Design and Implementation of Adaptive Turbo Encoder for Quantized Software Defined Low-Power DVB-RCS Radios

Sherif Welsen Shaker, *Member, IEEE* and Salwa Hussien Elramly, *Senior Member, IEEE*

Abstract — **Turbo codes are employed in every robust wireless digital communications system. Those codes have been adopted for the satellite return channel in DVB-RCS (Return Channel via Satellite) standard. In Software Defined Radios (SDRs), Field Programmable Gate Array technology (FPGA) is considered a highly configurable option for implementing many sophisticated signal processing tasks. The implementation for such codes is complex and dissipates a large amount of power. This paper studies the efficient implementation of quantized DVB-RCS turbo coding. Also, a low-power, turbo encoder for DVB-RCS is described using a VHDL code. The proposed encoder design is implemented on Xilinx Virtex-II Pro, XC2vpx30 FPGA chip. FPGA Advantage Pro package provided by Mentor Graphics is used for VHDL description and ISE 10.1 by Xilinx is used for synthetization.**

Keywords — **DVB, FPGA, Quantization, Software Defined Radios, Turbo codes, VHDL.**

I. Introduction

SDR is characterized by its flexibility so that modifying or replacing software programs can completely change its functionality. SDRs can reduce the cost of manufacturing and testing, while providing a quick and easy way to upgrade the product and take the advantage of new signal processing techniques and new wireless phone applications [1], [2]. In the early 1990's Field Programmable Gate Arrays (FPGAs) have become a considerable option in digital communication hardware where they were often applied as configurable logic cells to support memory controller tasks, complex state machines and bus interfacing [3]. Revolutionary changes have been made on FPGA technology in recent years. Complex real-time signal processing functions can yet be realized due to high clock speeds and huge gate densities provided by FPGA recent generations, like in Vertix-6 LXT FPGAs by Xilinx, which are optimized for high-performance logic and DSP with low power serial connectivity [4]. Many sophisticated signal processing tasks are performed in SDR that can be implemented on FPGA, including advanced compression algorithms, channel estimation, power control, forward error control, synchronization, and protocol management… etc [3].

The Digital Video Broadcasting (DVB) project was founded in 1993 by the European Telecommunications Standards Institute (ETSI) with the goal of standardizing digital television services. Its initial standard for satellite delivery of digital television, named DVB-S, used a concatenation of an outer (204,188) byte shortened Reed Solomon code and an inner constraint length 7, variable rate (r ranges from 1/2 to 7/8) convolutional code [5].
The same infrastructure used to deliver television via satellite can also be used to deliver Internet and data services to the subscriber. Internet over DVB-S is a natural competitor against cable modem and DSL technology, and its universal coverage allows even the most remote areas to be served. Because DVB-S only provides a downlink, an uplink is also needed to enable interactive applications such as web browsing. The uplink and downlink need not be symmetric, since many Internet services require a faster downlink.

One alternative for the uplink is to use a telephone modem, but this does not allow for always-on service, has modest data rates, and can be costly in remote areas. A more attractive alternative is for the subscriber equipment to transmit an uplink signal back to the satellite over the same antenna used for receiving the downlink signal. However, given the small antenna aperture and requirement for a low-cost, low-power amplifier, there is very little margin on the uplink. Therefore, strong FEC coding is desired. For this reason, the DVB Project has adopted turbo codes for the satellite return channel in its DVB-RCS (Return Channel via Satellite) standard [6]. At the same time that the DVB Project was developing turbo coding technology for the return channel, it was updating the downlink with modern coding technology. The latest standard, called DVB-S2, replaces the concatenated Reed-Solomon/convolutional coding approach of DVB-S with a concatenation of an outer BCH code and inner low density parity check (LDPC) code [7]. The result is a 30% increase in capacity over DVB-S. The outstanding coding performance of those codes requires the investigation of hardware implementation issues. For portable radio, low power consumption is a key implementation issue. Decoding algorithm simplification and quantization leads to reduction of power consumption in the radio receiver.

Sherif Welsen Shaker is a research scientist with the Communications and Signal Processing Research Lab., CSPRL, Faculty of Engineering, Ain-Shams University, Cairo, Egypt (phone: +20106444379; e-mail: welsen@ieee.org).

Salwa Hussien Elramly, is a professor and founder of CSPRL, Faculty of Engineering, Ain-Shams University, Cairo, Egypt; (e-mail: sramlye@netscape.net).

II. DVB-RCS

The DVB-RCS turbo code was optimized for short frame sizes and high data rates. Twelve frame sizes are supported ranging from 12 bytes to 216 bytes, including a 53 byte frame compatible with ATM and a 188 byte frame compatible with both MPEG-2 and the original DVB-S standard. The return link supports data rates from 144 kbps to 2 Mbps and is shared among terminals by using multi-frequency time-division multiple-access (MF-TDMA) and demand-assigned multiple-access (DAMA) techniques. Eight code rates are supported, ranging from $r = 1/3$ to $r = 6/7$.

Like the turbo codes used in other standards, a pair of constituent RSC encoders is used along with Log-MAP or Max-log-MAP decoding [8]. The decoder for each constituent code performs best if the encoder begins and ends in a known state, such as the all-zeros state. This can be accomplished by independently terminating the trellis of each encoder with a tail which forces the encoder back to the all-zeros state. However, for the small frame lengths supported by DVB-RCS, such a tail imposes a non-negligible reduction in code rate and is therefore undesirable. As an alternative to terminating the trellis of the code, DVB-RCS uses circular recursive systematic convolutional (CRSC) encoding [9], which is based on the concept of *tailbiting* [10]. CRSC codes do not use tails, but rather are encoded in such a way that the ending state matches the starting state.

Most turbo codes use binary encoders defined over GF(2). However, to facilitate faster decoding in hardware, the DVB-RCS code uses *duobinary* constituent encoders defined over GF(4) [11]. During each clock cycle, the encoder takes in two data bits and outputs two parity bits so that, when the systematic bits are included, the code rate is $r = 2/4$. In order to avoid parallel transitions in the code trellis, the memory of the encoder must exceed the number of input bits, and so DVB-RCS uses constituent encoders with memory three (a constraint length of four).

There are several benefits in using duobinary encoders. First, the trellis contains half as many states as a binary code of identical constraint length (but the same number of edges) and therefore needs half as much memory and the decoding hardware can be clocked at half the rate as a binary code. Second, the duobinary code can be decoded with the suboptimal but efficient Max-log-MAP algorithm at a cost of only about 0.1-0.2 dB relative to the optimal log-MAP algorithm. This is in contrast with binary codes, which lose about 0.3-0.4 dB when decoded with the max-log-MAP algorithm [12]. Additionally, duobinary codes are less impacted by the uncertainty of the starting and ending states when using tailbiting and perform better than their binary counterparts when punctured to higher rates.

III. DVB-RCS Turbo Code Efficient Implementation Issues

The most efficient hardware implementation of the DVB-RCS turbo code means to reach the best performance in terms of speed, area and low power consumption without loss of error correction capability. The most efficient implementation is always the tradeoff between hardware complexity and decoding ability. In order to achieve the expected performance, simplification must be attempted at difference abstraction levels. In this paper, application knowledge is exploited to significantly simplify high-level design towards lower implementation complexity.

A. Simplifying the Decoding Algorithm

The implementation of the MAP algorithm is difficult despite having the best performance. The implementation complexity of MAP decoding is due to the numerical representation of probability, non-linear functions, multiplications and additions. Converting the MAP into Log-MAP and substituting the logarithms by Jacobian logarithms avoids the numerical problems of the MAP decoding algorithm, while the performance of the Log-MAP keeps equivalent to MAP [8]. By avoiding the correction term of the Jacobian logarithms, the Max-Log-MAP will perform similar to Log-MAP for DVB-RCS [12].

B. Decoder Quantization

All decoding algorithms including Max-Log-MAP are usually specified in the floating-point domain. To get an efficient implementation, fixed-point number representation has to be used, which implies transformation from floating-point to fixed-point. The primary goal of this quantization for hardware implementation is to find a fixed-point model that has all bit-widths as small as possible under the condition of an acceptable degradation of the coding performance. In hardware implementation, the reduction of data-path bit-widths, control complexity and memory size leads to a reduction of area and power consumption and an increase of speed. The input data quantization as well as the inner data quantization have major influence on the control complexity and directly determine the bit-width of data-path and memory size. The smaller the bit-widths of quantization, the better the performance of the decoder in terms of speed, area and the power consumption. The quantization also has its effects on the decoding performance and thus, the optimized quantization is a key issue for the implementation complexity.

IV. DVB-RCS Turbo encoder

The block diagram of the turbo encoder that is used by DVB-RCS is shown in Fig. 1. The basic building blocks of the encoder are the following:

A. Recursive Systematic Convolutional Encoder

The CRSC constituent encoder used by DVB-RCS is shown in Fig. 2. The encoder is fed blocks of k message bits which are grouped into $N = k/2$ *couples*. The number of couples per block can be $N \in \{48, 64, 212, 220, 228, 424, 432, 440, 752, 848, 856, 864\}$. The number of bytes per block is $N/4$. In Fig. 2, A represents the first bit of the couple, and B represents the second bit. The two parity bits are denoted W and Y. For ease of exposition, subscripts are left off the figure, but below a single subscript is used to denote the time index $k \in \{0, ..., N-1\}$ and an optional second index is used on the parity bits *W* and *Y* to indicate which of the two constituent encoders produced them.

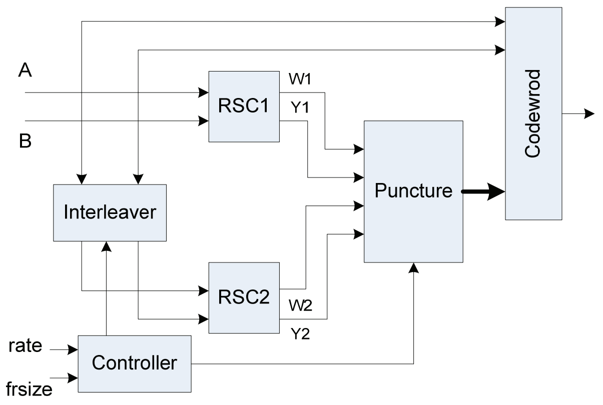

Fig. 1. Block diagram of DVB-RCS Turbo encoder.

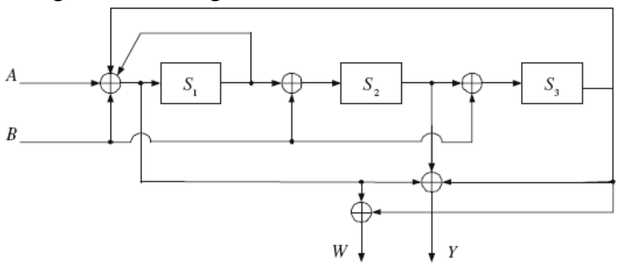

Fig. 2. Duobinary CRSC constituent encoder used by DVB-RCS.

Because of the tailbiting nature of the code, the block must be encoded twice by each constituent encoder. During the first pass at encoding, the encoder is initialized to the all-zeros state, $\mathbf{S}_0$ = [0 0 0]. After the block is encoded, the final state of the encoder $\mathbf{S}_N$ is used to derive the circulation state. The circulation state $\mathbf{S}_c$ is given by:

$$\mathbf{S}_c = (\mathbf{I} + \mathbf{G}^N)^{-1}\mathbf{S}_N \tag{1}$$

where

$$G = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{2}$$

In practice, the circulation state $\mathbf{S}_c$ can be found from $\mathbf{S}_N$ by using a lookup table [6]. Once the circulation state is found, the data is encoded again. This time, the encoder is set to start in state $\mathbf{S}_c$ and will be guaranteed to also end in state $\mathbf{S}_c$.

B. Turbo Code Permutation (Interleaver)

The first encoder operates on the data in its natural order, yielding parity couples $\{W_{k,1}, Y_{k,1}\}$. The second encoder operates on the data after it has been interleaved. Interleaving is performed on two levels. First, interleaving is performed within the couples, and second, interleaving is performed between couples. Let $\{A'_k, B'_k\}$ denote the sequence after the first level of interleaving and $\{A''_k, B''_k\}$ denote the sequence after the second level of interleaving. In the first level of interleaving, every other couple is reversed in order, i.e. $(A'_k, B'_k) = (B_k, A_k)$ if k is even, otherwise $(A'_k, B'_k) = (A_k, B_k)$. In the second level of interleaving, couples are permuted in a pseudorandom fashion. The exact details of the second level permutation are as follows [6]:

```
Set the permutation parameters $P0$, $P1$, $P2$ and $P3$
For $j$ = 0, ..., N-1
    – if $j$ mod. 4 = 0, then $P = 0$;
    – if $j$ mod. 4 = 1, then $P = N/2 + P1$;
    – if $j$ mod. 4 = 2, then $P = P2$;
    – if $j$ mod. 4 = 3, then $P = N/2 + P3$.
$i = P0 \times j + P + 1$ mod. $N$
```

Table 1 provides the combinations of the default parameters to be used. The interleaving relations satisfy the odd/even rule (i.e. when j is even, i is odd and vice-versa) that enables the puncturing patterns to be identical for both encodings.

TABLE 1: TURBO CODE PERMUTATION PARAMETERS.

Frame size in couples	*P*0	{*P*1, *P*2, *P*3}
N = 48 (12 bytes)	11	{24,0,24}
N = 64 (16 bytes)	7	{34,32,2}
N = 212 (53 bytes)	13	{106,108,2}
N = 220 (55 bytes)	23	{112,4,116}
N = 228 (57 bytes)	17	{116,72,188}
N = 424 (106 bytes)	11	{6,8,2}
N = 432 (108 bytes)	13	{0,4,8}
N = 440 (110 bytes)	13	{10,4,2}
N = 848 (212 bytes)	19	{2,16,6}
N = 856 (214 bytes)	19	{428,224,652}
N = 864 (216 bytes)	19	{2,16,6}
N = 752 (188 bytes)	19	{376,224,600}

After the two levels of interleaving, the second encoder (which is identical to the first) encodes the sequence $\{A''_k, B''_k\}$ to produce the sequence of parity couples $\{W_{k,2}, Y_{k,2}\}$. As with the first encoder, two passes of encoding must be performed, and the second encoder will have its own independent circulation state.

C. Rates and Puncturing block

To create a rate r = 1/3 turbo code, a codeword is formed by first transmitting all the un-interleaved data couples $\{A_k, B_k\}$, then transmitting $\{Y_{k,1}, Y_{k,2}\}$ and finally transmitting $\{W_{k,1}, W_{k,2}\}$. The bits are transmitted using QPSK modulation, so there is a one-to-one correspondence between couples and QPSK symbols. Alternatively, the code word can be transmitted by exchanging the parity and systematic bits, i.e. $\{Y_{k,1}, Y_{k,2}\}$, followed by $\{W_{k,1}, W_{k,2}\}$ and finally $\{A_k, B_k\}$.

Code rates higher than r = 1/3 are supported through the puncturing of parity bits. To achieve r = 2/5, both encoders maintain all the Y_k but delete odd-indexed $W_{k,}$. For rate 1/2 and above, the encoders delete all $W_{k,}$. For rate r = 1/2, all the Y_k bits are maintained, while for rate r = 2/3 only the even-indexed Y_k are maintained, and for rate r = 4/5 only every fourth Y_k is maintained. Rates r = 3/4 and 6/7 maintain every third and sixth Y_k respectively, but are only exact rates if N is a multiple of three.

V. Simulation and Results

A Matlab code is first driven to evaluate the performance of the DVB-RCS turbo coding. Fig. 3 shows the influence of the block size on the BER curve vs. E_b/N_o using AWGN channel for blocks of N = {48, 64, 212, 220, 228, 424, 752, and 864} message couples, or correspondingly {12, 16, 53, 55, 57, 106, 188, and 216} bytes. In each case, the code rate is r = 1/3, and ten iterations of Max-Log-MAP decoding are performed. Fig. 4 shows the influence of the code rate on the BER curve, results are shown for all seven code rates when the block size is N = 212 message couples, ten iterations of Max-Log-MAP decoding are performed.

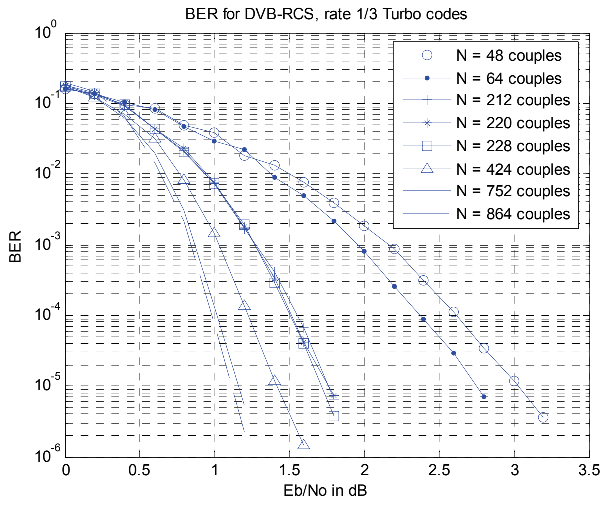

Fig. 3. Influence of block size on the BER performance of the DVB-RCS turbo code.

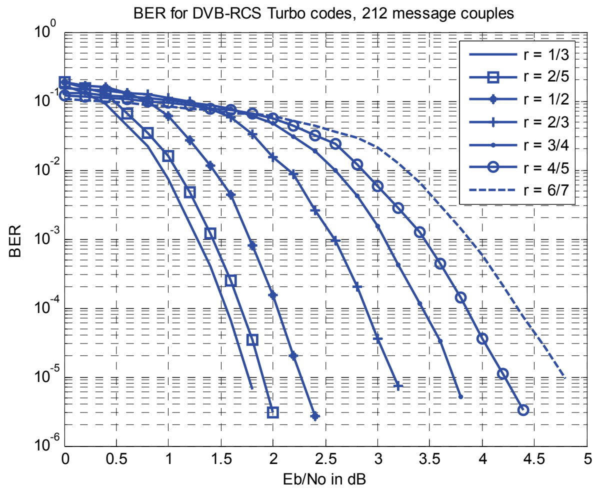

Fig. 4. Influence of code rate on the BER performance of the DVB-RCS turbo code.

The input Log Likelihood Ratios (LLRs) are fed to the decoding algorithm then multiplied by a scaling factor. The performance of the Max-Log-MAP algorithm with a constant correction factor under such scaling has been studied. The max* operator is executed twice, once for the forward sweep of the trellis and the other for the reverse, hence it constitutes a significant part of the decoder complexity [12]. On of several operations that estimate that term is given by:

$$\max^*(x,y) = \max(x,y) + f_c(|y\text{-}x|) \tag{3}$$

where x, y are the input LLRs and $f_c(|y\text{-}x|)$ is the correction factor. For the Max-Log-MAP with a constant correction factor, the max* operator can be approximated by:

$$\max^*(x,y) \approx \max(x,y) + \begin{cases} 0, if \; |y-x| > T \\ C, if \; |y-x| \le T \end{cases} \tag{4}$$

Varying the input LLR range as a function of SNR gives good results, but the best performance could be obtained when the code rate is also taken into account. For a given code rate, a smaller decoder input range affects the performance of the code, while a higher decoder input range affects the resolution and it is clear that there is a tradeoff between resolution and performance. So, to account for the performance of quantized decoder input codes for the variation in the parameters, it is important to have a quantized range that changes as a factor of SNR. To get that efficient range, we round the LLR values for a given code rate and then we made an estimate for linearly spaced values between the maximum and minimum quantized LLR values.

The performances of 48 block variable rate floating point codes were compared to the quantized version on those codes as shown in Fig. 5. As can be noticed from Fig. 5, there is a very little difference in the performance between the floating point codes and the quantized codes.

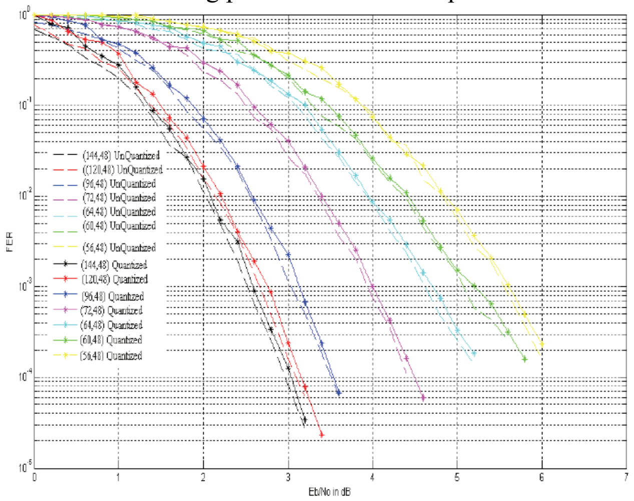

Fig. 5. Influence of decoder quantization on the FER performance of the DVB-RCS turbo code.

The proposed DVB-RCS turbo encoder is then described at the register transfer level with VHDL code, which considers the low power design by trying to reduce the switching activity of the encoder circuits. This can be done by switching off the second interleaver while the data is applied to the first. More power reduction can be

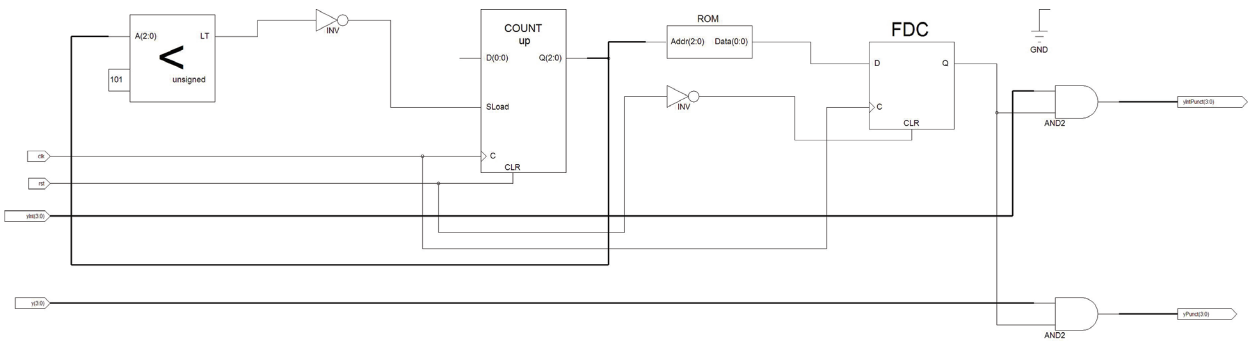

Fig. 8. RTL schematic of the puncture block.

achieved in the puncture block with code rates higher than 2/5 that W parity register is switched off because its contents do not have to be transmitted with the coded sequence

The simulation of the proposed encoder has been made using Modelsim SE 6.4b digital simulator to test the function of the different blocks for the implemented encoder. Fig. 6 shows the simulation waveforms for the puncture block when the code rate r = 1/3 while Fig. 7 shows the waveforms when r = 6/7.

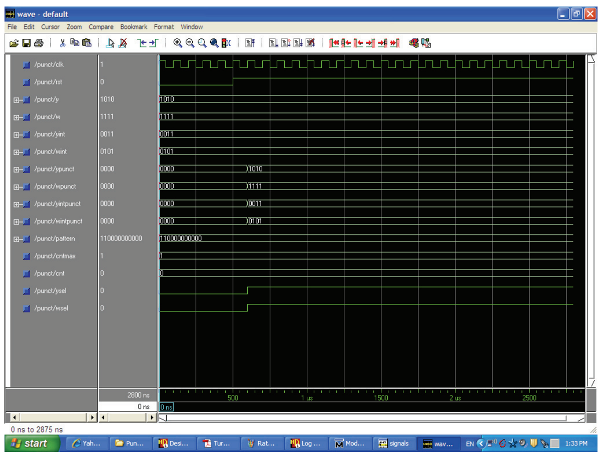

Fig. 6. Simulation of the puncture block, (rate 1/3).

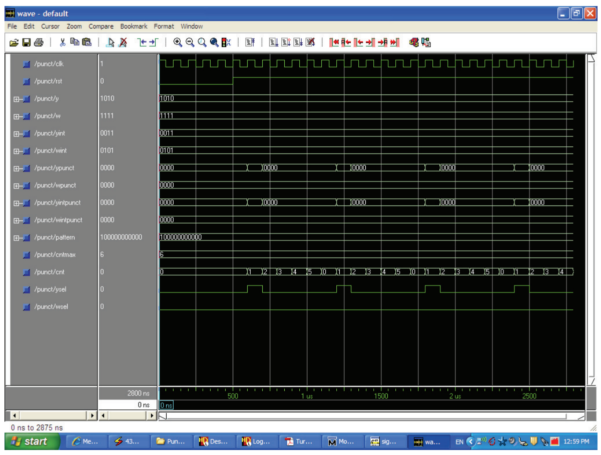

Fig. 7. Simulation of the puncture block, (rate 6/7).

Xilinx ISE 10.1 tools have been used for the synthesization process to map the design to the FPGA target technology. Xilinx Virtex-II Pro, xc2vpx30, with speed grade -6 has been selected; the design took about less than (3%) of the total chip resources, and the device utilization summary is shown in table 2. The results also showed that the implemented design can work with frequency up to 233.8 MHz. The RTL schematic diagram for the puncture is shown in Fig. 8.

TABLE 2: DEVICE UTILIZATION SUMMARY.

Selected Device	**xc2vp30-6fg676**	
Number of slices	373 out of 13696	3%
Number of slices Flip Flops	436 out of 27392	2%
Number of 4 input LUTs	367 out of 27392	2%

VI. CONCLUSION

In this paper, an efficient decoder quantization for DVB-RCS turbo coding has been made that reduces the power consumption while realizing DVB-RCS radios. Performance simulation for varying of different supported code rates for such codes has been presented, and results show that the quantized decoder essentially matches the performance of the floating point decoder. A design of low-power, turbo encoder for DVB-RCS has also been proposed. The design benefits from the concept of reducing the switching activity approach by means of register toggling for power reduction. The design has been described by VHDL using FPGA Advantage Pro by Mentor Graphics, simulated using Modelsim SE 6.4b, and then targeted on Xilinx Virtex-II Pro, XC2vp30 FPGA chip using ISE design suit 10.1 by Xilinx. The design took about 3% of the total chip logic elements. The maximum operating frequency is 233 MHz.

REFERENCES

[1] Joseph Mitola III, "Software Radios," *IEEE Communications Magazine*, volume: 33 no. 5, pp.24-25, May 1995.

[2] J. Kumagai, "Winner: Radio Revolutionaries," *IEEE Spectrum Magazine*, January 2007.

[3] M. Cummings and S. Huruyama, "FPGA in the Software Radio," *IEEE Communication Magazine*, volume: 37, no. 2, pp. 108-112, February 1999.

[4] Xilinx Inc., Virtex-6 SXT for DSP and memory-intensive applications with low-power serial connectivity, http://www.xilinx.com/products/v6s6.htm. Last visited: April. 2009.

[5] European Telecommunications Standards Institute. "Digital broadcasting system for television, sound, and data services." *ETS 200 421*, 1994.

[6] European Telecommunications Standards Institute. "Digital video broadcasting (DVB); interaction channel for satellite distribution systems;." *ETSI EN 301, 790 V1.5.1 (2009-05)*, 2009.

[7] European Telecommunications Standards Institute. ‘Digital video broadcasting (DVB) second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications.” *DRAFT EN 302 307 DVBS2-74r15*, 2003.

[8] P. Robertson, P. Hoeher, and E. Villebrun. “Optimal and sub-optimal maximum a posteriori algorithms suitable for turbo decoding.” *European Trans. On Telecommun.*, 8(2):119–125, Mar./Apr. 1997.

[9] C. Berrou, C. Douillard, and M. Jezequel. “Multiple parallel concatenation of circular recursive convolutional (CRSC) codes.” *Annals of Telecommunication*, 54(3-4):166–172, Mar.-Apr. 1999.

[10] H. H. Ma and J. K. Wolf. “On tail biting convolutional codes.” *IEEE Trans. Commun.*, 34:104–111, May 1986.

[11] C. Berrou and M. Jezequel. “Non binary convolutional codes for turbo coding.” *IEE Electronics Letters*, 35(1):39–40, Jan. 1999.

[12] M. C. Valenti and J. Sun. “The UMTS turbo code and an efficient decoder implementation suitable for software defined radios.” *Int. J. Wireless Info. Networks*, 8:203–216, Oct. 2001.