A high performance processor architecture for multimedia applications

Shafqat Khan\textsuperscript{a}, Muhammad Rashid\textsuperscript{b,\textasteriskcentered}, Faraz Javaid\textsuperscript{c}

\textsuperscript{a}COMSATS Institute of Information Technology Abbottabad, Pakistan
\textsuperscript{b}Umm Al-Qura University (UQU), Makkah, Saudi Arabia
\textsuperscript{c}Center for Advance Studies in Engineering (CASE), Islamabad, Pakistan

\textbf{A R T I C L E   I N F O}

Article history:
Received 17 October 2016
Revised 25 September 2017
Accepted 25 September 2017
Available online 29 September 2017

Keywords:
Reduced instruction-set computer (RISC)
Application-specific instruction-set processor (ASIP)
Multimedia processing
Sub-word parallelism
Re-configurable system

\textbf{A B S T R A C T}

In this paper, an efficient sub-word parallelism (SWP)-enabled Reduced instruction-set Computer (RISC) architecture is proposed. The proposed architecture can perform efficiently for both conventional and multimedia-oriented applications. Speed-up for multimedia applications is achieved by adding the customized SWP instructions in RISC processor core. Rather than operating on a single data, customized instructions perform parallel computations on multiple pixels, packed in word-size registers. The sub-word-sizes in SWP instructions are selected, based upon the pixel sizes (8, 10, 12, 16-bit) in modern multimedia applications. The SWP-RISC processor is designed and implemented on two different CMOS technology nodes (90 nm and 45 nm). The performance of processor is characterized for different multimedia applications and compared with the state-of-the-art TMS320C64X processor.

© 2017 Elsevier Ltd. All rights reserved.

1. Introduction

Nowadays, modern electronic devices like smart phones, digital cameras, automobiles etc. have to perform a variety of tasks. For the scheduling, controlling and smooth functionality of multiple tasks, a processor core is required in electronic devices. Most of the embedded processor designs are based upon Reduced instruction-set Computer (RISC) architecture because of its simplicity and optimization in instruction-set. The hardware architecture of RISC-based processor can be easily optimized for high speed applications using various performance enhancement techniques [1].

1.1. Related work

Several enhancements have been proposed in RISC architecture to increase the performance. For example, instructions pertaining to JPEG and H.264 encoding standards have been implemented and incorporated in the instruction-set of a RISC processor core \cite{2,3}. Similarly, a RISC instruction-set architecture for a flexible Multiply-Accumulate (MAC) unit of a Very-Large-Instruction-Word (VLIW) Digital Signal Processor (DSP) core for 32, 16 and 8-bit data computations is presented in \cite{4}. While Rashid et al. \cite{2,3} and Huu et al. \cite{4} advocate the use of instruction-set extension, a sub-word parallelism (SWP)
technique has been proposed in some multimedia processors for the enhancement of performance at operation level [5,6]. Consequently, several SWP-based multimedia extensions have been implemented in both general purpose as well as DSP architectures. The typical examples of this trend are the inclusion of Multi Media Extension (MMX) and the introduction of streaming Single-Instruction-Multiple-Data (SIMD) extensions SSE, SSE2 and SSE3 in IA-32 Intel architecture [7].

In addition to the Intel architecture, a SIMD-based general purpose datapath with efficient operation structure for motion estimation algorithm is presented in [8]. Similarly, a configurable and data-reusable instruction-set processor, exploiting the symmetry of search pattern to reduce the redundant data loading, is presented in [9]. However, most of these multimedia enhancements are based upon the conventional sub-word-sizes of 8, 16, 32-bit without considering the pixel sizes (8, 10, 12-bit) in most modern multimedia application [10,11]. As a result, the under utilization of processor resources occurs which ultimately reduces the performance. In [12], an extended RISC-V processor core, specifically designed for near-threshold (NT) operation in tightly coupled multi-core cluster, has been proposed for ultra-low power systems. Similarly, Hussain et al. [13] presents an integrated self-aware computing model, mitigating the power dissipation of a heterogeneous re-configurable multi-core architecture, by dynamically scaling the operating frequency of each core.

To summarize, in most of the customized processing cores, the main focus is to increase the performance by incorporating the instructions pertaining to the targeted applications without considering the characteristics of the data that needs to be processed. This lack of coordination between data and the underlying hardware architecture reduces the overall performance of the processor.

1.2. Our work

In this paper, a customized RISC processor core is proposed for multimedia applications. The performance of processor is increased, both at instruction level as well as at operation level. At instruction level, the customized instructions related to the operations required in modern multimedia applications, are incorporated in the RISC instruction-set. The customized multimedia instructions are implemented using synopsis processor design tool [14]. These instructions help to perform the multimedia-oriented operations like discrete cosine transform (DCT), motion estimation, discrete wavelet transform (DWT), image enhancement, pixel summations, sum of absolute difference etc. very efficiently.

At operation level, the performance of RISC processor is improved by using the sub-word parallelism (SWP) in the architecture of different arithmetic operators. In SWP, multiple pixels are packed in word-size registers and parallel computations are performed on the packed pixels. Consequently, the word-size resources are utilized to maximum extent even for the low precision pixel data. In this work, multimedia-oriented sub-word-sizes (8, 10, 12, 16-bits) are considered instead of classical sub-word-sizes (8, 16, 32-bits).

The implementation results of the proposed processor in terms of area, power and critical path prove the viability of our approach. Furthermore, the performance is compared with the recent TMS320C64x processor [15], having classical multimedia instruction-set. Due to efficient implementation and better coordination between pixel data and hardware architecture, the proposed processor in this paper provides higher speed-up for modern multimedia applications.

The rest of this paper is organized as follows: Section 2 sets the stage for the designing of SWP-RISC core, targeted at multimedia applications. Section 3 presents the architecture of customized multimedia instructions, added in the SWP-RISC core. Furthermore, the resources required for implementing each customized instruction are also analyzed. Section 4 describes the hardware implementation of SWP-RISC processor. Moreover, the chip area, timings and power consumption of the proposed processor are compared with the conventional RISC processor. Section 5 explains the performance of proposed processor when working on different multimedia applications. The performance is also compared with the state-of-the-art digital signal processor TMS320C64X, having conventional multimedia extension. Finally, the conclusion is presented in Section 6.

2. Setting the stage

This section starts with the brief overview of RISC architecture followed by the multimedia-oriented SWP technique which is used in the SWP-RISC architecture. Finally, the advantages of using ADL for processor designing are explained.

2.1. Overview of selected RISC architecture

RISC is a type of architecture that utilizes a small, highly-optimized set of instructions [1,4]. In this paper, SWP-enabled multimedia enhancements are proposed in five-stage RISC architecture. This RISC architecture is selected because of its simplicity and the hardware corresponding to multimedia instructions can easily be implemented on a RISC platform without major modifications. RISC under consideration in this article has 32 32-bit general purpose registers (GPRs) and few special-purpose registers (e.g. program counter and instruction register). In the selected RISC architecture, the range of program memory is kept limited to 32 Kilo word memory locations (0x0000 to 0xFFFF). Each memory location is 32-bits in size. The data memory stores the input data on which the program operates as well as the final results computed from the operations. Each memory location is 32-bits in length such that the data can be transferred from memory to GPR in one clock cycle. The defined range is kept limited to 32 Kilo word memory locations (0x8000 to 0xFFFF). Furthermore, the load-store
RISC architecture is used in which the data-processing takes place only on register contents. All the instructions are uniform and of fixed length (32-bit).

2.2. Sub-word parallelism technique

To increase the performance of RISC processors, an SWP technique [5] has been carried out on the dedicated arithmetic operators. These SWP operators perform parallel operations on sub-words which are conveniently compatible with the word-size of the processor. By doing this, the processor can achieve more parallelism rather than wasting the word-size data-path and register sizes when operating on low-precision data. Conventionally, the word-size of processor is a multiple of sub-word-sizes which helps to reduce the complexity of SWP operators. For instance, some of the conventional sub-word-sizes for 32-bit processors are 8, 16 and 32 bits. As an example, Fig. 1 shows four basic addition operations on 8-bit sub-word in a 32-bit processor.

In multimedia applications, the sizes of the input data for computations are 8, 10, 12 or sometimes 16-bits. These multimedia data sizes are not in coordination with the existing processor’s sub-word-sizes, resulting in the under utilization of processor’s resources [16]. To increase the performance of multimedia applications, our proposed SWP-RISC processor considers multimedia-oriented sub-word-sizes rather than the conventional sub-word-sizes. Consequently, these operators can perform operations on word-size operands (32-bits) as well as on sub-words (four 8-bit sub-words or three 10-bit sub-words or two 12-bit sub-words or two 16-bit sub-words), packed in word-size registers.

2.3. Processor designing using ADL

In this paper, an architectural descriptive language (ADL) [17] is used for the designing of SWP-RISC processor. In ADL, the processor resources and the architecture of a complete processor are described. Different architectural descriptive languages are available and for our SWP-RISC processor design, we have chosen LISA due to its flexibility for designing arbitrary processor architectures.

2.3.1. LISA overview

LISA is used to model any type of processor that is defined by an instruction-set. It allows easy modeling of cycle-accurate, single-instruction-multiple-data (SIMD) and very-long-instruction-word (VLIW) processors. LISA descriptions are composed of resource and operations declarations. The declared resources represent the storage objects of the hardware architecture (e.g. registers, memories, pipelines etc). Operation definitions collect the description of different properties of the system (e.g. operation behavior, instruction-set information and timing).

2.3.2. Processor Designer tool

In this work, Synopsys Processor Designer tool [14] is used for the designing of customized processor. As shown in Fig. 2, the input to Processor Designer tool is the LISA model of the processor. The LISA model describe the resources and architecture of instructions. The Processor Designer generates Register-Transfer-Level (RTL) code and software development tools such as assembler, dis-assembler, linker, instruction-set simulator, debugger and C-compiler. Combining hardware and software design in a processor design platform greatly reduces the complexity and time of modeling. Before the generation of RTL code, the LISA model of a processor is tested and debugged using the processor debugger tool. The designer can view the status of all the hardware resources, pipeline stages, registers and memories. After the LISA model is verified, the RTL code is generated automatically.
3. Customized SWP-RISC instructions

SWP-RISC processor is designed to enhance the performance of a RISC-based processor for multimedia applications. Along with the conventional instruction-set, it also contains customized instructions to perform computations on multimedia-oriented data. The customized instructions perform computations on multiple sub-words/pixels, packed in word-size registers. Consequently, computations are performed by a single instruction while utilizing the word-size resources. Moreover, the customized SWP instructions can perform computations on 8, 10, 12, 16, or 32-bit data. The operations performed by the SWP instructions are selected according to the computational requirements in multimedia applications like discrete cosine transform (DCT), motion estimation (ME), discrete wavelet transform (DWT), image enhancements, image/video compression and so on. Furthermore, the general purpose multimedia-oriented SWP instructions like ADD/SUB, multiply, sum-of-absolute-difference (SAD), sum-of-products (SOP) are also designed and included in the customized instruction-set. In other words, SWP instructions can perform computations on the pixel-size data as well as on the word-size data. Specific bits in the instruction will determine the sub-word-size. Consequently, it has the potential to meet the processing requirements of computationally intensive multimedia applications. Hardware architecture, instruction format and resources required to implement each SWP instruction are explained in the next sub-sections.

3.1. SWP ADD/SUB instruction

Addition is one of the most important and commonly used operation in conventional as well as in multimedia domain. SWP ADD/SUB instruction is used to add pixels in parallel. For 8-bit sub-word-size, the SWP ADD/SUB instruction performs four parallel 8-bit addition operations. Similarly, for 10, 12, 16-bits sub-word-sizes, it performs parallel additions on input pixel data accordingly. At the output, multiple sum values are available in the same clock cycle.

**Instruction format:** The assembly syntax of SWP ADD/SUB instruction is given below:

- **SWPADD dest, src1, src2, mode**
- **SWPSUB dest, src1, src2, mode**

SWPADD and SWPSUB are 8-bit opcodes. src1, src2 and dest are 32-bit registers specifiers. Each register is represented by five bits in the SWP instruction. Finally, the 4-bit mode value indicates the sub-word-size.
Instruction architecture: The hardware architecture of SWP ADD/SUB instruction for 8-bit sub-word-size is shown in Fig. 3. However, for other sub-word-sizes (10, 12, and 16-bits), the processor hardware will reconfigure itself accordingly.

3.2. SWP MULT instruction

SWP MULT instruction is used to multiply the corresponding sub-words from multiplicand and multiplier registers. The product sub-word has a bit-width, double to the input sub-word-sizes. For example, for 8-bit sub-word-size, each product sub-word consists of 16 bits. Four 16-bit product sub-words are stored in two 32-bit registers (Prod-Hi and Prod-Low). Similarly, for other sub-word-sizes (10, 12, 16-bit), the product sub-words are generated and stored accordingly.

Instruction format: The assembly syntax of 32-bit SWP MULT is given below:

\[
\text{SWPMULT dest-hi, dest-lo, src1, src2, mode}
\]

SWPMULT is an 8-bit opcode. The src1 and src2 are 32-bit input register specifiers. The dest-lo and dest-hi are 32-bit output product registers. The mode indicates the selected sub-word-size.

Instruction architecture: SWP control signals (sw8, sw10, sw12 and sw16) are utilized in the generation of partial products. Due to the dependence of partial products on sub-word control signals, the generated partial products remain valid for different selected sub-word-sizes. For 8-bit sub-word-size, the swp control signals are: sw8 = 1, sw10 = 0, sw12 = 0 and sw16 = 0. The same SWP multiplier will generate the product sub-words for the selected sub-word-sizes accordingly. The 32-bit multiplicand (src1), 32-bit multiplier (src2) and swp control signals (sw8, sw10, sw12 and sw16) are used to generate internal signals, as shown in the following equations:

\[
\begin{align*}
\text{az8i} &= \text{src1} \cdot \left(\text{1}_{\text{sw8}}\right) \\
\text{az10i} &= \text{src1} \cdot \left(\text{1}_{\text{sw10}}\right) \\
\text{az12i} &= \text{src1} \cdot \left(\text{1}_{\text{sw12}}\right) \\
\text{az16i} &= \text{src1} \cdot \left(\text{1}_{\text{sw16}}\right) \\
\text{az8_10i} &= \text{src1} \cdot \left(\text{1}_{\text{sw8 + sw10}}\right) \\
\text{az8_12i} &= \text{src1} \cdot \left(\text{1}_{\text{sw8 + sw12}}\right) \\
\text{az8_16i} &= \text{src1} \cdot \left(\text{1}_{\text{sw8 + sw16}}\right) \\
\text{az10_12i} &= \text{src1} \cdot \left(\text{1}_{\text{sw10 + sw12}}\right) \\
\text{az10_16i} &= \text{src1} \cdot \left(\text{1}_{\text{sw10 + sw16}}\right) \\
\text{az12_16i} &= \text{src1} \cdot \left(\text{1}_{\text{sw12 + sw16}}\right) \\
\text{az8_10_12i} &= \text{src1} \cdot \left(\text{1}_{\text{sw8 + sw10 + sw12}}\right) \\
\text{az8_10_16i} &= \text{src1} \cdot \left(\text{1}_{\text{sw8 + sw10 + sw16}}\right) \\
\text{az8_10_12_16i} &= \text{src1} \cdot \left(\text{1}_{\text{sw8 + sw10 + sw12 + sw16}}\right)
\end{align*}
\]

These internal signals are used in the generation of partial products in different regions. The internal signals valid in different regions of the partial products array are shown in Fig. 4.

Each partial product is generated by ANDing the appropriate internal signals with multiplier (src2) signal. For example, the partial product PP0 is generated by ANDing 32-bit src2 with the concatenation of the following internal signals: (az8_10_12_16i[31 : 14], az8_10_12i[13 : 12], az8_10i[11 : 10], az8i[9 : 8], srci[7 : 0]). Similarly, PP31 is generated by
ANDing the src2 with the concatenation of following signals: \((az8.10.12.16[31:14], az8.10.12[13:12], az8.10[11:10], az8[9:8], src1[7:0])\).

The remaining partial products are also generated in the same manner. Finally, the partial products are added to obtain the final product. The advantage of using this technique is that a separate hardware is not required for different sub-word-size multiplications. Through the efficient use of SWP control signals, same hardware is used for different sub-word-size multiplications.

### 3.3. SWP SAD instruction

Sum-of-absolute-difference (SAD) is used to find the difference between two image blocks. SAD is the most commonly used operations in multimedia applications like motion estimation, image enhancement and image filtering etc. In SAD, the absolute difference between the corresponding pixels in the two image blocks are computed and accumulated. Minimum value of SAD indicates the high level of resemblance between the image blocks and vice versa.

SWP SAD instruction is designed to accumulate the difference between multiple corresponding pixels instead of a single pixel. For 8-bit sub-word-size, SWP SAD instruction accumulates the absolute difference between four pixels in parallel. On \((8 \times 8)\) pixels image block, sixteen iterations of SWP SAD instructions are required to compare two images. However, on the conventional RISC processor, the same task is performed using 64 subtract, absolute and accumulate instructions. Similarly, for other sub-word-sizes, the parallelism in SAD operations are achieved accordingly.

**Instruction format:** The assembly syntax of 32-bit SWP SAD instruction is given below:

\[
\text{SWPSAD dest, src1, src2, mode}
\]

SWPSAD is an 8-bit opcode. Source registers src1 and src2 contain packed pixels from two different image blocks. The accumulated SAD value is stored in a 32-bit destination register dest. The mode determines the size of pixels stored in the input registers.

**Instruction architecture:** The hardware architecture of SWP SAD instruction is shown in Fig. 5.
In each clock cycle, the absolute value of difference is calculated on multiple pixels in parallel using SWP \(|a - b|\) unit. Instead of implementing absolute operation directly, the absolute difference operation is obtained by subtracting smaller pixel from the larger pixel value. Smaller pixel value is determined by using a comparator circuit. The SWP sub-word adder unit is used to add the sub-words, packed within the word-size register. Finally, the ACCUMULATOR is used to accumulate the absolute difference values recursively.

3.4. SWP ME instruction

Motion estimation is the most commonly used algorithm to remove the temporal redundancies. In motion estimation, the candidate block in the current frame is compared with different blocks in the reference frame and the best match is searched. Instead of transmitting the whole block, the difference between the candidate block in the current frame and the best-match block in the reference frame is transmitted in the form of a motion vector. At the receiving end, the motion vector indicates the location of best match-block in the reference frame [18]. There are different cost functions to estimate the differences between the blocks. In the proposed SWPME instruction for motion estimation, SWP SAD is used as a cost function due to its simplicity and efficiency.

**Instruction format:** The assembly syntax of 32-bit SWPME is given below:

```
SWPME motion-vec, ref-addr, cand-addr; mode
```

SWPME is an 8-bit opcode. ref-addr and cand-addr indicate the address of reference frame and candidate block in the memory. The mode indicates the size of pixels or sub-word-size.

**Instruction architecture:** The dedicated hardware architecture corresponding to SWP ME instruction is shown in Fig. 6. SWP-ctrl signal selects the sub-word-size based upon the pixel size (8, 10, 12 or 16-bit). The SWP read pixels units in Fig. 6 (for the search area image and the current block) are used to read the required pixel values from RAM and pack them in word-size register for the onward processing. The blocks in search area or reference frame are compared with the current block. SWP SAD operator unit is used to compute the SAD value corresponding to the current block and the search area block. The internal architecture of SWP SAD operator is shown with dotted lines. The SAD comparator is used for comparing the current SAD value with the previously obtained minimum SAD value to obtain the global minimum SAD value. The coordinates of the block in the search area corresponding to the global minimum SAD value gives the motion vector. The proper sequence of all the activities are controlled by a SWP state machine controller.

3.5. SWP DCT instruction

Discrete cosine transform (DCT) is used to convert the time domain image into frequency domain [19]. The mathematical expression for performing N-point DCT operation is given in equation below:

\[
X(K) = \alpha(K) \sum_{n=0}^{N-1} u(n) \cos \left(\frac{2\pi}{2N}K\right)n \\
\alpha(0) = \sqrt{\frac{1}{N}} \\
\alpha(k) = \sqrt{\frac{2}{N}} \quad 1 \leq k \leq N - 1
\]

For a 8-point DCT operation, \(N\) has a value of 8 in the equation. The inputs to 8-point DCT operator are eight pixel values \((u(0) \ldots u(7))\). After performing the DCT computations, eight DCT values \((X(0) \ldots X(7))\) are obtained at the output of operator. Instead of applying the DCT on the whole image, it is usually applied on the smaller blocks of \(8 \times 8\) size. To reduce the complexity of 2-D DCT, the DCT operation is first applied on the rows of the image block followed by the transpose operation and then the row-wise DCT operation is performed once again.

**Instruction format:** The assembly syntax of SWPDCT instruction is given below:
**SWPDCT** `DCT-img-addr, input-img-addr, mode`

`SWPDCT` is the opcode. `input-img-addr` indicates the address of an input `(8 \times 8)` image block in the memory. `DCT-img-addr` indicates the memory address where the output DCT matrix will be stored. The `mode` represents the sub-word/pixel size.

**Instruction architecture:** The hardware architecture corresponding to the implementation of `SWPDCT` instruction is shown in Fig. 7 [6].

As shown in Fig. 7, vector formation for 1D DCT unit is used to pack multiple pixels in a word-size register. Pre transpose `SWP 8-point DCT` unit is used to perform parallel computations on multiple pixel rows rather than single row. The arithmetic operations performed in each pipeline stage of the Pre transpose `SWP 8-point DCT` unit are also shown in Fig. 7 with dotted lines. After 1-D DCT process, packed values are arranged in a proper location using Arrangement of 1D DCT values unit. Transpose unit is used to convert rows into columns in 1-D DCT matrix. DCT operation is again applied on transposed 1-D DCT matrix to obtain 2-D DCT values. `SWP state machine controller` unit is used to control the sequence of all the activities in DCT operation.

---

**Fig. 6.** SWP motion estimation architecture.
Fig. 7. SWP DCT architecture.
3.6. SWP FIR/IIR instruction

Finite impulse response (FIR) and infinite impulse response (IIR) filters are required in signal and image processing applications [20]. Based upon the requirements, low pass, high pass or band pass filters can be designed and implemented. The general equations for FIR and IIR filters are given below:

\[
\text{FIR } y[n] = \sum_{i=0}^{P} b_i x[n-i] \\
\text{IIR } y[n] = \frac{1}{Q} \left( \sum_{i=0}^{P} b_i x[n-i] - \sum_{j=1}^{Q} a_j y[n-j] \right)
\]

where \(x[n]\) is the input signal, \(y[n]\) is the output signal, \(P\) is the feed forward filter order, \(Q\) is the feed back filter order, \(b_i\) is the feed forward filter coefficients and \(a_i\) is the feed back filter coefficients. As shown in the equations, the most commonly used operations in FIR and IIR filters are Multiply-and-Accumulate (MAC) operations. Efficient implementation of MAC operation highly increases the performance of processor [4]. In the conventional RISC architecture, the MAC operation is performed on single data value or pixel in each clock cycle. As a result, the word-size resources are under utilized. In our proposed SWP-RISC, MAC operation can be performed on multiple sub-words/pixels in parallel using the SWP multiplier and adder. The achieved parallelism depends upon the number of pixels or data values which can be packed in word-size registers.

**Instruction format:** The assembly syntax for SWP MAC instruction is given below:

\[\text{SWPMAC } \text{dest}, \text{src1}, \text{src2}, \text{mode}\]

SWPMAC is the opcode. \(\text{src1}\) and \(\text{src2}\) are input registers which contain packed pixels. The resultant MAC value is stored in dest register.

**Instruction architecture:** The hardware architecture of SWP MAC instruction is given in Fig. 8.

SWP MULT unit gives the product of corresponding sub-words in product register. SWP sub-word adder unit adds the sub-words, packed in word-size registers. Finally, the Accumulator unit is used to add the product values recursively.

3.7. SWP DWT instruction

Discrete wavelet transform (DWT) operation is widely used in communication and medical imaging systems, as it provides high data compression ratio in JPEG2000 standard [21,22]. DWT decomposes the image into frequency domain, keeping the temporal information intact. It greatly reduces the memory requirements and speed-up the communication by breaking an image into small blocks. 2D-DWT implementation equations are given below:

\[
\text{y}_{\text{LL}}[n] = ((x[n] \ast g[n]) \downarrow 2) \ast g[n] \downarrow 2 \\
\text{y}_{\text{LH}}[n] = ((x[n] \ast g[n]) \downarrow 2) \ast h[n] \downarrow 2 \\
\text{y}_{\text{HL}}[n] = ((x[n] \ast h[n]) \downarrow 2) \ast g[n] \downarrow 2 \\
\text{y}_{\text{HH}}[n] = ((x[n] \ast h[n]) \downarrow 2) \ast h[n] \downarrow 2
\]

Where \(x[n]\) is the input signal, \(y[n]\) is the output 2-D DWT signal, \(g[n]\) is the impulse response of the low-pass filter and \(h[n]\) is the impulse response of the high-pass filter. Filtering is the main operation in DWT which ultimately requires extensive MAC operations. In SWP-RISC, efficient DWT operator is implemented by using the SWP MAC operators.

**Instruction format:** The assembly syntax of 32-bit SWP 2D-DWT instruction is given below:

\[\text{SWPDWT } \text{DWT-img-addr, input-img-addr, mode}\]

SWPDWT is the opcode. input-image-addr indicates the address of an input \((8 \times 8)\) image block in the memory. DWT-image-addr indicates the memory address where the output DWT coefficients matrix will be stored. Finally, the mode represents the sub-word/pixel size.

**Instruction architecture:** The hardware architecture of 2D-DWT operation is shown in Fig. 9.
Horizontal data is filtered and down sampled by 2, extracting the approximation and detailed coefficients. The extracted coefficients are vertically filtered and down sampled by 2, to get four coefficients $y_{LL}(\text{low,low})$, $y_{HL}(\text{low,high})$, $y_{LH}(\text{high,low})$ and $y_{HH}(\text{high,high})$. The $y_{LL}$ is a coarser version of the original input data. On the other hand, $y_{LH}$, $y_{HL}$, and $y_{HH}$ are the high frequency coefficients containing the detailed information.

4. SWP-RISC hardware implementation

The previous section of this article has proposed the multimedia-oriented SWP instructions. This section describes the implementation of proposed instructions. For this purpose, the conventional RISC core implemented in LISA, is used from the IP library [17]. The SWP instructions for multimedia applications (explained in Section 3) are designed in LISA and incorporated in RISC IP. Consequently, the new SWP-RISC IP contains both conventional as well as multimedia-oriented SWP instructions.

Based upon the selected mode, SWP instructions can perform parallel computations on 8, 10, 12, 16-bit pixels, packed in word-size registers. Moreover, the SWP instructions can also perform computations on word-size data as well. Therefore, the SWP instructions can also replace the conventional instructions in RISC architecture. For the efficient utilization of resources, basic blocks of conventional RISC like general purpose registers (GPR) array, instruction register, program counter, data cache, instruction cache etc. are used in SWP-RISC architecture without any increase. However, the hardware related to SWP instructions is added in the RISC architecture. Synopsys Processor Designer tool [14] is used to generate the RTL code and software tools (compiler, debugger, assembler and dis-assembler) of SWP-RISC IP core.

4.1. Synthesis results of SWP instructions

The hardware corresponding to different SWP instructions is synthesized on ASIC platforms using 90nm and 45nm technology nodes. For this purpose, the standard cell library from CSMC [23] is used. Synthesis is done using Encounter digital design tool from Cadence [24]. The area, critical path and power consumption for implementing customize SWP instructions on different CMOS technology nodes are shown in Table 1.

Table 1

<table>
<thead>
<tr>
<th>Instruction</th>
<th>90 nm CMOS</th>
<th>45 nm CMOS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Area (mm²)</td>
<td>CP (ns)</td>
</tr>
<tr>
<td>ADD/SUB</td>
<td>0.01</td>
<td>0.29</td>
</tr>
<tr>
<td>MULT</td>
<td>0.23</td>
<td>0.97</td>
</tr>
<tr>
<td>MAC</td>
<td>0.24</td>
<td>1.08</td>
</tr>
<tr>
<td>ME</td>
<td>0.09</td>
<td>0.36</td>
</tr>
<tr>
<td>DCT</td>
<td>0.12</td>
<td>0.37</td>
</tr>
<tr>
<td>DWT</td>
<td>0.85</td>
<td>1.03</td>
</tr>
<tr>
<td></td>
<td>1.01</td>
<td>1.16</td>
</tr>
</tbody>
</table>
Minimum resources in terms of area, CP and power are required by SWP ADD/SUB instruction due to the simplicity of operation. On different technology nodes, SWP MULT and SWP MAC instructions almost require the same resources, as the design of both instructions is based upon the SWP multiplier. In SWP SAD implementation, SWP-enabled absolute difference and accumulator consume almost 75% and 25% resources respectively. The additional resources in SWP ME are consumed by the controller, RAM reading units and the comparator.

It can be observed from Table 1 that SWP DCT and SWP DWT instructions require more resources as compared to other SWP instructions due to their computational complexity. In SWP DCT instruction, almost 70% resources are consumed in implementing one dimensional DCT operator. Remaining units like vector arrangement, transpose unit and controller part consume almost 30% of the resources. Finally, in SWP DWT instruction, the maximum resources are consumed by filtering units which mainly consist of MAC units.

4.2. SWP-RISC implementation results

In order to analyze the resources, the conventional RISC core and our proposed SWP-RISC core are both implemented on ASIC platforms using 90nm and 45nm technology nodes. The instruction-set of conventional RISC IP consists of 32 instructions. These instructions include load/store, arithmetic and control instructions. On the other hand, SWP-RISC IP core contains both conventional as well as customized SWP instructions, explained in Section 3.

SWP-RISC core contains 100 I/O pads. Out of which, 36 are input pads and 64 are output pads. Electro static discharge (ESD) protection circuit is implemented in each pad to increase the reliability of chip. The major blocks inside the chip are register bank array, data memory, instruction memory, arithmetic and logic unit (ALU) etc. Technology libraries from CSMC [22], which includes standard cells and I/O cells, are used in the implementation of RISC processors. Extensive layout simulations are performed to validate the functionality of the chip. Layout Vs schematic (LVS) and design rule check (DRC) are performed before generating the GDS file for fabrication. The area, CP, power consumption and chip area comparisons of both conventional and SWP-RISC IP core on different technology nodes are given in Table 2.

As shown in Table 2, with the improvement in CMOS technology node (from 90nm to 45nm), the chip area reduces accordingly. In SWP-RISC, 67% gate area is due to the conventional RISC hardware and 33% gate area corresponds to the hardware related to SWP instructions for multimedia applications. The CPs of both conventional and SWP-RISC IPs on both technologies are almost same because the customized SWP instructions are pipelined in such a way that the CP corresponding to SWP instructions should not increase the overall CP of the chip. Finally, the dynamic power of SWP-RISC IP is more than conventional RISC due to increased gate count and activity.

5. Performance comparison

For multimedia applications, the performance of SWP-RISC core is compared with the state-of-the-art DSP core TMS320C64X from Texas Instruments [15]. The multimedia instructions in TMS320C64X are based upon the conventional sub-word-sizes (8, 16, 32-bit). On the other hand, multimedia instructions in SWP-RISC are based upon the multimedia-oriented pixel sizes (8, 10, 12 and 16-bit). Table 3 shows the performance comparison of SWP-RISC with DSP TMS320C64X for different multimedia applications like JPEG [24], JPEG2000 [26], MPEG [27], H.264 [28] and data encryption standard (DES) [29].

In Table 3, the clock cycles required to perform multimedia computations for a frame size of (640 × 480) are shown for conventional RISC core, TMS320C64X and our SWP-RISC core. The speed-up over the conventional RISC due to multimedia enhancement in both TMS320C64X and SWP-RISC core are shown when working on different pixel size data (8, 10, 12-bit). These pixel sizes are selected based upon the data in modern multimedia applications.

While performing word length computations, the performance of both SWP-RISC (single core) and TMS320C64X are almost the same as both of them use the conventional instruction-set. However, for multimedia applications, compiler of SWP-RISC core uses proposed customized instruction-set where as TMS320C64X uses SIMD based multimedia instruction-set. As the conventional RISC operates on the word-size data, the variation in pixel sizes do not effect the performance. However, in TMS320C64X and SWP-RISC, the number of sub-words increase the parallelism in computations. As a result, less number of clock cycles are required to perform multimedia tasks, which ultimately increases the speed-up.

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Conventional Vs SWP-RISC IP core.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMOS Tech.</td>
<td>CP (ns)</td>
</tr>
<tr>
<td>Conventional RISC</td>
<td>90 nm</td>
</tr>
<tr>
<td></td>
<td>45 nm</td>
</tr>
<tr>
<td>SWP Multimedia RISC</td>
<td>90 nm</td>
</tr>
<tr>
<td></td>
<td>45 nm</td>
</tr>
</tbody>
</table>
On JPEG standard, compared to conventional RISC, single core SWP-RISC gives the actual speed-up of 3.72, 2.59 and 1.96 when the pixel sizes are 8, 10 and 12-bit respectively. Where as TMS320C64X gives the speed-up of 3.63, 1.86 and 1.86. For 8-bit pixel size, both TMS320C64X and SWP-RISC use sub-word-size of 8-bit. Therefore, the speed-up is almost the same (3.72 for SWP-RISC vs 3.63 for TMS320C64X). The small increase in SWP-RISC speed-up is because the operators are designed specifically for multimedia applications. For 10-bit pixel data, the speed-up of single core SWP-RISC is almost 40% more as compared to TMS320C64X. The reason being that 10-bit sub-word-size is not available in TMS320C64X and it uses 16-bit sub-word-size to perform computations on 12-bit pixel data. As a result, under utilization of resources occur which ultimately reduces the speed-up. Similarly, for 12-bit pixel size, the speed-up of SWP-RISC is more due to the availability of corresponding sub-word-size (1.96 for SWP-RISC vs 1.86 for TMS320C64X). Ideal speed-up of SWP-RISC core are 4, 3 and 2 for 8, 10 and 12-bit sub-word-sizes respectively. The actual speed-up of SWP-RISC (e.g. 3.72 for 8-bit, 2.59 for 10-bit and 1.96 for 12-bit sub-words in JPEG) is little bit less as compared to the ideal speed-up because of SWP overheads like packing/unpacking and arrangement of pixels/sub-words in word-size registers.

For computationally intensive multimedia applications like MPEG, H.264 etc., the SWP overheads reduces further due to the high influx of input data and more arithmetic operations to perform. As a result, SWP operators are utilized to maximum extent and void sub-word slots (due to the non availability of data) are reduced. Therefore, the speed-up increases further for both TMS320C64X and SWP-RISC. However, the speed-up enhancement is more in SWP-RISC as compared to TMS320C64X because of better co-ordination between pixel data and the sub-word-sizes. For 8-bit pixel size, the speed-up obtained in JPEG, JPEG2000, MPEG, H.264 and DES are 3.72, 3.81, 3.84, 3.88 and 3.70 respectively when using SWP-RISC as a single processing core. Results show that even in the presence of small overheads, SWP-RISC gives substantial reduction in clock cycles compared to conventional RISC and TMS320C64X when working on different multimedia applications.

5.1. Performance on multi-core platform

The performance of proposed SWP-RISC core is evaluated in multi-core cluster environment for different multimedia applications. For this purpose, four SWP-RISC IP cores are used as processing elements (PE) and shared address space (SAS) is used to communicate with peripherals. Parallel programming is done using openMP. Block diagram of multi-core SWP-RISC architecture is shown in Fig. 10. As shown in Table 3, for computationally intensive application H.264, compared to single core conventional RISC, quad core SWP-RISC processor gives an overall speed-up of 14.35, 10.25, and 7.38 for pixel sizes of 8-bit, 10-bit and 16-bit respectively. High speed-up is achieved due to the exploitation of parallelism both at core level (by using 4 cores) and at operator level (by using SWP instructions). For same sub-word-sizes, the small variation in speed-up for different applications is due to the availability of more inherent parallelism in some applications.

5.2. Performance in power smartness mode

The performance of SWP-RISC is evaluated while operating in power smartness mode. In power smartness mode, the input voltage (VDD) level of standard cells are reduced from 3.3 V to 2.2 V and clock gating is used to reduce the transitions in less active area of the chip. SWP-RISC core consumes almost 40% less power as compared to the normal mode in power smartness mode for high end applications like MPEG and H.264. The reduced power consumption is achieved at the cost of
low input voltage and clock gating. The power consumption can further be reduced by operating the chip at lower frequency as per requirements. In power smartness mode, the SWP-RISC core can be used in low power applications like processing nodes in wireless sensor network (WSN) and Internet Of Things (IOT).

6. Conclusions

The implementation of SWP-RISC processor on different CMOS technology nodes has been presented in this paper. The proposed processor contains conventional as well as customized multimedia instructions. For customized multimedia instructions, sub-word parallelism (SWP) with sub-word-sizes of 8, 10, 12 and 16 bits are considered. As a result, SWP-RISC provides better performance for both conventional and multimedia-oriented applications. The customized instructions are designed in an architectural descriptive language. The corresponding hardware and software tools are generated using Synopsys Processor Designer tool. The performance of SWP-RISC is compared with the DSP core TMS320C64X. Due to the customized instruction-set, SWP-RISC performs computations on multimedia applications very efficiently and gives higher speed-up. In power smartness mode, the proposed SWP-RISC core can be used for low power applications like wireless sensor networks (WSN) and Internet Of Things (IOT).

Acknowledgments

This project is funded by NSTIP (National Science Technology, Innovative Plan), Saudi Arabia. We acknowledge the support of KACST (King Abdul-Aziz City for Science and Technology) and STU (Science and Technology Unit) Makkah.

References


[10] Chen SHF, Hsiao HCW. Fast detection of microaneurysms in color fundus images. IEEE international conference on multimedia big data (BigMM), Taiwan; 2016.


**Shafqat Khan** received his Ph.D. degree in System on Chip (SoC) from the University of Rennes, France in 2010. Currently, he is working as a senior design engineer in fable Integrated Circuit (IC) design center. His research interests include multimedia processing, arithmetic operator designing, speed enhancement in operator design, embedded processors, and re-configurable architectures.

**Muhammad Rashid** received his Ph.D. degree in embedded systems design from the University of Bretagne Occidentale, Brest, France, in 2009. Currently, he is working at the Computer Engineering Department of Umm Al-Qura University, Makkah, Saudi Arabia. Prior to joining Umm Al-Qura University, he worked with Thomson Research and Development, Paris, France. His research is mainly focused on electronic design automation for embedded systems.

**Faraz Javaid** received his MS degree in Electrical Engineering from Center for Advanced Studies in Engineering, Islamabad, Pakistan in 2017. Currently, he is working as a design engineer in Integrated Circuit (IC) design center. His research interests include multimedia processing, embedded processors, digital systems design, digital and mixed signal IC designing.