FPGA-Based Variable Length Decoders

Jari Nikara[†‡], Stamatis Vassiliadis[‡], Jarmo Takala[†], and Petri Liuha[§]

[†]*Inst. of Digital and Computer Systems*
Tampere University of Technology
Tampere, Finland
{*jari.nikara, jarmo.takala*}@tut.fi

[‡]*Computer Engineering Lab.*
Delft University of Technology
Delft, The Netherlands
S.Vassiliadis@et.tudelft.nl

[§]*Nokia Research Center*
Tampere, Finland
petri.liuha@nokia.com

Abstract

A straightforward and fair comparison of variable length decoders is extremely difficult due to different implementation approaches, e.g., different codeword tables, IC technologies, design styles, and compression ratios. On the other hand, reconfigurable platforms provide fast design iteration times to change the design variables. Therefore, the variable rate symbol-parallel Variable Length Decoding (VLD) approach has been compared to FPGA-based variable length decoders presented in literature. The behavioural non-optimized VHDL model of the decoder has been mapped onto the FPGAs used in the references in order to guarantee same technological features. The variable rate symbol-parallel decoder provides 16-100 % better throughput at 2-3.6 times lower frequencies than the referenced designs.

1. Introduction

In data compression, a set of symbols is represented with reduced number of bits. Variable Length Coding (VLC), e.g., Huffman code [1], is a lossless compression technique where frequently occurring symbols are represented with shorter codewords without explicit boundary information between codewords. Therefore, in Variable Length Decoding (VLD), the length of the current codeword should be known before the next codeword can be decoded. This feature complicates the decoder design substantially and limits the performance.

A traditional VLD method is to decode one symbol at a time in symbol-serial fashion using either bit-serial tree-based processing as in [2] or bit-parallel approach as in [3–6]. In symbol-parallel schemes like [7, 8], the major design issue is to break data dependencies between codewords. Another issue is the management of the increasing hardware and control complexity, especially when large codeword tables and long codewords are used.

A straightforward and fair comparison of variable length decoders is practically impossible due to different implementation approaches. Standards and different codeword tables distinguish decoders from each other. Furthermore, input data with different compression ratios affect decoders with variable output rate. Implementation platform, e.g., ASIC vs. FPGA, arises technology dependency while design styles (synchronous vs. asynchronous) set up different restrictions to decoders. Altogether the decoding performance of the used technique is not directly comparable although all the previous aspects can be considered as critical design issues as well.

In this paper, VLD scheme proposed earlier in [8] is compared to FPGA-based variable length decoders presented in literature by using uniform implementation approaches. Short design iteration times on FPGA allow the configuration of the decoder based on the scheme to match the reference decoders or to provide at least the same features with references. In other words, the uniformity is guaranteed with same codeword tables, compression ratios, implementation platform, and synchronous design style. Briefly, the decoder based on [8]:

- has variable output rate, 1-6 symbols per cycle;
- provides 16-100% better throughput than references;
- operates at 2-3.6 times lower frequencies than referenced decoders.

The remaining of the discussion is organized as follows. The background is overviewed in Section 2. In Section 3, the reported FPGA-based variable length decoders are introduced with the given performance figures. The presented decoders are compared and discussed in Section 4. Finally, the conclusions are presented in Section 5.

2. Background

In VLD, the incoming variable length coded input stream is buffered and a codeword length is detected from a block of the input stream as illustrated in Fig. 1. The detected codeword or corresponding pseudocode is used to determine the actual symbol with the aid of predefined codeword values, i.e., codeword table. Depending on the decoding approach the symbols may be buffered, e.g., for managing variable processing rate. The input stream is then aligned for the next decoding iteration according to the length of the codeword.

In general, there is no explicit boundary information for detecting the end or beginning of the codeword in the variable length coded data stream. This implies that the length of the current codeword should be known before

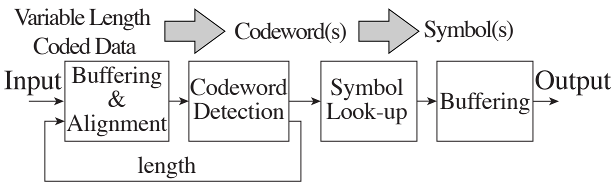

Figure 1. Block diagram of generalized VLD.

the next codeword can be decoded. This introduces data dependency, which complicates the decoder design substantially. Therefore, a traditional VLD method is to decode one symbol at a time in symbol-serial fashion by using either bit-serial or bit-parallel processing. In bit-serial decoding as in [2], input stream is compared bit-by-bit to a binary tree starting at the root of the tree until the entire codeword is detected in the leaf node. Instead in bit-parallel processing used in [3–6], the longest codeword length bits are buffered in order to guarantee a symbol for each cycle.

Due to the recursive dependencies between the codewords bit-serial processing is not applicable for symbol-parallel decoding when processing a single data stream. In symbol-parallel decoding schemes in [7, 8], the major design issue is to break the data dependencies between the codewords. Another issue is the management of the increasing hardware and control complexity, especially when large codeword tables and long codewords are used. The increasing complexity can be managed by either decoding only short codewords concurrently or limiting the number of symbols.

The performance of the variable length decoders depend not only on the chosen decoding technique but also implementation approach. Standards, like JPEG [9] or MPEG-2 [10], with different codeword tables set their own requirements for the decoder. Furthermore, input data with different compression ratios, i.e., ratios of the compressed data to original uncompressed data, affect decoders with variable output rate; the less compression, the longer codewords resulting in decreased throughput.

One main issue in comparing performance of different decoder implementations is how to equalize the effects of the different ASIC technologies or how to make different FPGAs and their specific features equivalent to each other or even to ASICs. However, the characteristic figures about the performance of the decoder are mostly given only for the chosen technology and without detailed variables. In other words, the results are technology dependent and consequently the performance of the chosen decoding technique is hidden behind technology.

3. Variable Length Decoders

In this section, the FPGA-based variable length decoders reported earlier are presented in three classes according to the decoding technique. All the decoders process data in bit-parallel manner but from symbol parallelism point of view they are different. In order to emphasize the difficulties in comparison with different technologies, let us remark the huge variation in the descriptive figures of the following decoders.

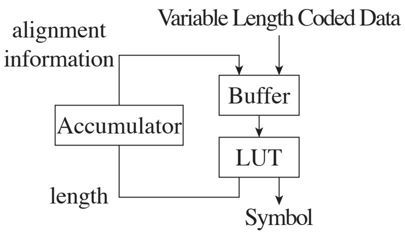

Figure 2. Symbol-serial decoder.

3.1. Symbol-Serial Approach

Symbol-serial variable length decoders to be presented in this paper are based on the approach presented by Lei and Sun in [3]. A principal block diagram of the decoders is depicted in Fig. 2. Briefly, the block of encoded bitstream is buffered and matched with the stored prespecified codewords in a Look-Up Table (LUT). The codeword length is accumulated by the accumulator and its value is used as a pointer to the correct location in buffered bitstream. Furthermore, the accumulator triggers the reloading of buffer with new data.

In the symbol-serial approach, a symbol is produced in every clock cycle regardless of the codeword length. The increased hardware complexity compared to bit-serial decoding is compensated by the constant output rate and guaranteed operation speed. However, the approach does not exploit the fact that most probably a block of bits in the input stream contains more than one codeword.

Aspar *et al.* reported a symbol-serial variable length decoder implemented on Altera's Flex 10K20RC240-4 and Flex 10K20RC240-3 FPGAs [4]. The decoder has been designed to support up to 16-bit codewords according to JPEG standard [9]. The achieved operation frequencies are 9.91 MHz for 10K20RC240-4 and 11.54 MHz for 10K20RC240-3. In both platforms, the utilization of logic cells is 1145 out of 1152 logic cells.

Another FPGA-based symbol-serial variable length decoder is proposed by Jeon *et al.* in [5]. In order to reduce the processing time in the critical path, the decoder exploits plane separation technique where input plane and OR plane performing the data buffering operate in parallel. The consecutive PLA-based matching process uses exactly the same method as previous decoder, i.e., the block of encoded bitstream is matched with all possible codewords stored into LUT.

The decoder using the plane separation technique has been realized on Altera's Flex 8000 FPGA. When applying the presented technique to MPEG-2 intra-frame decoding [10], the throughput of 15 million symbols per second has been achieved. From logical resources point of view, about 30 % performance improvement from Lei's approach doubles the required resources.

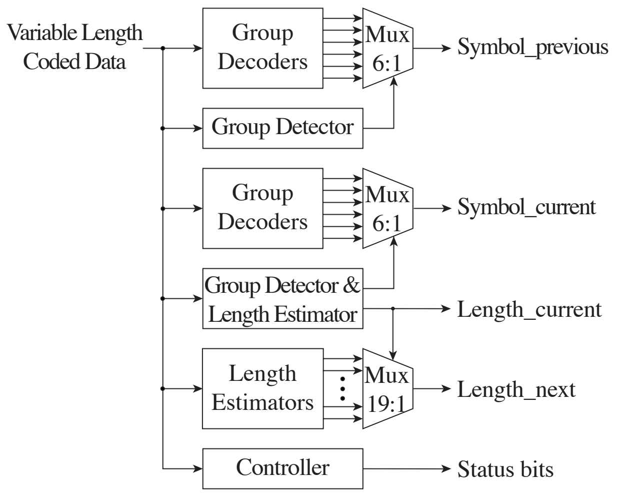

Figure 3. Constant rate symbol-parallel decoder.

3.2. Constant Rate Symbol-Parallel Approach

Sima *et al.* [6] introduced a scheme to improve the performance of the general purpose processor by augmenting it with an FPGA-based symbol-serial variable length decoder. In order to achieve higher throughput, the previous scheme has been extended with a constant rate symbol-parallel VLD on the FPGA-augmented processor in [7]. With such a VLD technique, the output rate is kept constant but in every cycle more than one symbol is produced.

The principal block diagram of the MPEG-2 decoder returning two symbols per cycle is illustrated in Fig. 3. Using the terms previous, current, and next in chronological order, the main idea is to determine the symbol and the length for the current codeword, and only the length for the next codeword during the same VLD call. Concurrently, the symbol for the previous codewords are determined. In other words, the generation of the next symbol is postponed to the subsequent cycle and the series of decoding cycles results in symbol-parallel decoding.

In more details, the operation can be described as follows. A codeword table is partitioned into LUTs referred as group decoders. The decoding is started by forwarding bit fields from the block of encoded bitstream to parallel group decoders and performing parallel matching with codewords in group decoders. Each group decoder returns the symbol value as if the generated symbol would have been valid. The selection of the proper symbol is done according to leading bits in group detector, which determine the correct group. Simultaneously, length estimators determine the length of the current codeword and the length candidates for the next codeword. The valid length is selected according to the current length.

Sima *et al.* considered an FPGA-augmented TriMedia processor running at 200 MHz [7, 11]. When mapped onto Altera's ACEX EP1K100 FPGA, the approach returning one symbol exhibits seven TriMedia cycles while two symbols can be returned in eight TriMedia cycles. The MPEG-2 compliant two-symbol decoder with constant output rate yields the throughput of about 50 Msymbols/s or 275 Mbits/s. The implementation requires all 12 Embedded Array Blocks, i.e. RAM blocks, and 51 % of the logic cells supporting either codeword table B.14 or B.15 in MPEG-2 standard.

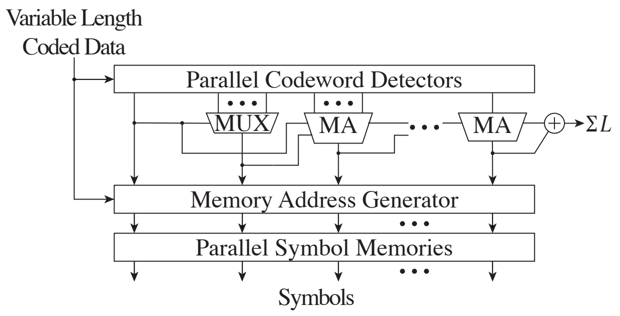

Figure 4. Variable rate symbol-parallel decoder.

3.3. Variable Rate Symbol-Parallel Approach

Nikara *et al.* introduced a variable rate symbol-parallel VLD technique in [8]. The approach exploits the fact that most probably a block of bits in input stream contains more than one codeword. Consequently, the technique is capable of decoding multiple symbols from arbitrary length buffer with variable rate. The decoding can be decomposed into two stages: parallel/serial codeword detection and symbol look-up. In addition, the symbol look-up can be performed in two stages: address generation and symbol fetch.

The operation of the decoder illustrated in Fig. 4 is as follows. A block of encoded bitstream is stored into a N-bit codeword buffer and codeword detection is performed by parallel matching units referred as parallel codeword detectors. All the matching units detect codewords simultaneously and return the lengths of the detected codewords. The procedure results in redundant number of codeword lengths from which incorrect values are removed by recursive selection; the length of the first codeword defines the index to the second codeword, the lengths of the first and second codeword define the index to the third codeword, etc. Such a recursive selection is realized with the aid of Multiplexed Add (MA) units, which perform addition and the selection of the proper length simultaneously.

The symbol look-up is started with determining the address for each symbol corresponding the detected codeword. The length is used to determine the page in the symbol table and the partial codeword, which can be extracted from the codeword buffer, defines the offset in the symbol table. The symbol look-up can be performed independently from symbol memory. The sum of the valid codeword lengths is provided to an external shifter aligning the encoded input stream for a new decoding cycle.

The variable rate symbol-parallel decoding scheme presented in [8] has been applied to MPEG-2 variable length code in [12]. The block diagram of MPEG-2 decoder returning up to six symbols per cycle is depicted

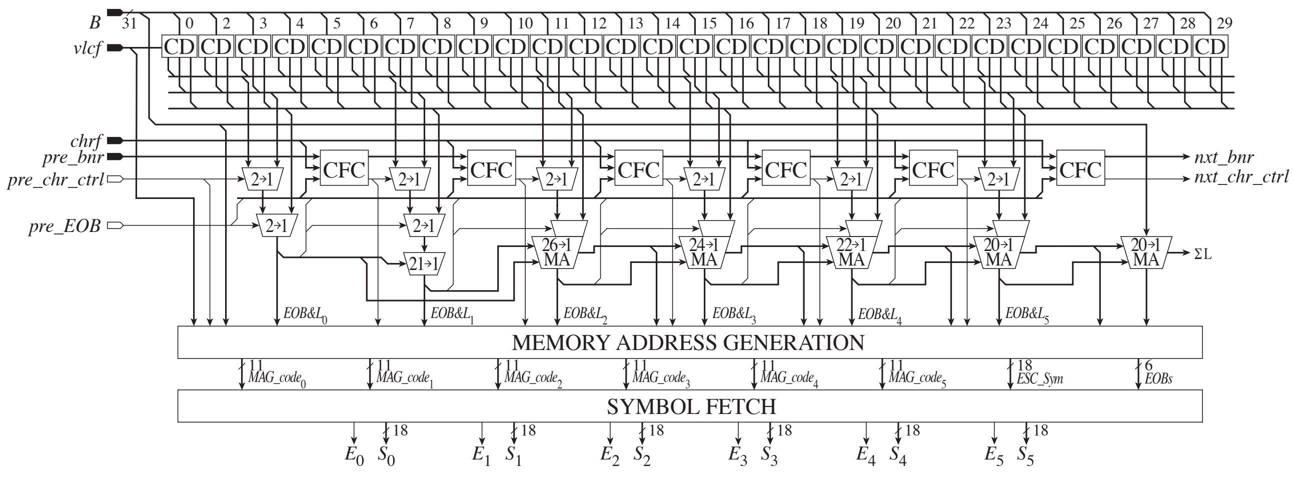

Figure 5. Variable rate symbol-parallel MPEG-2 decoder.

in Fig. 5. The codeword detection consists of 29 Codeword Detectors (CD), which have inputs from the 31-bit buffer B. Each CD returns three End-of-Block (EOB) statuses and codeword lengths from the location shown above the CD. The returned values are candidates for the DC coefficient (luminance and chrominance) and AC coefficient according to VLC format parameter *vlcf* (intra or non-intra). Chrominance Format Counter (CFC) selects between luminance and chrominance candidates according to chrominance format *chrf* (4:2:0, 4:2:2, 4:4:4, or non-intra) and macroblock number *pre_bnr*. The selection of the proper codeword length from DC and AC candidates is controlled by previous EOB status *pre_EOB*. The VLC format *vlcf*, chrominance controls *chr_ctrl*, the EOB statuses, and lengths of the codewords are forwarded to the memory address generation with the intermediate sums in order to generate the memory address and sign referred as *MAG_code* for each codeword. Apart from *MAG_codes*, the possible escape value *ESC_Sym* and the EOB statuses *EOBs* are returned. The symbol fetch is trivial read operation from parallel symbol memories.

Behavioural VHDL model of the decoder has been mapped onto Xilinx Virtex-II FPGA where 2 940 CLBs out of 23 040 were allocated in addition to three dual-port memories of 160 rows with 11-bit words. The decoder can decode at most six codewords per cycle at 22MHz frequency. When decoding data stream with 30% compression ratio, the average throughput of 585 Mbits/s, 105 Msymbols/s, or 4.8 symbols per cycle has been achieved.

4. Comparison

In order to compare the variable rate symbol-parallel decoding technique, the non-optimized model of the decoder in [12], where the specific features of implementation platform are not exploited, is mapped onto the same FPGA technologies as the references but without parallel symbol memories. The characteristics of the variable length decoders in [4, 5, 7] and the results of the decoder [12], which are obtained by using Exemplar LeonardoSpectrum, are summarized into three columns labeled as Altera Flex10K, Altera Flex8K, and Altera ACEX1K100 according to used FPGA in Table 1.

Although our original objective has been to uniform the decoders based on different decoding approaches, clear differences and restrictions still exist. The symbol-parallel decoders in [7, 12] require clearly more **hardware resources** than symbol-serial decoders in [4, 5]. Furthermore, the decoders in [4, 5] are independent whereas decoders in [7, 12] are clearly targeted to embedded system providing external resources for data buffering and aligment. The decoder in [12] does not compete in hardware resources with other decoders due to the high degree of parallelism and it does not fit into all FPGAs that are used in the references. On the other hand, there are already enough resources available in state of the art FPGAs and the integration density is increasing.

In the symbol-parallel decoding approaches used in decoders [7, 12], the **critical path** can be adjusted with requirements of the application by processing more data in a cycle; more time, more symbols per cycle. Therefore, they are advantageous when cycle time is specified according to environment. The critical path in decoder [12] is dominated by recursive selection of proper codewords and therefore, codeword properties reflecting to codeword detection delay have minor effect in total cycle time. In other words, the increase in codeword detection delay has relatively small part in total cycle time.

The decoders in [4, 5, 7] have constant output rate resulting in constant **throughput** in terms of symbols whereas source data statistics reflect to throughput in the decoder [12]. The decoder is sensitive to compression ratio due to variable processing rate, i.e., the worse compression ratio implies longer codewords and thus less codewords in the buffer returning less symbols per cycle. The throughput values in Table 1 are estimated by assuming the average codeword length of 5.5 bits.

The properties of the **decoders on Altera Flex10K** supporting JPEG standard are collected into column labeled as Altera Flex10K in Table 1. The decoder in [12]

Table 1. Comparison of FPGA-based variable length decoders

	Altera Flex10K		Altera Flex8K		Altera ACEX1K100	
	Aspar *et al.* [4]	Nikara *et al.*	Jeon *et al.* [5]	Nikara *et al.* [12]	Sima *et al.* [7]	Nikara *et al.* [12]
Standard	JPEG	JPEG	MPEG-2 Intra	MPEG-2	MPEG-2	MPEG-2
CWT	K.5	K.3-K.6	Not known	B.12-B.15*	B.14 or B.15	B.14 or B.15
Logic cells	1 145	5 833	Not known	6 397	51 %	35 %
Frequency	11.54 MHz	4.8 MHz	15 MHz	4.2 MHz	25 MHz	12.1 MHz
Throughput	Constant 1 S/C 11.54 MS/s 63 Mbits/s	Variable 1-6 S/C avg. 4.8 S/C 23.04 MS/s 127 Mbits/s	Constant 1 S/C 15 MS/s 82 Mbits/s	Variable 1-6 S/C avg. 4.8 S/C 20 MS/s 111 Mbits/s	Constant 2 S/C 50 MS/s 275 Mbits/s	Variable 1-6 S/C avg. 4.8 S/C 58 MS/s 319 Mbits/s

CWT: Codeword tables. S/C: Symbols per cycle. MS/s: Million symbols per second. * Support for 4:2:0, 4:2:2, and 4:4:4 chrominance formats.

is configured to support JPEG standard and is referred as Nikara *et al.* in the column. The structure of the decoder follows the structure of the MPEG-2 decoder in Fig. 5, only codeword tables are assumed to be typical Huffman tables from the standard [9]. The decoder requires about five times more hardware without symbol memories providing double throughput in time domain. In addition, the decoder supports four codeword tables.

The characteristics of the MPEG-2 **decoders on Altera's Flex8K** are summarized into the column Altera's Flex8K in Table 1. The MPEG-2 symbol-parallel decoder [12] illustrated in Fig. 5 is used in comparison although decoder in [5] supports only intra-frame decoding. This difference reflects to the critical path due to the different complexities of the codeword tables in LUT and CDs and larger multiplexers and MA units. However, the symbol-parallel decoder results in better throughput than symbol-serial decoder assuming 5.5-bit codewords on average, i.e., compression ratio of about 30%.

When comparing symbol-parallel **decoders on ACEX EP1K100 FPGAs** supporting codeword table B.14 or B.15 in MPEG-2, we assume again the compression ratio of 30%. The decoder in [12] is simplified to support only a single codeword table, which is reflected in the required resources. The achieved results are given in the column ACEX EP1K100 FPGAs in Table 1. The throughput of the decoder in [7] is constant, two symbols per cycle. Instead the decoder in [12] is capable of returning up to six symbols per cycle although less symbols would imply shorter cycle time.

5. Conclusions

In this paper, FPGA-based variable length decoders have been reviewed and compared. When presenting the method and estimating its performance with the aid of certain implementation, the performance of the method is extremely difficult to distinguish from the technological performance. Therefore, exact design variables or characteristics that are independent from technology should be also provided. Without these independent facts, the results can always be speculated. However, according to experimental results, the performance of the variable rate symbol-parallel approach can be considered promising for future applications.

6. References

[1] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, Sept. 1952.

[2] A. Mukherjee, N. Ranganathan, and M. Bassiouni, "Efficient VLSI designs for data transformation of tree-based codes," *IEEE Trans. Circuits Syst.*, vol. 38, no. 2, pp. 306–314, Mar. 1991.

[3] S. M. Lei and M. T. Sun, "An entropy coding system for digital HDTV applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 1, no. 1, pp. 147–155, Mar. 1991.

[4] Z. Aspar, Z. M. Yusof, and I. Suleiman, "Parallel Huffman decoder with an optimized look up table option on FPGA," in *Proc. TENCON 2000*, vol. 1, Kuala Lumpur, Malaysia, Sep. 24–27 2000, pp. 73–76.

[5] J. H. Jeon, Y. S. Park, and H. W. Park, "A fast variable-length decoder using plane separation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 5, pp. 806–812, Aug. 2000.

[6] M. Sima, S. Cotofana, S. Vassiliadis, J. T. J. van Eijndhoven, and K. Visser, "MPEG macroblock parsing and pel reconstruction on an FPGA-augmented TriMedia processor," in *Proc. IEEE Int. Conf. Comput. Design*, Austin, Texas, USA, Sep. 24–26 2001, pp. 425–430.

[7] ——, "MPEG-compliant entropy decoding on FPGA-augmented TriMedia/CPU64," in *Proc. IEEE Symp. Field-Programmable Custom Computing Machines*, Napa Valley, CA, USA, Apr. 21–24 2002, pp. 261–270.

[8] J. Nikara, S. Vassiliadis, J. Takala, M. Sima, and P. Liuha, "Parallel multiple-symbol variable-length decoding," in *Proc. IEEE Int. Conf. Comput. Design*, Freiburg, Germany, Sept. 16–18 2002, pp. 126–131.

[9] International Telecommunication Union, "Information technology – Digital compression and coding of continuous-tone still images – requirements and guidelines, CCITT Recommendation T.81," Sept. 1992.

[10] ——, "Information technology – Generic coding of moving pictures and associated audio information: Video, ITU-T Recommendation H.262," Feb. 2000.

[11] M. Sima, personal correspondence, Apr. 2003.

[12] J. Nikara, S. Vassiliadis, J. Takala, , and P. Liuha, "Multiple-symbol parallel decoding for variable length codes," accepted to *IEEE Trans. VLSI Syst.*