

# Design of an Adaptive Architecture for Energy Efficient Wireless Image Communication\*

Clark N. Taylor, Debashis Panigrahi, and Sujit Dey

Electrical and Computer Engineering Department  
University of California, San Diego  
La Jolla, CA 92093 USA  
{cntaylor, dpani, dey}@ece.ucsd.edu

**Abstract.** With the projected significant growth in mobile internet and multimedia services, there is a strong demand for next-generation wireless appliances capable of image communication. However, wireless image communication faces significant bottlenecks including high energy and bandwidth consumption. Past studies have shown that the bottlenecks to wireless image communication can be overcome by developing adaptive image compression algorithms and dynamically adapting them to current channel conditions and service requirements [1, 2].

In this paper, we present the design of an adaptive hardware/software architecture that enables adaptive wireless image communication. Through intelligent co-design of the proposed architecture and algorithms, we achieve an architecture which enables not only power and performance efficient implementation, but also fast and efficient run-time adaptation of image compression parameters. To achieve efficient image compression and run-time adaptation, we characterized the adaptation needs of an adaptive image compression algorithm in terms of parameters, and implemented an adaptive hardware/software architecture capable of executing JPEG image compression with different parameters. We present experimental results demonstrating that the proposed architecture enables low overhead adaptation to current wireless conditions and requirements while implementing a low cost (energy and performance) implementation of adaptive image compression algorithms.

## 1 Introduction

With the forthcoming introduction of 2.5G and 3G cellular telephony, together with the increasing popularity of the internet, there is a growing need for devices capable of wireless image communication. However, the transmission of images over wireless channels face significant bottlenecks such as severely limited bandwidth and energy consumption. Therefore, to effectively design systems capable of communicating image information over wireless channels, the bandwidth and energy consumption bottlenecks must be overcome.

---

\* This work is supported by the UCSD Center for Wireless Communications, UC CoRe, and SRC

A characteristic of wireless communication which can be used to overcome the bandwidth and energy bottlenecks is the varying wireless channel conditions (such as changing Signal-to-Noise Ratio (SNR) over time) and diverse service requirements (such as latency) of different applications. Instead of designing a multimedia system which assumes worst-case wireless communication conditions and requirements, significant savings can be achieved by designing a system which can adapt to differing conditions and requirements.

Several previous research works have addressed adapting to current wireless conditions to conserve energy and bandwidth. In [3], the authors adapt the channel coding parameters to match current channel conditions, thereby increasing the average bandwidth available. An algorithm is proposed in [4] to modify the broadcast power of the RF power amplifier to meet quality of multimedia data requirements, thereby lowering energy consumption. In [5] a method is presented for setting both channel coding and power amplifier settings to adjust for current conditions, thereby lowering energy consumption. A methodology for adapting JPEG image compression to current wireless conditions to minimize energy and bandwidth is presented in [2]. In all of these research studies, algorithms were presented to minimize energy consumption and/or bandwidth through adaptation to current wireless and application conditions. However, none of the previous research work has addressed the architecture used for implementing an adaptive algorithm. In this paper, we present a novel hardware/software architecture which is adaptive to help conserve energy and bandwidth during wireless image communication.

Past research has presented various reconfiguration architectures 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 or custom-fit [6] processors provide the highest amount of flexibility, but is performance limited in terms of execution time and power. Datapath level configurability, like the RAPID architecture [7], as well as logic level configurability, like FPGA based systems [8, 9, 10], provide better execution time and power characteristics than general purpose processors, but still cannot match the performance of an ASIC implementation for a specific application. In addition, the overhead associated with run-time reconfiguration of an FPGA based architecture (reconfiguration time and storage of configurations in memory) can be significant. Due to the need for high-performance, low power image compression which rapidly adapts to wireless communication conditions and requirements, software, datapath, and logic-level reconfigurable architectures may not be sufficient for wireless image communication.

To achieve an architecture that provides the necessary flexibility and performance for adaptive wireless image communication, we propose a hardware/software (hw/sw) adaptation methodology that considers co-design of adaptive algorithms and architectures together. In the first step, we characterize the adaptation needs of an adaptive algorithm in terms of parameters that need to be configured during the execution of the algorithm. We then develop a hw/sw architecture that provides the required adaptability by implementing portions of

the algorithm in software as well as designing parameterizable 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 adaptive architecture that provides software-like configuration overhead and ASIC-like performance. In this paper, we focus on developing such a hardware/software adaptive architecture for the adaptive image compression algorithms for wireless communication presented in [2, 1].

The paper is organized as follows. In section 2, we review the process of making an algorithm adaptive through parameter identification and configuration. Section 3 describes our hardware/software adaptive architecture for wireless image communication. In section 4, we present experimental results demonstrating the efficiency of the proposed architecture in dynamically executing adaptive image compression and run-time adaptation algorithms. Section 5 concludes the paper.

## 2 Adaptive Image Compression

To implement an efficient hardware/software adaptive architecture for wireless image communication, it is necessary to understand the adaptive image compression and run-time adaptation algorithms being implemented. While the design of algorithms has been described earlier in [1, 2], we present a brief review in this section. First, we present an overview of the JPEG image compression algorithm [11], followed by a description of the parameters used to create an adaptive image compression algorithm. We also review two run-time adaptation algorithms developed to dynamically select the proper image compression parameters.

### 2.1 JPEG Image Compression Algorithm

To review the creation of an adaptive image compression algorithm, we first present a brief overview of the JPEG image compression algorithm [11]. To implement JPEG image compression, 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 which needs to be transmitted. These quantized values are then encoded using a Huffman-encoding based technique to reduce the size of the image representation.

### 2.2 JPEG Image Compression Parameters

To modify the JPEG image compression algorithm to create an adaptive image compression algorithm, two parameters which can be modified at run-time were

selected [1, 2]. The first parameter of JPEG which can be dynamically adapted 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, as does the amount of data to be transmitted wirelessly. The decrease in amount of data to be transmitted affects both the *communication energy* (energy consumed in the wireless transmission of the data), and the bandwidth and latency requirements of transmitting the image.

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 [12]. 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. For example, when the VBS is 8, all frequency information is computed. On the other hand, if the VBS is 5, all data outside the low-frequency 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, while the amount of data needed to transmit the computed image also decreases as the zero values can be encoded to result in a more compact representation.

### 2.3 Run-time Adaptation Algorithms

Once an adaptive image compression algorithm has been developed, it is necessary to implement run-time adaptation algorithms which dynamically select the proper parameters for the adaptive image compression algorithm. The run-time adaptation algorithms are responsible for monitoring the current wireless and application conditions and requirements and selecting image compression parameters. The image compression parameters chosen should minimize the bandwidth and energy consumed during the compression and communication of the image.

In Fig. 1 we present a methodology for selecting the optimal parameters for the adaptive image compression algorithm at run-time while minimizing the run-time cost of executing the run-time adaptation algorithm. The methodology is divided into two sections, precomputation and run-time computation. The precomputation step generates a lookup table(s) which is used at run-time to select optimal image compression parameters (VBS and quantization level). By performing most of the computation outside of the wireless appliance (in the pre-computation step), the run-time cost of adaptation to wireless and application conditions and constraints is minimized. In [1] and [2], two different algorithms were presented utilizing our parameter selection methodology to dynamically select image compression parameters that minimize energy and bandwidth consumption.

In [1], the algorithm presented (*LowOverhead*) assumes that the transmission energy/bit is constant over time. To find the optimal image compression parameters at run-time, it performs a simple lookup yielding the optimal VBS and quantization level parameters. However, assuming that the transmission energy per bit is constant may not be applicable with a changing wireless channel. In

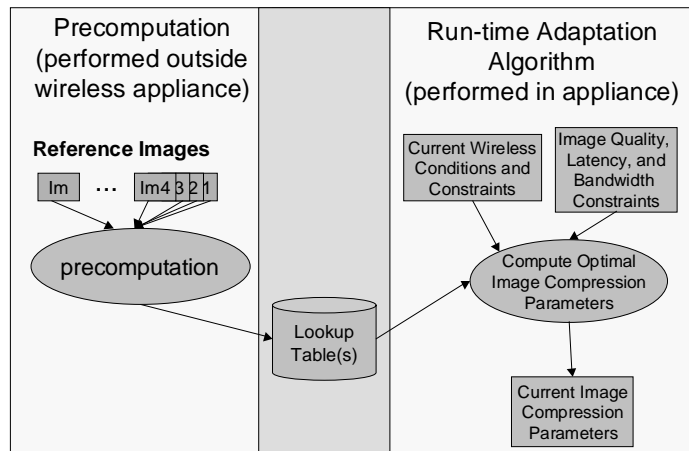


Fig. 1. Methodology for selecting optimal JPEG image compression parameters

[2], an algorithm is presented (*Variable*) for selecting optimal image compression parameters with a varying transmission energy/bit. To adjust to a varying transmission energy/bit, a different lookup table must be generated during the precomputation step. During run-time, multiple table lookups and some computation must be performed to determine the optimal image compression parameters. Therefore, the *Variable* algorithm consumes more resources than the *LowOverhead* algorithm, but can adapt to changing transmission energy conditions, making it more flexible.

With an understanding of the adaptive image compression and run-time adaptation algorithms, it is possible to design an adaptive architecture which meets the performance constraints of wireless communication while still enabling the adaptivity necessary to overcome the bandwidth and energy bottlenecks to multimedia wireless communication. In the next section, we present our hardware/software adaptive architecture allowing for algorithm-level adaptation to current wireless communication conditions and constraints.

### 3 Adaptive Architecture

To leverage the advantages of adapting wireless image communication to current communication conditions and requirements, we propose an adaptive architecture for wireless multimedia appliances shown in Fig. 2. Our new architecture includes some components of a traditional wireless appliance (unshaded), such as a source coder, channel coder, RF modulator, and power amplifier. It also includes two new components that we propose for adaptive image 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 (as discussed in section 2.3). The adaptive image coder is designed to reconfigure as required to execute adaptive image compression algorithms with varying parameters over time. In the rest of this paper, we concentrate on our proposed hardware/software adaptive architecture for the adaptive image coder and run-time adaptation algorithms.

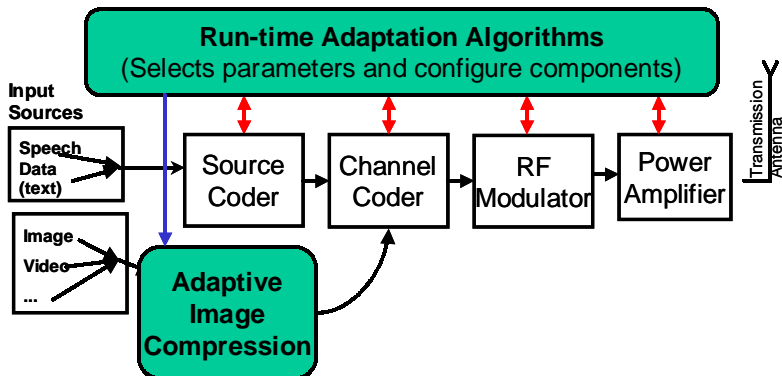


Fig. 2. Proposed adaptive architecture for multimedia wireless appliances

### 3.1 Hardware/Software Mapping of Algorithm Tasks

To implement an adaptive image coder which can efficiently and dynamically execute different image compression algorithms and parameters together with run-time adaptation algorithms, we developed an adaptive hardware/software (hw/sw) architecture. Traditional hw/sw co-design methodologies try to perform hw/sw mapping of tasks in order to optimize for performance and power requirements. In addition to the above objectives, we considered adaptability as an objective in mapping the tasks to a hw/sw architecture. It is well known that any software component of a system is easily configurable whereas 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.

Additionally, we do not have to sacrifice adaptivity through mapping the transform step to hardware. Through co-design of the adaptive image compres-

sion algorithms with the hw/sw architecture, we can know a-priori which parameters to include in the transform hardware accelerators (such as VBS in a DCT accelerator) for the adaptive algorithms. Therefore, the parameters which are used to modify the adaptive image compression algorithm at run-time can still be dynamically modified with the transform step mapped to hardware.

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, the SPIHT[13], AWIC[14], and JPEG2000[15] image compression algorithms all use the same transform step (DWT) even though their encoding and quantization steps vary greatly. Therefore, we can map image data transforms (DWT or DCT) to hardware, achieving the energy and performance gains desired, 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, so long as the hardware unit is parameterizable to configure the quantization multipliers with any value.

To map the encoding of transformed and quantized data to our hardware/software architecture, we first considered the adaptivity necessary for our image compression embedded system. Between image compression algorithms which use the same transform, the encoding step widely varies. For example, within the JPEG image compression standard, there are two possible methods for encoding information. One uses pre-defined probabilities to perform a quasi-Huffman coding method which runs much quicker, but does not achieve as good of compression. There is also a method for using true Huffman coding, requiring a pass to determine signal probabilities, which 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. Therefore, the encoding step of image compression was mapped to software.

### 3.2 Adaptive Hardware/Software Architecture

To implement our hardware/software mapping of algorithm tasks, we developed the architecture shown in Fig. 3. Our architecture includes a general-purpose picoJava processor core [16], a hardware accelerator (DCT), an on-chip memory, and the PI-BUS[17] on-chip system bus.

To implement the software portion of the image compression algorithms, along with the run-time adaptation algorithms of our proposed mobile multimedia appliance, our architecture includes the picoJava soft core from Sun Microsystems [16]. We implement the computationally intensive tasks of adaptive image compression algorithms in parameterizable hardware units, termed as hardware accelerators, to assist in achieving high performance multimedia

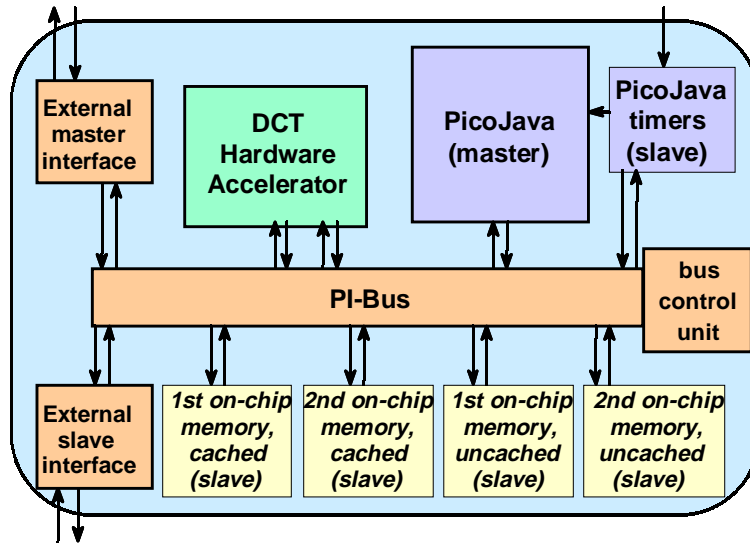


Fig. 3. Hardware/software architecture of image compression SoC

communication. The architecture of the DCT hardware accelerator is shown in Fig. 4.

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 the run-time selection of image compression parameters while maximizing the performance of the accelerator. To enable the flexibility necessary for run-time parameter selection, 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. Therefore, by co-design of the algorithms and architecture, we were able to implement a DCT hardware accelerator which does not sacrifice the needed flexibility for wireless image communication.

On the other hand, it is imperative that our architecture be capable of meeting the real-time constraints of wireless image communication. Therefore, it is important to note that 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. Because memory accesses are the performance limiting factor for the DCT hardware unit, we decided to include two local 64-element memories to reduce the need for external memory accesses. The DCT hardware accelerator is designed such that while the computational unit is computing the information in one memory, the bus interface and control units are unloading and loading the other memory. Using this design for the hardware accelerator significantly boosts the performance of the JPEG image compression architecture while maintaining the flexibility necessary for adaptive image compression.

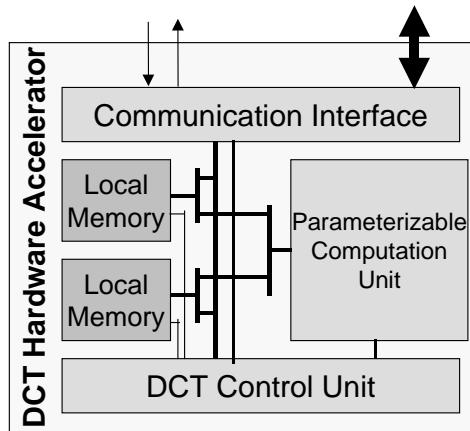


Fig. 4. Architecture of DCT hardware accelerator

We have designed the DCT hardware accelerator in VHDL code. Using Synopsys' Design Compiler to map the DCT to UMC's  $.18\mu\text{m}$  Cu technology [20], we found that the area of the DCT block (excluding memories) was equivalent to 66,176 NAND gates, with a critical path delay of 3.52ns, leading to a clock speed of  $>250$  MHz. The high performance enabled by the DCT hardware accelerator design allows for ASIC-like performance during execution of the JPEG image compression algorithm.

To enable communication of the picoJava core with the DCT hardware accelerator as well as the on-chip memories, we have used the PI-Bus [17] and the associated bus control unit and master/slave interfaces. Using the PI-Bus, we are able to effectively meet all the on-chip inter-component communication needs as well as the need for external communications.

By judiciously performing hw/sw mapping with a knowledge of adaptive image compression algorithms, we have designed an architecture with good performance, while still allowing for the adaptivity necessary to reduce bandwidth and energy through adaptive image compression. 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 adaptive wireless image communication architecture. We demonstrate the low adaptation cost enabled by the design of our hw/sw adaptive architecture, followed by the effects of adaptation on wireless image communication. Finally, we demonstrate the performance advantage of our architecture over an embedded software architecture.

The performance (execution time) of our hardware/software architecture reported in the subsections below were obtained using a fast hw/sw co-simulation framework we developed [18]. The energy consumption results presented for hardware units were obtained using Synopsys’ 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’ RTL power estimation tool. For all the experiments we use UMC’s .18 $\mu\text{m}$  Cu technology [20].

#### 4.1 Adaptation Overhead

To determine the cost of run-time adaptation in our architecture, we first present the costs of finding the correct parameters to configure the image coder with. The correct parameters are determined by the run-time adaptation algorithms discussed in section 2.3. In Table 1, we compare the execution time and energy consumed in executing the run-time adaptation algorithms 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, while the second and third row correspond to two different run-time adaptation algorithms used for determining the optimal parameters for image compression. The algorithm *LowOverhead*[1] consists of a simple lookup at run-time, but assumes a constant transmission energy per bit. The second algorithm, *Variable*[2] requires some computation in addition to multiple table lookups, but is more flexible in considering the current wireless conditions. As shown in Table 1, the energy cost of determining the adaptation parameters is at most 6 $\mu\text{J}$ , which is less than 1% of the energy cost of compressing a 16x16 image (765 $\mu\text{J}$ ), while the execution time overhead is at most 2%. These results show that the costs of run-time adaptation are very low in our architecture, thereby enabling adaptive image compression to conserve energy and bandwidth.

**Table 1.** Resources required to determine optimal configuration of image compression SoC

Algorithm	Time (# cycles)	Energy ( $\mu\text{J}$ )
JPEG	455981	765
<i>LowOverhead</i> [1]	986	0.5
<i>Variable</i> [2]	11,246	5.4

Once the run-time adaptation algorithms have determined the appropriate configuration for the adaptive image coder, the DCT block must be configured with the appropriate parameters. To configure 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 configuration time is less than 30 $\mu\text{s}$ , with an energy consumption of less than 1 nJ. Compared with FPGA reconfiguration times which are on the order of milliseconds and millijoules, our architecture demonstrates extremely inexpensive run-time adaptation costs.

## 4.2 Energy and Bandwidth Savings of JPEG Image Compression Adaptation

In order to evaluate the efficiency of varying the quantization level and VBS parameters to reduce energy consumption and bandwidth requirements for wireless image communication, we performed several experiments using our hardware implementation of the DCT and quantization portions of the JPEG image compression algorithm. Table 2 shows the results of our experiments based on configuring the adaptive image coder between possible Virtual Block Sizes (VBS) and quantization levels for the image *Monarch*[21]. For each VBS and quantization level combination, we report PSNR (denoting quality of image), energy consumed in the transformation and quantization steps (CompE), the bandwidth required to transmit the image in number of bytes (Bandwidth), and the communication energy (CommE) consumed in transmitting those bytes using Bluetooth access technology [19]. To compute communication energy, we assume a maximum Bluetooth transmission power of 1mW, and a bandwidth of 200 kbits/s. For each VBS (1-8), columns 2 through 5 present the results for a quantization level of 28, where as columns 6 through 9 correspond to a quantization level of 50.

**Table 2.** Effect of parameters on energy consumption

VBS	Quantization=28				Quantization=50			
	PSNR (dB)	CompE ( $\mu$ J)	Bandwidth (# bytes)	CommE ( $\mu$ J)	PSNR (dB)	CompE ( $\mu$ J)	Bandwidth (# bytes)	CommE ( $\mu$ J)
8	34.87	380	52499	2100	33.21	377	39712	1588
7	34.82	358	52358	2094	33.20	356	39669	1587
6	34.46	338	51774	2071	33.08	336	39509	1580
5	33.30	318	50167	2007	32.42	316	38461	1538
4	31.15	298	46489	1860	30.73	297	36334	1453
3	28.47	275	39042	1562	28.31	275	31301	1252
2	24.99	254	26204	1048	24.96	253	22209	888
1	20.61	235	11214	449	20.61	234	10346	414

It can be seen from Table 2 that our architecture enables tradeoffs between image quality, computation energy, bandwidth required for image transmission, and communication energy through the selection of VBS and quantization level parameters. As the VBS decreases, the computational energy decreases, as does the number of bytes to be transmitted, leading to a decrease in communication energy and bandwidth requirements. Therefore, varying the VBS in our architecture enables a tradeoff between image quality and the bandwidth required and computation and communication energy consumption. For example, for a quantization level of 28, decreasing the VBS from 8 to 4 decreases the image quality (PSNR) and Computation Energy (CompE) from 34.87dB to 31.15dB and  $380\mu$ J to  $298\mu$ J, respectively. The bandwidth required to transmit the image also decreases from 52499 to 46489 bytes, with a corresponding decrease in communication energy of  $2100\mu$ J to  $1860\mu$ J.

In addition, by varying the quantization level parameter, we observe that the computation energy does not change significantly, but the bandwidth required for image transmission, together with the communication energy change significantly with differing quantization levels. Therefore, varying the quantization level in our architecture enables a tradeoff between image quality, bandwidth required for image transmission, and communication energy. For example, with a VBS of 8, changing from a quantization level of 28 to 50 changes the image quality (PSNR) from 34.87dB to 33.21dB, while the bandwidth required for image transmission (Bandwidth) decreases from 52499 to 39712 bytes, with a corresponding decrease in communication energy (CommE) from 2100 $\mu$ J to 1588 $\mu$ J.

### 4.3 Hardware/Software Mapping Results

In the previous two subsections, we have demonstrated the adaptivity enabled by, and the low-cost approach of varying parameters at run-time in, our adaptive hw/sw architecture. We have also demonstrated the tradeoffs between image quality, computation and communication energy, and amount of data to be transmitted enabled by adapting the image compression parameters in our hw/sw architecture. In this subsection, we also demonstrate that our hw/sw architecture has significant performance and power advantages over a pure embedded software implementation.

In Table 3, we compare our architecture with a software implementation in terms of execution time 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 hardware/software 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. These results show that it is possible to develop a high performance, low power architecture while enabling the adaptation necessary for wireless multimedia communication.

**Table 3.** Comparison of our hardware/software architecture with an embedded software implementation

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

## 5 Conclusion

One of the major challenges to the future growth of wireless image communication is the significant energy and bandwidth consumption required. One method for overcoming these bottlenecks is to use run-time adaptation of the wireless radios transmitting multimedia data. In this paper, we present an adaptive hw/sw architecture which enables run-time adaptivity through efficient execution of adaptive image compression and run-time adaptation algorithms. We have reviewed some image compression parameters which can be selected at run-time to conserve energy, and presented our methodology for performing hw/sw mapping to enable adaptivity while maintaining ASIC-like performance. We developed the adaptive hw/sw architecture through co-design of adaptive algorithms and architectures. The adaptive algorithms and architectures presented in this paper ensure minimal adaptation costs while maintaining time and power performance which is close to an ASIC implementation.

## References

- [1] C. N. Taylor, S. Dey, and D. Panigrahi, "Energy/Latency/Image Quality Trade-offs in Enabling Mobile Multimedia Communication", in *Software Radio - Technologies and Services* (E. D. Re, ed.), pp. 55-66, Springer Verlag, 2001.
- [2] C. N. Taylor and S. Dey, "Adaptive Image Compression for Enabling Mobile Multimedia Communication", in *In Proceedings of IEEE International Conference on Communications*, 2001.
- [3] S. Kallel, S. Bakhtiyari, and R. Link, "An Adaptive Hybrid ARQ Scheme", *Wireless Personal Communications*, vol. 12, pp. 297-311, March 2000.
- [4] P. Cherriman and L. Hanzo, "Error-rate Based Power-controlled Multimode H.263-Assisted Video Telephony", *IEEE Transactions on Vehicular Technology*, vol. 48, pp. 1726-38, September 1999.
- [5] M. Goel, S. Appadwedula, N. R. Shanbhag, K. Ramchandran, and D. L. Jones, "A Low-power Multimedia Communication System for Indoor Wireless Applications", in *1999 IEEE Workshop on Signal Processing Systems. SiPS 99*, pp. 473-82, October 1999.
- [6] 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.
- [7] C. Ebeling, D. C. Cronquist, and P. Franklin, "RaPiD - Reconfigurable Pipelined Datapath", in *The 6th International Workshop on Field-Programmable Logic and Applications*, 1996.
- [8] 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.
- [9] R. D. Wittig and P. Chow, "OneChip: An FPGA Processor With Reconfigurable Logic", in *IEEE Symposium on FPGAs for Custom Computing Machines*, April 1996.
- [10] 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.

- [11] G. K. Wallace, "The JPEG still picture compression standard", in *IEEE Transactions on Consumer Electronics*, vol. 38, February 1992.
- [12] J. Bracamonte, M. Ansoorge, 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*, vol. 2952, pp. 591–6, October 1996.
- [13] 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*, vol. 6, June 1996.
- [14] M. A. Lepley and R. D. Forkert, "AWIC: Adaptive Wavelet Image Compression", tech. rep., MITRE, September 1997.
- [15] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG2000 Still Image Coding System: An Overview", *IEEE Transactions on Consumer Electronics*, vol. 46, pp. 1103–1127, November 2000.
- [16] "PicoJava MicroProcessor Core," Sun Microsystems, <http://www.sun.com/microelectronics/picoJava>.
- [17] PI-Bus Toolkit, <http://www.sussex.ac.uk/engg/research/vlsi-Jan97/projects/pibus>.
- [18] 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.
- [19] The Official Bluetooth Info Site, <http://www.bluetooth.com>.
- [20] UMC Group, *0.18um 1P6M Logic Process Interconnect Capacitance Model (Rev. 1.2)*, July 1999.
- [21] Waterloo Repertoire ColorSet, <http://links.uwaterloo.ca/colorset.base.html>.