ISSN 2319-8885,Volume01,Issue No. 02

International Journal of Scientific Engineering and Technology Research

www.semargroups.org

Jul-Dec 2012, P.P. 89-96

Design prototype of DVB Symbol for Multibank Memory

T.SARITHA KUMARI[1],MOHD.SHAHBAZ KHAN [2]

[1]M.Tech Student of CMRIT, Ranga Reddy(Dt), AP-India,e-mail: sarithatalapally@gmail.com,

[2]Asst Professor, ECE Dept, JCMRIT, Ranga Reddy(Dt), AP-India,

Abstract: In this paper, an efficient symbol-deinterleaver architecture compliant with the digital-video-broadcasting (DVB) standard is proposed. By partitioning the entire symbol buffer into four separate parts with a special low-conflict access control strategy, the symbol deinterleaver can be implemented with four-bank single-port on-chip memory blocks with slight overhead. The experimental result shows that 30% savings of hard-ware cost can be achieved compared with the conventional double-buffer approach. In addition, a look ahead online circuit of a symbol permutation-address generator is also proposed, which can provide the required permutation addresses every cycle to avoid either the use of a lookup table or an extra temporary buffer. Being the major part of the entire DVB forward-error-correction decoder, the proposed symbol deinterleaver can contribute a great saving of the overall decoder cost.

Keywords: Digital video broadcasting (DVB), symbol interleaving.

I. INTRODUCTION

The digital-video-broadcasting (DVB) standard [1], [2] is one of the major advanced video broadcasting standards which have been adopted in Europe and many Asian countries. In these areas, due to this new technology, the traditional home television systems will soon be phased out and replaced by the new mobile television service has also emerged. Therefore, the design of efficient DVB receivers has attracted a lot of attention. Among the entire receiver system, the forward-error-correction (FEC) decoder is one of the key modules. The FEC scheme adopted by the DVB standard is based on the concatenated code which consists of a Reed–Solomon (RS) code, convolution interleaving, convolutional code, bitwise interleaving, and symbol interleaving. Many researches have been devoted to explore the design of convolutional and RS de-coders. The design of deinterleavers, due to their relative simplicity in operation, has seldom been addressed. However, the silicon cost of these, particularly the symbol deinterleaver, can occupy more than 35% area of the entire FEC decoders [3]–[5]. Therefore, this paper aims to address the area-efficient VLSI architecture of the symbol deinterleaver.

II. REVIEW OF THE SYMBOL-DEINTERLEAVER DESIGN

The DVB symbol interleaving, belonging to the category of the block interleaving scheme, is used to map the v-b words onto the 1512 (2k mode), 3024 (4k mode), or 6048 (8k mode) active carriers per orthog-onal frequency division multiplexing (OFDM) symbol. The number of words for each block and the number of bits ¿ for each word depend on the OFDM model and the modulation scheme adopted in the system,

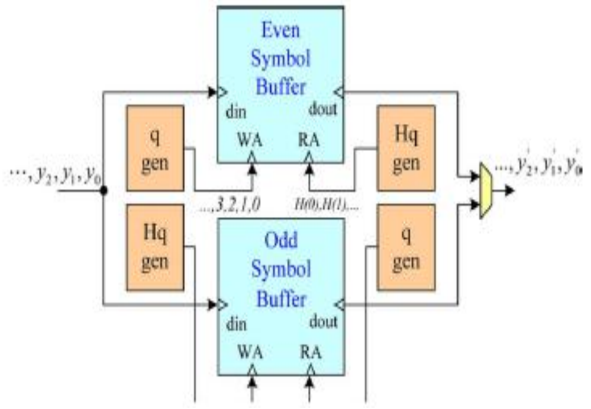


Fig. 1. Block diagram of the typical symbol-deinter leaver design.

respectively. Assuming that the data block input to the symbol interleaver is represented as

$Y' = (y'_0, y'_1, y'_2, \ldots, y'_{N_{\max}-1})$, it will be permutated to a new data block $Y = (y_0, y_1, y_2, \ldots, y_{N_{\max}-1})$ according to the following definition [1], [2]: $y_{H(q)} = y'_q$, for even symbols for $q = 0, \ldots, N_{\max} - 1$. $y_q = y'_{H(q)}$, for odd symbols for

$$q = 0, \ldots, N_{\max} - 1 \qquad (1)$$

Where $N_{\max}$ is equal to 1512, 3024, and 6048 for 2k, 4k, and 8k modes, respectively. $H(q)$ is a special permutation function which can be used to produce a pseudorandom sequence of length $N_{\max}$ consisting of nonrepetitive numbers from 0 to $N_{\max} - 1$. For the receiver of the DVB signals, it will require a symbol-deinterleaver module to reverse the order of the data back to the original. Fig. 1 shows the block diagram of the typical symbol-deinterleaver design [4], [6] which consists of two ping-pong symbol buffers.

The function of the buffer is to hold an entire incoming symbol of data such that they can be retrieved afterward according to their deinterleaving order. Each buffer can be realized by an on-chipSRAMblock with $N_{\max}$ cells and accessed under the control of the associated address generator. The storing and fetching pattern of the data will vary with the even and odd symbols. For even symbols, the buffer operations can be described by

$$\begin{aligned} mem_{\text{even}}(q) &= y_q, \quad \text{for } q = 0, \ldots, N_{\max} - 1 \\ y'_q &= mem_{\text{even}}(H(q)), \quad \text{for } q = 0, \ldots, N_{\max} - 1 \end{aligned} \qquad (2)$$

while, for the odd ones, they can be described as

$$\begin{aligned} mem_{\text{odd}}(H(q)) &= y_q, \quad \text{for } q = 0, \ldots, N_{\max} - 1 \\ y'_q &= mem_{\text{odd}}(q), \quad \text{for } q = 0, \ldots, N_{\max} - 1 \end{aligned} \qquad (3)$$

where the notation $mem_{\text{odd}}(i)$ represents the th cell of the odd symbol buffer.

Since only either a memory read or memory write operation will take place at any time for each buffer, the port number for the on-chip SRAM used for the buffer can be only one. In addition, the size of each memory cell required is equal to the word length of the deinterleaved data, which is usually multiple of the word length v of the original transmitted data. It depends on the number of bits used to sample the received data at the receiver. Due to the noise incurred during the transmission, the receiver will usually sample more bits for the better recovery of the data. The use of three to four bits for the soft inputs is verycommon in practice [3], [4]. In Fig. 1, two separate buffers dedicated to the even and odd symbols are employed.

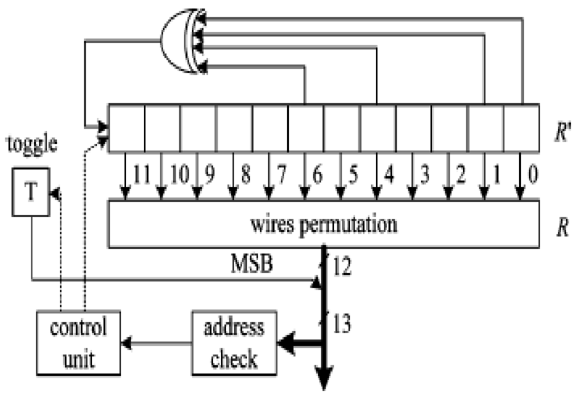


Fig. 2. Symbol interleaver address-generation scheme for the 8k mode.

These ping-pong buffers, however, can be further replaced by a single buffer if their operations can be synchronized properly. It can be observed from (2) and (3) that the write address of the qth input data of the current symbol and the read address of the qth output data of the previous symbol are the same. They are both equal to q if the current symbol is even and $H(q)$ if the current symbol is odd. Therefore, if the qth data write and qth data read operations can be scheduled to the neighboring time slots, the number of buffers as well as the sets of address generators required can be reduced from two to one. However, the memory port counts for the single buffer will increase to two since one memory read and one memory write operation will occur every cycle.

In Fig. 1, two types of address generators are used for the control of data accesses. The block q-gen can produce the sequential sequence order 0,1,2 . . . .. , while the block Hq–gen can produce the permutated sequence order $H(0), H(1), H(2), \ldots$. The q–gen module can be simply realized by a single adder; however, the implementation of Hq–gen is much more complicated. The definition of the permutation function $H(\cdot)$ according to [1] and [2] is generated by the following algorithm:

$$q = 0, \quad \text{for}\,(i = 0; i < M_{\max}; i = i + 1) \left\{ H(q) = (i \mod 2)2^{N_r - 1} + \sum_{j=0}^{N_r - 2} R_i(j)2^j; \quad \text{if}\,(H(q) < N_{\max})\, q = q + 1; \right\}. \qquad (4)$$

Here, $M_{\max}$ equals 2048, 4096, and 8192 for 2k, 4k, and 8k modes, respectively. In addition, N_r is equal to $\log_2 M_{\max}$. The Hq–gen module can be directly realized by a lookup table which contains all the possible precomputed $H(\cdot)$ values. However, the cost of this implementation will be very high because the table will have to contain 10 584 entries in total for different OFDM modes. A more efficient implementation is to generate the address online by the logic circuits, as shown in Fig. 2 and given in [1] and [2]. The term R_i shown in (4) can be represented by $\mathrm{WP}(R_i')$, where R_i' can be iteratively derived from $LSFR(R_{i-1}')$. Here, LSFR(.) denotes the function of a linear-shift-feedback-register (LSFR) operation used to generate a pseudorandom number sequence stored in register R'.The contents of register R' will be transformed by a wire-permutation function $\mathrm{WP}(\cdot)$ to another value R which will be appended with an additional signal produced by a toggle register T to form a candidate value for the next $H(\cdot)$ result. This candidate value can be as large as $M_{\max} - 1$ such that it will be discarded if it exceeds the bound of $N_{\max}$.

The main drawback of this online address-generator circuit is that it cannot guarantee that a valid next $H(\cdot)$ value will be produced each cycle. Therefore, once the generation of a valid $H(\cdot)$ is postponed, it will further affect the writing and reading of the deinterleaved data and cause the requirement of the additional input–output buffering circuits.

III. PROPOSED DESIGN OF SYMBOL DEINTERLEAVER

As described in the previous section, there are two main issues for the design of a symbol deinterleaver conforming to the DVB standard.

TABLE I
DATA STORED AT THE ADDRESS j FOR EACH MEMORY BANK

Bank			Data set	
number	name	size	even symbol	odd symbol
#1	EL	2048	$e_{2\times j}$	$o_{H^{-1}(2\times j)}$
#2	OL	2048	$e_{2\times j+1}$	$o_{H^{-1}(2\times j+1)}$
#3	EH	1024	$e_{2\times j+\frac{M_{max}}{2}}$	$o_{H^{-1}(2\times j+\frac{M_{max}}{2})}$
#4	OH	1024	$e_{2\times j+\frac{M_{max}}{2}+1}$	$o_{H^{-1}(2\times j+\frac{M_{max}}{2}+1)}$

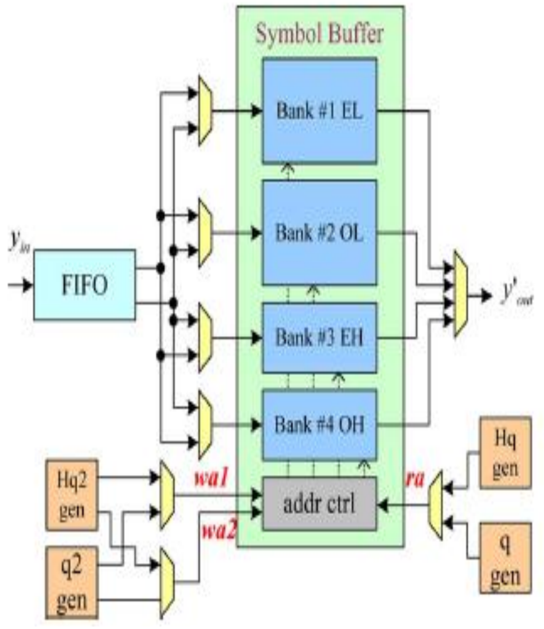


Fig. 3. Overall architecture of the proposed symbol-deinterleaver design.

The first one is the design of a symbol buffer, and the other one is the design of the Hq–gen module. In the following, our proposed design methods for these two modules will be described in detail. Fig. 3 shows the overall architecture of the proposed DVB symbol deinterleaver.

A. Design of Symbol Buffer

The symbol buffer is used to store the incoming symbol of data in order to generate the deinterleaved output sequence afterward. As shown in Fig. 1, two separate buffers can be used, and each buffer is realized by a single-port SRAM block. The other approach is to use only a single buffer, but the buffer has to be realized by a dual-port SRAM block. This paper proposes a more efficient design of the buffer by using multibank single-port SRAM blocks.

As shown in the proposed DVB symbol deinterleaver in Fig. 3, the proposed symbol buffer consists of four single-port SRAM blocks. The size of the first two memory banks is 2048 words, while the size of the last two is 1024 words. These four banks are labeled by EL, OL, EH, and OH because they are responsible for the storage of even-low, odd-low, even-high, and odd-high indexed data in a symbol. Table I lists the detailed data-storage method of each bank. The proposed data distribution over the banked memory blocks can help reduce the possibility that the buffer data read and write operations occur on the same memory bank. For example, when buffering the input even symbol, the data will be written to the memory in a sequential order. Therefore, for the first $(M_{\max}/2)$ data, the memory banks EL and OL will be accessed alternately, while for the remaining data, the memory banks EH and OH will be accessed alternately. Since the previous odd symbol data stored in the buffer will be fetched out while buffering the input even symbol, the memory read and write operations can just take place alternatively in two separate memory banks.

TABLE II
DETAILED OPERATIONS OF THE PROPOSED SYMBOL BUFFER FOR 8k MODE

	IN: even symbol, OUT: odd symbol					IN: odd symbol, OUT: even symbol										
Cycle	0	1	2	3	⋯	0	1	2	3	⋯	28	29	30	31	32	33
read	0	1	2	3	⋯	0	4096	128	4128	⋯	1712	216	4643	3204	4408	2147
bank	EL	OL	EL	OL	⋯	EL	EH	EL	EH	⋯	EL	EL	OH	EL	EH	OL
1st write		0	1	2	⋯		0	4096	128	⋯	4167		1712		216	3204
bank		EL	OL	EL	⋯		EL	EH	EL	⋯	OL		EL		EL	EL
2nd write					⋯					⋯					4643	4408
bank					⋯					⋯					OH	EH
data_in	y_0^e	y_1^e	y_2^e	y_3^e	⋯	y_0^o	y_1^o	y_2^o	y_3^o	⋯	y_{28}^o	y_{29}^o	y_{30}^o	y_{31}^o	y_{32}^o	y_{33}^o
data_out	y_0^o	y_6^o	y_{1624}^o	y_{4040}^o	⋯	y_0^e	y_{4096}^e	y_{128}^e	y_{4128}^e	⋯	y_{1712}^e	y_{216}^e	y_{4643}^e	y_{3204}^e	y_{4408}^e	y_{2147}^e
output	y'_0	y'_1	y'_2	y'_3	⋯	y'_0	y'_1	y'_2	y'_3	⋯	y'_{28}	y'_{29}	y'_{30}	y'_{31}	y'_{32}	y'_{33}

The buffering of the input odd symbol data will be more complex compared with that of the even symbol because they are written following the permutated order. The division of the memory banks according to the even and odd indexed data can work perfectly for buffering the even symbol; however, such ideal division does not exist for the odd symbol such that the chance of the memory read and write accesses to the same memory bank cannot be eliminated. In order to solve the bank conflict, our memory banks are also divided according to whether the index of the stored data is larger than $(M_{\max}/2)$ or not. This division can help in reducing the possibility of bank conflict because, according to the permutation function generation algorithm shown in Fig. 2, there will be no consecutive accesses to the memory location over $(M_{\max}/2)$ due to the operation of toggle bit. On the other hand, consecutive accesses to the lower memory location below $(M_{\max}/2)$ can indeed happen and result in a bank conflict. In such situation, we will give a higher priority to the memory read operation to avoid memory congestion. However, the input data which cannot be written to the memory due to the conflict have to be, in turn, temporarily stored into a first-in–first-output (FIFO) buffer, as shown in Fig. 3. To avoid too many data being accumulated in FIFO, multiple data in FIFO have to be written to the memory simultaneously if possible.

```
$q_R = 0; q_W = 0; S_{type} = 0;$
$for\ (t = 0;\ t < M_{\max};\ t++)\{$
  $if\ (S_{type} == 0)\{$// even symbol in;odd symbol out
    $ra = q_R; wa1 = q_w; wa2 = q_w + 1;\}$
  $else\{$// odd symbol in;even symbol out
    $ra = H(q_R); wa1 = H(q_w); wa2 = H(q_w + 1);\}$
  $if\ (in\_valid())\{$// new input $y_{in}$ arrives
   $y'_{out} = mem(ra); FIFO\_push(y_{in});\}$
  $if\ (bank(ra)\,!= bank(wa1))\{$// no bank conflict
    $if\ (!FIFO\_empty())\ \ mem(wa1) = FIFO\_pop();$
    $q_w++;\}$
  $elseif\ ((bank(ra)\,!= bank(wa2))\ \&\&(bank(wa1)\,!= bank(wa2)))\{$
    $if\ (!FIFO\_empty())\ \ mem(wa2) = FIFO\_pop();$
    $q_w++;\}$
  $q_R++;$
  $if\ ((q_R == N_{\max})\ \&\&(q_W == N_{\max}))\{$//end of symbol
      $q_R = 0; q_w = 0; S_{type} = 1 - S_{type}; t = 0;\}$
}
```

Fig. 4. Algorithm for the data read and write operations used in the proposed symbol-deinterleaver architecture.

In our proposed design, at most two sets of data will be written to those banks which are idle. The detailed operation can be represented by the pseudocode shown in Fig. 4. The function bank(i) returns the bank number which the address i belongs to. Table II illustrates the detailed operations for the first several clock cycles. According to our simulation result, the maximum number of data stored in FIFO for the deinterleaving operation of 2k, 4k, and 8k modes is 15, 12, and 31, respectively, assuming that the consecutive $N_{\max}$ input data enter the deinterleaver. Therefore, a FIFO of size 31 will be used in our symbol deinterleaver.

B. Design of Permutation Function Generator

As mentioned before, the main drawback of the permutation-function- generator circuit shown in Fig. 2 is that it cannot guarantee a valid address to be generated every cycle because the range of permutated pseudonumbers created may exceed the maximum address bound. If the valid address cannot be generated at the cycle when there is data entering the buffer, the data cannot be written into the buffer and has to be stored in some other temporary registers instead.

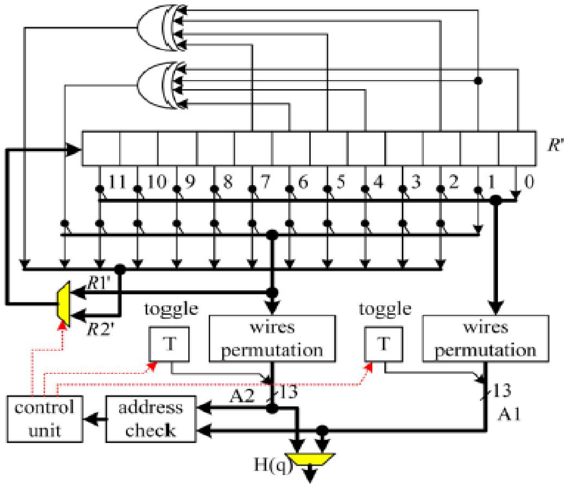
Similarly, the data fetch operation of the previous symbol in the buffer will also be affected by the invalid address cycle. The invalid address will occur 2144 times during the 8192 processing cycles for an 8k symbol, which will lead to a large extra temporary register overhead. Therefore, how to guarantee the timing production of valid permutation addresses each cycle is very important. The output generated by the circuit shown in Fig. 2 is not always valid because it will sometimes exceed the address bound. However, it can be verified from (4) that, if the current output is not valid, the next generated value will definitely be valid due to the operation of toggle bit **T**. Based on this observation, a look ahead permutation-address generation algorithm can be adopted, as shown in Fig. 5(a) and the corresponding circuit diagram in Fig. 5(b), where two candidate next addresses A1 and A2 are produced each cycle. If A1 is valid, it will be the next address output; otherwise, A2 will be chosen. The proposed design methodology can be further extended for the implementation of the $Hq2\_gen$ module used in Fig. 3, where two valid addresses may be required in one cycle. Three candidate addresses will be generated based on the LSFR, and only two of them will be selected.

IV. EXPERIMENTAL RESULT

Table III compares the implementation cost of different architectures of the symbol deinterleaver using 0.18- m technology. The memory modules used are created by the Artisan memory generator. The first design is based on the double buffer with two single-port SRAM modules while the second one is based on a single buffer with one dual-port SRAM module of 6048 words. This table lists the core size and its equivalent gate count reported by electronic design automation tools. It can be found that the proposed design is the most area efficient and can save about 30% in both the chip area and gate count compared with the second best design. The critical path of the proposed design is about 1 ns longer than the other two, but these three can all run up to 100 MHz which is well above than the DVB specification. The results given in Table III is based on the hard-in scheme. If multiple-bit soft inputs are considered, the saving achieved by the proposed method will become of more impact. It should be noted that the first two designs adopt the same proposed Hq_gen design, as shown in Fig. 5. If the direct design of Fig. 2 is used, assuming that consecutive data enter the deinterleaver, an extra buffer of size up to 2144 will be required to hold the input due to the failure of the valid-address generation.

```
$q = 0;$
$for\ (i = 0;\ i < M_{max};\ i++)\{$
    $R1' = LSFR(R');\ R2' = LSFR(LSFR(R'));$
    $A1 = (i_{mod2} \cdot 2^{N_r-1} + \sum_{j=0}^{N_r-2} WP(R')_j \cdot 2^j;$
    $A2 = ((i+1)_{mod2} \cdot 2^{N_r-1} + \sum_{j=0}^{N_r-2} WP(R1')_j \cdot 2^j;$
    $if(A_1 < N_{max})\ \{H(q) = A1; R' = R1';\}$
    $else\ \{H(q) = A2; R' = R2'; i = i+1;\}$
    $q = q + 1;$
$\}$
```

(a)



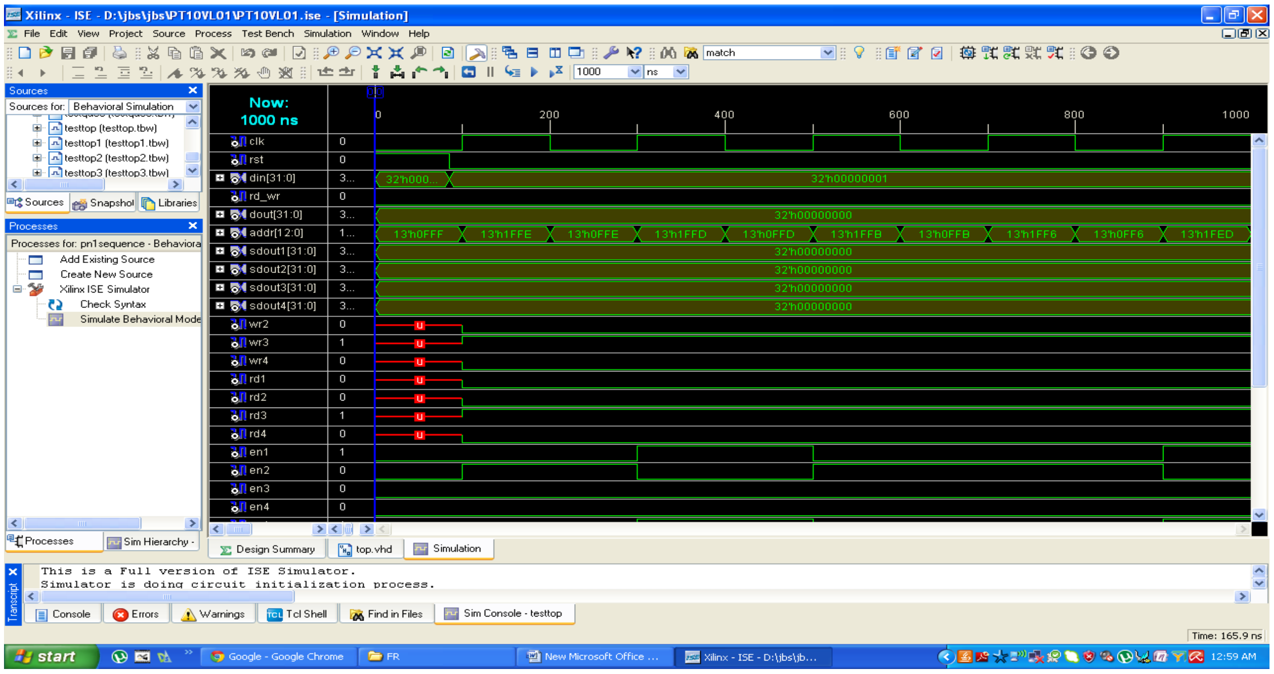
(b)

Fig. 5. Proposed design of the permutation function generator: (a) the proposed lookahead algorithm and (b) the detailed circuit diagram for 8k mode.

TABLE III
AREA COMPARISON FOR DIFFERENT SYMBOL-DEINTERLEAVER DESIGNS

Architecture	Double Buf. [4][6]	Single Buf.	Multi-bank (Prop.)
Core size	$0.95um^2$	$1.06um^2$	$0.66um^2$
Overall gate#	66.5K	74.3.6K	46.4K
Memory only	65.6.3K	73.4.7K	42.4K

V. SIMULATION RESULTS



```
This is a Full version of ISE Simulator.
Simulator is doing circuit initialization process.
```

Fig. 6. Simulation results of top level design

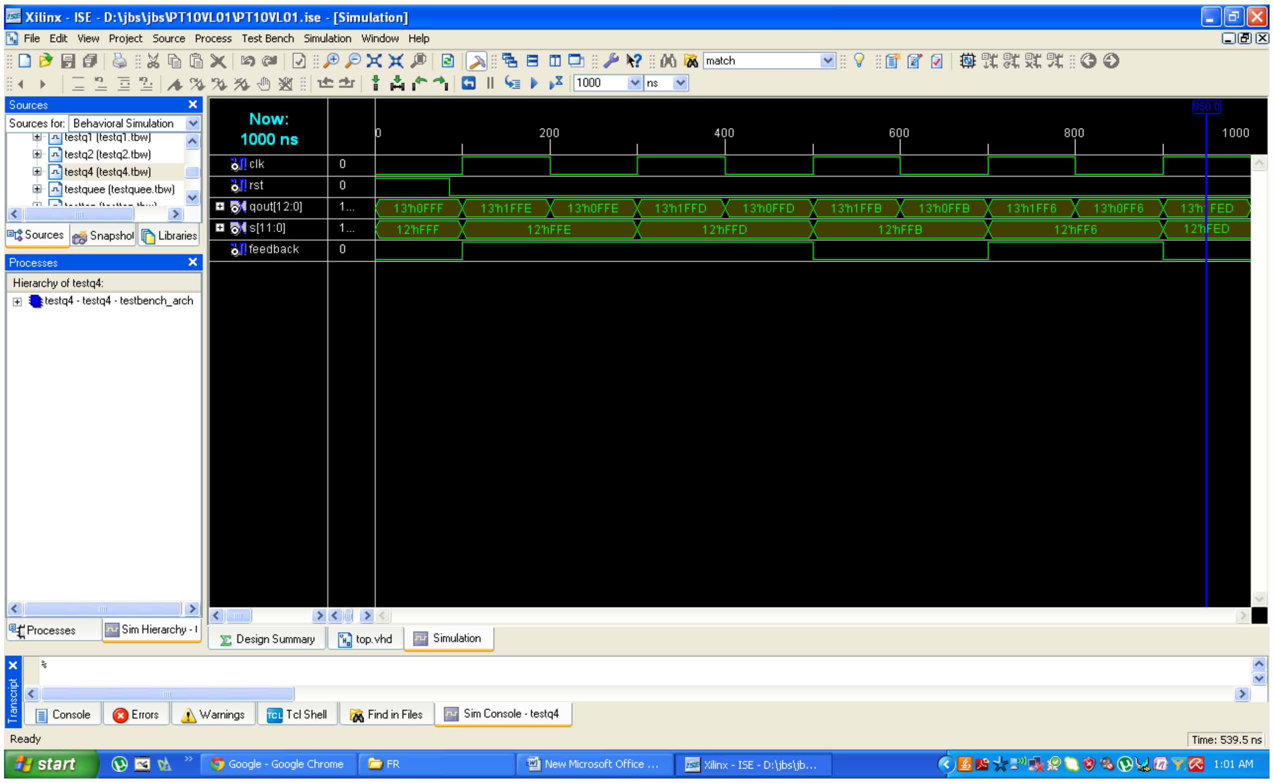


Fig. 7. simulation results of q

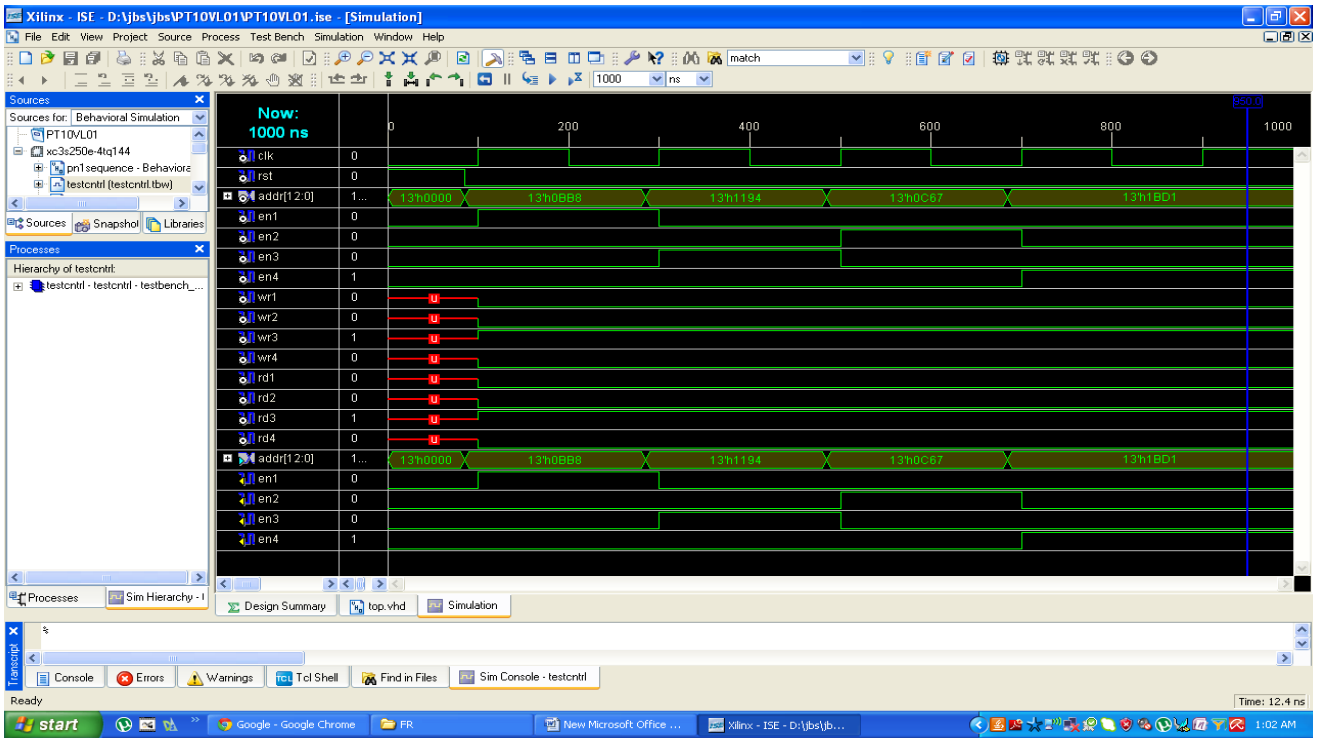


Fig.8. simulation results of control unit

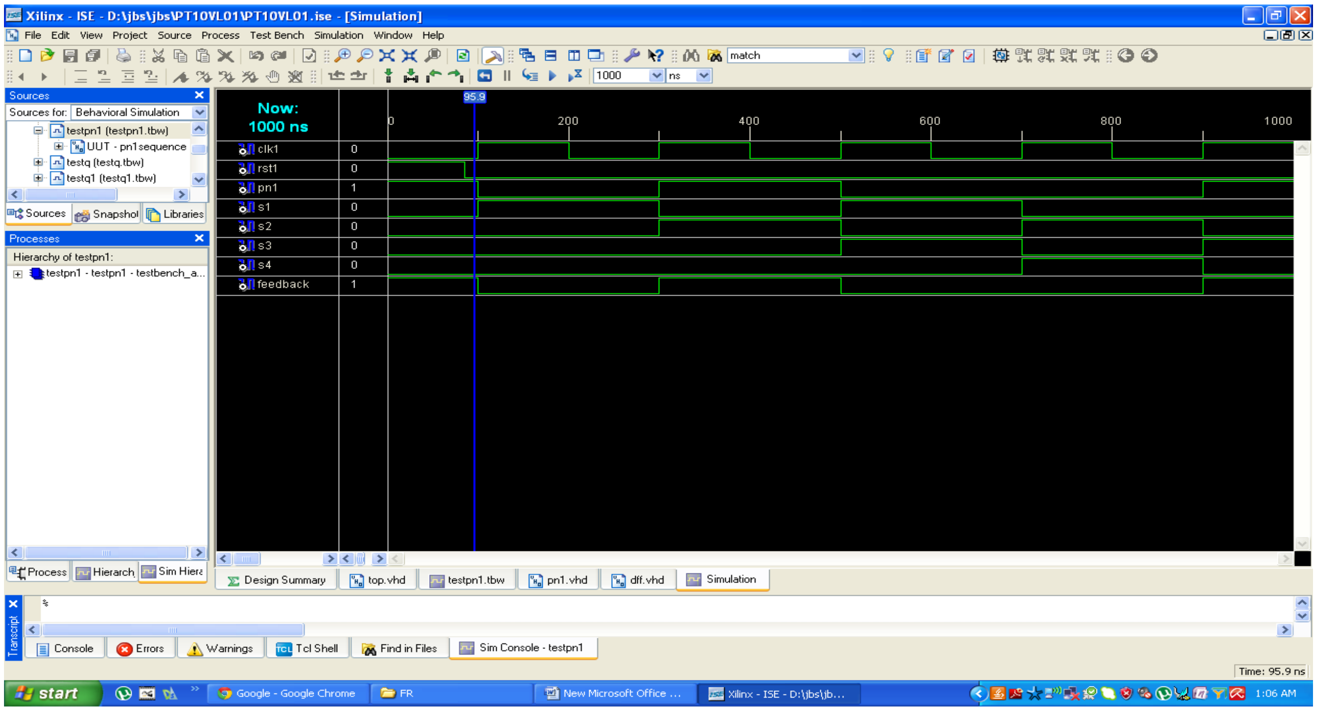


Fig.9. simulation results of Lfsr

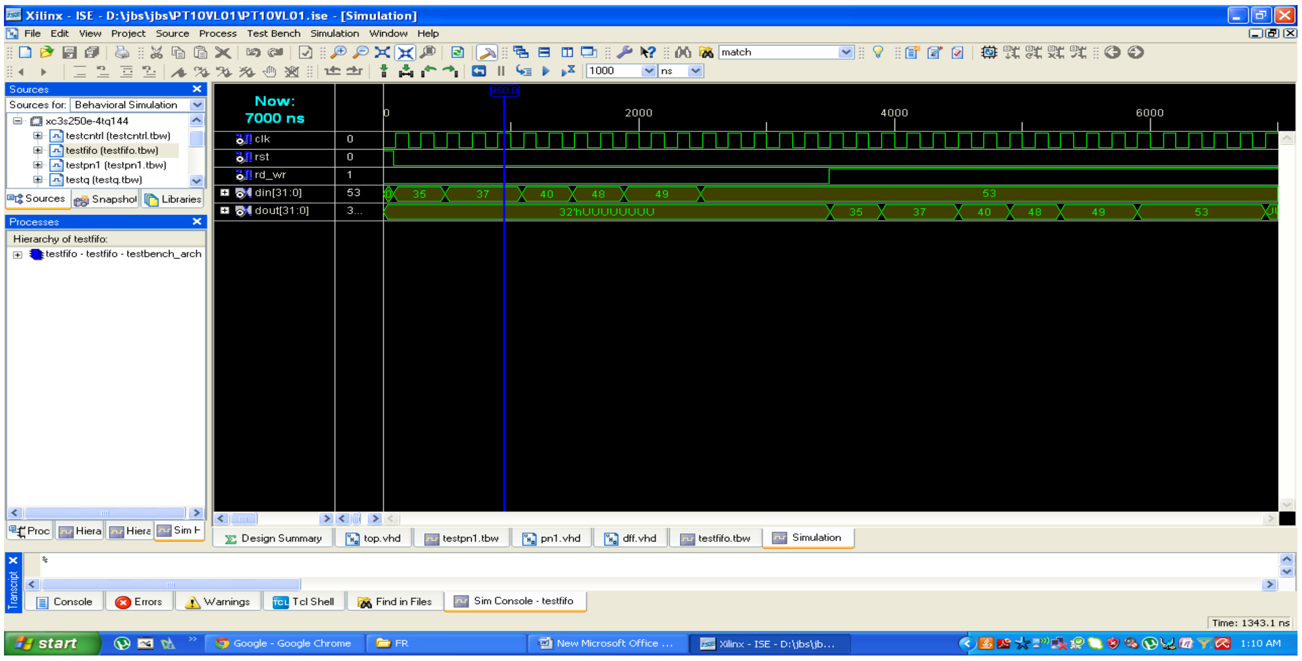


Fig.9. Simulation results of fifo

VI. CONCLUSION

This paper has presented an efficient VLSI design of the symbol deinterleaver based on a multibank single-port memory architecture. By the proposed data partitioning and access approach, the chance of memory conflict can be highly reduced such that only one additional FIFO of length 31 is required. About 30% savings of hardware cost can be achieved compared with the traditional approach. This paper also addressed the lookahead DVB permutation address generator which can supply a valid address per cycle to avoid the extra use of a temporary buffer.

VII. REFERENCES

[1] Digital Video Broadcasting (DVB): Framing Structure, Channel Coding and Modulation for Digital Terrestrial Television, EN 300 744 Vl.4.1., Jan. 2001, ETSI.
[2] Digital Video Broadcasting (DVB): Transmission System for Handheld Terminals (DVB-H), EN 302 304 VI.1.1., Nov. 2004, ETSI.
[3] C. Del Toso, P. Combelles, J. Galbrun, L. Lauer, P. Penard, P. Robertson, F. Scalise, P. Senn, and L. Soyer, "0.5m CMOS circuits for demodulation and decoding of an OFDM-based digital TV signal conforming to the European DVB-T standard," IEEE J. Solid-State Circuits, vol. 33, no. 11, pp. 1781–1792, Nov. 1998.
[4] L.-F. Chen and C.-Y. Lee, "Design of a DVB-T/H COFDM receiver for portable video applications," IEEE Commun. Mag., vol. 45, no. 8, pp. 112–120, Aug. 2007.
[5] Y.-N. Chang, "Design of an efficient memory-based DVB-T channel decoder," in Proc. IEEE Int. Symp. Circuits Syst., Kobe, Japan, May 2005, pp. 5019–5022.
[6] L. Horvath, I. Dhaou, H. Tenhunen, and J. Isoaho, "A novel highspeed reconfigurable demapper symbol deinterleaver architecture for DVB-T," in Proc. IEEE ISCAS, Orlando, FL, Jun. 1999, vol. 4, pp. 382–385.