# A Lifting Instruction for Performing DWT in LEON3 Processor based System-on-Chip

Rajul Bansal[1,2], Mahendra Kumar Jatav[2] and Abhijit Karmakar[1,2]

[1] Academy of Scientific and Innovative Research (AcSIR), CEERI Campus, Raj, India 333031
[2] CSIR - Central Electronics Engineering Research Institute (CEERI), Pilani, Raj, India 333031

**Abstract.** Discrete Wavelet Transform (DWT) calculations form an inherent part of many signal processing applications. Application specific instructions provide a means to increase performance and efficiency of System-on-Chip (SoC) requiring DWT operations. In this paper, lifting scheme based hardware for efficient DWT calculation, is implemented as an instruction to enhance the performance of an SoC. The hardware is integrated using the coprocessor interface of the SPARCv8 ISA based LEON3 processor. This method for attaching lifting hardware is found to be much more efficient than the prevalent system-bus based integration. The performance measure is provided in terms of CPI and MIPS along with FPGA and ASIC implementation results of the SoC.

**Keywords:** Lifting Scheme, DWT, LEON3, SPARCv8, System-on-Chip, SoC.

## 1 Introduction

Application specific instructions of extensible processor as well as coprocessor extensions of any processor, provide significant area and performance improvements by introducing application specific hardware functionality [1]. Custom instructions require architectural modification of processor pipeline that necessitates time consuming reverification of entire processor pipeline. Contrary to this, a coprocessor extension enables addition of application specific hardware as coprocessor instructions, without the need for modifying existing processor pipeline. It is reasoned that custom instructions are suitable for fine grained operations whereas coprocessor extensions are best suited for coarse grained tasks [2]. In this paper, we have proposed an architecture for computation of Discrete Wavelet Transform (DWT), incorporating lifting scheme [3] based custom hardware coprocessor instructions.

DWT is used in modern day audio, video, image and various signal processing applications due to inherent benefits, such as, multiresolution representation, sequential processing [4] and absence of blocking artifacts noticeable in Discrete Cosine Transform (DCT) based image processing. SoCs catering to advanced applications ranging from compressing 3D data such as that of an MRI scan [5] to combustion failure detection of IC engines through vibration signal analysis [6], demand higher processing and low power requirement. Thus, making efficient hardware implementation a necessity rather than a requirement.

Most of the lifting-based architectures, including some of the recent ones [7-9], focus on memory reduction, throughput, latency, cycle count, pipeline equalization, frequency of operation, hardware utilization, area and low power. However, the effect of their integration in a modern SoC environment and corresponding results have not been provided. Our paper presents a case where lifting hardware is integrated as an instruction in the open source LEON3 processor pipeline, utilizing the coprocessor interface rather than the standard bus-based approach. The performance analysis is computed and compared both at software as well as hardware implementation level.

## 2 Lifting Hardware

Discrete wavelet transform gained widespread use for various signal processing applications after the proposition of lifting scheme by W. Sweldens [3] that reduced the hardware requirement for wavelet filter implementation by half. In this paper, we have implemented the lifting hardware for one dimensional, single level DWT and integrated it as an application specific custom coprocessor instruction, in order improve the system level performance.

In our work, a lossless (5, 3) filter is chosen as the wavelet filter for implementation. After breakup of poly-phase matrix and converting it to banded matrices multiplications, the final spatial domain equations formed after all the mathematical steps [3], can be written as (1) and (2) where $j$ represents the DWT level and $i$ represents indexing of input samples. Based on these lifting equations, the designed hardware constituting three lifting steps namely: Split, Predict and Update is as shown in Fig. 1.

$$odd_{j+1,i} = odd_{j,i} - (even_{j,i} + even_{j,i+1})/2 \quad (1)$$

$$even_{j+1,i} = even_{j,i} + (odd_{j+1,i-1} + odd_{j+1,i})/4 \quad (2)$$

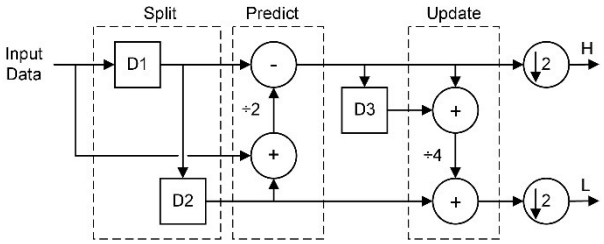


**Fig. 1.** Lifting hardware

## 3 Integration with LEON3 Processor

The integration of lifting hardware with SPARCv8 compliant 7-stage pipelined LEON3 processor is accomplished using only the load, store and operate coprocessor instructions. In the detailed integration diagram shown in Fig. 2, FE, DE, RA, EX, MA, XC and WB represent fetch, decode, register access, execute, memory access,

exception and write back stages of the LEON3 pipeline. Among the seven stages, control signals only from decode, execute and exception stages are required for lifting coprocessor integration. The signal, *cpo_store* is the only output and carries the store data back to integer unit which is subsequently stored back to data cache and in turn to RAM. The *cpi_lddata* carries data from data cache or RAM, to be loaded in the coprocessor register file.

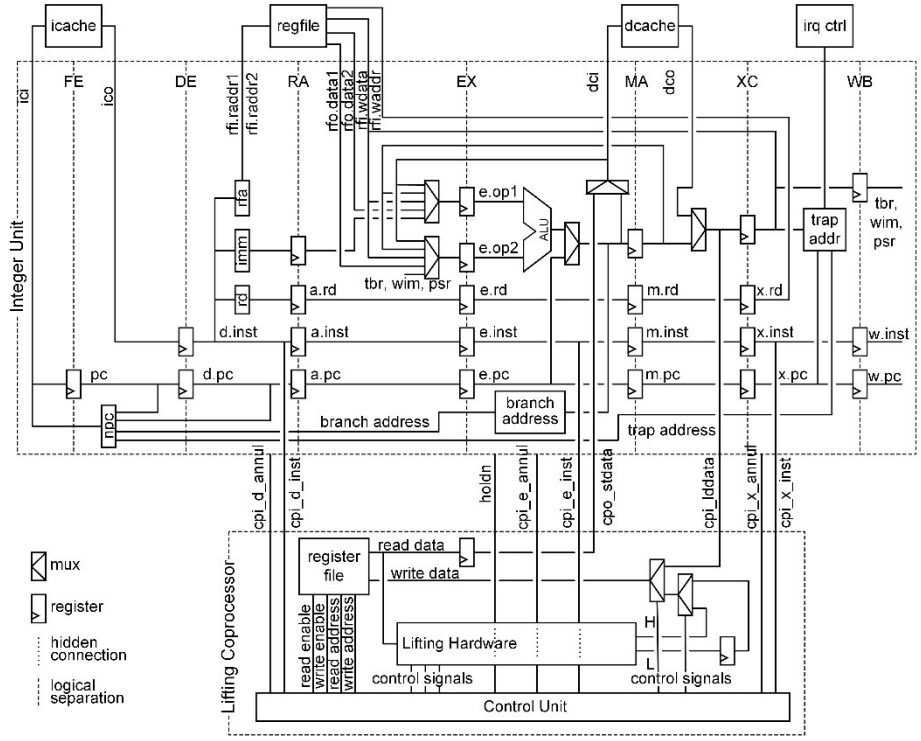


**Fig. 2.** Integration of lifting hardware with LEON3 pipeline

The lifting hardware gets *read data* from register file only when coprocessor operate instruction is issued. The resulting H and L are saved back to register file in alternate cycles. Since DWT in an in-place transform, the input data values can be overwritten and as a result, 32 samples can be operated upon in a single sequence of coprocessor instructions. In case, there is a situation when LEON3 pipeline gets stalled, the stalled status of the processor pipeline is forwarded through *holdn* signal, upon whose arrival the control unit stalls the lifting hardware and prevents loss of synchronization. This is required only when performing coprocessor load and store that must be synchronized with respective stages in the LEON3 pipeline. However, if LEON3 pipeline gets stalled during the coprocessor operate instruction, the lifting hardware operations proceed without any pause and this parallel execution of both the pipelines increases the CPI of the system.

## 4 Performance Analysis and Comparison

System modelling has been done on TSIM2 Instruction Set Simulator (ISS) at higher abstraction level for faster execution of application codes. The application code performs single-level, one-dimensional DWT of an image using coprocessor instructions integrated as assembly code snippets in the embedded C code of the application. To compare the results of our implementation, we have modelled three different SoC configurations. The first configuration (*Config* 1) computes the DWT operation using generic SPARCv8 instructions i.e. without custom lifting instructions. *Config* 2 computes the same using the proposed custom lifting hardware instructions, whereas, *Config* 3 computes the same through lifting hardware attached via AHB system bus with requisite bus interface logic.

Based on the results shown in Table 1 for various sizes of an image, *Config* 2 provides an average of 30% reduction in number of cycles from pure software based implementation and 18% reduction in number of cycles from bus-based implementation. The percentage of reduction in number of cycles increases for larger image sizes. Moreover, for DWT using custom instructions, the number of cycles, instructions, CPI and Million Instructions Per Second (MIPS) are found to be better than both *Config* 1 and *Config* 3. It is further seen from Table 1 that the CPI/MIPS benefits grow with increase in the size of the image in case of designs with custom lifting instructions, whereas the CPI/MIPS values remain almost the same for DWT operations done using generic SPARCv8 instructions. It can be reasoned that as the image size is increased, the percentage of coprocessor instructions in the machine code increases, which subsequently causes the decrease of CPI values.

**Table 1.** Cycles per Instruction (CPI) performance

| Image Size | Parameter | DWT using generic SPARCv8 instructions | DWT using custom instructions | AHB bus coupled hardware based DWT |
|---|---|---|---|---|
| 32x32 | Cycles / Instructions | 114992 / 47456 | 78650 / 36121 | 94183 / 42593 |
| | CPI / MIPS | 2.42 / 41.23 | 2.18 / 45.93 | 2.21 / 45.22 |
| 64x64 | Cycles / Instructions | 323771 / 130210 | 178494 / 85340 | 237035 / 111012 |
| | CPI / MIPS | 2.49 / 40.22 | 2.09 / 47.81 | 2.14 / 46.83 |
| 128x128 | Cycles / Instructions | 1148029 / 460068 | 570390 / 283678 | 800963 / 386150 |
| | CPI / MIPS | 2.49 / 40.07 | 2.01 / 49.73 | 2.07 / 48.21 |
| 256x256 | Cycles / Instructions | 4424701 / 1777188 | 2116118 / 1073182 | 3034819 / 1482854 |
| | CPI / MIPS | 2.49 / 40.16 | 1.97 / 50.71 | 2.05 / 48.86 |
| 512x512 | Cycles / Instructions | 17490685 / 7041060 | 8261142 / 4228126 | 11932355 / 5866598 |
| | CPI / MIPS | 2.48 / 40.25 | 1.95 / 51.18 | 2.03 / 49.17 |

**Table 2.** FPGA and ASIC synthesis results

| Parameter | Standard design (*Config* 1) | With custom instruction (*Config* 2) | With bus-based integration (*Config* 3) |
|---|---|---|---|
| No. of slice registers (93296) | 5666 | 5963 | 6014 |
| No. of slice LUTs (46648) | 20891 | 21210 | 21311 |
| No. of RAM blocks (172) | 42 | 46 | 46 |
| FPGA Power (mW) | 310 | 306 | 314 |
| Total ASIC cell area ($um^2$) | 12758050 | 12879652 | 12884508 |
| ASIC Power (mW) | 549.1 | 554.9 | 555.7 |

The three design cases are also implemented in HDL targeting Spartan 6 XC6SLX75 FPGA device using Xilinx ISE 13.4. They are further targeted for standard cell based ASIC flow using UMC 180nm based Faraday standard cell library. Synthesis results presented for both in Table 2 show that there is only a slight increase in resource utilization on integrating additional lifting hardware and this increase can be justified for the amount of performance improvement it provides. Moreover, the increase in the hardware is lesser than the traditional integration approach using system bus. This is because of reduced address decode and control logic within the arbiter unit of AHB bus. The proposed integration achieves lower power consumption than the bus-based design making it power efficient as well.

## 5 Conclusion

In this paper, we have presented a unique case where lifting hardware is integrated as an instruction in a processor pipeline utilizing the coprocessor interface. Analysis of results suggest that the proposed hardware modification provides significant performance and power benefits. The lifting hardware that is integrated in our design can be used recursively to implement multilevel as well as multidimensional DWT. Moreover, a dual scan architecture can also be integrated to achieve higher hardware utilization and further performance benefits can be attained.

## References

1. S. O'Melia and A. J. Elbirt: Enhancing the Performance of Symmetric-Key Cryptography via Instruction Set Extensions. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 18(11), 1505-1518 (2010).
2. F. Sun, S. Ravi and N. K. Jha: A Synthesis Methodology for Hybrid Custom Instruction and Coprocessor Generation for Extensible Processors. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 26(11), 2035-2045 (2007).
3. W. Sweldens: The Lifting Scheme: A Custom-Design Construction of Biorthogonal Wavelets. Applied and Computational Harmonic Analysis 3(2), 186-200 (1996).
4. V. M. Potdar, S. Han and E. Chang: A survey of digital image watermarking techniques. In: IEEE International Conference on Industrial Informatics, (2005).
5. W. Badawy, M. Weeks, G. Zhang, M. Talley and M. A. Bayoumi: MRI Data Compression Using a 3-D discrete wavelet transform. IEEE Engineering in Medicine and Biology Magazine, 21(4), 95-103 (2002).
6. F. A. Shirazi and M. J. Mahjoob: Application of Discrete Wavelet Transform (DWT) in Combustion Failure Detection of IC Engines. In: International Symposium on Image and Signal Processing and Analysis ISPA, (2007).
7. B. K. Mohanty, A. Mahajan and P. K. Meher: Area- and power-efficient architecture for high-throughput implementation of lifting based 2-D DWT. IEEE Transactions on Circuits and Systems-II: Express Briefs 59(7), 434-438 (2012).
8. W. Zhang, Z. Jiang, Z. Gao and Y. Liu: An efficient VLSI architecture for lifting-based discrete wavelet transform. IEEE Transactions on Circuits and Systems II: Express Briefs 59(3), 158-162 (2012).
9. Y. Hu and C. C. Jong: A memory-efficient high-throughput architecture for lifting-based multi-level 2-D DWT. IEEE Transactions on Signal Processing 61(20), 4975-4987 (2013).