAs have increased by 16 times to about 500 Giga Multiply Accumulate operations per second (GMACS)[4]. Multiply-Accumulate (MAC) is a common operation in many DSP applications, where operands are multiplied and then added into a running total in an accumulator register. The commonality of MAC in DSP applications makes GMACS a standard benchmark when testing the capabilities of a core.

Table 1: Comparison between TI DSP and Altera FPGAs using MAC per second benchmark[4]

Total 16x16 Multiply Accumulators/Sec						
Year	**1990**	**1995**	**2000**	**2002**	**2005**	**2007**
TI DSP	33	160	600	2,400	8,000	17,000
Altera FPGAs	35	436	5,016	47,520	255,960	565,400

Table 1 compares the total number of 16x16 MAC/Sec between TI DSP and Altera FPGAs. By the year 2000 we can notices the exponential growth of FPGA capabilities compared to traditional DSP processors[4]. In 2007 the average TI DSP could perform 17,000 MAC per second versus 565,400 MAC per second for an Altera FPGA.

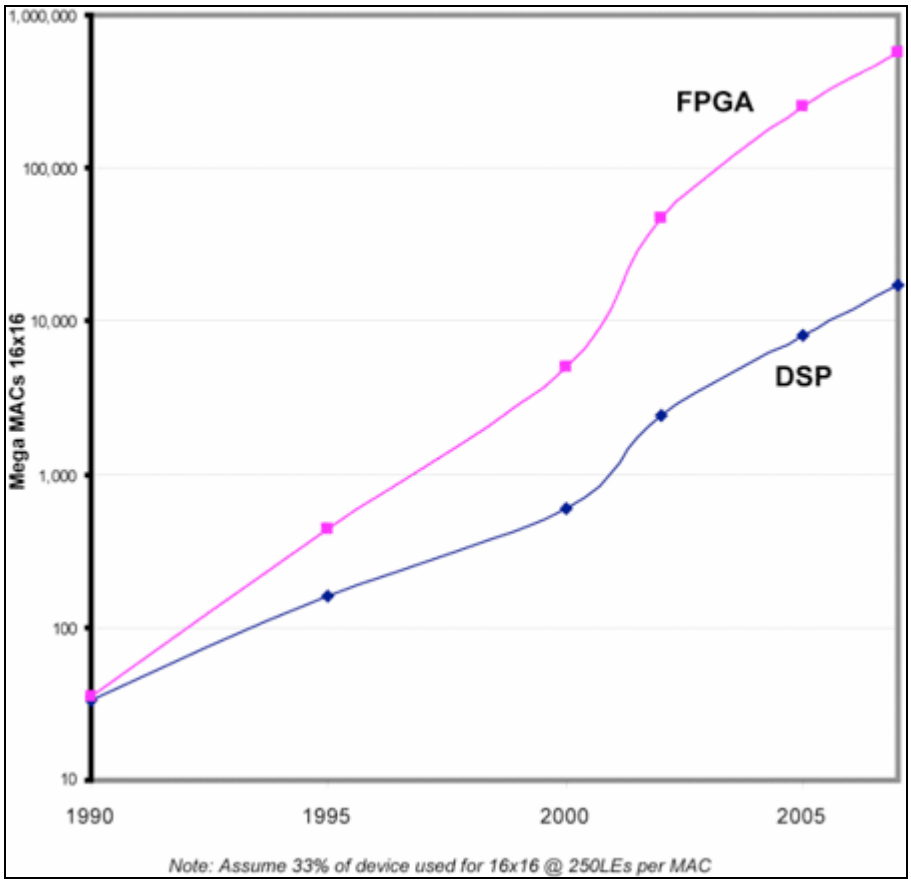

Figure 1: TI DSP vs. Altera FPGA using MAC /Second benchmark[4]

Figure 1 emphasizes the exponential growth in performance for FPGAs over DSP in executing DSP related algorithms. The major caveat is that the FPGA may be limited by signal propagation delays, especially for complex algorithms. This prevents optimal utilization of it's capabilities resulting in rated speeds well below the advertised specifications.

For comparison between the performance of DSP and FPGA we chose to implement the Discrete Wavelet Transform (DWT). DWT is the core transform in JPEG2000. JPEG2000 is a new image compression standard developed by the Joint Photographic Experts Group[5]. The JPEG2000 standard provides a better compression ratio than the original JPEG standard [6.] JPEG2000 also includes extra features which were not available in previous standards such as allowing different compression schemes for digital images. These powerful features come at the expense of process complexity. JPEG2000 implementations are up to six times more computationally complex than JPEG[8]. Software implementations of JPEG200 have the advantage of being flexible, however they may not suitable to meet the hard deadlines of most real time systems. On the other hand, hardware implementations such as ASICs offer high performance in terms of speed but lack the flexibility of software implementations on general purpose microprocessors. The traditional compromise between generic microprocessors and ASIC is the use of programmable Digital Signal Processors (DSPs). DSPs are superior to general microprocessors in terms of power dissipation, high performance, and the availability of efficient logics on the DSP boards, which commonly needed for signal processing [7].

The Field Programmable Gate Arrays (FPGAs) rapid growth in computing power has given digital signal developers an alternative to DSPs. FPGAs and DSPs can be used as coprocessors along with other software components to execute the time consuming stages of the encoding process in JPEG2000. These stages are designed in hardware and loaded into the reconfigurable processor for fast execution. This approach preserves the flexibility of software and the high performance of hardware[9]. The entropy encoding and DWT are the most time-consuming stages of the standard[10]. Hardware can accelerate the execution of DWT tremendously. The entropy encoding component on the other hand is better suited to a software implementation. It would be prudent to implement the DWT component on a coprocessor FPGA or DSP when designing a real-time system that incorporates the JPEG2000 standard.

In this paper, we specifically investigate FPGA and DSP implementations of 2-D lifting-based Daubechies 5/3 transform (also called Le Gall 5/3). The goal is to compare the performance of a traditional DSP design using C language to an FPGA implementation using hand written VHDL code. The FPGA used in this experiment is an Altera DE2 board with 8MB of SDRAM, 4MB of flash memory, and a 50MHz Cyclone II FPGA chip. The DSP processor used in this experiment is a TI TMS320C6416 with a 600MHz clock rate, 32KB L1 cache, and a 1024KB L2 cache. Both implementations are compared in terms of development effort, and processing performance. The results of the experiment proves the superiority of FPGA's over traditional DSP processors in terms of time execution and hardware utilization, nevertheless this advantage comes at the cost of a higher development effort.

2. A CLASSIC APPLICATION - DWT

One of the classical signal processing applications are transforms in general. Due to the growing interest in wavelet transforms we decided to use Le Gall 5/3 as our bench test to compare the performance of TI DSP to FPGA. The wavelet transform is the representation of a signal using wavelets. Wavelets are special small waves of varying frequency and limited duration that mathematically define the basis functions of the wavelet transform. The wavelet transform uses wavelet bases $\psi_{a,b}(t)$ to represent a function such as f as following:

$$f = \sum_{t} a_{a,b} \psi_{a,b}(t) \tag{2.1}$$

Where $a_{a,b}$ are the wavelet coefficients. There are many different wavelet families, which mainly varies due to the trade-offs between localization in space domain and the smoothness of the wavelet[11].

The wavelet basis for DWT can be obtained through equation (2.2). The variable j is the scaling factor, which controls the width of the wavelet, while k is the dilation variable that varies the location of the wavelet.

$$\psi_{j,k}(t) = 2^{\frac{-j}{2}} \psi(2^{-j}t - k) \tag{2.2}$$

The scale and translate parameters are chosen so that the resulting wavelet basis form an orthogonal set (the inner product of the wavelet basis $\psi_{j,k}$ equal to zero), and the dilation factor is a power of two[11].

A 1-D wavelet transform can be extended to a 2-D wavelet transform using separable wavelet filters[8]. Using Separable filters; the 2-D transform can be computed by applying 1-D transform to the rows of the original image, and then applying the same transform on the columns[12]. Figure 2 shows a 2 level 2-D transform on a Lena image.

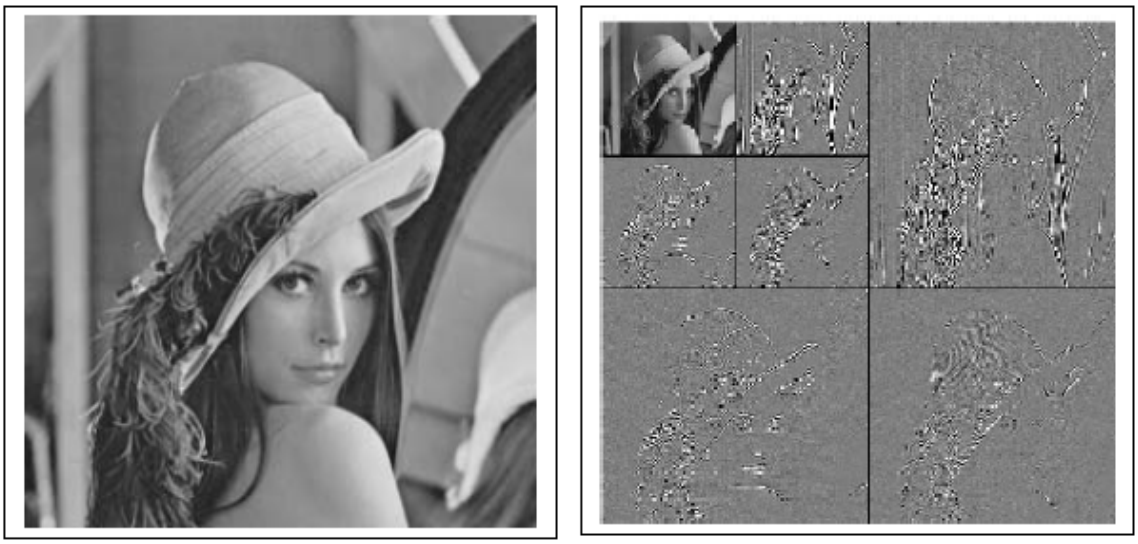

Figure 2: 2 level 2-D transform on a Lena image

2.1 The lifting scheme

Lifting scheme is an efficient implementation of the wavelet transform algorithms described in Swelden [13]. The scheme was primarily developed to implement first generation wavelets in special domain by factorizing the transform into the lifting steps. Later the lifting scheme was extended to a generic method to construct the so-called second generation wavelets. Second generation wavelets do not usually translate and dilate on one mother wavelet as the first generation wavelets do, yet enjoys all the main properties of the first generation such as fast transform, localization and good approximation.

The lifting scheme de-correlates data to achieve a more compact representation. The lifting scheme captures the information contained in the image, or an approximation with an acceptable margin of error using fewer coefficients. The process starts by splitting the input data into odd and even samples (*split phase*). The odd set comprises of all the input data that has an odd index, where the even set comprises of all the input data that has an even index. Assuming that the data is smooth, we use the even samples to predict the value of the odd samples by means of interpolation (*predict phase*). We replace the odd samples by the difference between the prediction and the original value of the odd samples. With a good prediction, we can assume that difference should be small and can be represented with less number of bits. Until now the even samples are only a sub-sample of the original data, which causes the data to lose certain properties that we would like to preserve. In the case of images it is important to keep the mean of the samples constant. Therefore the (*update phase*) updates the even samples using the newly calculated odd samples such that the mean of the data is preserved. These three steps are repeated on the even samples which results in splitting the even samples into half in each level, until all samples are transformed.

2.2 Daubechies 5/3

The Daubechies 5/3 transform uses the lifting based implementation. Daubechies 5/3 is an implementation of reversible integer-to-integer wavelet transform which is constructed using the algorithm described in Calderbank *et al* [14]. The first

stage splits the input image pixels into halves separating the even samples from the odd ones. This is called also the lazy wavelet transform because it does not decorrelate the data, but rather it splits it into odd and even samples [15].

The second stage is to use the even samples to predict the odd ones with the use of equation (2.3). The more correlation present in the data the closer will be the predicted value to the odd samples.

$$Y(2n+1) = X_{ext}(2n+1) - \left[\frac{X_{ext}(2n) + X_{ext}(2n+2)}{2} \right] \tag{2.3}$$

The final scaling coefficients are the difference between the original odd samples and the predicted ones, which is the outcome of equation (2.3). If the input data had perfect correlation the outcome of equation (2.3) would be zero. Therefore, it is apparent that the wavelet coefficients capture the high frequency (deviation from DC level) in the image.

In the final phase the even samples are lifted using the scaling coefficients and computed as following:

$$Y(2n) = X_{ext}(2n) + \left[\frac{Y(2n-1) + Y(2n+1) + 2}{4} \right] \tag{2.4}$$

The scaling coefficients are the outcome of equation (2.4). The coefficients preserve the DC level of the input data capturing the low frequency component of the image.

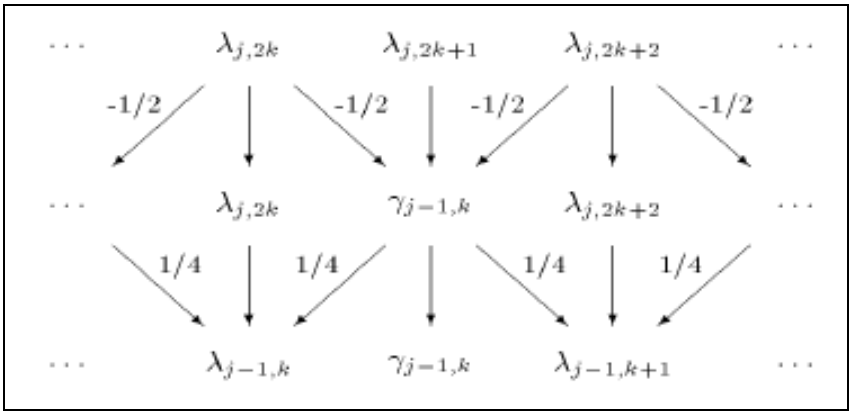

Figure 3: The lifting scheme, predict $\gamma_{-1,k}$ and update $\lambda_{-1,k}$

Figure 3 shows the different steps of the lifting scheme. To reiterate, the wavelet transform consist of the following steps: we first calculate the wavelet transforms as a failure to be linear, which is predicting the odd samples $\gamma_{-1,k}$. Second we lift the sub-sampled coefficients with the help of the wavelet coefficients $\lambda_{-1,k}$.

Real signals in everyday life are time-limited and do not extend infinitely. The lifting scheme assumes an infinite signal, which becomes a problem towards the edges of the real finite signal. This problem is solved by extending the signal at the left and right edges of the signal. In order to avoid discontinuity at the edges, the signals are extended periodically and symmetrically. JPEG200 uses the *1D_EXTR* procedure described in the standard [5].

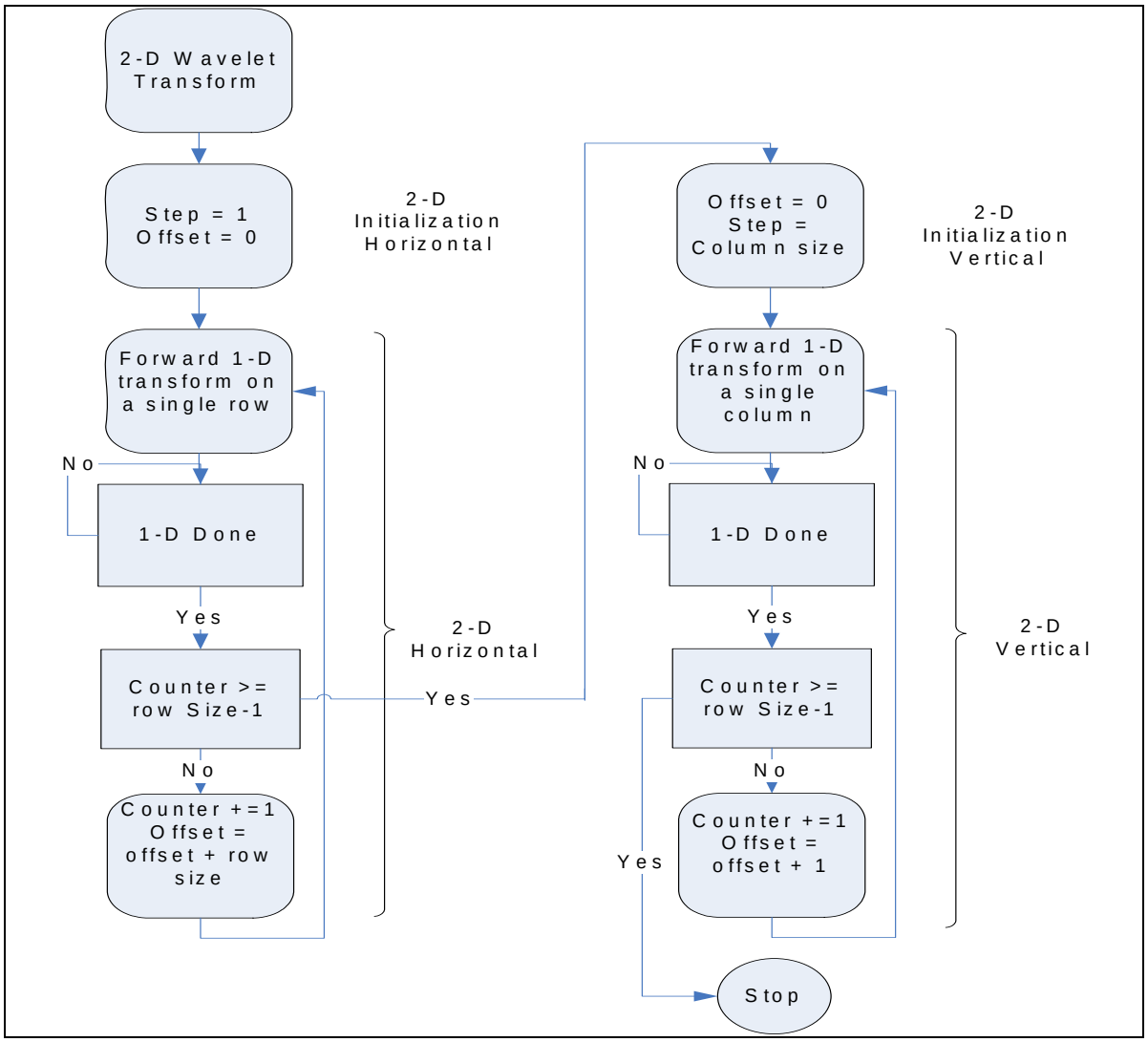

Figure 4: Design structure of the DWT implementation

4. EXPERIMENTS

The objective of the experiments in this section is to evaluate FPGAs as candidates to replace the traditional DSP processors. To do so, we implement Daubechies 5/3 in hand-written VHDL and C language and compare the performance of each implementation running on its dedicated platform in terms of speed, size, and development effort.

The decomposition level was set to one in all implementations in order to normalize the comparison. The size of the testing image was set to 128x128 pixels, where each pixel is 8 bits wide.

Figure 4 shows the general topology of both designs. The implementation starts by initializing the step and offset variables before starting to perform 1-D forward transform on the rows of the image. The rows are continuous with a step size equal to one between each element. The offset starts with zero and gets incremented by an entire size of a row each time the code process a new row. Once the 1-D forward transform module finishes a row it sends a signal back to the 2-D main module. The 2-D process checks if the variable *counter* is larger the number of rows. If the previous condition is met then the forward transform for all rows in the image are found. If not then the *counter* variable is incremented as well as the *offset*

variable until all rows are traversed. Once the coefficients of the rows are found, the same process is repeated on the columns of the image. When processing the columns, the step size to retrieve an element from RAM is equal to a column size while the offset between each column is one. After Applying 1-D forward transform on the rows and the columns of the image we find the discrete wavelet transform of the image.

One of the advantages of the lifting scheme is that it does not require auxiliary memory to store the transform coefficients. Rather it replaces the original values of the image with the coefficients of the DWT. In order to visualize the results of the transform we have to reorder the coefficients contained in the rows and columns of the image. The implementation does not reorder the results of the 1-D transform since we are only interested in performance comparison. The design extends the data as mentioned earlier accordingly and applies the sliding window method to calculate the scaling and wavelet coefficients [16]. The standard approach to implementing DWT requires several passes for each row to implement each lifting and scaling stage in the transform. The Daubechies 5/3 would require two passes for each row to avoid the need to buffer the entire row of data [16]. The sliding window method only requires one pass over a row of data regardless of the number of stages of the DWT. The size of the window depends on the number of data points needed to process the final scaling and wavelet coefficients at each window slide. The technique reduces the number of data pixels that need to be buffered. For example, Daubechies 5/3 requires only a 4 pixel buffer in each window slide [16].

The DSP design was implemented using DSP/BIOS to ensure the efficiency of the implementation. DSP/BIOS is a scalable real-time kernel that uses real-time scheduling and synchronization. DSP/BIOS Application Programming Interface (API) provides over 150 functions, some which are C standard library functions that supersede the library functions bundled with the C compiler. The DSP implementation uses *MEM_alloc* function from the DSP/BIOS library to allocate contiguous blocks of memory from the memory segment, initialized by the DSP/BIOS configuration file. Also, DSP/BIOS allows real-time probing, tracing, and monitoring of the design. The design uses the DSP/BIOS statistics view analysis tool to display real-time hardware and time statistics stored in the STS objects. The STS objects are created using the DSP/BIOS configuration tool.

The VHDL design mimics exactly its DSP counterpart. The design uses a synchronous state machine to implement the steps of the DWT design shown in figure 4. To ensure equality in the performance of both designs the test data was stored on-chip for both implementations. The DSP/BIOS configuration tool declares a heap for data in the 1024kb on-chip memory. The VHDL implementation uses on-chip M4K memory blocks. As the name indicates the M4K blocks are memory segments of size 4KB that are synchronous and can be configured to use as on-chip RAM.

5. RESULTS

The outputs of each design were compared and verified to ensure correctness of the design. Table 2 shows the results of the synthesis report of the HDL design while table two shows the results of the TI DSP processor. The Cyclone II FPGA executed the Daubechies 5/3 transform in 164,354 clock cycles at 50 MHz clock rate. On the other hand the TI 6416 DSP required 10,770,432 clock cycles at 600 MHz clock rate. At 50 MHz clock rate the FPGA completed the transform of a 128x128 pixels in 3.2 ms, while the DSP required 17.9 ms to do the same job. In terms of execution performance, the FPGA has clearly outperformed the DSP by a wide margin. The difference in time between the FPGA and the DSP implementations is 14.7ms. In other words The Cyclone II FPGA can implement DWT on 4.5 tiles of size 128x128 at the same time that it takes the DSP design to complete one tile. A time difference in favor of the FPGA was expected due to its increased capabilities to perform complex calculations as well as its increased processing speed.

The synthesis report of the VHDL design shows the maximum frequency (f_{max}) achieved by the FPGA design. f_{max} is the maximum clock frequency that can be achieved without violating setup and hold time requirements. The following is the equation to compute *fmax*.

$$f_{\max} = \frac{1}{\text{LogicDelay} + \text{ProgrammableRoutingDelay}} \tag{5.1}$$

Table 2: FPGA design results

FPGA	
Number of clock cycles	**164354 cycles**
Execution time at 50MHz clock rate	**3.2 ms**
Execution time at 120MHz clock rate	**1.3 ms**
Execution time/pixel	**0.19µs**
HW utilization	**742 Logical Element**
Lines of code	**132**
Core power dissipation	**79.97 mW**

Table 3: TI DSP 6416DSK results

TI DSP 6416DSK	
Number of clock cycles	**10770432 cycles**
Execution time at 600MHz clock rate	**17.9 ms**
Execution time/pixel	**1 µs**
HW utilization	**67.2KB**
Lines of code	**429**
Core power dissipation	**540 mW**

The logic propagation delay is directly proportional to the number of logic levels in an FPGA design. The more logic levels there are, the longer it takes for data to propagate through. The routing delay is directly proportional to the number of routing segments used in the critical path. Additional routing segments increase the routing delay. The synthesis report indicated that f_{max} for the design is 120MHz. The Cyclone II is considered a Low-cost FPGA. Many other mid-range FPGAs can easily accommodate the design's maximum frequency and therefore, deliver the theoretical expected execution time of 1.3 ms (as indicated in table 2) for the same DWT transform.

The FPGA design requires 742 Logical Elements (LEs) to implement the design. The Cyclone II FPGA provides 33,216 LEs, making the designs HW utilization 2.2% of the total available hardware. The DSP implementation occupies 67.2KB of the 1024KB on-chip memory, making the designs hardware utilization 6.5% of the total available size. These results indicate that the VHDL design is more efficient in terms of hardware utilization of the system than the DSP design.

Tables 2 and 3 list the lines of code required by each design. The VHDL design line count is 429 lines while the C line count was 132. These results indicate that the FPGA design required 3.2 times more lines of code. This gives a close estimate on the ratio of hours required to implement and debug the two designs. Without doubt, the familiarity of the C language and linear programming versus HDL and parallel programming to most signal processing engineers reduces the time required to design for DSP platforms. Also, the linear progression of the C language drastically reduced the debugging effort. The ability to step through the code while executing in real-time and observe the variable's values using DSP/BIOS statistical tools is a big advantage for the DSP design. In contrast, debugging concurrent code such as the VHDL code by tracing register values throughout the simulation plot is time consuming, specially when the design become more complex. With the current demand on FPGAs, major vendors like Xilinx and Altera are constantly trying to bridge the gap between system-level design and physical FPGA implementation. Among their attempts is to introduce SIMULINK tools to support their FPGA design in matlab environment. This approach is favored by both vendors due to the large base of MATLAB programmers which is over 1 million around the world [16]. Also, there are currently many high level hardware design tools such as System-C, Handel that allows simulating concurrent processes described by ordinary C++ syntax. These tools are intended to provide a seamless path from system level algorithm design to FPGA implementation using tools and languages favored by signal processor engineers.

Finally, the result tables show the power dissipation in each implementation. The DSP core dissipates 540 mW while the FPGA core dissipates 79.97 mW. This means that the FPGA is almost 6 times more efficient than the TI 6416DSK. The power saving was expected since the FPGA runs at a clock rate of 50MHz compared to 600Mhz clock rate by the DSP. The low clock rate leads to lower toggling rate, which results in less power dissipation.

6. CONCLUSION

The wavelet transform is currently being used in many engineering fields. The real time implementation of the Discrete Wavelet Transform (DWT) is a current area of research as it is one of the most time consuming steps in the JPEG2000 standard. FPGAs are revolutionizing image and signal processing. In recent years DSP engineers have generated interest in Field Programmable Gate Arrays (FPGAs) as a replacement for the traditional DSP processors. FPGAs are indeed revolutionizing image and signal processing due to the advanced capabilities that FPGAs hold. The results of this experiment have demonstrated that the FPGA design has a better quality of results than traditional DSP processors and is a viable alternative to DSP solutions. However, the time and effort needed to create and debug the FPGA design was much greater than the time needed when designing for the DSP platform. The continuous improvement of system-level design tools for FPGAs, should shorten the development and debugging stage of hardware design.

REFERENCES

1. Ciletti, M. D., Advanced Digital Design with the Verilog HDL, New Jersey: Prentice Hall 2003.
2. Kehtarnanaz, N. and Simsek, B., C6x-Based Digital Signal Processing, New Jersey: Prentice Hall
3. Malagamba, A., "Assembly All Ye IP," *FPGA and Structured ASIC Journal*, [online]. Available: http://www.fpgajournal.com/articles_2005/pdf/20051115_ip.pdf, November 2005.
4. Ekas, Paul (2007). FPGAs rapidly replacing high-performance DSP capability. Retrieved July 31, 2008, from DSP-FPGA.com Web site: http://www.dsp-fpga.com/articles/ekas/.
5. ISO/IEC FCD 15444-1:2000 V1.0, 16 March 2000), http://www.jpeg.org, (Last visited December, 12, 2007).

6. Santa-Cruz, D. and Ebrahimi, T., “An Analytical Study of JPEG 2000 Functionalities,” Proc. Int’l Conf. Image Processing, IEEE, New Jersey, 2000.
7. Andreas, S., Richard, C., “Discrete Wavelet Transform Core for Image Processing Applications,” Proc. of SPIE-IS&T Electronic Images, SPIE, vol. 5671, 2005.
8. Cantineau, O., “Enabling Real-Time JPEG2000 with FPGA Architecture,” Global Signal Processing Conferences & Expos (GSPx), International Signal Processing Con., CF-JPG031505-1.0, March 2005.
9. Zafarifar, B., “Micro-codable Discrete Wavelet Transform,” Computer Engineering Laboratory, Delft University of Technology, July 2002.
10. Shirvaikar, M and Bushnaq, T., “VHDL Implementation of Wavelet Packet Transforms Using SIMULINK Tools ,” Proc. Of SPIE-IS&T Electronic Images, SPIE, vol. 6811, 2008.
11. Grapes, A., “An Introduction to Wavelets”. IEEE Computational Science and Engineering, summer 1995, vol. 2 num. 2 published by the IEEE Computer Society.
12. Usevitch, Bryan E., “A Tutorial on Modern Lossy Wavelet Image Compression: Foundations of JPEG2000,” IEEE Signal Processing Magazine, September 2001.
13. Sweldens, W., “The lifting scheme: A new philosophy in biorthogonal wavelet constructions,” Proc. SPIE, vol. 2569, pp. 68-79, Sept. 1995.
14. Calderbank, A. R., Daubechies, I., Sweldens, W. and Yeo, B. –L, “Wavelet transforms that map integers to Integers,” Appl. Comput Harmon. Anal, vol. 5 pp. 332-369, July 1998.
15. Shim M., and Laine, A., “Overcomplete lifted wavelet representations for multiscale feature analysis,” IEEE International Conference on Image Proc., vol. 2, pp. 242-246, Oct 1998.
16. Meyer-Baese, U., Vera, A., Mayer-Baese, A., Pattichis, M. and Perry, R., “Discrete Wavelet Transform FPGA Design using Matlab/Simulink ,” ECE Dept., FAMU-FSU, 2004.