Implementation of Dynamic Reconfigurable Audio Player and Spectrum Analyzer

Vaibhawa Mishra
CSIR-CEERI
Digital Systems Group
Pilani, Rajasthan, India-333031

Kota Solomon Raju
CSIR-CEERI
Digital Systems Group
Pilani, Rajasthan, India-333031

Pramod Tanwar
CSIR-CEERI
Digital Systems Group
Pilani, Rajasthan, India-333031

ABSTRACT
This work presents the implementation of FPGA based run time reconfigurable audio player system that is targeted to ML507 board with frequency component display of the stored audio data. The presented work combines both HW and SW flows where the complex parts of the architecture are designed to HW modules. These HW modules can be reconfigured on the fly by using partial dynamic reconfiguration. In our work, we are using filter block as partial reconfiguration module, 16 point FFT to decimate audio sample data in frequency in SW logic and displaying the spectrum of the audio signal on the display monitor. Our aim is to implement reconfigurable complex audio player as an application of reconfiguration computing, proving how this technique may be helpful to decrease resource utilization of the device with negligible performance overhead.

Keywords
Partial Reconfiguration, Spectral Analysis, Audio Player, Audio Signal Processing.

1. INTRODUCTION
Dynamic Reconfiguration technique for the FPGA devices can be adapted to increase the device utilization for complex application on small FPGA with lesser logic cells. Now days, FPGA are coming with dedicated HW blocks that directs a new method of realization for high performance embedded system design. These platforms with FPGAs not only help system architect to realize complex systems using processor and configurable IP core but also provide facility of swapping in and out of used and unused modules on demand.
In this paper, reconfigurable audio player and spectral display system having dual processor based approach using internal partial reconfiguration feature of Xilinx FPGA has been presented. The various filter blocks of the system have been mapped to a single region and used as Partial Reconfiguration Module. The filtered audio data has been given to the audio coder and also stored in the shared memory where it can be used to calculate its frequency component by using Fast Fourier Transform algorithm.
This paper is organized as follows. Some of the previous works are mentioned Section 2. Section 3 presents brief description of proposed architecture. Architecture analysis of dual processor based self-reconfigurable platform has been presented in Section 4. Implementation analysis of the idea has been discussed in Section 5. The results and conclusions have been presented in Section 6 and 7 respectively.

2. PREVIOUS WORK
In the Dynamic Partial Reconfiguration area, a lot of research has been made especially reconfigurable processor [1], video processing system [2], security system [3] and image processing [4] and [5]. An audio signal based work has been presented in [6]. The research related to spectral analyser has been presented in [7]. A reconfigurable filter design with partial reconfiguration approach has been presented in [8]. Now days, FPGAs are being used for prototyping a complete computing system as in [9].
In this presented work, audio data filtering mechanism has been implemented on the stored audio file with spectral analysis of the audio data and that spectral information has been displayed on DVI monitor. Of course, Partial Reconfiguration method has been used to reconfigure the filter IP core dynamically.
A dual processor based system has also been used in our application where a shared block memory has been used for audio data transfer between two processors.

3. PROPOSED ARCHITECTURE
A dual processor internal self-reconfiguration approach has been proposed for our application. The design has been composed of one static region and two reconfigurable regions. The static part of the design has mainly one hard core PowerPC processor and one soft core micro-blaze processor. The Hard core PowerPC has been used to control reconfiguration and audio processing. It is taking the audio data from WAV audio file stored in external FLASH memory and pumps data to both reconfigurable filter blocks dedicated for left and right channels respectively. This processor is also used as reconfiguration controller whenever user interrupts its execution. The other processor, that is micro-blaze in soft core, is sharing the sampled data through block memory and runs the FFT algorithm to decompose the data into its frequency component and transfers the spectral information to character- graphics mapped controller IP core for DVI monitor. The schematic of the proposed reconfigurable architecture is shown in Fig. 1
The application running on 1st processor allows reading and locating partial bit files for various filters module from an external Flash memory and dynamically reconfiguring the part of FPGA after the initial configuration. The HwIcap module, used for reconfiguration, has been driven through s/w API (ICAP API). The application running on 2nd processor handles the 16 – point FFT computation and communicates spectral information with display controller IP core to view on monitor.

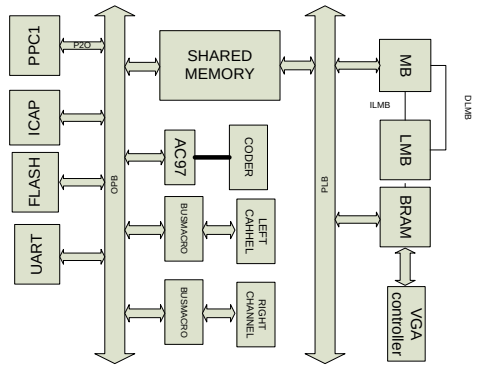

Fig. 1 Schematic of proposed reconfigurable architecture

4. ARCHITECTURE ANALYSIS

Proposed architecture can be described in four processing units as shown in Fig. 2. The proposed architecture is a mixed of hardware and software co-design. Two of the processing units like audio reading processing and FFT computing unit are mapped as software tasks. Rest of the processing units i.e. filterization of audio data and spectral display unit are implemented as hardware modules. The suggested architecture is using two processors, so the execution of the application can be easily portioned in between both of the processors to minimize the overall execution overhead.

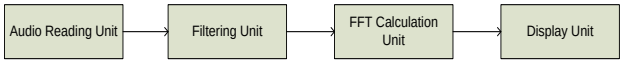

Fig. 2 Processing flow for audio data

As discussed earlier, an audio reading unit of the system is performed by the hard core PowerPC processor. The main work of this processing unit is collecting the audio data from the WAV file stored in external memory at the rate of 48kHz. The coming data is in 32-bit format that is combination of 16 bit left and right channel data format. These data then passed to reconfigurable filter IP core for further process. After processing, the 1st processor collects the data again from the both filter IP core. The data depends on which type of filtering mechanism is available at that time of execution in both reconfigurable filter IP core. The left channel filtered audio data is then stored in a dual ported shared block memory from where other processor can share it for FFT calculation unit and display unit. When user needs to reconfigure one type of filter core module with other type of filter module, he has to send control through HyperTerminal and menu will be displayed asking to select which type of filter he likes to use. Menu will ask to perform reconfiguration for both type of channel or only one. Depending on the user inputs, processor reads appropriate partial bit files from the FLASH memory and replaces the previous module by new one. It is also important to mention here that each filter region having low – pass, high – pass, band – stop and all – pass functionality as a Partial Reconfigurable Module. At the time when reconfiguration is in progress, the playing back of audio data is stopped. Once the reconfiguration is completed, the process starts again.

In this proposed architecture, the filter unit for audio is implemented as two reconfigurable cores dedicated for both left and right channels. The audio filters used for our implementation are capable to filter 4 kHz tone in the audio input stereo data. The specifications of the filters are shown in Table 1.

Table 1 Filter specification used in design

Filter Specification	Filter Type			
	Band Stop		High Pass	Low Pass
Sampling Frequency (kHz)	48		48	48
Pass Band Frequency (Hz)	1st Pass Band	1600	6200	2000
	2nd Pass Band	4200		
Stop Band Frequency (Hz)	1st Stop Band	3800	3800	4200
	2nd Stop Band	6400		
Order	46		44	43
Attenuation (dB)	60		60	60
Pass band ripple factor	1		1	1

The all pass filter model has 13 clock cycles delayed output. The filters are modeled with System Generator having single channel and the model is used twice in the design i.e. for left channel and right channel. The System Generator token is used to generate synthesized file. The block diagram for our filter model is as shown in Fig. 3.

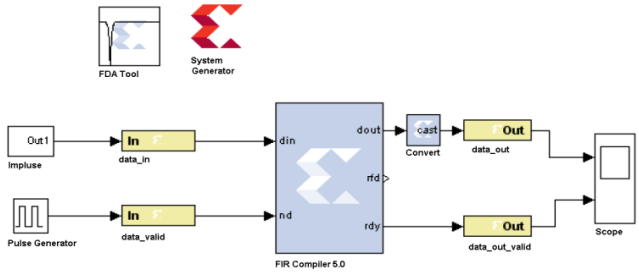

Fig. 3 Block diagram of filter model used in System Generator

The audio data from the filter block is now ready to play and to do spectral analysis. The frequency analysis unit uses 16 sets of filtered data for its computation as we are using 16 point FFT [10]. We are giving only real values to FFT function being executed by soft core micro-blaze keeping imaginary part of the input value as zero. We are calculating 16 inputs to the output of the FFT in different ways using 10 multiplications and 79 additions. Real and imaginary parts of the output from X(0) to X(8) depend on simple addition and subtraction of values from inputs. For the real and imaginary values of the output from X(9) to X(15) we are using formulae as shown in expression (1) and (2) where n varies from 9 to 15.

$$\mathrm{Re}\{X(n)\} = \mathrm{Re}\{X[(16-n) \bmod 16]\} \quad (1)$$

$$\mathrm{Im}\{X(n)\} = -\mathrm{Im}\{X[(16-n) \bmod 16]\} \quad (2)$$

The concept that is working for this software FFT implementation is to compute X [2k] and X [4k+1], individually. To compute X [2k] we are taking 8 point real FFT of input x(n) + x(n+8), where n is from 0 to 7. To calculate F[4k+1] we are taking 4 point complex FFT as shown in expression (3) where n ranges from 0 to 3. We are

also calculating twiddle factor multiplies for the 4 point complex FFT.

$$\exp(-2*pi*j*n/16)*\{x(n)-x(n+8)+j(x(n+12)-x(n+4))\} \quad (3)$$

FFT function used in this design is not only computing FFT values of the inputs but it also produces the magnitude of the output values with proper windowing of the input sampled data. Blockman coefficient has been used for windowing of the input sample to FFT. The output of the FFT is stored and displayed graphically on an 800x600 DVI resolution display using Fairchild FMS 3818 D/A [11] to take data from FPGA to the DVI output.

To calculate frequency resolution of N point FFT, formulae can be given as (4), where f_s is sampling frequency and N is number of FFT points. For the FFT algorithm that we are using in our design, the resolution frequency will be 3000 Hz (48000/16). The output of the FFT is continuously scaled and stored to a dual ported memory by the 2nd processor. One port of the memory is attached with processor through bus and other port is used by display IP core. Each vertical bar displayed on the monitor represents FFT bins of 3000 Hz. The left most display is from 0 to 3000 Hz and next one is from 3000 Hz to 6000 Hz. There are 16 bins being displayed so the right most bar represents 45000 Hz to 48000 Hz.

$$f_{rs} = f_s / N \quad (4)$$

The software code for the 2nd processor continuously reads the audio data from the memory, calculating its Fourier transforms, converting it to bar character and then writes to memory shared with display IP core.

5. IMPLEMENTATION ALANYSIS

The idea has been implemented and tested on ML507 board having XC5VFX70T-1FF1136 device of Xilinx Virtex family. Audio output has been routed to stereo speaker. The spectral chart of the audio has been displayed on monitor having 800x600 resolution. The complete set-up has been shown in Fig. 4.

Since proposed approach is based on HW/SW co-design having dual processor, the base system has been designed with Xilinx Platform Studio tools. This tool is also capable to merge the software application for the both processors. Partial Reconfiguration flow has been adapted using EAPR [12].

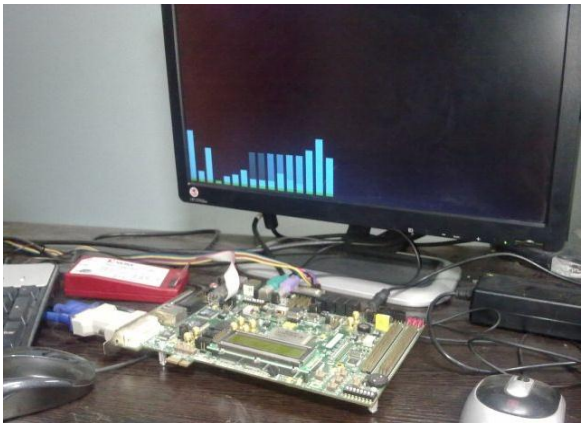

Fig. 4 Experimental Setup of the System

The filter's net-lists has been designed and generated using system generator tool. The rest of the system net-lists has been generated using Xilinx EDK. Once the net-list of the base system and filters are ready, they are integrated in PlanAhead tools to generate initial bit file and partial bit files. As in Fig. 1, two processors have been used and both of them have been clocked on 300 MHz. Rest of the IP cores are running at PLB Bus Frequency that is 100 MHz. Only Display controller IP core is running at 40 MHz.

It is very important to describe here that perfect synchronization mechanism is required between both processors and the monitor display. Since the 2nd processor's speed is much higher than display core, we can keep processor busy in computing Fourier transforms. There are two reconfigurable filter IP cores for two channels. One can internally reconfigure them by loading partial bit files using ICAP port. User can interact with system using HyperTerminal. To establish communication through HyperTerminal, RS232 UART IP has been used. To read audio data from the FLASH memory, SYSACE controller has been used. Partial bit files are also stored in Compact Flash card.

6. RESULTS

Successful implementation of the proposed architecture validates that using partial reconfiguration; a complex real time application can be easily tested. It has been also proved the benefit of partial reconfiguration in minimum resource utilization. It has been successfully showed that the application may be partitioned in between two processors with proper data synchronization. Without using PR flow, eight parallel filter blocks are required in the circuit. The implemented system has only two of the many filter blocks in use at any time of execution. Resource utilization comparison between with PR flow and without PR flow is as shown in Table 2. As in Table II, it can be easily seen that a marginal reduction in resource utilization has been achieved, but the architecture demands more resources in terms of DSP48E block. So, our architecture cannot be implemented using "without PR" flow.

Table 2 Comparison of Resource Utilization

Type of resources	Resource Utilization (no. of Blocks)	
	With PR	Without PR
Slice LUTs	2582	18320
Full Used LUT FF Pair	2340	9360
Slice Registers	2516	10338
DSP48Es	48	192 (cross the max. Limit)

Reconfiguration time taken by ICAP running at 100 MHz has been also measured. This reconfiguration time has been compared with that of external partial reconfiguration method where the downloading of partial bit files has been achieved through JTAG running at 6 MHz. The comparison of the above has been shown as Table 3. As shown in Table 3, one can easily draw an idea that reconfiguration time parameters depends on the size of the partial bit files. Sizes of various bit files have been also given as in Table 3. Since only "With PR" flow has been implemented for our architecture so related information like size of bit files and reconfiguration time has not been given in the Table 3.

Table 3 Comparison of Sizes of the Bit Files and Configuration Time

Design method	Bit File		Size of Bit file	Reconfig. Time Using	
				JTAG	ICAP
Without PR	Static bit file		-	-	-
With PR	Static bit file		3.22 MB	6 scc	-
	P. B. (Left channel)	Low Pass Filter	314 KB	1 sec	370 mS
		High Pass Filter	314 KB	1 sec	370 mS
		All Pass Filter	314 KB	1 sec	375 mS
		Band Stop Filter	314 KB	1 sec	375 mS
	P. B. (Right channel)	Low Pass Filter	316 KB	1 sec	390 mS
		High Pass Filter	316 KB	1 sec	390 mS
		All Pass Filter	316 KB	1 sec	392 mS
		Band Stop Filter	316 KB	1 sec	395 mS

7. CONCLUSION

In this paper, an internal partial dynamic reconfiguration implementation of the real time reconfigurable audio spectrum display has been presented. This implementation involves a mixed HW/SW co design method which uses two instead of all complex hardware cores on FPGA by taking reconfiguration advantages. It has been shown that this architecture proves better in terms of area and performance if compared with fully software implementation or hardware implementation.

Spectral analysis has been performed on left channel audio data only. But this idea may lead to both channel analysis. Fourier transform has been computed in software and it is feasible here because processor is running much faster than display IP core and VGA monitor clock. 16- point FFT computation has been used but 32 or 64 point FFT may be used for better performance for spectral analysis. The hardware implementation of FFT can be also feasible with proper attention to audio data synchronization. The current version of the proposed system is having standalone application, but our next version of the system may integrate a complete real time GUI libraries as well as more complex application like run time video processing, MPEG decoding. Current implementation targets Virtex-5 FX FPGA from Xilinx. However, this may be realized on other FPGA based higher reconfigurable platforms.

8. ACKNOWLEDGMENTS

The authors would like to thank Dr. Chandra Shakher, Director CSIR-CEERI for allowing utilizing the resources of the institute, as well as Dr. P. Bhanu Prasad, Group Leader, Digital Systems Group, for valuable contributions and discussions.

9. REFERENCES

[1] L. Bauer, M. Shafique and J. Henkel,"Efficient resource utilization for an extensible processor through dynamic instruction set adaptation," IEEE Trans. Very Large Scale Integr. Syst., Vol. 16, pp. 1295 – 1308, October 2008. [Online]. Available: http://portal.acm.org/citation.cfm?id=1515843.1515848

[2] Bhandari, S.U.; Subbaraman, S.; Pujari, S.S.; Mahajan, R.; , "Real Time Video Processing on FPGA Using on the Fly Partial Reconfiguration," 2009 International Conference on Signal Processing Systems , pp.244-247, 15-17 May 2009

[3] Z. E. A. A. Ismaili and A. Mousa, "Self-partial and dynamic reconfiguration implementation for aes using fpga," CoRR, vol. abs/0909.2369, 2009.

[4] Tumeo, A.; Monchiero, M.; Palermo, G.; Ferrandi, F.; Sciuto, D.; , "An Internal Partial Dynamic Reconfiguration Implementation of the JPEG Encoder for Low-Cost FPGAsb," VLSI, 2007. ISVLSI '07. IEEE Computer Society Annual Symposium on , pp.449-450, 9-11 March 2007.

[5] Raikovich, T.; Feher, B.; , "Application of partial reconfiguration of FPGAs in image processing," Ph.D. Research in Microelectronics and Electronics (PRIME), 2010 Conference on , pp.1-4, 18-21 July 2010.

[6] Feilen, M.; Ihmig, M.; Zahlheimer, A.; Stechele, W.; , "Real-time signal processing on low-cost-FPGAs using dynamic partial reconfiguration," Integrated Circuits (ISIC), 2011 13th International Symposium on , pp.110-113, 12-14 Dec. 2011

[7] Youmu Zhang; Jie Yang; , "Design of spectral analyzer based on FPGA," Mechanic Automation and Control Engineering (MACE), 2011 Second International Conference on , pp.4008-4011, 15-17 July 2011.

[8] Chang-Seok Choi; Hanho Lee; , "An Reconfigurable FIR Filter Design on a Partial Reconfiguration Platform," Communications and Electronics, 2006. ICCE '06. First International Conference on , pp.352-355, 10-11 Oct. 2006

[9] Vaibhawa Mishra, Kota Solomon Raju, Pramod Tanwar, "Implantation of Dynamically Reconfigurable Systems on Chip with OS Support", International Journal of Computer Applications (IJCA), Vol. 49, No 6, July, 2012, pp. 33-35.

[10] The MIT webpage. [Online]. Available: http://www.mit.edu/~emin/source_code/fft/index.html

[11] Xilinx Inc., "UG347:ML505/ML506/ML507 Evaluation Platform," May, 2011.

[12] Xilinx Inc., "Early Access Partial Reconfiguration User Guide," March, 2005.