An Energy-Delay Efficient Subword Permutation Unit

Giorgos Dimitrakopoulos, Christos Mavrokefalidis, Costas Galanopoulos, and Dimitris Nikolos

Technology and Computer Architecture Lab [*]
Computer Engineering and Informatics Dept.
University of Patras, 26500 Patras, Greece

Abstract

Subword permutations are useful in many multimedia and cryptographic applications. Specialized instructions have been added to the instruction set of general-purpose processors to efficiently implement the required data rearrangements. In this paper, the design of a new energy-delay efficient subword permutation unit is examined. The proposed architecture has been derived by mapping the functionality of one of the most powerful permutation instructions (GRP) to a new enhanced linear sorting network. The introduced subword permutation unit is fast and achieves significant area and energy reductions compared to previous implementations. Also its regularity and its reduced wiring enables efficient VLSI implementations. The efficiency of the proposed architecture has been validated using static CMOS implementations in a standard performance 130nm CMOS technology.

1 Introduction

General-purpose computing workload is increasingly taken by multimedia applications [1]. One of the main characteristics of multimedia applications is that they deal with low precision data that exhibit high levels of data parallelism. In most cases, multimedia data are packed into subwords of one or two bytes that are processed in parallel in word-oriented processors of 32 or 64 bits, according to the SIMD paradigm [2], [3]. Several new instructions have been introduced and added to the instruction set of modern microprocessors, in order to efficiently handle subword operations and enhance the performance of software-implemented multimedia algorithms.

Apart from subword-parallel arithmetic operations, the subwords need to be efficiently rearranged inside the registers in order to enhance the computation and fully exploit the data parallelism available in multimedia applications. Efficient bit and subword permutations are also needed for the software implementation of cryptographic algorithms [4]. The selection of new permutation instructions and the design of fast permutation units have recently attracted a lot of interest [4]–[6]. Bit permutations are the most difficult form of subword permutations. The difficulty lies in the large number of distinct possible results, $n!$ permutations of a n-bit value are possible. When subword permutations are considered, the problem is easier since $(n/b)!$ outcomes are possible, where b denotes the subword size. Driven by application requirements most instruction sets limit the use of more complex permutation operations to the subword level, while only simpler shift and rotate operations are supported at the bit level. In this way the hardware implementation is simplified and the permutation operations can be used in high-speed microprocessors with single-cycle latency.

Several instructions and their hardware implementations have been proposed to efficiently perform arbitrary permutations [4], [7]. One of the most powerful instructions is GRP [5]. It has a general functionality and its use is versatile. It achieves significant speedup when used in cryptographic algorithms, while its benefits when used in other applications, such as subword sorting and the generation of DCT coefficients has been analyzed in [7]. GRP R_D, R_S, R_C takes two source operands, the data and the control bits stored in R_S and in R_C, respectively, and generates one result that is stored in the destination register R_D. The instruction divides the bits of register R_S in two groups according to the values of the control bits of R_C. If a control bit is 1, then the corresponding data bit of R_S is put in the first group. Otherwise, the bit of R_S is put in the second group. In the result the relative position of the bits in each group remains unchanged. An example execution of GRP is shown in Fig. 1.

In this paper a new hardware implementation of the GRP instruction is proposed that is better suited for the most practical cases of multi-bit subwords. The novel architecture has been designed and simulated in a standard performance 130nm CMOS technology. The derived designs are fast and achieve significant area and energy reductions com-

[*]We thank the European Social Fund (ESF), Operational Program for Educational and Vocational Training II (EPEAEK II), and particularly the Program PYTHAGORAS, for funding the above work.

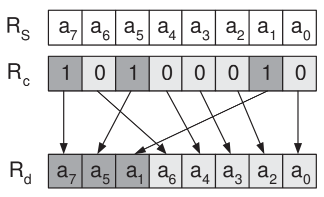

R_S
a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0
R_c
1 0 1 0 0 0 1 0
R_d
a_7 a_5 a_1 a_6 a_4 a_3 a_2 a_0

Figure 1. An example of GRP execution.

pared to the most efficient already presented architecture.

The rest of the paper is organized as follows. Section 2 gives a brief review of the previous GRP implementations. Section 3 gives the main idea behind our work and Section 4 describes in detail the proposed architecture and its circuit implementation. Experimental results are given in Section 5. Finally, conclusions are drawn in Section 6.

2 Prior Work

Three architectures have been proposed for the hardware implementation of the GRP instruction [8], [9], [10]. In all cases, data associated with different control bits were separately extracted from the input operand and aligned at the output. The general form of the two-datapath architecture followed is shown in Fig. 2. The left datapath is responsible for concentrating the data bits with a control bit equal to one to the left side of the result register. In the same manner, the right datapath that assumes complemented control bits concentrates the rest data bits to the right side of the result register. The partial results of the two extraction units are unified with a logical OR operation. In order to allow the OR unification at the output, the input data bits are first masked with the corresponding control bits. After the masking operation the bits with a control bit equal to zero are also set to zero. The two extraction units have almost identical structure, since the concentration direction of the data bits slightly alters their design. The difference of the already presented architectures lies in the design of the two extraction units.

In the first implementation a recursive shifting approach is followed [8]. At first, for the case of the left extraction unit, groups of two bits are examined and those with a control bit equal to one are shifted to the left. Then the two-bit groups are merged to form a correctly aligned group of four bits where the bits with a control bit equal to one have moved to the left side of the group. In order to merge the two parts, the bits of the right group with a control bit equal to one replace the bits of the left group with a zero control bit. In parallel the number of zero control bits of the new group is also counted. This is performed by adding the precomputed number of zeros of the two smaller groups. The derived number, controls the shift-merge procedure of the next level. This procedure of counting and merging continues recursively until a single group of n bits is derived.

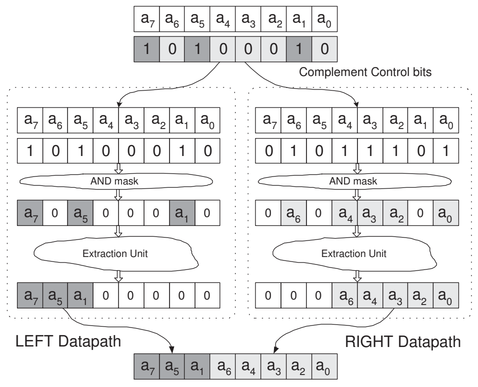

a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0
1 0 1 0 0 0 1 0
Complement Control bits
a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0
1 0 1 0 0 0 1 0
AND mask
a_7 0 a_5 0 0 0 a_1 0
Extraction Unit
a_7 a_5 a_1 0 0 0 0 0
a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0
0 1 0 1 1 1 0 1
AND mask
0 a_6 0 a_4 a_3 a_2 0 a_0
Extraction Unit
0 0 0 a_6 a_4 a_3 a_2 a_0
LEFT Datapath
RIGHT Datapath
a_7 a_5 a_1 a_6 a_4 a_3 a_2 a_0

Figure 2. The two-datapath architecture for the implementation of GRP.

The general functionality of recursively counting the number of positions that a bit requires to be shifted to the left, without altering its relative significance with the rest bits with the same control bit, has also been used in the second implementation [9]. The main characteristic of this design is that the shifting amount for each bit is counted in a carry-save fashion directly from the input control bits. Then the carry-save representation of the shifting amount is encoded in such a way, so as to correctly configure the multiplexers of an inverse butterfly network that performs the alignment of all the bits with a control bit equal to one. The carry-save computation of the shifting amount and the configuration of the routing network is partially overlapped in time leading to faster solutions compared to the first approach [8].

The third approach [10] for the implementation of the GRP instruction does not rely on counting the required shifting amount. The computation of the control information that rearranges the data, is performed in parallel with the movement of the data bits to their correct position. The design of [10] slightly resembles the architecture proposed in this paper since the GRP operation is also viewed as a sorting operation for the control bits. The datapath of [10] that implements GRP, is a variation of bitonic sorters, where the control bits are sorted and the data bits are moved to the left using cascaded butterfly networks without altering their relative significance. This design has been proven to be more than 20% faster than the previous architectures [8], [9] requiring almost the same amount of area. The main differences between the proposed approach and the fastest architecture presented in [10] will be clarified in the next section.

3 Main Idea

According to the definition of the GRP instruction, the data bits, that have a corresponding control bit equal to one, are concentrated to the left side of the output. This action resembles a sorting operation for the control bits, where the

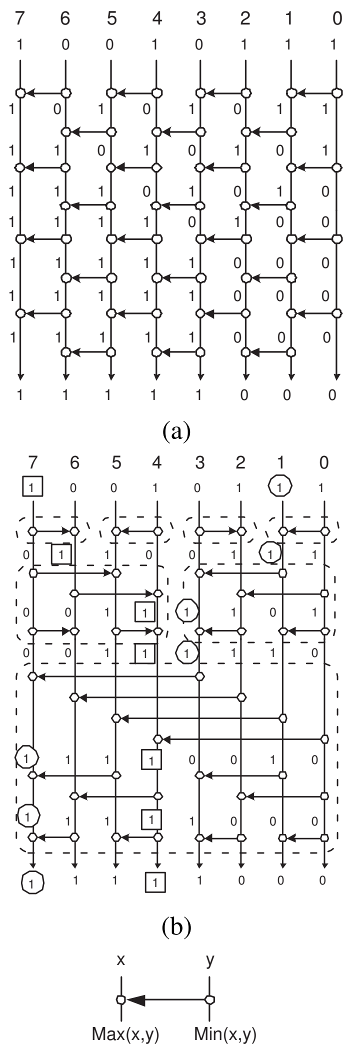


Figure 3. (a) The linear odd-even transposition and (b) the bitonic sorting networks used for the implementation of GRP.

largest bits, i.e., bits equal to one, are gathered to the left. Therefore, the problem of designing a hardware unit that would directly execute the GRP instruction is equivalent to the design of a sorting network that would simultaneously sort the control bits and exchange appropriately the positions of the corresponding data bits. The main problem that arises is that apart from sorting the control bits, we must also ensure that the relative position of the data bits remains unchanged. This is the main constraint that complicates the design of the circuit implementing GRP functionality. We consider two cases of sorting networks [11], the linear odd-even transposition network and the bitonic-sorting network, shown in Fig. 3. Both sorters are composed of the compare-and-exchange elements also illustrated in Fig. 3. The direction of the arrow denotes the final position of the maximum element. When only two bits are compared, their maximum is given by the boolean OR function, while the corresponding minimum is given by the boolean AND function.

In the first case (Fig. 3(a)), i.e., the odd-even transposition network, only neighbor values are compared and the number of levels required to perform a sorting operation is equal to the number of input bits. At each level of the linear network the ones move to the left side of the output, while the zeros move to the right. The inputs of each compare element are swapped whenever the most significant bit is 0 and the less significant bit is 1. Due to the topology of the odd-even network a less significant bit cannot pass over a more significant bit with the same value. Therefore when the bits are appropriately sorted using the linear network, their relative significance is preserved.

In the second case, the circuit is composed of appropriately connected subnetworks, called bitonic sorters [12]. A string of bits is bitonic when it has the form $111\ldots00\ldots011\ldots1$ or $00\ldots01\ldots1100\ldots0$. A bitonic sorter is effectively a butterfly network that sorts bitonic sequences. In order to construct a general bitonic sorting network $\log_2 n$ stages of bitonic sorters are used. At each stage, two bitonic sequences are merged and a new double size bitonic sequence is produced. Bitonic sorting networks cannot preserve at the output the relative significance of the bits under comparison. Due to the butterfly structure of the bitonic sorting networks the bits are recursively divided in two independent halves. When a bit is put in one half because of a swap, it cannot regain its relative significance compared to the rest bits with the same value that were placed to the other half. As an example you can see the two marked ones in Fig.3(b). Although they are correctly sorted at the output (no zero bit exists in a more significant position) the route that they followed, has altered their relative significance. Therefore, in case that the corresponding data bits followed them while they were sorted, the data bits would be in wrong order according to GRP.

Bitonic sorters have been employed for the GRP implementation in [10]. Additional circuitry has been added to preserve the relative significance of the data bits with control bits equal to one. However it is impossible using a single bitonic sorter to simultaneously preserve the relative significance of the data bits with a zero control bit that need to be placed to the right part of the result. This is the reason why in [10] two extraction units (Fig. 2) are used. Since GRP requires the relative significance of the data bits with the same control bit to remain unchanged, the odd-even transposition network is better suited for its implementation.

4 A Novel Sorter-Based GRP Unit

The proposed GRP unit is based on an enhanced odd-even transposition network. The circuit accepts the data bits

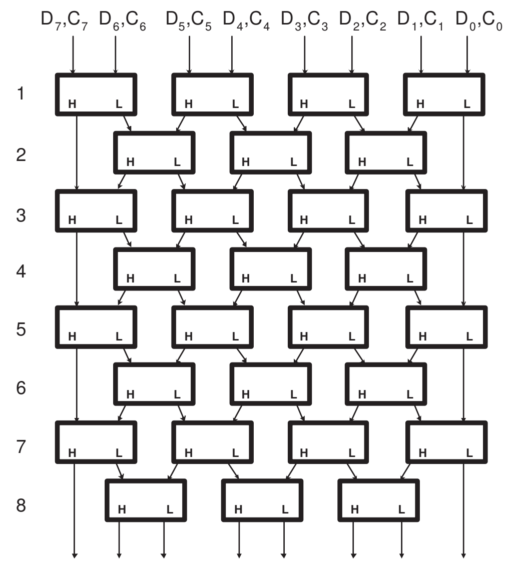


Figure 4. An 8-bit GRP unit based on the odd-even transposition network.

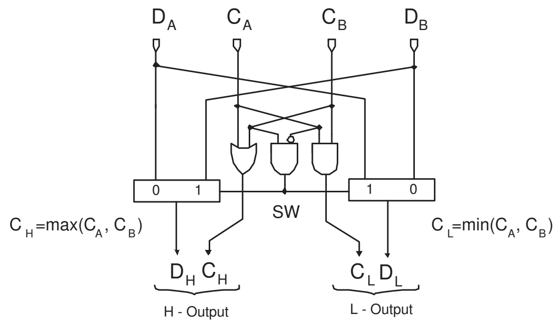


Figure 5. The circuit level implementation of the proposed compare-and-exchange cells.

that need to be separated to two groups at the output, and the corresponding control bits that denote, which data bits should be selected for each group. The block diagram of the enhanced linear transposition network is shown in Fig. 4. The compare and exchange cells of Fig. 4 assume as input two pairs of data and control bits. At the high output (H) of the cell, the maximum of the two sorted control bits appears along with the corresponding data bit. The opposite holds for the low output (L) of the cell. The circuit level implementation of the proposed compare-and-exchange cells is illustrated in Fig. 5. At each stage the control bits are sorted using OR gates to select the maximum and AND gates to select the minimum of the two bits. Whenever the left control bit C_A is 0 and the right control bit C_B is 1, then a swap between the data bits D_A and D_B should take place. In all other cases the control bits are already sorted and no data exchange is needed. The exchange of the data is controlled by the swap-enable signal SW which is equal to

$$SW = \overline{C}_A \cdot C_B \tag{1}$$

where $\cdot$ and $\overline{x}$ denote the logical AND and NOT operations, respectively. In the following, $+$ denotes the logical OR operation. The last level of the compare-and-exchange cells is simplified since no extra control bits are required at the output.

4.1 Subword-Oriented GRP Unit

By definition, GRP can directly support multi-bit subword operations. This can be achieved if the bits that belong to the same subword have identical control bits. In this paper, we investigate the case that the minimum supported subword size b is larger than a single bit. In this case the maximum number of subwords that exist in a n bit word are equal to n/b. Therefore in order to separate the subwords in two groups using GRP, n/b control bits are required. The enhanced linear sorting network can be simplified when used for multi-bit subwords. In each exchange operation, according to the value of the derived swap-enable signal, b-bit subwords need to be swapped. Therefore b 2-to-1 multiplexers are required in each compare-and-exchange cell that are all controlled by the same swap-enable signal. The already proposed GRP units [8], [9], [10] can be simplified in the same way, when they are designed to support a minimum subword size greater than one bit.

In the sequel, the GRP unit of n bits that supports a minimum subword size of b bits is denoted as $(n/b) \times b$ GRP unit. The design of a 8×8 GRP unit that functions on byte subwords is exactly the same with the one shown in Fig. 4, assuming that each D_i represents an 8-bit quantity and the multiplexers of the compare cells are 8-bits wide. The enhanced linear sorting network, when used for multi-bit subword GRP operations has three main advantages over the already presented solutions.

- A single sorting network can simultaneously separate the data bits in two groups according to GRP definition. In this way the hardware complexity is roughly reduced to half compared to the GRP units that use two separate extraction units [8], [9], [10].
- All wires connect to neighbor cells thus increasing the regularity of the design and reducing both the buffering requirements and the size of the driving gates. This is also supported by the fact that both the data path (subwords) and the control path (sorted control bits) of the circuit follow the same topology. This is not the case for the GRP units of [8], [9].
- The number of stages n/b is not prohibitive and is almost the same with previous solutions for the most common case of modern datapaths that support a minimum subword size of one byte.

These observations concerning the benefits of the proposed architecture will be better substantiated by the experimental results given in Section 5.

4.2 Fast Subword-Oriented GRP Unit

In the first version of the compare-and-exchange cells (Proposed I), the swap-enable signal, which drives the subword multiplexers, is computed via an AND gate with one input inverted. A faster solution would be derived, if the swap operation in each cell is controlled directly from the input control bits and the multiplexers are replaced by simpler gates. Such a modification is possible if we combine the two-datapath architecture shown in Fig. 2 and the enhanced linear sorting network. In this case, the compare and exchange cells should have four data inputs/outputs while the number of control bits remains unchanged. Following the architecture of Fig. 2, at first, the subwords are masked with the corresponding control bits and their alignment at the output is performed by the two extraction units. Because of masking, the bits of a subword D that assume the same control bit C are equal to $D \cdot C$ for the left extraction unit and $D \cdot \overline{C}$ for the right extraction unit. The enhanced linear sorting network can directly perform the functionality of each extraction unit. In this case several simplifications are possible.

Assume at first the case of the left extraction unit. The data inputs of the compare cells are denoted as $D_{A\to L}$ (left input) and $D_{B\to L}$ (right input). Following the multiplexing function of each cell (Fig. 5) and Eq. (1), the data at the H output $D_{H\to L}$ and the L output $D_{L\to L}$ are given by:

$$D_{H\to L} = D_{A\to L} \cdot \overline{(\overline{C}_A \cdot C_B)} + D_{B\to L} \cdot \overline{C}_A \cdot C_B \quad (2)$$

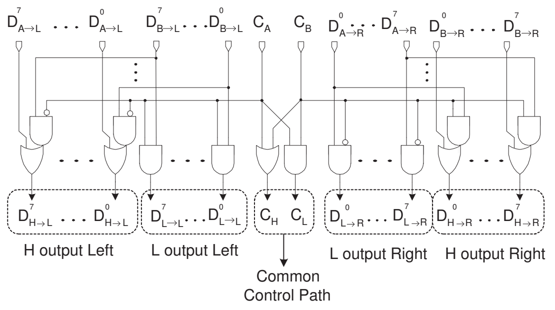
$$D_{L\to L} = D_{A\to L} \cdot \overline{C}_A \cdot C_B + D_{B\to L} \cdot \overline{(\overline{C}_A \cdot C_B)} \quad (3)$$

Since at the beginning the data bits of the left extraction unit $D_{A\to L}$ and $D_{B\to L}$ have already been masked with the control bits C_A and C_B respectively, equations (2) and (3) that describe the high and low data outputs of each compare cell can be written as follows:

$$D_{H\to L} = D_{A\to L} + D_{B\to L} \cdot \overline{C}_A \quad (4)$$

$$D_{L\to L} = D_{B\to L} \cdot C_A \quad (5)$$

In the right extraction unit the H and L outputs are interchanged because of the opposite alignment direction and the control bits are inverted. If again index A refers to the left input and B to the right input of a compare cell, then a swap is performed in the right extraction unit when $(\overline{C}_A, \overline{C}_B) = (1, 0)$ since data move gradually to right. This is equivalent to the swap-enable condition of the left extraction unit, i.e., $(C_A, C_B) = (0, 1)$, already described by Eq. (1). This fact allows us to share the same control network between the two extraction units. Following this conclusion and Eq. (1) the data $D_{A\to R}$ and $D_{B\to R}$ are selected



$D^7_{A\to L}$ . . . $D^0_{A\to L}$
$D^7_{B\to L}$ . . . $D^0_{B\to L}$
C_A
C_B
$D^0_{A\to R}$ . . . $D^7_{A\to R}$
$D^0_{B\to R}$ . . . $D^7_{B\to R}$
$D^7_{H\to L}$ . . . $D^0_{H\to L}$
$D^7_{L\to L}$ . . . $D^0_{L\to L}$
C_H C_L
$D^0_{L\to R}$ . . . $D^7_{L\to R}$
$D^0_{H\to R}$ . . . $D^7_{H\to R}$
H output Left
L output Left
L output Right
H output Right
Common Control Path

Figure 6. Optimized implementation of the compare-and-exchange cell for an 8×8 unit.

at the H and L outputs of the right extraction unit as follows:

$$D_{H\to R} = D_{B\to R} \cdot \overline{(\overline{C}_A \cdot C_B)} + D_{A\to R} \cdot \overline{C}_A \cdot C_B \quad (6)$$

$$D_{L\to R} = D_{B\to R} \cdot \overline{C}_A \cdot C_B + D_{A\to R} \cdot \overline{(\overline{C}_A \cdot C_B)} \quad (7)$$

Since in this case $D_{A\to R}$ and $D_{B\to R}$ have already been masked in the beginning by the complement of the control bits $\overline{C}_A$ and $\overline{C}_B$, the outputs of the compare cells can be simplified as follows:

$$D_{H\to R} = D_{B\to R} + D_{A\to R} \cdot C_B \quad (8)$$

$$D_{L\to R} = D_{A\to R} \cdot \overline{C}_B \quad (9)$$

The two partial results of the left and the right extraction unit are unified with an OR operation. The modified cell that implements the newly derived equations for the case of an 8×8 GRP unit is shown in Fig. 6 (Proposed II). Without increasing the area of the circuit we managed to make the generation of the swap-enable signal unnecessary, and also to equally split the fanout between C_A and C_B that guide independently the two sorting directions. Also the AND/OR gates that sort the control bits are efficiently shared by the two extraction units. Both extraction units are placed as a single circuit using the topology of Fig. 4.

5 Experimental Results

The proposed architectures have been compared to the fastest previous solution [10] using static CMOS implementations in UMC 130nm standard performance CMOS technology. All measurements were performed for the typical process corner at a temperature of 70^{o}C, assuming a nominal supply of 1.2V (1FO4 $\sim$ 62ps). In order to explore the energy-delay space for each design we performed gate sizing for several delay targets, beginning from the circuit's minimum achievable delay. Optimization is performed with MatLab and using the methodology presented in [13]. For the derived gate sizes the energy and the delay of each circuit have been measured in HSpice. During optimization

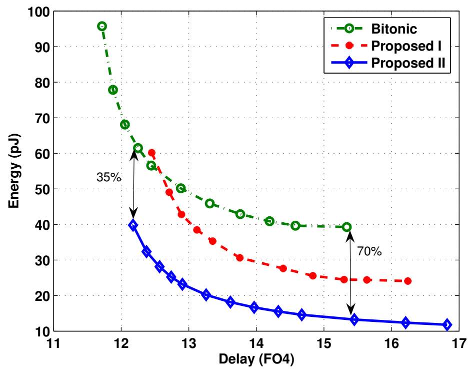


Figure 7. The energy-delay curves of the circuits implementing an 8×8 GRP unit.

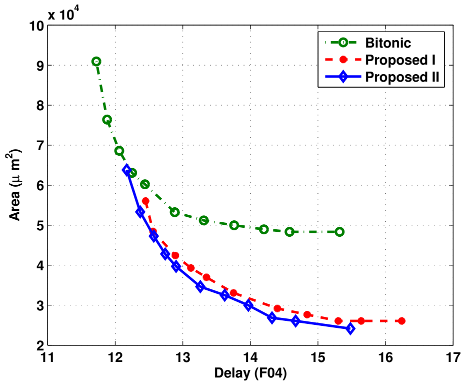


Figure 8. The area-delay curves of the circuits implementing an 8×8 GRP unit.

and measurements we assumed that the outputs of the circuit are loaded with a capacitance of 100fF that roughly corresponds to a 350m metal-2 wire in our technology. Interstage wiring loads, both capacitance and resistance, have also been taken into account, assuming for the design a bit slice of 16 metal-1 tracks. In order to get reasonable delays all compared solutions have been optimized assuming that the maximum allowed input capacitance is less than 25fF.

Fig. 7 illustrates the energy-delay behavior of the two variants of the proposed GRP unit and the architecture presented in [10] for an 8×8 configuration. The energy per operation required by the proposed designs is significantly smaller. For almost all delay values the new circuits achieve energy reductions that range between 35% and 70%. The subword-oriented design of [10] achieves a minimum delay, which is only 0.4FO4 less than the minimum delay achieved by the proposed permutation unit with two extraction units. The first version of the proposed GRP unit is slower mostly because of the higher fanout requirements and the increased delay of the multiplexers compared to the AND-OR gates of the second implementation. For larger ratios of output to input capacitance more inverter stages should be added in all circuits under comparison. In this case their delay and energy is uniformly increased and does not alter the energy-delay trend illustrated by Fig. 7.

For each one of the derived solutions the area-delay tradeoff has been also investigated. Fig. 8 shows the area-delay characteristics of all circuits under comparison. It can be observed that the proposed architectures require significant less area than the GRP unit of [10] (more than 40% for 14FO4 delay). The area of the proposed architectures is almost the same. The second approach is slightly more efficient due to the reduced gate sizes and the more dense layout of the static CMOS AND-OR gates compared to the transmission-gate multiplexers of the first implementation.

6 Conclusions

An efficient subword-oriented GRP unit has been presented in this paper. The design is based on an enhanced linear sorting network that offers an energy-delay efficient solution with reduced wiring requirements, and yields efficient implementations in deep-submicron technologies.

References

[1] K. Diefendorff and P. K. Dubey, "How Mutimedia Will Change Processor Design," *Computer*, vol. 30, no. 9, pp. 43–45, Sept. 1997.

[2] T. Conte et al., "Challenges to Combining General-Purpose and Multimedia Processors," *Computer*, vol. 12, no. 30, pp. 33–37, Dec. 1997.

[3] N. T. Slingerland and A. J. Smith, "Multimedia Extensions for General Purpose Microprocessors: A Survey," *Microprocessors and Microsystems*, , no. 29, pp. 225–246, 2005.

[4] R. B. Lee, Z. Shi, and X. Yang, "Efficient permutation instructions for fast software cryptography," *IEEE Micro*, vol. 21, no. 6, pp. 56–69, Dec 2001.

[5] Z. Shi and R. B. Lee, "Bit permutation instructions for accelerating software cryptography," in *IEEE ASAP*, July 2000, pp. 138–148.

[6] X. Yang and Ruby B. Lee, "Fast Subword Permutation Instructions Using Omega and Flip Network Stages," in *Intern. Conference on Computer Design*, Sep. 2000, pp. 15–22.

[7] Z. J. Shi, *Bit Permutation Instructions: Architecture, Implementation and Cryptographic Properties*, PhD Thesis, Princeton Univ., 2004.

[8] Z. J. Shi and R. B. Lee, "Implementation complexity of bit permutation instructions," in *Proc. of Asilomar Conference on Signals, Systems and Computers*, Nov 2003, pp. 879–886.

[9] Y. Hilewitz, Z. J. Shi, and R. B. Lee, "Comparing fast implementations of bit permutation instructions," in *Proc. of Asilomar Conference*, Nov 2004, pp. 1856–1863.

[10] G. Dimitrakopoulos et. al., "Fast Bit Permutation Unit for Media Enhanced Microprocessors," in *IEEE ISCAS*, May 2006, pp. 49–52.

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, 2nd Edition*, MIT Press, 1990.

[12] K. E. Batcher, "Sorting networks and their applications," in *AFIPS Joint Computer Conference*, 1968, pp. 307–314.

[13] S. P. Boyd, S. J. Kim, D. Patil, and M. A. Horowitz, "Digital Circuit Optimization via Geometric Programming," *Operations Research*, vol. 53, no. 6, pp. 899–932, Nov.-Dec. 2005.