ISSN ONLINE (2395-1680)
DOI: 10.21917/ijme.2015.0001

IMPLEMENTATION OF SOFT PROCESSOR BASED SOC FOR JPEG COMPRESSION ON FPGA

K.S.V. Swarna[1] and Y. David Solomon Raju[2]
[1]*School of Engineering, Faculty of Science, Engineering and Built Environment, Deakin University, Australia*
E-mail: ssrungar@deakin.edu.au
[2]*Department of Electronics and Communication Engineering, Holy Mary Institute of Technology and Science, India*
E-mail: davidsolomonraju131@gmail.com

Abstract

With the advent of semiconductor process and EDA tools technology, IC designers can integrate more functions. However, to reduce the demand of time-to-market and tackle the increasing complexity of SoC, the need of fast prototyping and testing is growing. Taking advantage of deep submicron technology, modern FPGAs provide a fast and low-cost prototyping with large logic resources and high performance. So the hardware is mapped onto an emulation platform based on FPGA that mimics the behaviour of SOC. In this paper we use FPGA as a system on chip which is then used for image compression by 2-D DCT respectively and proposed SoC for image compression using soft core Microblaze. The JPEG standard defines compression techniques for image data. As a consequence, it allows to store and transfer image data with considerably reduced demand for storage space and bandwidth. From the four processes provided in the JPEG standard, only one, the baseline process is widely used. Proposed SoC for JPEG compression has been implemented on FPGA Spartan-6 SP605 evaluation board using Xilinx platform studio, because field programmable gate array have reconfigurable hardware architecture. Hence the JPEG image with high speed and reduced size can be obtained at low risk and low power consumption of about 0.699W. The proposed SoC for image compression is evaluated at 83.33MHz on Xilinx Spartan-6 FPGA.

Keywords:

Prototype, System-on-Chip, JPEG Compression, Micro Blaze Soft Core Processor

1. INTRODUCTION

Continuing advances in latest technologies allow the implementation of ever larger and more complex systems on a single chip. This concept is referred to as System-on-Chip (SoC). For this we use FPGA as system on chip because of its Re-configurable hardware architecture. To explain this concept we consider JPEG compression. So the SoC for JPEG compression is built on FPGA Spartan-6 SP605 evaluation board using Xilinx platform studio. System on chip refers to integrating all components of an electronic into single integrating circuits. Application of soc is field of digital, analog, mixed-signal & various fields. A typical application is in area of embedded system. Embedded system is a special-purpose computer system designed to perform one or few dedicated task with real-time computing constraints for proper, reliable, controlled, and timely operation some suitable real time operating system is used. Hence to develop reconfigurable SoC we use FPGA because of its high programmability and low risk. A platform FPGA can be defined as a device that in addition to the field programmable logic cells integrates a predetermined collection of resources such as embedded CPUs, SRAM, versatile general purpose IO ports, high speed serial links, various standard peripherals and others. Collection of these functionalities that may be implemented as hard or soft IP cores makes the platform FPGAs extremely flexible reconfigurable SoC devices where they can be customized to a big variety of complex applications by adequately configuring and programming a needed set of available on-chip components. For our developments we opted for devices from the Xilinx Spartan-6 FPGA family because their features and evolution path seemed most adequate for our needs when technology choices had to be done. So we consider a real time application i.e. JPEG compression to develop a system on chip platform on FPGA using Xilinx Platform Studio.

In this paper we use FPGA as a system on chip which is then used for image compression proposed hardware platform using Xilinx IP catalogue. For JPEG compression the soft processor i.e. Microblaze is generated using Xilinx platform studio. Finally the bit file and elf file are obtained and these are combined into download.bit file which will then be downloaded to program the FPGA. This procedure is explained briefly in the following sections.

Section 1 gives the introduction and section 2 literature survey and basic concept of soft processor i.e. Microblaze. Here mainly we discuss about architecture of Microblaze and the functioning of each block. Then we will discuss the method of generating hardware platform and software platform for JPEG compression which is nothing but system on chip for JPEG compression in section 3. In section 4, we will discuss the complete information about jpeg compression i.e. origin of JPEG, block diagram of JPEG compression and the compression technique. We use discrete cosine transform based compression technique which is nothing but a lossy compression. In section 5, we give information about implementation of JPEG compression on Spartan-6 by considering any BMP image as raw image. In section 6, the results for this JPEG compression using FPGA as system on chip is shown, and how the size is reduced and the time taken for the compression is shown.

2. LITERATURE SURVEY

This section gives an overview of soft processors, different types of soft processors, the significance of Microblaze soft-core processor, its features and brief information about its architecture. A soft microprocessor (also called soft-core microprocessor or a soft processor) is a microprocessor core that can be wholly implemented using logic synthesis. It can be implemented via semiconductor devices containing programmable logic (e.g., ASIC, FPGA, CPLD), including both

high-end and commodity variations. There are different types of soft core processors, some of which are Nios-II, picoblaze, Microblaze etc. The details of these soft core processors are given below.

Nios II is a 32-bit embedded-processor architecture designed specifically for the Altera family of FPGAs. Nios II incorporates many enhancements over the original Nios architecture, making it more suitable for a wider range of embedded computing applications, from DSP to system-control. Nios II is a successor to Altera's first configurable 16-bit embedded processor Nios.

Pico Blaze is the designation of a series of three free soft processor cores from Xilinx for use in their FPGA and CPLD products. They are based on an 8-bit RISC architecture and can reach speeds up to 100 MIPS on the Vertex 4 FPGA's family. The processors have an 8-bit address and data port for access to a wide range of peripherals. The license of the cores allows their free use, albeit only on Xilinx devices, and they come with development tools. All instructions execute in two clock cycles, making performance of the core instruction set deterministic.

The Microblaze embedded processor soft core is a reduced instruction set computer (RISC) optimized for implementation in Xilinx Field Programmable Gate Arrays (FPGAs). Compared to other general purpose processors, Microblaze is quite flexible with a few configurable parts and capable of being extended by customized co-processors. There are a number of on-chip communication strategies available including a variety of memory interfaces. The operating frequency of Microblaze on spartan-6 SP605 kit is 83.33 MHz Hence we use Microblaze soft-core processor in order to develop hardware platform for JPEG compression application. The Microblaze soft core processor is highly configurable, allowing selecting a specific set of features required by the design. Microblaze is a soft-processor containing 32-bit RISC architecture. It has 32-bit 32 × 32 general purpose registers. It is supported in Vertex and Spartan family devices.

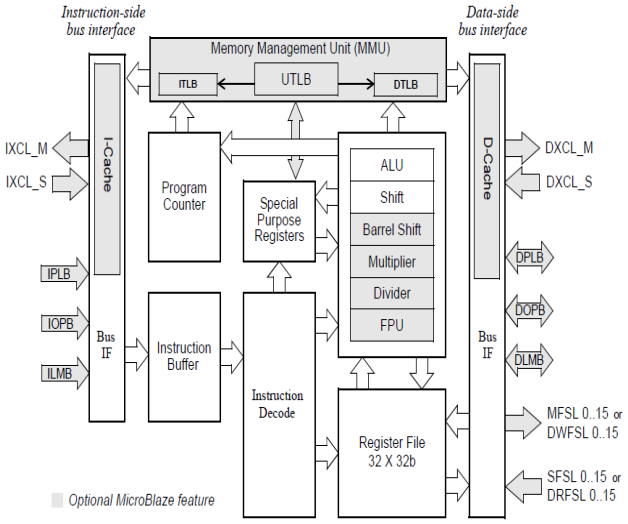


Fig.1. Microblaze soft-core processor architecture

Microblaze processor has an instruction decoding unit, 32 × 32 bit general purpose register file, arithmetic unit and special purpose registers. In addition, it has an instruction prefetch buffer. The arithmetic unit is configurable, as shown in core block diagram. The Barrel Shift, Multiplier, Divider and FPU are optional features. Microblaze processor has a three- stage pipeline: fetch, decode and execute. For most of instructions, each stage takes one clock cycle. There is no branch prediction logic. Branch with delay slot is supported to reduce the branch penalty. Microblaze is a Harvard architecture processor, with both 32-bit I-bus and D-bus. Cache is also an optional feature. Three types of buses, FSL, LMB and OPB are available. FSL bus is a fast co-processor interface. LMB is one-clock-cycle, on-chip memory bus while OPB is a general bus with arbitration. Microblaze has an orthogonal instruction set architecture. It has thirty-two 32-bit general purpose registers and up to eighteen 32-bit special purpose registers, depending on configured options.

3. SYSTEM ON CHIP FOR JPEG COMPRESSION

In this section we discuss about the proposed Hardware and Software platform for JPEG compression. System on chip implementation for jpeg compression involves three layers in it. The three layers are hardware platform followed by operating system (OS) and the required application that is to be carried out. Pictorially it is shown below.

APPLICATION	→	JPEG
OPERATING SYSTEM	→	STAND_ALONE
HARDWARE PLATFORM	→	MICROBLAZE BASED SYSTEM

Fig.2. Layered structure of the SoC

First layer is hardware platform. It is generated using Microblaze. Microblaze is flexible and can be configurable customized soft core processor. Here the components required for the application are configured and the hardware platform for the platform is generated. Second layer is operating system. We use standalone OS for our paper. It is involved in the software part of our paper. On this OS we develop our application i.e., JPEG compression. Third and the final layer of our paper is the application part, where we develop our application in embedded-C language for system on chip.

3.1 HARDWARE PLATFORM FOR JPEG COMPRESSION

Hardware platform for the JPEG compression requires components like, BRAM, ILMB_cntlr, DLMB_cntlr, MDM, UART, Sys_ACE Compact flash, MCB-DDR3, PLB-Bus. All these hardware components are configured for JPEG compression. Its architecture is shown in the below Fig.3.

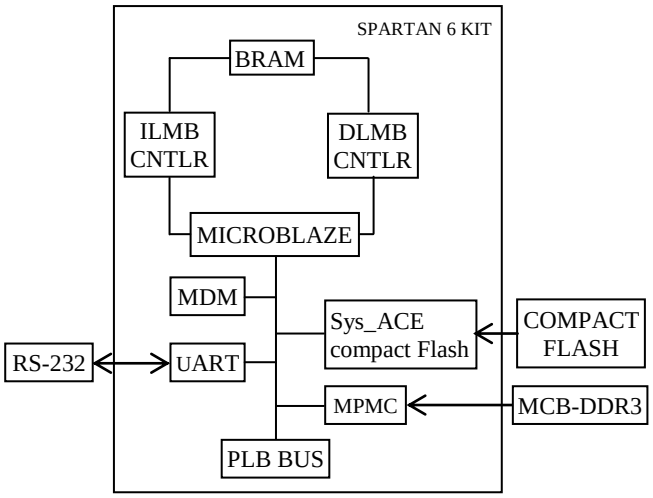


Fig.3. Hardware platform for JPEG compression

The Hardware platform generated in the Xilinx Platform Studio is exported to SDK i.e., Software Development Kit where the remaining process of the application is carried out.

3.2 SOFTWARE PLATFORM FOR JPEG COMPRESSION

Proposed System on Chip is simple embedded system. In-order to implement application on the designed hardware we require an operating system. Standalone OS is used to develop JPEG compression algorithm on proposed hardware platform. To access compact flash memory Xilinx fatfs library is enabled in standalone OS. Using the library functions like sysace_fopen, sysace_fread, sysace_fwrite and sysace_fclose BMP image can be accessed and processed resulting in JPEG image. In SDK, source files are not stored under Workspace.

3.3 SDK APPLICATION DEVELOPMENT FLOW

The development flow of the application in Software Development Kit is shown clearly stage by stage in the below flow chart.

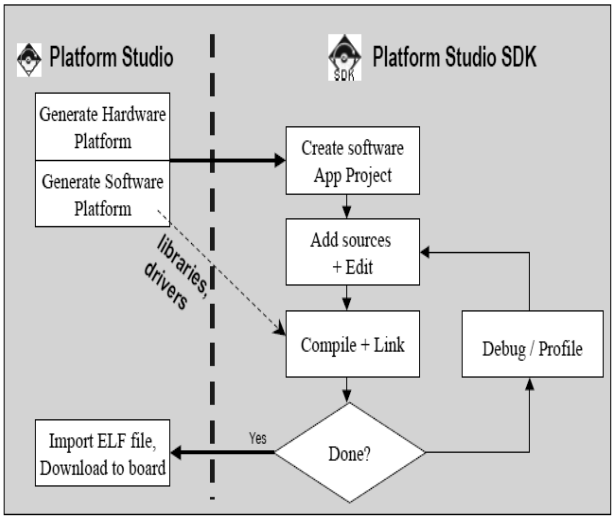


Fig.4. SDK Development flow

The process involved here is hardware platform is exported to SDK. Then the software platform is created which is the standalone board support package. The drivers and libraries which are included during the generation of software platform are used during the compiling and linking process. Compiling refers to transferring high level language i.e., embedded-C language to machine level language. Further on this software platform Xilinx c-paper is created for the application. Edit the application and add the required source files to it. Then compile and run it. If there are no errors then ELF file i.e., executable file is generated which can be further downloaded to the Spartan-6 SP605 evaluation board. If there are any errors then those are debugged and the above process is repeated further. Thereafter executable file i.e., elf file is generated. The bit file from the hardware platform and elf file from the software are further combined and dumped on to the Spartan kit.

4. JPEG COMPRESSION

One of the hottest topics in image compression technology today is JPEG. The acronym JPEG stands for the Joint Photographic Experts Group, a standards committee that had its origins within the International Standard Organization (ISO). The JPEG specification defines a minimal subset of the standard called baseline JPEG, which all JPEG-aware applications are required to support. This baseline uses an encoding scheme based on the Discrete Cosine Transform (DCT) to achieve compression. DCT-based algorithms have since made their way into various compression methods. It gives a lot of flexibility so as to obtain a desired compression ratio (CR). DCT-based encoding algorithms are always lossy by nature. DCT algorithms are capable of achieving a high degree of compression with only minimal loss of data. This scheme is effective only for compressing continuous-tone images in which the differences between adjacent pixels are usually small. As presented in Fig.5, the base principle of JPEG compression for color images considers the five main operations:

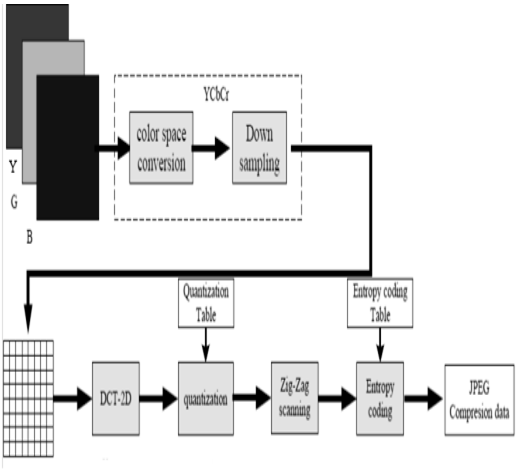


Fig.5. Base Principle Architecture of JPEG Compression

Color space conversion and down sampling: Here the image is transformed into an optimal color space and down sample is applied to chrominance components by averaging groups of pixels together. DCT-2D: Apply a Discrete Cosine Transform (DCT) to blocks of pixels, thus removing redundant image data.

Quantization: Quantize each block of DCT coefficients using weighting functions optimized for the human eye. Zig-Zag scanning: The zig-zag scanning pattern for run-length coding of the quantized DCT coefficients was established here. The pattern is used for luminance and for chrominance. Entropy coding: Encode the resulting coefficients (image data) using a Huffman variable word-length algorithm to remove redundancies in the coefficients.

4.1 COLOR SPACE CONVERSION

The JPEG algorithm is capable of encoding images that use any type of color space. JPEG itself encodes each component in a color model separately, and it is completely independent of any color-space model, such as RGB, HSI, or CMY. The best compression ratios result if a luminance/chrominance color space, such as YUV or YCbCr, is used. Most of the visual information to which human eyes are most sensitive is found in the high-frequency, gray-scale, luminance component (Y) of the YCbCr color space. The other two chrominance components (Cb and Cr) contain high-frequency color information to which the human eye is less sensitive. Most of this information can therefore be discarded. Hence it is necessary to convert RGB color space into YCbCr color space, which is given by,

$$\begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix} = \begin{pmatrix} 0.29900 & 0.58700 & 0.11400 \\ -0.16874 & -0.33126 & 0.50000 \\ 0.50000 & -0.41869 & -0.08131 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} + \begin{pmatrix} 0 \\ 128 \\ 128 \end{pmatrix}$$

In comparison, the RGB, HSI, and CMY color models spread their useful visual image information evenly across each of their three color components, making the selective discarding of information very difficult. All three color components would need to be encoded at the highest quality, resulting in a poorer compression ratio. Gray-scale images do not have a color space as such and therefore do not require transforming.

4.2 DOWN SAMPLING

The simplest way of exploiting the eye's lesser sensitivity to chrominance information is simply to use fewer pixels for the chrominance channels. For example, in an image nominally 1000 × 1000 pixels, we might use a full 1000 × 1000 luminance pixels but only 500 × 500 pixels for each chrominance component. In this representation, each chrominance pixel covers the same area as a 2 × 2 block of luminance pixels. We store a total of six pixel values for each 2 × 2 block (four luminance values, one each for the two chrominance channels), rather than the twelve values needed if each component is represented at full resolution. Remarkably, this 50 percent reduction in data volume has almost no effect on the perceived quality of most images. Equivalent savings are not possible with conventional color models such as RGB, because in RGB each color channel carries some luminance information and so any loss of resolution is quite visible. The JPEG standard allows several different choices for the sampling ratios, or relative sizes, of the down sampled channels. The luminance channel is always left at full resolution (1:1 sampling). Typically both chrominance channels are down sampled 2:1 horizontally and either 1:1 or 2:1 vertically, meaning that a chrominance pixel covers the same area as either a 2 × 1 or a 2 × 2 block of luminance pixels. JPEG refers to these down sampling processes as 2h1v and 2h2v sampling, respectively. Another notation commonly used is 4:2:2 sampling for 2h1v and 4:2:0 sampling for 2h2v; this notation derives from television customs (color transformation and down sampling have been in use since the beginning of color TV transmission). 2h1v sampling is fairly common because it corresponds to National Television Standards Committee (NTSC) standard TV practice, but it offers less compression than 2h2v sampling, with hardly any gain in perceived quality.

4.3 DISCRETE COSINE TRANSFORM-2D

Discrete Cosine Transform (DCT) represents the image as the sum of sinusoids of varying magnitude and frequencies, the DCT calculation is fairly complex; in fact, this is the most costly step in JPEG compression. We can discard high-frequency data easily without losing low-frequency information. The DCT step itself is lossless except for round off errors. DCT is used to produce uncorrelated coefficients, allowing effective compression as each coefficient can be treated independently without risk of affecting compression efficiency. The human visual system is very dependent on spatial frequencies within an image. In fact it is more sensitive to the lower frequencies than to the higher ones. Thus we can discard information that is not perceptible to the human visual system and keep the information that is important to it. The DCT-2D is computed as follows: first, the image data is divided into non-overlapped 8 × 8 matrix blocks; second, all of the 8 × 8 matrix blocks are transformed by the two dimensional Discrete Cosine Transform, which is given by the following equation.

$$F(u,v) = \frac{1}{4} C(u) C(v) \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos\left[\frac{\pi(2x+1)u}{16}\right] \cos\left[\frac{\pi(2y+1)v}{16}\right]$$

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}} & w = 0 \\ 1 & others \end{cases}, C(v) = \begin{cases} \frac{1}{\sqrt{2}} & w = 0 \\ 1 & others \end{cases}$$

The result of this equation is an 8 × 8 matrix representing the frequency domain of the pixel values in the original 8 × 8 block. Most of the image data will be retained in only a portion of the matrix.

4.4 QUANTIZATION

Quantization is used to allow for a better compression ratio, the quantization is the operation that introduces information losses in the JPEG compression process. To discard an appropriate amount of information, the compressor divides each DCT output value by a quantization coefficient and rounds the result to an integer. The larger the quantization coefficient, the more data is lost, because the actual DCT value is represented less and less accurately. Each of the 64 positions of the DCT output block has its own quantization coefficient, with the higher-order terms being quantized more heavily than the low-order terms (that is, the higher-order terms have larger quantization coefficients). Furthermore, separate quantization tables are employed for luminance and chrominance data, with the chrominance data being quantized more heavily than the luminance data. This allows JPEG to exploit further the eye's differing sensitivity to luminance and chrominance. Hence the Quantization is defined as division of each DCT coefficient by

the corresponding quantization value $S(u, v)$, followed by rounding to the nearest integer, which is given by equation,

$$F'(u,v) = Round\left(\frac{F(u,v)}{S(u,v)}\right)$$

$F'(u, v)$ is also called as the coefficient of DCT, in above equation, the coefficient of $F'(0, 0)$ is referred as the DC coefficient, the others are referred as the AC coefficient. The compressor starts from a built-in table that is appropriate for a medium-quality setting and increases or decreases the value of each table entry in inverse proportion to the requested quality. The complete quantization tables actually used are recorded in the compressed file so that the decompressor will know how to (approximately) reconstruct the DCT coefficients. Selection of an appropriate quantization table is something of a black art. DCT coefficient is actually multiplied by 5041 which is stored in the proposed implementation as the corresponding quantization value and then the least significant 16 bits are discarded by a shift operation.

4.5 ZIG-ZAG SCANNING AND ENTROPY CODING

Entropy coding is a special form of lossless data compression. It involves arranging the image components in a zigzag order employing run-length encoding (RLE) algorithm that groups similar frequencies together, inserting length coding zeros, and then using Huffman coding on what is left. Hence after quantization is used, the DC coefficient and AC coefficient of each 8 × 8 block should be read in a Zig-Zag order, as depicted in Fig.6.

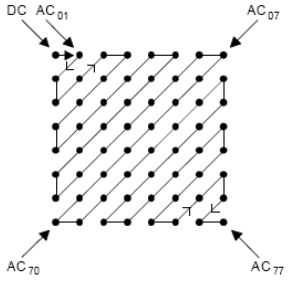


Fig.6. Zig-Zag Scanning

Huffman compression will losslessly remove the redundancies, resulting in smaller JPEG data. The JPEG standard provides general-purpose Huffman tables; encoders may also choose to generate Huffman tables optimized for the actual frequency distributions in images being encoded. Finally, all of the DC coefficient and the AC coefficients should be coded by Huffman code. After this, the JPEG data stream is ready to be transmitted across a communications channel.

5. IMPLEMENTATION ON SPARTAN-6 SP605

The SP605 board enables hardware and software developers to create or evaluate designs targeting the Spartan-6 XC6SLX45T-3FGG484 FPGA. The SP605 provides board features common to many embedded processing systems. Some commonly used features include: a DDR3 component memory, sysACE-compact flash, general purpose I/O and a UART, instruction local memory bus, data local memory bus. Additional user desired features can be added through mezzanine cards attached to the onboard high speed VITA-57 FPGA Mezzanine Connector (FMC) low pin count (LPC) connector. Spartan-6 FPGA delivers an optimal balance of low risk, low cost, and low power for cost-sensitive applications, now with 42% less power consumption and 12% increased performance over previous generation devices. Part of Xilinx's All Programmable low-end portfolio, Spartan-6 FPGAs offer advanced power management technology, up to 150K logic cells, integrated PCI Express blocks, advanced memory support, 250MHz DSP slices, and 3.2Gbps low-power transceivers. The SP605 is powered from a 12V source that is connected through a 6-pin (2 × 3) right angle Mini-Fit type connector J18. The AC-to-DC power supply included in the kit has a mating 6-pin plug.

5.1 INTERFACING SPARTAN-6 WITH PC

The Development takes place on one machine (host) and is downloaded to the embedded system (target). For this we connect the Spartan-6 SP605 evaluation kit to the PC using USB cables and power on the kit. The external memory card which is of 2GB is inserted at the slot provided on the kit. This external memory card is used to load the input BMP image into the kit, and stores the output JPEG image which is a compressed image. The Fig.7 shows the interfacing of PC with Spartan-6 kit.

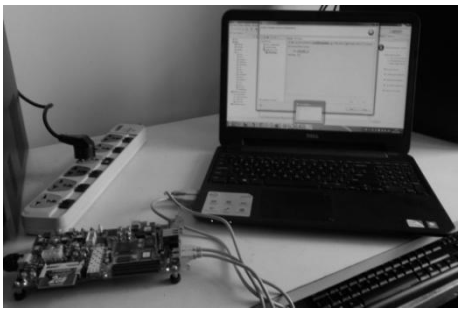


Fig.7. Interfacing of Spartan-6 SP605 evaluation kit with PC

After all the connections are made, go to the Xilinx tools in SDK and click on the program FPGA. Here the elf file and the system.bit file will be combined into download.bit file which will then be downloaded to program the FPGA. Prior to download, the instruction memory (FPGA Block RAM) will be updated in the bit stream with the executable generated using the GNU compiler. Fig.8 shows the window appeared when we click on program FPGA. Here browse the bit stream and BMM files which are attached and then click program.

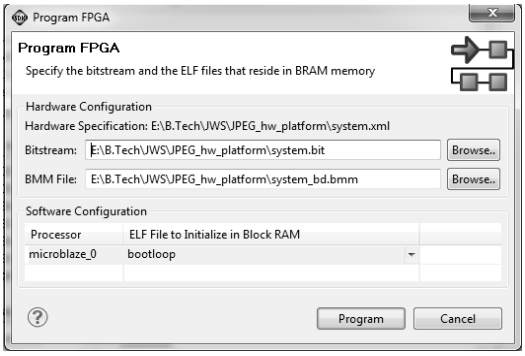


Fig.8. Settings for programming the FPGA

After the step program FPGA, go to the run in SDK and then click on run configuration in which studio connection is selected. Here the configuration settings should be selected as COM3 and baud rate will be 9600. After this click on apply and then ok. Finally the message is generated in console about the size of the BMP image i.e. image height, width, number of rows, number of columns, JPEG header information, size of JPEG image and the time taken for the compression of BMP image to JPEG image. The input BMP image size and output JPEG image size can be seen by connecting the external memory card to the PC using the flash card. Hence the output compressed JPEG image can be obtained.

6. RESULTS

Finally the input BMP image is compressed and the output JPEG image is obtained using FPGA as system on chip. Here we analysed the time taken for the compression and reduction in size. This analysis is done by considering two input images and the output JPEG images were obtained as the figures shown below.

(a). BMP image-1(364 KB)

(b). JPEG image-1(12.5 KB)

(c). BMP image-2(1.37 MB)

(d). JPEG image-2(43.4 KB)

Fig.9. Results for the image compression

We also analyzed that, if the cache memory which was included in the design was disabled then the speed of compression was also reduced as the image should be fetched from external memory DDR3 SDRAM 128MB.

6.1 DESIGN UTILISATION SUMMARY

Table.1. Design Summary with Synthesis Report

Slice Logic Utilization	Used	Available	Utilization
Number of Slice Registers	2,549	54,576	4%
Number used as Flip Flops	2,542		
Number used as AND/OR logics	7		
Number of Slice LUTs	3,020	27,288	11%
Number used as logic	2,759	27,288	10%
Number used as Memory	228	6,408	3%
Number used as Dual Port RAM	136		
Number of occupied Slices	1,354	6,822	19%
Number of LUT Flip Flop pairs used	3,808		
Number with an unused Flip Flop	1,324	3,808	35%
Number with an unused LUT	788	3,808	20%
Number of fully used LUT-FF pairs	1,666	3,808	43%

```
Slice Logic Distribution:
  Number of occupied Slices:              1,354 out of   6,822   19%
  Nummber of MUXCYs used:                   344 out of  13,644    2%
  Number of LUT Flip Flop pairs used:     3,808
    Number with an unused Flip Flop:      1,354 out of   3,808   35%
    Number with an unused LUT:              788 out of   3,808   20%
    Number of fully used LUT-FF pairs:    1,666 out of   3,808   43%
    Number of slice register sites lost
      to control set restrictions:            0 out of  54,576    0%

IO Utilization:
  Number of bonded IOBs:                     73 out of     296   24%
    Number of LOCed IOBs:                    73 out of      73  100%
    IOB Flip Flops:                          38
```

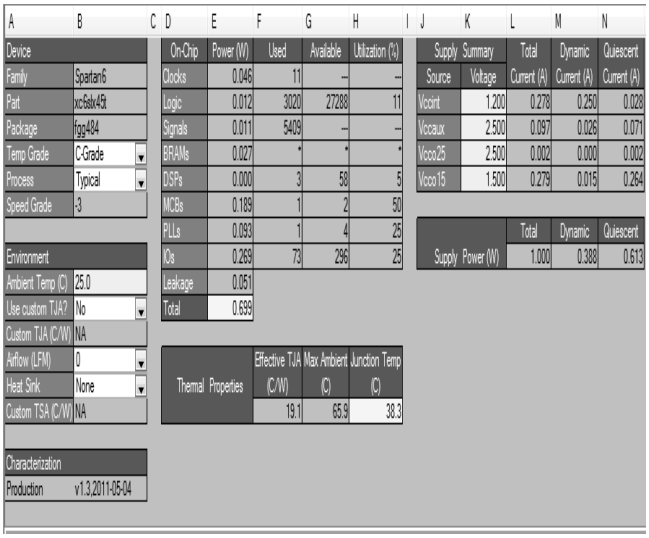
From this design summary we analysed that the JPEG compression design occupied a very less space on Spartan-6 FPGA i.e., only 19%. Hence we can implement many other complex applications by utilizing the remaining space of the Spartan-6 FPGA.

6.2 POWER ANALYSIS REPORT

Here we can analyse the power utilised by the on-chip peripherals. They are shown in the below Fig.10 and Fig.11 clearly.

On-Chip	Power (W)
Clocks	0.046
Logic	0.012
Signals	0.011
BRAMs	0.027
DSPs	0.000
MCBs	0.189
PLLs	0.093
IOs	0.269
Leakage	0.051
Total	0.699

Fig.10. Power utilised by the on-chip peripherals



Device	
Family	Spartan6
Part	xc6slx45t
Package	fgg484
Temp Grade	C-Grade
Process	Typical
Speed Grade	-3

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0
Heat Sink	None
Custom TSA (C/W)	NA

Characterization	
Production	v1.3 2011-05-04

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.046	11	--	--
Logic	0.012	3020	27288	11
Signals	0.011	5409	--	--
BRAMs	0.027	*	*	*
DSPs	0.000	3	58	5
MCBs	0.188	1	2	50
PLLs	0.093	1	4	25
IOs	0.269	73	296	25
Leakage	0.051			
Total	0.695			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	19.1	65.9	38.3

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.200	0.278	0.250	0.028
Vccaux	2.500	0.097	0.026	0.071
Vcco25	2.500	0.002	0.000	0.002
Vcco15	1.500	0.279	0.015	0.264

		Total	Dynamic	Quiescent
Supply Power (W)		1.000	0.388	0.613

Fig.11. Power utilised by the on-chip peripherals with temperature

6.3 TIMING REPORT

The timing details of the application are shown in the below Fig.12 report in detailed.

Constraint	Check	Worst Case Slack	Best Case Achievable	Timing Errors
TS_clk_666_6667MHz180PLL0_nobuf = PERIOD TIMEGRP "clk_666_6667MHz180PLL0_nobuf" TS_sys_clk_pin * 3.33333333 PHASE 0.75 ns HIGH 50%	MINPERIOD	0.001ns	1.499ns	0
TS_clk_666_6667MHzPLL0_nobuf = PERIOD TIMEGRP "clk_666_6667MHzPLL0_nobuf" TS_sys_clk_pin * 3.33333333 HIGH 50%	MINPERIOD	0.001ns	1.499ns	0
TS_clock_generator_0_clock_generator_0_SIG_PLL0_CLKOUT2 = PERIOD TIMEGRP " clock_generator_0_clock_generator_0_SIG_PLL0_CLKOUT2" TS_sys_clk_pin * 0.4 16666667 HIGH 50%	SETUP HOLD	0.043ns 0.239ns	11.956ns	0 0
TS_sys_clk_pin = PERIOD TIMEGRP "sys_clk_pin" 200 MHz HIGH 50%	MINLOWPULSE	2.200ns	2.800ns	0
NET "SysACE_CompactFlash/SysACE_CLK_BUFGP/IBUFG" PERIOD = 30 ns HIGH 50%	SETUP HOLD	25.716ns 0.467ns	4.284ns	0 0
PATH "TS_TIG_MCB_DDR3_S6_DONE_CAL_SYNCH_path" TIG	SETUP	N/A	3.203ns	N/A
PATH "TS_TIG_MCB_DDR3_S6_SYS_RST_SYNCH_path" TIG	SETUP	N/A	2.179ns	N/A

Fig.12. Timing Details

For the compression of image from BMP to JPEG on proposed SoC it took 45sec when cache is enabled and when cache is disabled it took about 240sec. It is operated at 83.33 MHz, consumed very less power of 0.699 watts and occupied only 19% of available resources on the Spartan-6 SP605 evaluation board.

7. CONCLUSION AND FUTURE SCOPE

In this paper new SoC has been proposed for JPEG compression using single soft processor i.e., single Microblaze, implemented on Spartan-6 SP605 evaluation board using Xilinx Platform Studio. For compression of image from BMP to JPEG format on proposed SoC required 45sec when cache is enabled and 240sec when cache is disabled. While compared to dual core based PC it consumed very less power of 0.699Watts. System on chip for JPEG occupied only 19% of available resources on the Spartan-6 SP605 evaluation board and evaluated at a frequency of 83.33 MHz. To improve the system performance, the proposed SoC can be extended to Multi soft processors and hardware accelerators like FSL (fast simplex link) and DMA (direct memory access) controllers can be included. Further, we can generate up to 4 number of Microblaze on the FPGA based Spartan-6 SP605 evaluation kit and the entire task can be divided among all the processors resulting in high computation speed. Complex applications can be implemented for the efficient utilization of the remaining space on the Spartan-6 FPGA.

REFERENCES

[1] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG2000 still image coding system: an overview", *IEEE Transactions on Consumer Electronics*, Vol. 46, No. 4, pp. 1103-1127, 2000.

[2] H. Anas, S. Belkouch, M. El Aakif, and N. Chabini, "FPGA implementation of a pipelined 2D-DCT and simplified quantization for real-time applications", *International Conference on Multimedia Computing and Systems*, pp. 1-6, 2011.

[3] J. Ahmad, K. Raza, M. Ebrahim and U. Talha, "FPGA based implementation of baseline JPEG decoder", *Proceedings of the 7th International Conference on Frontiers of Information Technology*, Article No. 29, pp. 1-6, 2009.

[4] Joris van Emden, Marcel Lauwerijssen, Sunwei and Cristina Tena, "JPEG Codec Library base don Microblaze processor", Available at: http://www.opencores.org/projects.cgi/web/mb-jpeg/overview.

[5] N. Ahmed, T. Natarajan and K. R. Rao, "Discrete Cosine Transform", *IEEE Transactions on Computers*, Vol. C-23, No. 1, pp. 90-93, 1974.

[6] R. Uma, "FPGA Implementation of 2-D DCT for JPEG Image Compression", *International Journal of Advanced Engineering Sciences and Technologies*, Vol. 7, No. 1, pp. 001-009, 2011.

[7] Sun Wei, "A FPGA-based Soft Multiprocessor System for JPEG Compression", Technical University Eindhoven, the Netherlands, 2006.

[8] V. A. M. Prakash and K. S. Gurumurthy, "A Novel VLSI Architecture for Digital Image Compression using Discrete Cosine Transform and Quantisation", *International Journal of Computer Science and Network Security*, Vol. 10, No. 9, pp. 175-182, 2010.

[9] Xilinx Inc., "Platform Studio and EDK", http://www.xilinx.com/ise/embedded_design_prod/platform_studio.html

[10] Xilinx Inc., "Microblaze Microcontroller Reference Design User Guide", 2005.