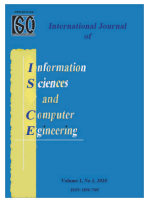
Information Sciences and Computer Engineering, Vol. 1, No. 1, (2010) 45–52

PHARUS

International Journal of
Information Sciences and Computer Engineering

journal homepage: http://www.ijisce.org



High Speed Reconfigurable SWP Operator for Multimedia Processing Using Redundant Data Representation

Shafqat Khan*, Emmanuel Casseau, Daniel Menard

INRIA/IRISA, ENSSAT - University of Rennes 1, 6 Rue de Kerampont - BP 80518 - 22305 Lannion - France

***Abstract*– For better performance and efficiency, high-speed reconfigurable computation units are required in processor design. However the reconfiguration overheads like interconnection cost and reconfiguration time reduce the benefits of reconfigurable processors. At the same time within the arithmetic operators, the speed of operations on binary data cannot be increased beyond certain limits because of the inherited carry propagation at any stage of the addition. In this paper to address both reconfiguration and computation time issues, a high-speed reconfigurable operator is proposed for multimedia applications. This operator provides reconfigurability at both the operation level (different multimedia oriented operations) and at the data size level (different pixel data sizes) through the use of multimedia oriented subword parallelism (SWP). The speed of the different arithmetic operations is improved through the use of a carry propagation free addition on a redundant (barrow save) data representation. For multimedia applications, this operator ensures reconfigurability with high resource utilization along with high-speed operations.**

Keyword: Multimedia processing, Data level parallelism, Carry propagation free addition, Redundant number system, Borrow save number system, Reconfigurable system, Embedded system.

1. Introduction

Efficiency of processors can be increased through the use of reconfigurability as well as high-speed arithmetic units. In [1, 2, 3, 4] different high-speed reconfiguration methods are discussed for dynamically reconfigurable processors. All these methods focus on reducing the reconfiguration time for performance enhancement. In conventional reconfiguration methods [5, 6, 7, 8], the processor reconfigures itself for a new set of applications without concentrating on the internal reconfigurability of processing units. This would result in a high reconfiguration cost as well as the under utilization of processor resources especially for multimedia applications with low-precision pixel data (8, 10, 12 or sometimes 16 bits). Besides the reconfiguration issues, the overall computation speed of any arithmetic unit is highly dependent upon the speed of the addition algorithm used. When processing on conventional binary data, the speed of arithmetic units cannot be increased beyond certain limits due to the propagation of carry at any stage irrespective of the type of the binary adder used.

Keeping in view the low-precision pixel data and high-performance requirements in multimedia domain, a coarse grain pipelined reconfigurable operator is proposed for multimedia applications in this paper. This operator provides reconfigurability at both the operation level and the data size level through the use of SWP [9, 10]. Reconfiguration at this level do not increase the complexity of the interconnection network as well as no reconfiguration time is required. For a better resource utilization, multimedia oriented subword sizes (8, 10, 12 or 16 bits) are considered rather than existing conventional subword sizes (8, 16 and 32 bits etc.)[11, 12, 13, 14, 15]. To increase the speed of different processing units, arithmetic operations are performed using a redundant or borrow save representation rather than the conventional binary representation. Latter in this paper, we will use BS for borrow save or redundant representation and CB for conventional binary representation. The borrow save (BS) representation allows to use a carry propagation free addition [16, 17] in the arithmetic units. Compared to conventional binary (CB) adders, carry propagation free adders increase the overall speed of the reconfigurable operator when performing different multimedia operations like sum of absolute value difference SAD, discrete cosine transform DCT etc. Moreover the SWP overheads when performing on BS data are less compared to CB data. The proposed multimedia operator can be used as a dedicated core (co-processor) to speedup multimedia processing [18]. For multimedia applications, the main processor transfers control to co-processor which will perform the computations on pixel data more efficiently compared to conventional computational operators.

In our earlier publication [19], we have discussed about the design of an embedded processor for multimedia applications. The architecture of the operator used in the processor design was the preliminary design effort which uses multimedia oriented subword sizes to increase the overall performance of the processor. In [20], architectures of different basic operators (ADD, SUB, MULT and MAC) using SWP are described. In [21], the design of a SWP based multimedia operator is described. In these

*Corresponding author:
Email address: Shafqat.Khan@irisa.fr, Ph: +33 296469179

previous works, the performance of the operator was improved through the use of SWP on multimedia oriented pixel sizes without focusing on the internal speed of the different processing units (ADD, SUB, MULT etc.). These operators use SWP on CB data which has the inherited carry propagation feature at any stage of the arithmetic operation resulting in an overall limited speed for the different multimedia operations. In this paper, we emphasize on the designing of a multimedia operator which not only uses multimedia oriented subword sizes to increase the resource utilization but also increases the speed of the different processing units through the use of BS number system. The carry propagation free addition property of BS number systems increases the speed of the different basic operations which ultimately increases the speed of multimedia operations like SAD, DCT etc. To our best knowledge, it is the first time redundant number system is applied to SWP design.

The paper is organized as follows. Section 2 gives a brief overview of subword parallelism SWP. The next three sections give details of basic arithmetic units that are used in proposed multimedia operator design. In Section 3, a multimedia oriented SWP adder using a BS representation is presented. In Section 4, a SWP BS multiplier unit is described. In Section 5, the architecture of a borrow save SWP SAD unit and its advantages over a conventional binary SWP SAD unit are also explained. The pipelined architecture of a coarse grain reconfigurable SWP operator using a BS representation is presented in Section 6. The synthesis results of the proposed reconfigurable operator for multimedia processing and its performance compared to the state of the art DSP chips are also presented. Finally conclusions are presented in Section 7.

2. Subword parallelism SWP

To increase the performance of processors, SWP techniques have been carried out on the basic arithmetic operators (ADD, SUB, MULT etc.). These basic SWP operators perform parallel operations on subwords 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 datapath and register sizes when operating on low-precision data [22, 23]. Conventionally the word size of processor is a multiple of subword sizes which helps to reduce the complexity of SWP operators. For instance, some of the conventional subword sizes for 64-bit processors are 8, 16 and 32 bits. Figure 1 shows four basic addition operations on 16-bit subword in a 64-bit processor.

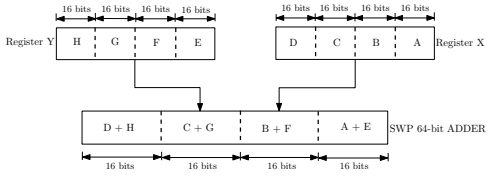


Fig. 1. Adder with 16-bit subwords

In multimedia applications, the sizes of the input data (pixels) for computations are 8, 10, 12 or sometimes 16 bits. These multimedia data sizes are not in coordination with existing processor's subword sizes resulting in the under utilization of processor's resources. To increase the performance of multimedia applications, our proposed reconfigurable operator considers multimedia oriented subword sizes rather than conventional subword sizes. This operator can perform operations on word size operands (40-bit) as well as on subwords (five 8-bit subwords or four 10-bit subwords or three 12-bit subwords or two 16-bit subwords) packed in word size registers. Word size of 40 bits is chosen because it gives good efficiency/complexity trade-off and ensures better resource utilization with different multimedia oriented pixel sizes. With these multimedia oriented word and subword sizes, the minimum resource wastage (0%) occurs when the selected subword size is either 8 bits or 10 bits and the worst case resource wastage (20%) occurs when selected subword size is 16 bits.

3. SWP adder using redundant numbers

The most commonly used operations in multimedia applications are addition, subtraction, absolute value, multiplication, sum of absolute value differences SAD $\sum |a - b|$ for motion estimation, sum of products $\sum |a \times b|$ for discrete cosine transform DCT etc. Almost all these operations require addition/subtraction at some stage of their internal computation. Therefore any efficient adder scheme can increase the overall performance of all the arithmetic units in the multimedia operator. Different adder architectures on conventional binary data have been proposed to reduce the carry propagation time and increase the efficiency. However in all these schemes some delay due to the carry logic remains there which increases with the word size.

To overcome the delay due to the carry logic, a carry propagation free addition algorithm is used in our proposed multimedia operator. Carry propagation free adders perform the addition on data represented in the borrow save format rather than the CB format. In the BS representation [17, 24], numbers are represented using radix-2 digits from the digit set $\{-1, 0, 1\}$. Digit -1 is denoted by $\bar{1}$ in this paper. The value of a number X is $\sum_{i=0}^{n-1} x_i 2^i$ where $x_i \in \{-1, 0, 1\}$. Using the BS representation, some numbers have several representations, this is a *redundant* number system. For instance the number 9, denoted by $\{1001\}_{-CB}$ in CB, has several BS representations: $\{101\bar{1}\}_{-BS}$ and $\{11\bar{1}\bar{1}\}_{-BS}$ and $\{1001\}_{-BS}$. BS representation allows constant time addition/subtraction. In order to increase the parallelism as well as the speed of our reconfigurable operator, multimedia oriented SWP capability is implemented on the BS representation. For this purpose, the three main steps of the computing process are detailed in next subsections.

3.1. SWP CB to BS converter

In the BS representation, each digit is represented by one of the digit from the digit set $\{\bar{1}, 0, 1\}_{-BS}$. For unsigned binary numbers no computation is required to convert them to the equivalent BS representation as both have the same value (CB 0 and 1 bits are converted to 0 and 1 BS digits respectively). However for 2's complement numbers the MSB of each subword has negative weight therefore it is replaced by digit $\bar{1}$ (when MSB = 1) or it keeps same value 0 (when MSB = 0).

3.2. SWP BS adder

The *SWP BS adder* is used to perform the SWP addition of subword data in the BS representation. Its architecture is based

upon the breaking of adder chain at subword boundaries. Based upon the selected subword size, the adder chain at subword boundaries are either break or continued as shown in Figure 2.

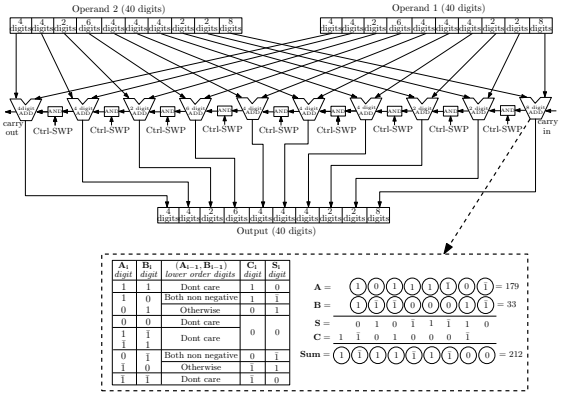


Fig. 2. SWP BS ADD architecture

In the first step intermediate sum and carry digits are generated for each digit and are arranged as $\{S_i, 2C_i\}$(carry left shifted by one rank). These S_i and C_i digits are generated based upon the value of the input digits at ith and (i-1)th ranks. In the second step, the intermediate sum and carry digits at each rank are added to generate the final sum. The intermediate sum and carry signals are generated in such a way that no carry is generated while adding S_i and C_{i-1} digits at any rank. In this way the propagation of the carry is avoided and a parallel addition can be performed in a constant time irrespective of the word length of the vectors to be added. To avoid any overflow, one extra digit must be allocated to each resultant subword which would result in a output of 45 digits for subword size of 8 bits. Subtraction is the same as addition except that in subtraction two's complement of a operand which needs to be subtracted is taken before addition.

Without considering the BS conversions, the critical path (CP) of the SWP adder on the BS representation is less compared to the CP of a SWP adder on CB data of same word and subword sizes (multimedia oriented). Compared to a CB SWP adder, the CP of the BS SWP adder is almost 51% less for 8-bit subwords up to 85% less for 16-bit subwords (see Section 6.2 for details about the hardware technology used). Although the BS conversions also consume some critical path but its value is very small compared to the CP saved while performing several arithmetic operations. The area and dynamic power overhead for this high speed are only 29% and 16% respectively.

3.3. SWP BS to CB converter

After performing BS computations, the output is converted back to CB representation. Depending on the output of operator, either simple (for single value) or SWP (for packed subwords) *BS to CB converter* unit is required at the output of arithmetic operator. This unit converts BS numbers into CB representation by subtracting the negative digit binary number from the positive digit binary number. This process is shown in Figure 3.

The overhead of this conversion is very small compared to the overall computations performed by the operator. For instance, in the SWP reconfigurable multimedia operator explained in Section 6, the *SWP BS to CB converter* unit only consumes 3%, 6% and 4% of total area, CP and dynamic power respectively.

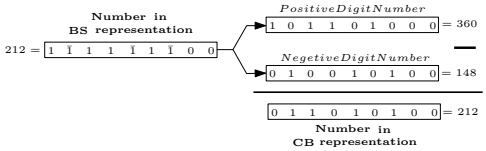


Fig. 3. BS to CB conversion

4. SWP BS multiplier unit

Multiplier is one of the most important basic processing unit in the design of any arithmetic operator. As the subwords are represented in BS representation, therefore *SWP BS (a × b) signed/unsigned* unit is used to perform SWP multiplications of signed as well as unsigned numbers. The implementation of this multiplier is based upon an extension of the SWP binary multiplier proposed in [12]. Compared to other SWP multipliers which are based on algorithms like Booth recoding etc., the SWP multiplier proposed in [12] gives good efficiency as it does not require any detection and suppression of carries at subword boundaries. This SWP multiplier supports only classical subword sizes (8, 16 and 32 bits etc.). However in our implementation of a SWP BS multiplier for reconfigurable processing, the multimedia oriented subword sizes of 8, 10, 12 and 16 bits are considered which do not have any uniform arithmetic relation with the word size (40 bits) of the SWP operator.

As multiplier and multiplicand are in BS representation, therefore partial products (PPs) are also generated in the same representation to get the benefits of the carry propagation free addition while adding them. PPs are generated corresponding to each digit in the multiplier digit vector. The PP digits are all zero when the multiplier digit is '0'. The PP digits are equal to the multiplicand digits when the multiplier digit is '1'. The PP digits are negation of the multiplicand digits when the multiplier digit is '$\bar{1}$'. Due to different arrangement of the PP matrix for each subword size multiplication, the PP digits generated for the multiplication of one subword size data are not valid for other subword size data. For instance the arrangement of PP digits for the subword size of 8 bits are different from the arrangement of PP digits for subword size of 10 bits. In our implementation of *SWP BS (a×b)*, the generated PP digits remain valid for each selected subword size multiplication. For this purpose the PP matrix is divided into two parts. The first part contains the digits of PPs which remain unchanged irrespective of different subword size selections and are generated directly by the multiplier and the multiplicand digits. The second part contains the PP digits whose values change for different selections of subword sizes and are generated indirectly by the multiplier digits and the modified multiplicand digit vector. The modified multiplicand digit vector changes its digit values corresponding to each selected subword size and hence the PP digits update automatically for each selected subword size. Therefore PP digits are generated for different subword sizes using the same PP generation hardware.

The distinction between signed and unsigned subwords are made at the time of the CB to BS conversion therefore no sign extension, zero padding or correction vector is required for signed/unsigned PPs. The PP digit matrix for a subword size of 8 bits is shown in Figure 4. Unfilled circles represent unused (0's) PP digits whereas filled circles represent used PP digits for

a subword size of 8 bits. Each block of filled circles represents one 8x8 multiplication block. PPs for other subword sizes (10, 12, 16 or 40) are arranged in the same way.

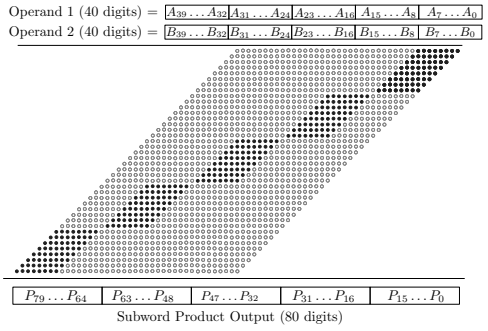


Fig. 4. Arrangement of PPs in SWP BS multiplier for 8-bit subwords

After generation, PPs in BS representation are added using BS adders. These adders add the PPs digits in a smaller time compared to CB adders. Compared to the SWP CB multiplier of same word and subword sizes, the SWP BS multiplier computes the product in 40% less time (for 8-bit subwords) up to 51% less time (for 16-bit subwords). Compared to a simple 40-bit BS multiplier without SWP capability, the area and critical path overhead for incorporating multimedia oriented SWP capability in the multiplier architecture is approximately 5% and 9% respectively.

5. SWP SAD using BS representation

Sum of absolute value differences (SAD) is one of the most commonly used operation in video applications for motion estimation etc. SAD operation is used in this section to explain that how computations are handled in our multimedia operator using the basic arithmetic units explained so for. SAD operation is given by Equation 1.

$$SAD = \sum_{i=0}^{N-1} |a_i - b_i| \tag{1}$$

As the SAD operation is normally applied to low precision pixel data in multimedia applications so subword parallelism SWP can enhance the performance of a SAD unit. The main functions in the calculation of the SAD are finding of the absolute values of the difference and their accumulation. A pipelined architecture of the SWP SAD operator using BS representation is shown in Figure 5.

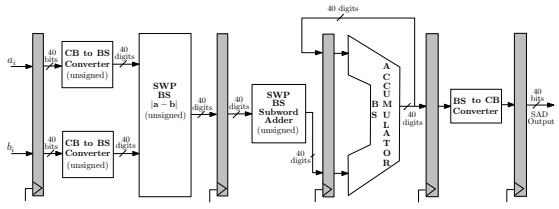


Fig. 5. SWP BS SAD unit

A dedicated control signal, not shown in Figure 5 is used to select the subword size for the SWP units. After converting numbers from CB to BS representation, a *SWP BS* $|a - b|$ *unsigned* unit is used to compute the absolute value of the differences of the packed subwords in the BS format. To achieve maximum advantages of the carry propagation free addition, this unit implements the absolute value operation using SWP BS adders/subtractors. The *SWP BS* $|a - b|$ *unsigned* unit either calculates $a - b$ (when $a>b$) or $b - a$ (when $b>a$) on BS subword data. Therefore in order to get more advantage (high-speed) of carry propagation free addition/subtraction, *SWP BS* $|a - b|$ *unsigned* operation is implemented using BS subtractors instead of implementing absolute value calculation directly.

Due to subtraction operation overflow is not possible hence 40 digits are sufficient to store the resultant subwords from the *SWP BS* $|a - b|$ *unsigned* unit. The output of this unit is in the form of packed subwords. The *SWP BS subword adder* unit adds the subwords packed in one register to obtained a single value. Based upon the selected subword size, the *SWP BS subword adder* unit separates the N_{sa} subwords x_i packed in its input register and then performs the addition of these subwords using a BS adder. The expression of the *SWP BS subword adder* output z_{sa} is given by Equation 2.

$$z_{sa} = \sum_{i=0}^{N_{sa}-1} x_i \tag{2}$$

To obtain the SAD value in the BS format, the output of the *SWP subword BS adder* unit is accumulated recursively using the *BS accumulator*. All these units ($|a - b|$, subword adder and accumulator) involve additions/subtraction of subwords which are performed on BS digits rather than CB bits which increases the speed of the overall SWP SAD operation. Finally the BS SAD output is converted to its CB representation using *BS to CB Converter* unit. Due to the use of fast BS adders, the BS pipelined SWP SAD unit can operate at 50% faster frequency (for 8-bit subwords) upto 70% faster frequency (for 16-bit subwords) compared to a CB SWP SAD unit. The area and power overheads of this speed enhancement is only 31% and 30% respectively.

6. Reconfigurable multimedia operator

6.1. Architecture of the operator

Our reconfigurable multimedia operator can perform different basic and complex multimedia operations on data of different sizes (8, 10, 12, 16, 40 bits). Maximum parallelism with high resource utilization is attained through the use of SWP with these multimedia oriented subword sizes. Along with parallelism high-speed multimedia operations are performed through the use of BS representation. Efficient results are obtained for four stage pipelined architecture of this operator and is shown in Figure 6. This operator is made up of the basic units which we have presented above.

Control bits used to select the subword size are communicated to all the units which contain SWP capabilities. To clarify the schematic, these control bits are not shown in Figure 6. Through the use of this control word, this multimedia operator can be configured for both the computation it executes and the size of the data. In the beginning all the input data vectors are converted from CB to BS representation so that all the arithmetic computations can be performed in BS. The reconfigurable SWP operator

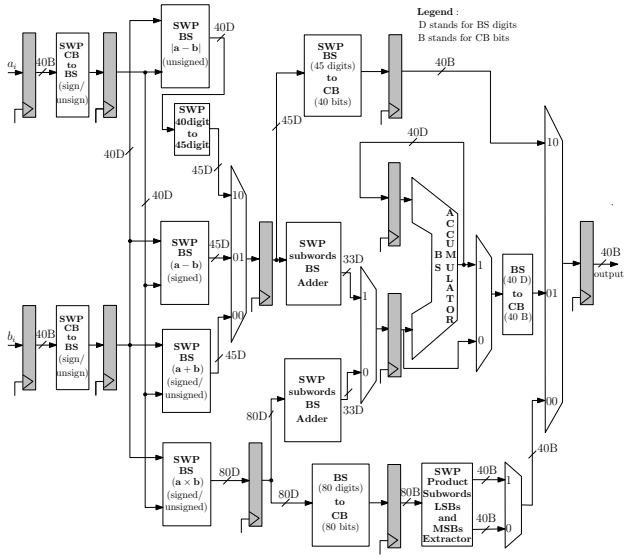


Fig. 6. Reconfigurable multimedia operator

can perform operations on signed as well as unsigned data values and gives the results in the required format. Based on the selected operation, the output of the reconfigurable operator can be in the form of a subwords or single accumulated value. After performing computations on BS values, the output is converted back to CB representation. As different operations produce different data length outputs, therefore different data length simple and SWP *BS to CB converter* are used in the operator design. Like the inputs, the output data of all the basic SWP units can be represented by subwords packed in 40-bit registers except for *SWP BS (a × b) unit* whose output consists of subwords packed in a 80-digit register. These 80 digits can be further converted to 80 bits using *BS to CB converter*. As the output data length is limited to 40 bits, therefore the 80-bit product is divided into 40-bit MSB and LSB parts using *SWP product subwords LSB and MSB extractor* unit. Based upon the selected subword size this unit extracts MSBs and LSBs of 80-bit product subwords. For instance for subword size of 8 bits, LSB and MSB extractor unit extracts $(71\ldots64, 55\ldots48, 39\ldots32, 23\ldots16, 7\ldots0)$ and $(79\ldots72, 63\ldots56, 47\ldots40, 31\ldots24, 15\ldots8)$ 40-bit parts of product respectively. Hence the complete product is obtained at the output of the reconfigurable operator in the form of subwords LSBs and MSBs in two successive clock cycles.

In addition to basic arithmetic operations (signed/unsigned), the reconfigurable operator can perform multimedia operations like SAD for motion estimation, dot product for DCT, $\sum(a+b)$ signed/unsigned, $\sum(a-b)$ signed/unsigned etc. Based upon the requirements, any combination of these operations can also be obtained such as $\sum(a\times b)+\sum(a+b)$ etc. Rather than subwords which sometimes provide loss of bit, these operations produce lossless single accumulated value at the output of the reconfigurable operator. If the accumulated value from the *SWP BS accumulator* is small, it is zero padded to 40 digits. As an example, consider the computation of the multiplication-accumulation used for dot product given by Equation 3.

$$dotproduct = \sum_{i=0}^{N-1}(a_i \times b_i) \tag{3}$$

The inputs are two 40-bit values and the selected subword size which is assumed to be 8 bits for this case. Hence each input vector contains five 8-bit packed subwords. First of all the input CB values are converted to their corresponding BS representation using *SWP CB to BS* units. Then, the *SWP BS (a×b)* unit produces a 80-digit product value in the form of five 16-digit product subwords. These packed product subwords are added together using the *SWP subword BS adder* unit and generate a 33-digit value. This 33-digit data length is selected based upon the worst case of 16-bit subword size for $\sum(a\times b)$ operation with no digit loss. For other subword sizes and operations, the data length requirements at the output of the *SWP subword BS adder* units are less. At each clock cycle the *BS accumulator* accumulates the 33-digit value with the previous values to generate a 40-digit $\sum(a\times b)$ term at the output. The input to the *BS accumulator* is a 33-digit value and the output of the *BS accumulator* is a 40-digit accumulated value resulting in extra seven digits. These extra seven digits are used as guard digits to avoid overflow. For other operations and smaller subword data sizes, the numbers of guard digits are greater and thus the number of accumulations which can be performed increases further.

6.2. Synthesis results

For the analysis of the area, speed and power consumption, the overall SWP reconfigurable BS operator has been synthesized to ASIC standard cell 130nm (CORE9GPLL_HCMOS9_TEC_4.1, 1.2V, 25 deg. C, low leakage standard cell library from ST Microelectronics) and 90nm (fsd0t_a standard performance low voltage threshold cell library from UMC) technologies using Synopsys *Design Compiler*. The area, speed and power consumption have been obtained for both target technologies.

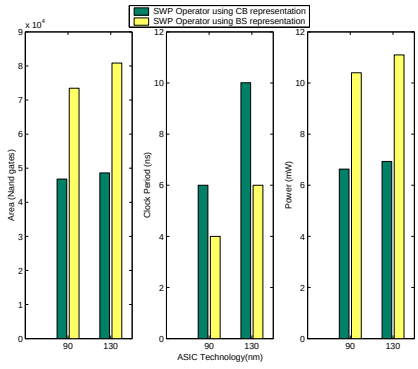


Fig. 7. Comparison of SWP operators using CB and BS representations

In order to get the best possible clock frequency for the reconfigurable operator, synthesis were performed for different clock periods on each ASIC technology and the results are considered for those clock frequencies which give a high efficiency (smallest product of gates, clock period and consumed dynamic power). On both target technologies, due to the use of the carry propagation free adder, the clock frequencies that give maximum efficiency are higher compared to the same SWP multimedia operator using the CB representation (clock periods are 6ns on 90nm ASIC technology and 10ns on 130nm ASIC technology). The

maximum design resources are consumed by the SWP BS multiplier which consumes almost 67% of total area and 57% of total power. Due to this reason we were able to increase the flexibility of the reconfigurable operator by adding other arithmetic operators without increasing the area to a larger extent. As a result the operator can perform variety of multimedia operations depending upon the requirements. Figure 7 shows the comparison of the area, clock period and dynamic power of the SWP multimedia operators when using CB and BS representations. On both target technologies, the clock frequency of the SWP operator using BS representation increases due to the use of high-speed BS arithmetic units. The area overhead for this high-speed is mainly due to the redundant arithmetic units, conversion units and additional glue logics used in BS operators. The probabilistic dynamic power overheads corresponds to the increase in the number of gates.

To perform any particular multimedia operation, only the required units are enabled. All the remaining units are disabled to reduce the switching activity. On the 130nm ASIC technology, the percentage of total power consumed by the SWP reconfigurable BS operator (with 8-bit selected subword size and clock period of 6ns) to perform different SWP operations is shown in the Figure 8. During these experiments, statistical power is estimated by monitoring the switching activity on each node while performing different operations on random test vectors.

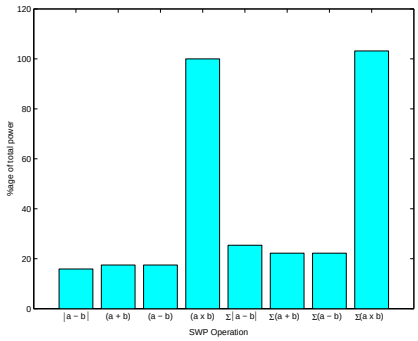


Fig. 8. Power consumption of operations

We assume reference power (100%) is consumed by the more complex basic operator i.e the multiplier ($a \times b$). All the SWP operations consume some percentage of reference power based upon the power consumption of the arithmetic units needed to be enabled for a particular operation. Obviously the power consumed by the complex operations which involve the accumulation is slightly larger compared to the operations which involve basic SWP arithmetic units. Additional power is mainly due to subword adder units. Maximum power consumed by the reconfigurable operator is 104% of the reference power when performing $\sum(a \times b)$ operation.

On average, the area overhead due to the use of BS number system in different units like ADD, SUB, MULT, absolute value of difference etc. is between 30% to 40%. Extra area is required due to the increase in the number of registers required for BS operator. Compared to the CB operator, the BS operator requires almost twice number of registers as each digit is represented by two bits. However the corresponding speed increase for basic SWP BS units varies between 70% to 80%. Due to these high-speed basic arithmetic units, the computation time of multimedia operations is also reduced by the same percentage when operating on different pixel sizes.

6.3. Performance on video applications

Compared to state-of-the-art Texas Instruments (TI) TMS320C64x DSP chip which uses binary arithmetic units and conventional subword sizes (8, 16 and 32-bit), our proposed SWP BS reconfigurable operator gives a reduced number of cycles due to the use of multimedia oriented subword sizes (8, 10, 12 and 16-bit) as well as high speed due to the use of carry propagation free additions on BS data. The processing unit of the TI DSP is made up of two clusters. Each cluster consists of four functional units along with one multiplier and two arithmetic and logic units. For a fair comparison, one reconfigurable operator is considered for our processor and one cluster is considered for the TI DSP. Compared to each cluster of the DSP chip, the percentage reduction in number of cycles (N_{cycles}) of our reconfigurable operator on different multimedia kernels (SAD, DCT and discrete wavelet transform DWT) when applied on a (16x16) image size is shown in Table 1. In addition to computation cycles shown in Table 1, TI DSP also requires loop control cycles when performing different multimedia operations. However these loop control cycles are not considered. With our reconfigurable operator, no extra control cycle is required.

Table 1. Percentage reduction in number of cycles

	Pixel size (bits)			
	8	**10**	**12**	**16**
%age reduction for SAD	0%	50%	25%	0%
%age reduction for DCT	0%	52%	33%	0%
%age reduction for DWT	0%	58%	38%	0%

In practice, processing is spread on two clusters with TI DSP so N_{cycles} is divided by two. In our case, as per the requirements several reconfigurable SWP operators can be used in the processor's design to further increase the efficiency through parallel processing.

7. Conclusion

Efficient reconfiguration along with high-speed arithmetic units improve the performance of processors for several multimedia applications. The benefits of both parallelism and high-speed computation can be combined in the operator design by using multimedia oriented SWP on BS representation. In SWP, supported subword sizes that are in coordination with the pixel size of multimedia applications can further improves the performance through better resource utilization. Our work shows that the speed of almost all the SWP arithmetic units used in the operator design can be improved by using the barrow save representation rather than conventional binary representation.

Acknowledgments

This work is supported by the French *Architectures du Futur* ANR program ANR-06-ARFU-004.

References

[1] R. David, D. Chillet, S. Pillement, and O. Sentieys, "Dart a dynamically reconfigurable architecture dealing with next generation telecommunications constraints," in *Proceedings of the Reconfigurable Architecture Workshop (RAW 02)*, April 2002.

[2] T. Sano, Y. Saito, and H. Amano, "Configuration with self-configured datapath: A high speed configuration method for dynamically reconfigurable processors," in *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA'09)*, pp. 112–118, July 2009.

[3] V. M. Tuan, N. Katsura, H. Matsutani, and H. Amano, "Evaluation of a multicore reconfigurable architecture with variable core sizes," in *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'09)*, pp. 1–8, July 2009.

[4] H. Amano, Y. Hasegawa, S. Tsutsumi, T. Nakamura, T. Nishimura, V. Tanbunheng, A. Parimala, T. Sano, and M. Kato, "Muccra chips: Configurable dynamically-reconfigurable processors," in *Proceedings of the IEEE Asian Solid-State Circuits Conference (ASSCC'07)*, pp. 384–387, 2007.

[5] S. Yeo, T. Roh, and J. Kim, "High energy efficient reconfigurable processor for mobile multimedia," in *Proceedings of the Int. Conf. on Circuits and Systems for Communications (ISSCC'08)*, pp. 618–622, June 2008.

[6] I. Skliarova and A. B. Ferrari, "Design and implementation of reconfigurable processor for problems of combinatorial computations," in *Proceedings of the Euromicro Symposium on Digital Systems Design*, pp. 112–119, June 2001.

[7] H. Amano, "A survey on dynamically reconfigurable processors," in *Institute of Electronics, Information and Communication Engineers (IEICE) Transactions on Communications*, pp. 3179–3187, 2006.

[8] Y. Saito, T. Sano, M. Kato, V. Tunbunheng, Y. Yasuda, and H. Amano, "A real chip evaluation of MuCCRA-3: A low power dycamically reconfigurable processor array," in *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA'09)*, pp. 283–286, 2009.

[9] J. Fridman, "Subword parallelism in digital signal processing," in *Proceedings of the IEEE Signal Processing Magazine*, pp. 27–35, March 2000.

[10] A.Wang and A. Chandrakasan, "A 180-mv subthreshold FFT processor using a minimum energy design methodology," in *Proceedings of the Solid-State Circuits, IEEE Journal*, pp. 310–319, Jan 2005.

[11] A. A. Farooqui, V. Oklobdzija, and F. Chechrazi, "64-bit media adder," in *Proceedings of the IEEE Int. Symp. Circuits and Systems*, (Orlando), May 1999.

[12] S. Krithivasan and M. Schulte, "Multiplier architectures for media processing," in *Proceedings of the Thirty seventh Asilomar Conference on signals, systems and computers*, pp. 2193–2197, 2003.

[13] C. Brunelli, P.Salmela, J.Takala, and J.Nurmi, "A flexible multiplier for media processing," in *Proceedings of the IEEE Workshop on Design and Implementation*, pp. 70–74, 2005.

[14] A. Danysh and D. Tan, "Architecture and implementation of a vector/SIMD multiply-accumulate unit," in *Proceedings of the IEEE Computer Society*, pp. 284–293, 2005.

[15] P. Corsonello1, S. Perri, M. Iachinoi, and G. Cocorullo, "Variable precision arithmetic circuits for FPGA based multimedia processors," in *IEEE Transactions on very large scale integration (VLSI) systems*, September 2004.

[16] A. Avizienis, "Signed-digit number representations for fast parallel arithmetic," *IRE Transactions on Electronic Computers*, vol. 10, pp. 389–400, 1961. Reprinted in E. Swartzlander (ed.), Computer arithmetic, vol.II, IEEE Computer Society Press, 1990.

[17] M. D. Ercegovac and T. Lang, *Digital Arithmetic*. Morgan Kaufmann, 2003.

[18] X. Zhang and X. Shen, "A power-efficient floating-point coprocessor design," in *Proceedings of the International Conference on Computer Science and Software Engineering*, pp. 75–78, 2008.

[19] D. Menard, E. Casseau, S. Khan, O. Sentieys, S. Chevobbe, S. Guyetant, and R. David, "Reconfigurable operator based multimedia embedded processor," in *Proceedings of the International Workshop on Reconfigurable Computing: Architectures, Tools and Applications*, pp. 39–49, 2009.

[20] S. Khan, E. Casseau, and D. Menard, "SWP for multimedia operator design," in *Proceedings of the 2nd Colloque Nationale of GDR SoC-SIP*, (Paris, France), June 2008.

[21] S. Khan, E. Casseau, and D. Menard, "Reconfigurable SWP operator for multimedia processing," in *Proceedings of the International Conference on Application-specific Systems, Architectures and Processors(ASAP'09)*, pp. 199–202, 2009.

[22] M. O. Cheema and O. Hammami, "Customized SIMD unit synthesis for system on programmable chip - a foundation for hw/sw partitioning with vectorization," in *Proceedings of the IEEE Design Automation conference*, pp. 54–60, Jan 2006.

[23] D. Esftathiou, J. Fridman, and Z. Zvonar, "Recent developments in enabling technologies for the software defined radio," in *Proceedings of the IEEE Communication Magazine*, pp. 112–117, August 1999.

[24] A. Guyot, Y. Herreros, and J.-M. Muller, "JANUS, an on-line multiplier/divider for manipulating large numbers," in *Proc. 9th IEEE Symposium on Computer Arithmetic*, pp. 106–111, 1989.

Shafqat Khan received the BSc degree in Electrical Engineering from the University of Engineering and Technology Peshawar, Pakistan in 2000, the M.S degree in Computer Engineering from the National University of Science and Technology Rawalpindi, Pakistan in 2005 and the M.S degree in Embedded System design from the University of Nice-Sophia Antipolis, France in 2007. He worked for several years in the research and development (R&D) organization as research engineer. Currently he is doing his research work in INRIA/ IRISA France. He is a member of the CAIRN (Computing ArchItectures embedding Reconfigurable resources for eNergy-efficient systems-on-chip) research team. His research interests includes multimedia processing, arithmetic operator designing, speed enhancement in operator design, embedded processors and reconfigurable architectures.

Emmanuel Casseau received the M.S. degree in Electrical Engineering in 1990 and the Ph.D degree in Electrical and Computer Engineering from the University of West Britanny, France, in 1994. From 1994 to 1996 he was a research engineer in the French National Telecom School, ENST Bretagne, France, where he developed high-speed Viterbi decoder architectures for turbo-code VLSI implementations. From 1996 to 2006 he was an Associate Professor in the Electronic Department at the University of South Britanny, France, where he led the IP project of the LESTER Laboratory. He is currently a Professor in INRIA/IRISA (French National Institute for Research in Computer Science and Control), University of Rennes1, France. His research interests include system design, high-level synthesis, SoCs design methodologies and reconfigurable architectures for multimedia applications.

Daniel Menard received the Engineering degree and M.Sc. degree in Electronics and Signal Processing Engineering from the University of Nantes Polytechnic School in 1996, the Ph.D. degree in Signal Processing and Telecommunications from the University of Rennes, in 2002. From 1996 to 2000, he was a Research Engineer at the University of Rennes. He is currently an Associate Professor of Electrical Engineering at the University of Rennes (ENSSAT) and a member of the CAIRN (Computing ArchItectures embedding Reconfigurable resources for eNergy-efficient systems-on-chip) research team at the INRIA/IRISA Laboratory. His research interests include implementation of signal processing and mobile communication applications in embedded systems, floating-to-fixed-point conversion, low power architectures and arithmetic operator design.