A Hardware/Software Reconfigurable Architecture for Adaptive Wireless Image Communication*

Debashis Panigrahi, Clark N. Taylor, and Sujit Dey

Dept. of Electrical and Computer Engg.
University of California, San Diego
E-mail: {dpani,cntaylor,dey}@ece.ucsd.edu

Abstract

With the projected significant growth in mobile internet and multimedia services, there is a strong demand for next-generation appliances capable of wireless image communication. One of the major bottlenecks in enabling wireless image communication is the high energy requirement, which may surpass the current and future capabilities of battery technologies. Past studies have shown that the bottlenecks can be overcome by developing adaptive multimedia compression algorithms which can adapt to dynamic channel conditions and service requirements [13, 12].

In this paper, we present an application-specific hardware/software reconfigurable architecture to support adaptive image compression algorithms. We present a design methodology which considers co-design between adaptive algorithms and architectural design leading to a reconfigurable architecture for image compression algorithms. Co-design of the proposed architecture aims not only at performance and power efficient implementation, but also towards fast and efficient run-time adaptation of an adaptive image compression algorithm. Finally, we present experimental results demonstrating that the proposed architecture provides a low cost (performance, energy) implementation for the adaptive image compression algorithm, and necessary run-time adaptation to current wireless conditions and requirements with very low overhead.

1. Introduction

With the introduction of third generation wireless systems and growing popularity of new wireless multimedia applications, there will be a strong demand for systems capable of wireless multimedia communication. One of the major difficulties in designing systems for wireless multimedia communication is the high energy requirement, which may surpass the current and future capabilities of battery technologies. Therefore, to effectively design systems capable of communicating multimedia information over wireless channels, the energy consumption bottleneck must be overcome.

The characteristics of wireless communication that can be used to overcome the energy consumption bottleneck are the varying wireless channel conditions, such as changing Signal-to-Noise Ratio (SNR) over time, and diverse service requirements, such as latency requirements of different applications. Instead of designing a multimedia system that assumes worst-case wireless scenarios, significant savings can be achieved by designing a system, which can adapt to the dynamic conditions and requirements. For wireless image communication, an adaptive image compression algorithm has been proposed in [12]. It has been shown that significant energy savings can be achieved by adapting appropriate parameters of an image compression algorithm according to the channel conditions and service requirements. In order to exploit the adaptivity provided by the adaptive image compression algorithm, it is critical to develop an efficient reconfigurable architecture, which can provide the required run-time adaptation with minimal overhead.

Past research has presented various reconfiguration methodologies at different levels of design, namely software-level, datapath-level, and logic-level, to dynamically reconfigure across different algorithms. Software level reconfigurability using general purpose processors as well as custom-fit processors [5] provides the highest amount of flexibility, but is performance limited in terms of execution time and power. Datapath level configurability, like RAPID architecture [3], as well as logic level configurability, like FPGA based systems [4, 15, 7], provide better performance than general purpose processors, but still cannot match performance of an ASIC implementation for a specific application. In addition, the overhead associated with reconfiguration of an FPGA based architecture (reconfiguration time and storage of configurations in memory) can be significant. An architecture supporting adaptive image compression algorithms needs to implement the algorithms efficiently (low-energy, low-latency), and should provide the necessary low overhead (adaptation time and energy) run-time adaptation. Due to the performance limitations and/or

*This work is supported by Center for Wireless Communication (CWC), University of California at San Diego

high reconfiguration overhead, software, FPGA, and datapath level reconfigurable architectures may not be sufficient to support the needs of adaptive wireless image coder.

To achieve an architecture that provides the necessary flexibility and performance for adaptive image compression algorithms, we propose a hardware/software (hw/sw) reconfiguration methodology, which considers co-design of adaptive algorithms and reconfigurable architecture together. In the first step, we characterize the adaptation needs of an adaptive algorithm in terms of parameters, which need to be configured during the execution of the algorithm. Then we develop a hw/sw architecture that provides the required configurability by provisioning for configurable software module as well as parameterizable ASIC hardware components. Since the software modules and hardware accelerators are designed considering the adaptation needs, the overhead of dynamic adaptation is minimal. In addition, we support efficient execution of run-time adaptation algorithms to select the appropriate parameters and configure the components accordingly. The above design methodology leads to a hardware/software reconfigurable architecture that provides software-like configuration overhead and ASIC-like performance. In this paper, we focus on developing such a hardware/software reconfigurable architecture for the adaptive image compression algorithm for wireless communication presented in [12].

The rest of the paper is organized as follows. In section 2, we review the adaptive image compression algorithm, as presented in [12]. Section 3 describes our hw/sw reconfigurable architecture for adaptive image compression. In section 4, we present experimental results demonstrating the efficiency of the proposed architecture in implementing adaptive image compression algorithms. Section 5 concludes the paper.

2. Adaptive Image Compression

To implement an efficient hw/sw reconfigurable architecture for wireless image communication, it is necessary to understand the adaptive image compression. While the design of algorithms has been described earlier in [13, 12], we present a brief review in this section. First, we present an overview of JPEG image compression algorithm, followed by a description of the parameters used to create an adaptive JPEG image compression algorithm.

Figure 1 shows a basic flow diagram of the JPEG algorithm. In JPEG image compression algorithm, the input image is divided into blocks of size 8 pixels by 8 pixels. Each of these 8x8 pixel blocks is transformed by a Discrete Cosine Transform (DCT) into its frequency domain equivalent. After the transform stage, each frequency component is quantized (divided by a certain value) to reduce the amount of information that needs to be transmitted. These quantized values are then encoded using a Huffman-encoding based technique to reduce the size of the image representation. Next, we briefly describe two parameters of the JPEG algorithm, which can be used to affect a run-time tradeoff between image quality, energy consumption, and the bandwidth required to transmit the image [13].

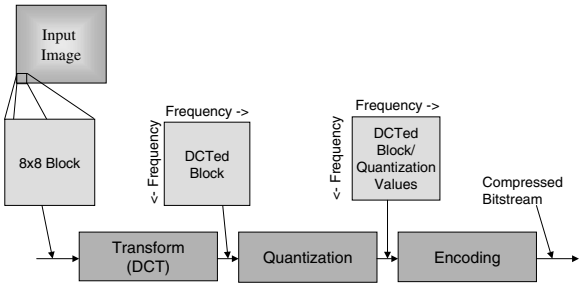

Figure 1. Basic flow of JPEG image compression algorithm

One parameter of the JPEG image compression algorithm that can be dynamically adapted during wireless image communication is the *quantization level*. The JPEG standard defines default quantization tables, which can be scaled up or down depending on the desired quality of the final image. As the quantization level is increased, the image quality decreases, so does the amount of data to be transmitted wirelessly, thereby decreasing communication energy.

The second parameter that can be dynamically adapted is Virtual Block Size (VBS). This parameter affects the DCT portion of JPEG as first introduced in [2]. To implement VBS, the DCT still inputs the entire 8x8 block of pixels, but outputs a VBSxVBS amount of frequency information rather than an 8x8 block. Figure 2 shows an example of setting the VBS to 8 and 5. When the VBS is 8, all frequency information is computed. When the VBS is 5, all frequency data outside the 5x5 block is set to 0. By setting the frequency values outside the VBSxVBS block to zero, computation energy is reduced because the elements set to zero do not have to be computed or quantized. Additionally, the bandwidth and energy needed to transmit the computed image also decreases as the zero values can be encoded to result in a more compact representation.

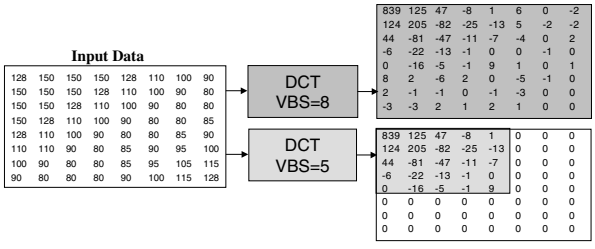

Figure 2. An Example of virtual block sizes 8 and 5

In order to achieve the energy savings facilitated by adapting image compression parameters, it is critical to develop an architecture supporting the necessary run-time adaptations. In the next section, we present a hardware/software reconfigurable architecture design, which allows for run-time adaptations of image compression parameters and efficient implementation of the algorithms.

3. Reconfigurable Architecture

To leverage the advantages of adapting a mobile multimedia radio to current communication conditions and requirements, we propose a mobile multimedia radio architecture shown in Figure 3. Our new architecture includes some components of a traditional radio transceiver (unshaded), such as a speech/text coder, channel coder, RF modulator, and power amplifier. It also includes two new components that we propose for adaptive multimedia communication, the adaptive image coder, and run-time adaptation algorithms, indicated by the shaded regions. The run-time adaptation algorithms are responsible for understanding the current network conditions and service requirements, and configuring the adaptive image source coder accordingly. The adaptive image coder is designed to support different multimedia data compression algorithms, along with their parameters. In the rest of this paper, we concentrate on our proposed hw/sw configurable architecture for the adaptive image coder.

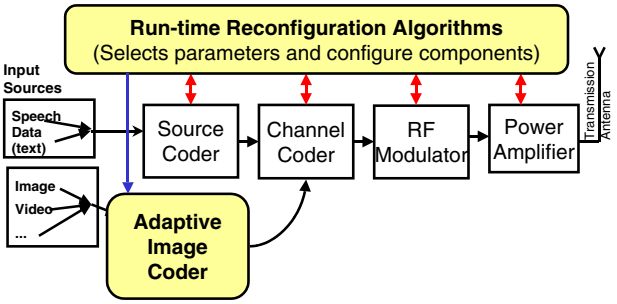

Figure 3. Our Proposed Multimedia Radio Architecture

3.1. Hardware/Software Mapping of Algorithm Tasks

To implement an adaptive image coder that can efficiently execute different image compression algorithms and parameters, we developed a hardware/software (hw/sw) architecture. Traditional hw/sw co-design methodologies attempt to perform hw/sw mapping of tasks in order to optimize for performance and power requirements. In addition to the above objectives, we considered configurability as an objective in mapping the tasks to a hw/sw architecture in this design. It is well known that any software component of a system is easily configurable, whereas the ASIC hardware components offer less dynamic configurability. However, implementing a task in hardware results in better performance and reduced energy consumption. We discuss below how we mapped the image compression algorithm tasks to different hw/sw components to maximize performance without sacrificing flexibility.

In an image compression algorithm, the transform step is the most compute intensive portion, whereas the quantization and encoding steps are more control-intensive. Our preliminary studies on JPEG (implemented fully in software) show that the DCT transform step consumes more than 60% of the computation requirements of the JPEG algorithm. Therefore, we can obtain a significant energy and performance improvement by mapping the transform step to hardware.

Through co-design of image compression algorithm and hw/sw architecture, we can know a-priori which parameters to include in the hardware accelerators (such as VBS) for a particular algorithm. Therefore, the parameters that are used to modify the adaptive image compression algorithm at run-time can still be dynamically configured with transform step mapped to hardware, thereby do not have to sacrifice flexibility.

In addition, even though image compression algorithms may differ greatly in their encoding and quantization steps, they often have an identical transform step. For example, SPIHT[11], AWIC[6], and JPEG2000[1] all use the same wavelet transform, even though their encoding and quantization steps vary greatly. Therefore, we can map image data transforms (DWT and DCT) to hardware without a loss in flexibility.

Across different image compression algorithms, quantization is performed by multiplying the values to be quantized by pre-determined values. This operation is repeated several times within the quantization phase. Therefore, it is possible to obtain a significant performance improvement by mapping the quantization step to hardware without limiting the flexibility of the system, as long as the hardware unit is parameterizable to configure the quantization multipliers by any value for any DCT-based compression algorithms.

To map the encoding step, we first considered the adaptivity necessary for our image compression embedded system. Between image compression algorithms that use the same transform, the encoding step widely varies. For example, within the JPEG image compression standard, there are two possible methods of encoding. One method uses pre-defined probabilities to perform a quasi-Huffman coding that runs much quicker, but does not achieve as good of compression. The other method uses true Huffman coding, which requires a pass to determine signal probabilities, and requires more computation, but yields better compression performance. Therefore, an attempt to map encoding

algorithms to hardware would result in multiple hardware units. In addition, encoding is the most control-intensive portion of image compression. Because of the above mentioned reasons, the encoding step of image compression was mapped to software.

3.2. Hardware/Software Architecture

To implement our hardware/software mapping of adaptive image compression tasks, we developed the architecture shown in Figure 4. Our architecture includes a general-purpose picoJava processor core [10], a hardware accelerator (DCT), an on-chip memory, and the PI-BUS[9] on-chip system bus.

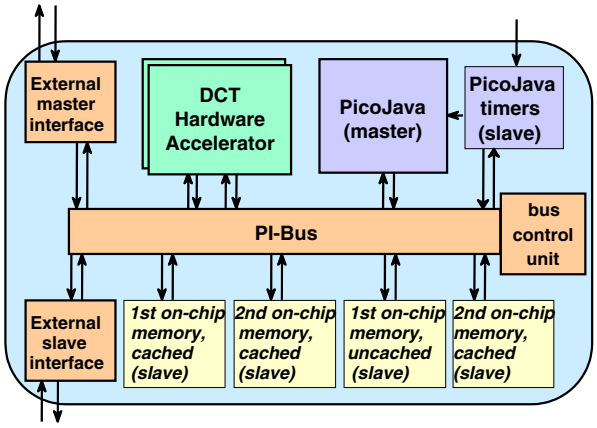

Figure 4. Hw/sw architecture of image compression SoC

To implement the software portion of the image compression algorithms, along with the run-time adaptation algorithms of our proposed multimedia radio, our architecture includes the picoJava soft core from Sun Microsystems [10]. We implement the computationally intensive tasks, such as transformation in image compression, in parameterizable hardware units, termed as hardware accelerators, to assist in achieving high performance multimedia communication. To enable communication of the picoJava core with the hardware accelerators as well as the on-chip memory, we used the PI-Bus [9] and the associated bus control unit and master/slave interfaces.

The architecture of the DCT hardware accelerator is shown in Figure 5. The DCT hardware accelerator consists of four main sections: a parameterizable computation unit, two 64 element memories, a bus interface unit, and the DCT control unit. The architecture of the DCT hardware accelerator was designed to enable run-time adaptation while maximizing the performance of the accelerator. To enable flexibility, the communication interface, control unit, and computation unit were designed to accept parameters from the picoJava processor core, and execute the DCT with different Virtual Block Sizes.

During the computation of the DCT, each pixel value is read and written to twice, quickly becoming the performance limiting factor of the DCT computation. To reduce external memory accesses, we decided to implement two 64-element memory blocks to reduce the need for external memory accesses. The DCT unit is designed such that while the computational unit is performing computation using values stored in one memory block, the bus interface and control units are unloading and loading the other memory block. Using this design for the hardware accelerator significantly boosts the performance of the JPEG image compression architecture while maintaining the flexibility necessary for adaptive JPEG image compression.

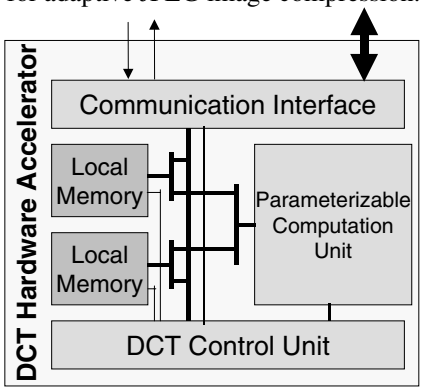

Figure 5. Design of DCT hardware accelerator

By judiciously performing hw/sw co-design with a knowledge of the adaptive JPEG image compression algorithm, we have designed an architecture with good performance, while still allowing for fast and dynamic adaptation at the algorithm level. In the next section, we present experimental results demonstrating the performance and flexibility of our architecture.

4. Experimental Results

In this section, we demonstrate the effectiveness of the proposed reconfigurable architecture in minimizing energy consumption for wireless multimedia communication. Our experimental results are divided into three main sections. First, we compare the time and energy performance results of our hw/sw mapping with an embedded software solution. Next, we present the energy savings enabled by run-time adaptation of the image coder. Last, we present the performance and energy costs of reconfiguration, demonstrating that they are minimal compared to the energy savings obtained by reconfiguration.

The performance (execution time) of our hw/sw architecture reported in the tables were obtained using a fast hw/sw co-simulation framework we developed [8]. The energy consumption results presented for hardware units were obtained using Synopsys's RTL power estimation tool. For

software energy estimates, the software code is run using the RTL model of the picoJava processor core, and estimated using Synopsys's RTL power estimation tool. For all the experiments we use UMC's 0.18μm Cu technology.

4.1. Hardware/Software Mapping Results

Using our hw/sw co-simulation framework, we compared the performance of the proposed hw/sw architecture against a standard, embedded-software implementation. Table 1 presents the results of our comparison in terms of time performance and energy consumed for the JPEG compression of an image (CutCowPI) of size 16x16 pixels. The first row corresponds to an implementation where all steps of the JPEG algorithm, namely Transform(T), Quantization(Q), and Encoding(E), are performed in software (Java) on the picoJava processor. The second row presents results for the proposed architecture, which implements Transform(T) and Quantization(Q) in hardware, and Encoding(E) in software. It can be seen that the proposed hw/sw architecture achieves a 4X timing performance improvement, and a 6X improvement in energy consumption. The improvements in execution time and energy consumption are primarily due to the parameterizable hardware implementation of the most compute-intensive portions of the algorithm. The results are for a small image file as we were constrained by the size of the class file supported by the picoJava processor.

Table 1. Comparison of the Hw/Sw Architecture with an Embedded SW Implementation

Software	Hardware	Performance (cycles)	Energy (mJ)
T,Q,E	–	1981287	4.520
E	T,Q	455981	0.765

4.2. Energy Effects of JPEG Image Compression Parameters

In order to evaluate the efficiency of varying the quantization level and VBS parameters in reducing energy consumption, we performed several experiments using a hardware implementation of the DCT and quantizer of the JPEG image compression algorithm. Table 2 shows the results of our experiments based on configuring the DCT hardware unit between possible Virtual Block Size (VBS) and quantization levels for the image *Monarch*[14]. For each VBS and quantization level combination, we report PSNR (denoting quality of image), energy consumed in the transformation and quantization steps (Computational Energy), and the communication energy/latency in terms of number of bytes to be transmitted. For each VBS (1-8), columns 2, 3 and 4 present the results for a quantization level of 28, where as columns 5, 6, and 7 correspond to a quantization level of 50.

It can be seen from Table 2 that as the VBS decreases, the computational energy decreases, as does the number of bits to be transmitted, leading to a decrease in communication energy. However, the quality of image (PSNR) decreases for smaller VBS. For instance, looking at columns 2, 3, and 4, with the quantization level of 28, the computation energy consumed decreases from 380 μJ to 275 μJ as we change the VBS from 8 to 3. Also, the bandwidth requirements reduce from 52499 bytes to 39042 bytes, while the image quality degrades from 34.87dB to 28.47dB in the process.

Table 2 also shows that dynamic tradeoffs between computational energy requirements, communication energy requirements and quality of image can be achieved by adaptively configuring two parameters of JPEG. For example, to compress the image with a quality of image of at least 32.5dB, either a VBS of 5 with a quantization level of 28, or a VBS of 6 with a quantization level of 50 can be chosen. While choosing a VBS of 5 consumes less computation energy, choosing a VBS of 6 will consume less communication energy because the quantization level is higher, leading to fewer bytes to be transmitted. Depending on the current conditions, the system may select either of the above two options.

4.3. Reconfiguration Overhead

While the savings obtained by adapting to the current image quality requirements are significant, we must insure that the cost of run-time adaptation does not outweigh the benefits of reconfiguration. To measure the cost of run-time adaptation, we first present the costs of selecting the correct parameters for the adaptive image coder.

In Table 3, we compare the execution time and energy consumed in running the run-time algorithms for determining the optimal JPEG image compression parameters with that of the JPEG image compression algorithm. The first row corresponds to the cost of executing the JPEG image compression algorithm on a 16x16 image, where as the second and third row correspond to two different algorithms used for determining the optimal parameters for image compression. The algorithm **Algor1**[13] computes a lookup table outside the wireless appliance that is loaded into the mobile appliance for use at run-time. This lookup table allows the appliance to find the optimal parameters through a simple table lookup according to current image quality and latency requirements. The algorithm **Algor2**[12] determines the optimal parameters according to image quality and latency requirements, together with current transmission energy/bit conditions. In the wireless appliance, executing **Algor2** consists of both a table lookup and some computation to determine the optimal parameters. As shown in Table 3, the energy cost of determining the reconfiguration parameters is at most 6μJ, which is less than 1% of the energy cost of compressing a 16x16 image (765μJ), and associated

Table 2. Effect of parameters on energy consumption

	Quantization=28			Quantization=50		
VBS	PSNR(dB)	Computation Energy(μJ)	Communication Energy(# bytes)	PSNR(dB)	Computation Energy(μJ)	Communication Energy(# bytes)
8	34.87	380	52499	33.21	377	39712
7	34.82	358	52358	33.20	356	39669
6	34.46	338	51774	33.08	336	39509
5	33.30	318	50167	32.42	316	38461
4	31.15	298	46489	30.73	297	36334
3	28.47	275	39042	28.31	275	31301
2	24.99	254	26204	24.96	253	22209
1	20.61	235	11214	20.61	234	10346

execution time overhead is at most 2%.

Once the VBS and quantization level parameters for reconfiguration have been determined, the DCT hardware block must be configured with the computed parameters. To reconfigure the DCT block to different parameters requires 70 writes across the system bus. Assuming 4 cycles per memory write and a bus speed of 100 MHz, the reconfiguration time is less than 30μs, with an energy consumption of less than 1 nJ.

Table 3. Parameter Selection Overhead

Algorithm	Time (# cycles)	Energy (μJ)
JPEG	455981	765
Algor1[13]	986	0.5
Algor2[12]	11,246	5.4

As shown, the costs of run-time adaptation, both in terms of energy consumed and time required, is insignificant compared to the possible energy savings of adapting to different image qualities.

5. Conclusion

In this paper, we presented a hardware/software reconfigurable architecture for adaptive JPEG image compression algorithm which adapts to current channel conditions in order to minimize energy consumption. The architecture achieves time and power performance which is close to an ASIC implementation while allowing for run-time adaptation to minimize energy dissipation. The possible energy savings resulted from adapting the image coder far outweigh the associated adaptation costs.

References

[1] O. K. Al-Shaykh, I. Moccagatta, and H. Chen. Jpeg-2000: A new still image compression standard. In *Conference Record of Thirty-Second Asilomar Conference on Signals Systems and Computers*, volume 1, pages 99–103, 1998.

[2] J. Bracamonte, M. Ansorge, and F. Pellandini. Vlsi systems for image compression. a power-consumption/image-resolution trade-off approach. In *Proceedings of the SPIE - The International Society for Optical Engineering*, volume 2952, pages 591–6, October 1996.

[3] C. Ebeling, D. C. Cronquist, and P. Franklin. Rapid - reconfigurable pipelined datapath. In *The 6th International Workshop on Field-Programmable Logic and Applications*, 1996.

[4] J. R. Hause and J. Wawrzynek. Garp: A mips processor with a reconfigurable coprocessor. In *The 5th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, April 1997.

[5] P. F. Joseph A. Fisher and G. Desoli. Custom-fit processors: Letting applications define architectures. In *Proceedings of the 29th IEEE/ACM International Symposium on Microarchitecture*, 1996.

[6] M. A. Lepley and R. D. Forkert. Awic: Adaptive wavelet image compression. Technical report, MITRE, September 1997.

[7] Y. Li, T. Callahan, E. Darnell, R. Harr, U. Kurkure, and J. Stockwood. Hardware-software co-design of embedded reconfigurable architectures. In *Proceedings, 37th Design Automation Conference*, June 2000.

[8] D. Panigrahi, C. N. Taylor, and S. Dey. Interface based hardware/software validation of a system-on-chip. In *Proceedings of 5th IEEE HLDVT Workshop*, November 2000.

[9] PI-Bus Toolkit. http://www.sussex.ac.uk/engg/research/vlsi-Jan97/projects/pibus.

[10] "PicoJava MicroProcessor Core," Sun Microsystems. http://www.sun.com/microelectronics/picoJava.

[11] A. Said and W. A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3), June 1996.

[12] C. N. Taylor and S. Dey. Adaptive image compression for enabling mobile multimedia communication. In *In Proceedings of IEEE International Conference on Communications*, 2001.

[13] C. N. Taylor, S. Dey, and D. Panigrahi. Energy/latency/image quality trade-offs in enabling mobile multimedia communication. In E. D. Re, editor, *Software Radio – Technologies and Services*, pages 55–66. Springer Verlag, 2001.

[14] Waterloo Repertoire ColorSet. http://links.uwaterloo.ca/colorset.base.html.

[15] R. D. Wittig and P. Chow. Onechip: An fpga processor with reconfigurable logic. In *IEEE Symposium on FPGAs for Custom Computing Machines*, April 1996.