FPGA Based Implementation of Convolutional Encoder- Viterbi Decoder Using Multiple Booting Technique

Dr. Dhafir A. Alneema[1] **Yahya Taher Qassim**[2]
Lecturer **Assistant Lecturer**
Computer Engineering Dept. University of Mosul, Mosul, Iraq
[1]**E-Mail: dhafiraln@yahoo.com** [2]**E-Mail: yahyataherqassim@yahoo.com**

Abstract

Convolutional encoding is considered one of the forward error correction schemes. It is an essential component of wireless communication systems such as the third generation (3G) mobile systems, which utilize some formulation of Convolutional encoding usually decoded via Viterbi decoders. There are different structures of Convolutional encoding which impart different requirements on the decoder. The Viterbi decoder segments with slight modifications can be used on systems with different constraint lengths, frame size and code rates. In this research, the design and implementation of Convolutional encoder with constraint length 3 and rate 1/2, and Viterbi decoder on Spartan 3E FPGA Starter kit (supported with XC3S500E) using multiple booting technique has been presented. VHDL language is used as a design entry. In the starter kit mentioned above, two designs are implemented on the flash memory using the multiple booting technique: the Convolutional encoder and the Viterbi decoder. The FPGA is configured with the specified design depending on the loaded program from the Intel flash memory. With this way of configuration, the FPGA itself can operate as a Convolutional encoder or Viterbi decoder that gain benefit through the reuse of the same hardware.

Key words : Convolutional encoder, Viterbi decoder, multiple booting technique and FPGA.

تنفيذ مشفر لافوفي-حلال شفرة فيتربي باستخدام تقنية الإقلاع المتعدد على رقاقة FPGA

د. ظافر عبد الفتاح النعمة (مدرس) يحيى طاهر قاسم (مدرس مساعد)
قسم هندسة الحاسبات / كلية الهندسة / جامعة الموصل

الخلاصـــــة

يعتبر المشفر اللافوفي أحد طرق تصحيح الخطأ الناتج عن إرسال البيانات. وهو جزء ضروري لأنظمة الأتصالات اللاسلكية مثل الجيل الثالث لأجهزة الهاتف المحمول والتي تستخدم بعض الصيغ لهذا المشفر وبالتالي تستخدم حلال الشفرة نوع فيتربي لأستعادة البيانات المرسلة. هنالك هياكل مختلفة للمشفر اللافوفي والتي تنقل متطلبات مختلفة لفاتح الشفرة. بتحوير بسيط للمقاطع الخاصة لفاتح شفرة فيتربي يمكن أن تستخدم في أنظمة ذات معدل شفرة وحجم بيانات وأطوال إرسال مختلفة. هذا البحث يستعرض تصميم وتنفيذ مشفر لافوفي ذو طول محدد 3 ومعدل سرعة مشفر 1/2 ، مع حلال شفرة فيتربي على لوح حاوي لـــ مصفوفة البوابات القابلة للبرمجة حقليا نوع Spartan 3E (والمدعمة بـــالرقاقة XC3S500E) باستخدام تقنية الإقلاع المتعدد. استخدمت لغة VHDL كوسيلة للتصميم المنطقي. في اللوح المذكور أعلاه وباستخدام تقنية الإقلاع المتعدد، تم تنفيذ تصميمين باستخدام ذاكرة الوميض الموجودة على اللوح المذكور، أحد التصميمين هو المشفر اللافوفي والاخر حلال شفرة فيتربي. تم تشكيل التصميم المحدد على مصفوفة البوابات القابلة للبرمجة حقليا أعتمادا على البرنامج المحمل من ذاكرة الوميض. بطريقة التشكيل هذه، فإن مصفوفة البوابات نفسها يمكن أن تعمل كمشفر لافوفي أو حلال شفرة فيتربي بالإستفادة من اعادة استخدام نفس الكيان المادي.

Received : 21 – 4 – 2009 Excepted : 14 – 1 - 2010

1. Introduction

Generally, there are two schemes of channel coding: one of them is the block coding that does not require memory, and the second is Convolutional coding that requires memory. Convolutional coding is used for the channel coding in digital communication systems, because it facilitates a better error correction compared with the block coding although it requires an extensive memory in decoding [1]. Viterbi algorithm is one of the Convolutional coding algorithms that has been widely applied to decode and estimate the information in communications and signal processing units[2].

In previous works [3][4][5], numerous methods have been introduced for reducing complexity and power consumption employing trellis based decoders and increasing the speed for Viterbi decoder in 3GPP, DUB and wireless communications. These methods used techniques to simplify the receiver trellis in a fixed manner for a given complexity or modify trellis searching in some way based on the specific received values. Some of the researchers compared between the use of FPGA, ASIC and DSP to find which one is suitable for the application. Other researchers studied the different methods for back trace unit to find the correct path and other tried to work with high frequency by using parallel operations of decoder units. The complexity of these decoders increased with the increasing of the constraint length.

This research presents the design and implementation of Convolutional encoder and Viterbi decoder on Spartan 3E FPGA using multiple booting technique [6][7]. This technique allows reusing the same hardware for implementing the Convolutional encoder and the Viterbi decoder.

This paper is organized as follows: Section 2 reviews the background theory of Convolutional encoder and Viterbi decoder. Section 3 describes the multiple booting technique. The simulation and implementation of this design is demonstrated in Section 4. Finally, section 5 gives the conclusions.

2. Background Theory

2.1 Convolutional Encoder

Convolutional encoder consists of linear shift register and XOR gates; and is generally characterized in (n, k, m+1) format, where: n represents the number of outputs of the encoder; k is the number of inputs of the encoder, m is the number of memory elements (Flip- Flops) and m+1 represents the constraint length. The rate of this encoder can be represented as k/n. Figure (1) shows the Convolutional encoder with (2, 1, 3) format and the generating polynomials:

$G0(D) = 1 + D + D^2$
$G1(D) = 1 + D^2$

Where D represents the delay element. Polynomial selection is important because each polynomial has different correction properties. Selecting polynomials that provide the highest degree of orthogonality maximizes the probability of finding the correct sequence [8].

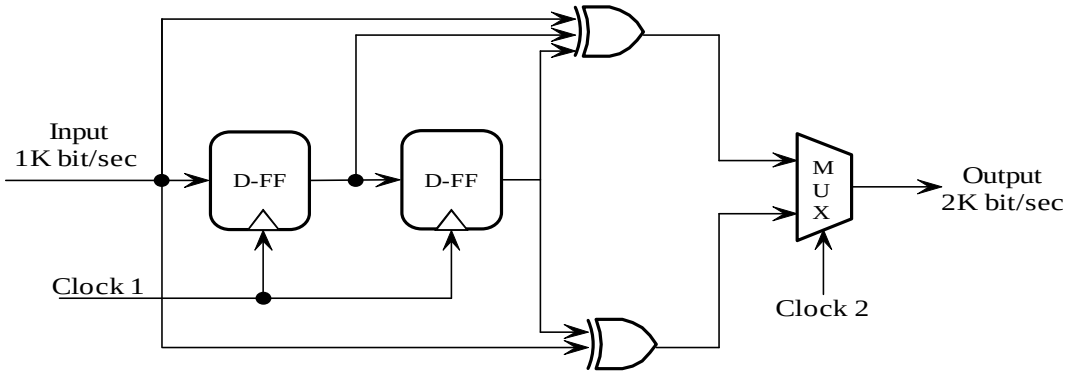


Figure (1) Non-Recursive, Non Systematic Convolutional Encoder (2, 1, 3)

2.2 Viterbi Decoder

The Viterbi decoding algorithm is a decoding process for Convolutional codes. This algorithm is based on maximum likelihood algorithm. When the received signal is sampled and quantized into two levels, either zero or one, the hard decision decoding will be used. In such decoding, the path through the trellis diagram is determined using hamming distance measure. Where the trellis represents the extension of the state diagram that explicitly demonstrates the state diagram with the time. The hamming distance is defined as a number of bits that are different between the observed symbol at the decoder and the sent symbol from the encoder [8].

The length of the trellis is equal to the length of the input sequence, which consists of the information bits followed by the reset sequence. The reset sequence forces the trellis into the initial state, so the trace back can be started at the initial state.

Figure (2) shows the simplified Viterbi decoder block diagram. The Viterbi decoding process consists of the following main tasks: branch metric computation, state metric update, survivor path recording and output decision generation. The branch metric computation compares the received code symbol with the expected code symbol using bit wise XOR and counts the number of different bits. Figure (3a) shows the implementation of the branch metric and figure (3b) shows the VHDL program steps that compute the branch metric.

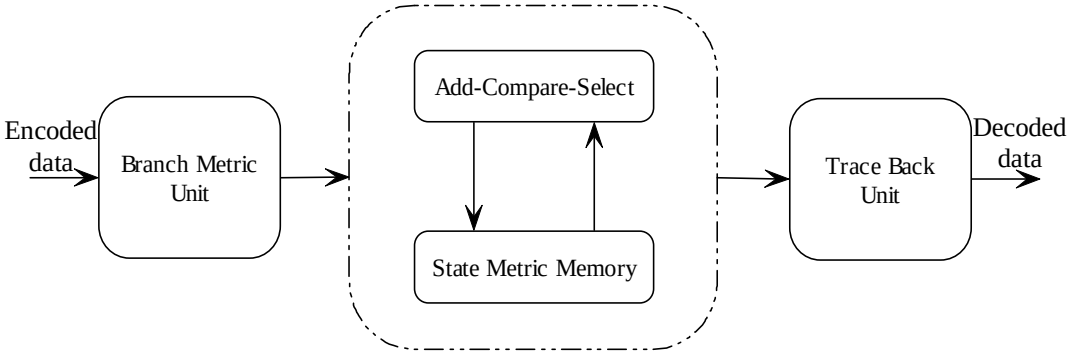
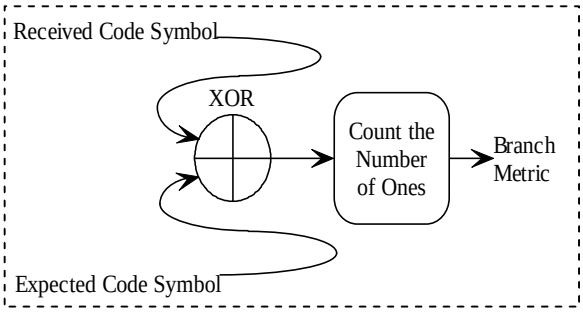


Figure (2) Simplified Viterbi Decoder Block Diagram

```
----------------------
       if (i <= no_states)  then

          if j="000" then
          br_met_temp0<=out_pat0(i) xor (rx(t)&rx(t+1));
          br_met_temp1<=out_pat1(i) xor (rx(t)&rx(t+1));
             j<=j+"001";
          elsif j<="010" then
             if flg1='0' and ii<2 then
                if br_met_temp0(ii)='1' then
                   br_cnt0 <= br_cnt0+'1';
                end if;
                if br_met_temp1(ii)='1' then
                   br_cnt1 <= br_cnt1+'1';
                end if;
                ii<=ii+1;
                if ii=1 then
                flg1<='1';
                end if;
             end if;
             j<=j+"001";
          else
          br_met0(i)<=br_cnt0;
          br_met1(i)<=br_cnt1;
          br_cnt0 <="00";
          br_cnt1 <="00";
          j<="000";
          ii<=0;
          flg1<='0';
             if (i < no_states)  then
              i<=i+1;
             else
              i<=0;
              t<=t+2;
             end if;--i < no_states
          end if;--j
       end if;--i
----------------------
```

(b)



(a)

Figure (3) : (a) Branch Metric Computation Block (b) The VHDL Program for

State metric update is achieved by computing the partial path metrics through accumulating the new values of branch metric to the previous values of state metric. There are two values of state metric due to the two branches enter each node in the trellis. So, the updated value of state metric at each node will take the minimum value. This can be done by using Add Compare Unit as shown in figure (4).

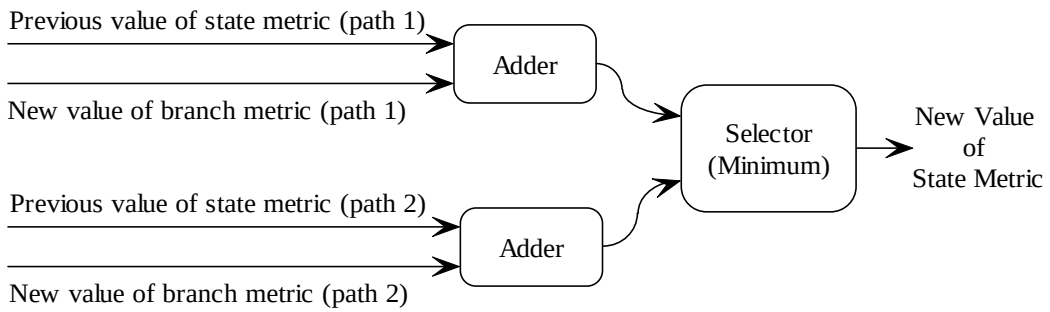


Figure (4) The Add Compare Unit

After updating the state metric for each node in the trellis, the survivor path can be identified by passing through the minimum value of state metric. Then, the partial path metrics will be updated and survivor information will be stored.

The survivor management unit is responsible for keeping track of the information bits associated with the surviving paths. The trace back approach records the survivor branch of each state. A flip-flop is assigned to each state to record '0' or '1' according to survivor path. Concatenating the decoded output bits in reversed order of time, the decoded output sequence will be formed [8].

3. Multiple Booting Technique

The Spartan-3E Starter Kit board [9] supports a variety of FPGA configuration options. One of these options is programming the on-board 128Mbit Intel Strata Flash (parallel NOR Flash PROM), then configuring the FPGA from the image stored in this Flash PROM using BPI Up or BPI Down configuration modes. Moreover, an FPGA can be dynamically loaded with two different FPGA configurations using the Spartan-3E FPGA's Multi Boot mode. Figure (5) demonstrates the Multiple Booting Technique for two different applications.

The multiple booting requires some steps that are different from those in the case of a single design implementation. In brevity, these steps require building more than one VHDL designs that include an additional selector input bits in the entity of each VHDL design program. The additional input bits are constrained for some of the slide switches in the Spartan 3E FPGA kit through the user constraint file (UCF). This gives the advantage for selecting the design configuration on the FPGA; for example: if n is the number of selection inputs, then 2^n will be the number of the total different designs that can be configured on the FPGA through the Intel Strata Flash.

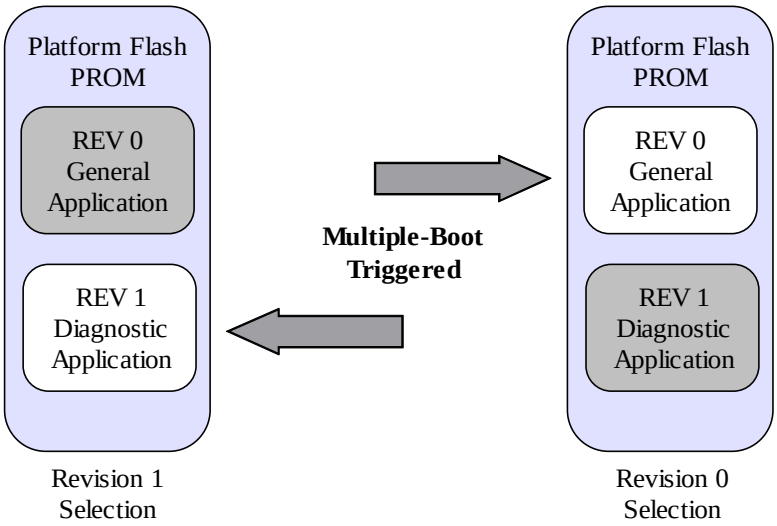


Figure (5) the Multiple Booting Technique

In this research, two different designs were considered on the mentioned flash and then applied on the FPGA separately to be either Convolutional encoder or Viterbi decoder,

so one input bit is needed for the switching between these two configurations. This type of configuration is different from the traditional method in the programming file type and the steps of programming. The PROM is retained with the design after power- off, then the FPGA will load the design from the PROM after power- on by pressing the programming pulse switch on the kit.

Additionally, before final implementation; two bit files generated from the two designs must be merged in one MCS file which is downloaded on the parallel NOR flash that contains the two different designs configurations. The resulting MCS file should be send to the FPGA kit through the RS232 serial link. The Hyper Terminal program is adequate for this task and available on most PCs. A new Hyper Terminal session can be started and configured in some steps, i.e., the communication settings and protocol required by an alternative terminal utility[6]. The Hyper Terminal file has some commands that help for erasing, reading and programming the Intel Strata Flash. Figure (6) illustrates a list of these simple commands and explains the execution of the program command (P), where entering the P command is followed by sending the MCS file from the transfer menu.

An MCS file contains additional information to define the storage address which PicoBlaze interprets as well as obtaining the configuration data [7]. The first few lines of the MCS file defining an FPGA BPI-UP configuration from NOR FLASH will be associated with address zero (000000) and each line contains 16 data bytes to be stored in sequential locations.

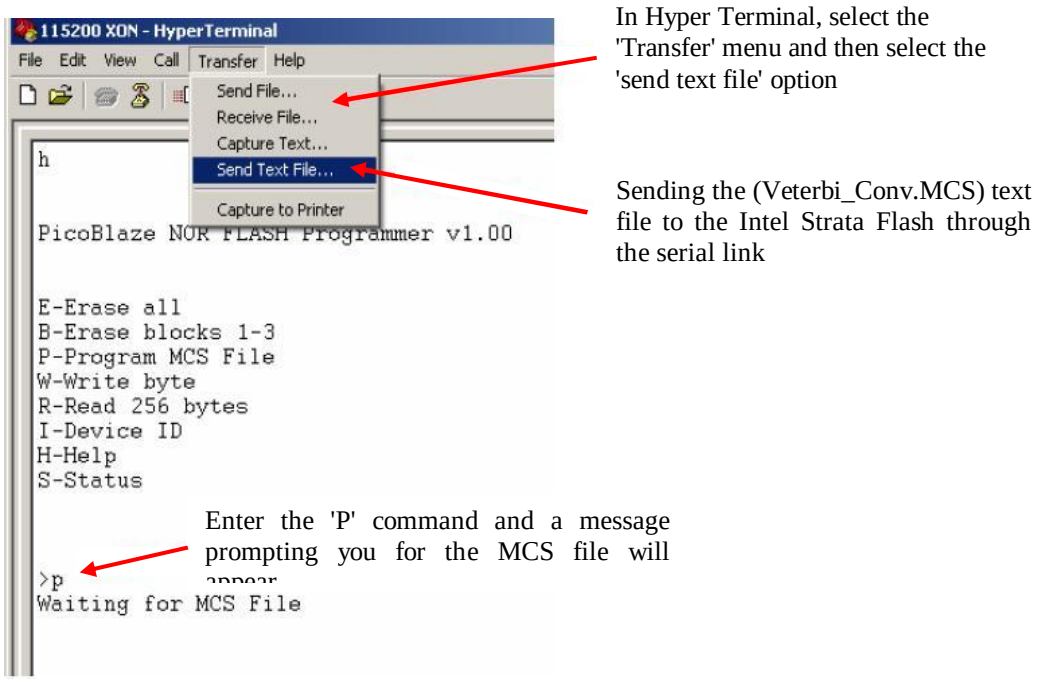


Figure (6) The Execution of The P Command in The Hyper Terminal Program

The final design implementation step on the FPGA is shown in figure (7) which also shows the programming window that gives an indication about the programming situation. Generating the required MCS file is done through the ISE (Integrated Software Environment) tools in some special guided software steps. Figure (8) shows the final step of generating such file.

The advantages of this configuration are lower board cost due to the reduced number of interface lines needed, faster configuration and no third device required for reconfiguration [7].

4. Design Simulation and Implementation

Timing Results

This research presents the implementation of Convolutional encoder- Viterbi decoder architecture on XC3S500E chip [10], using Spartan-3E FPGA Starter Kit, Project Navigator 8.2i Software (for modeling, verification and implementation) and Model Sim XE II/ Starter 5.8c Software which is used for timing simulation.

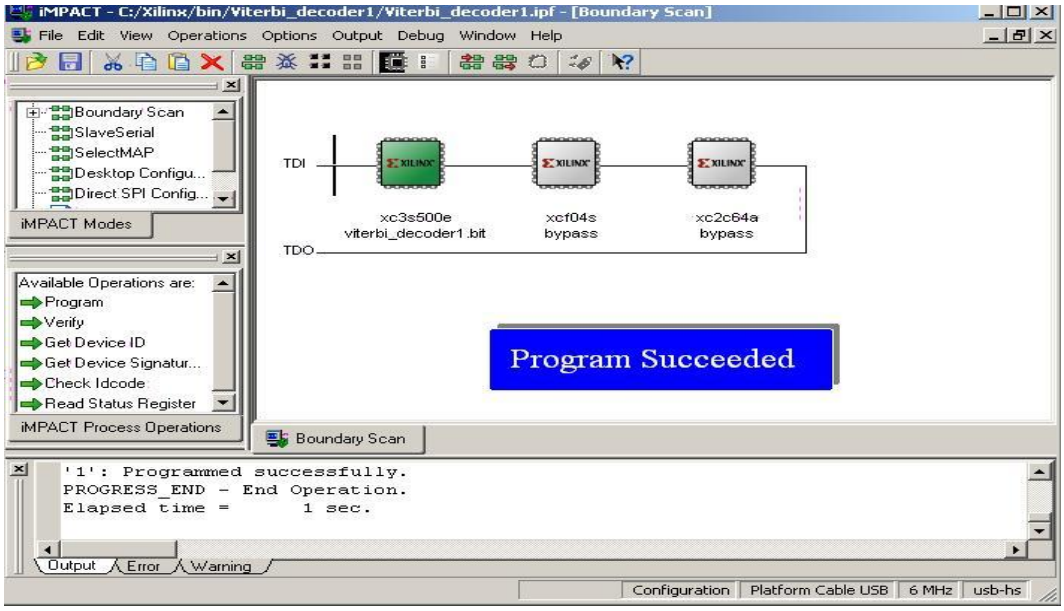


Figure (7) The Situation of The Programming Window

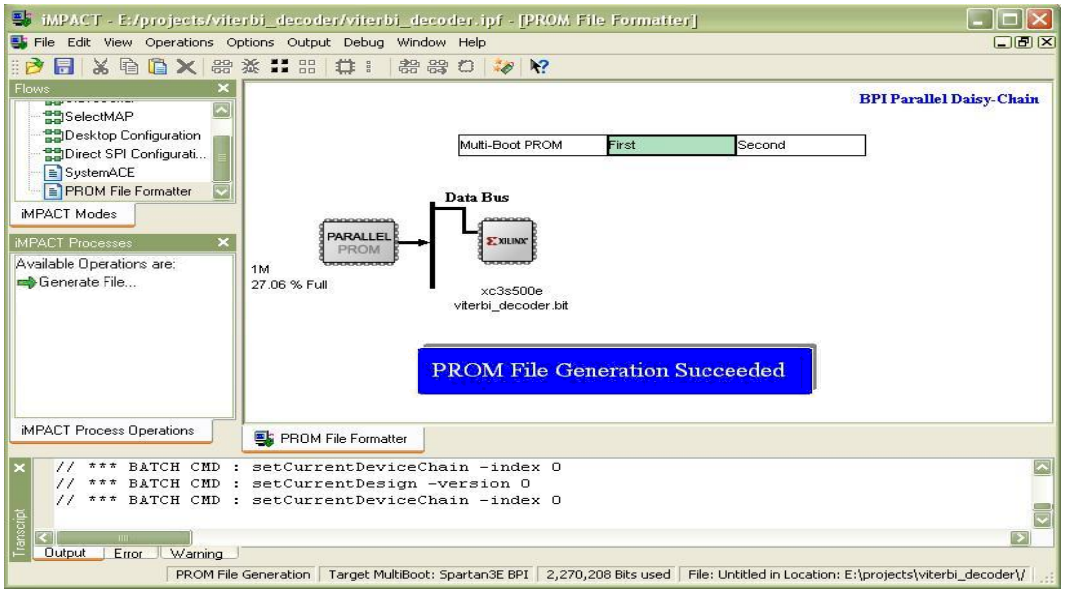


Figure (8) The Final Step in The Generation of MCS File

Each design is simulated separately and then implemented by applying the multi- boot technique. The generation of MCS file was necessary to combine the two designs. Then this file has been transferred to the Spartan-3E FPGA kit and stored in the PROM through the serial link using the Hyper Terminal windows program.

Figures (9,10) illustrate the timing simulation of the two designs. Figure(9) demonstrates all the signals of Convolutional encoder parts. The input data and the outputs of each flip- flop, generating polynomials and the encoded data were presented. The encoded data were produced by picking the polynomial values for each input sample; consequently, the output frequency is doubled. Figure (10) shows all the main signals of Viterbi decoder parts. The signals Out_pat0 and Out_pat1 represent the patterns of the state transitions of the system when the input is zero or one, respectively. The branch metrics (br_met0 and br_met1) for zero and one, respectively and the accumulated error metric (Acc_err_met) of each state are also shown in this figure. Bit_cnt, in the same figure, represents the bit counter whereas State_cnt represents the state counter. In addition to all these signals, the figure illustrates the origin_data which is Convolutionally encoded producing the transmitted_data. Then, after passing the channel, the received_data that contains the errors will be passed into the Viterbi decoder. This decoder detects and removes these errors and retrieve the original data. All the results of Convolutional encoder and Viterbi decoder were verified with those produced by Matlab.

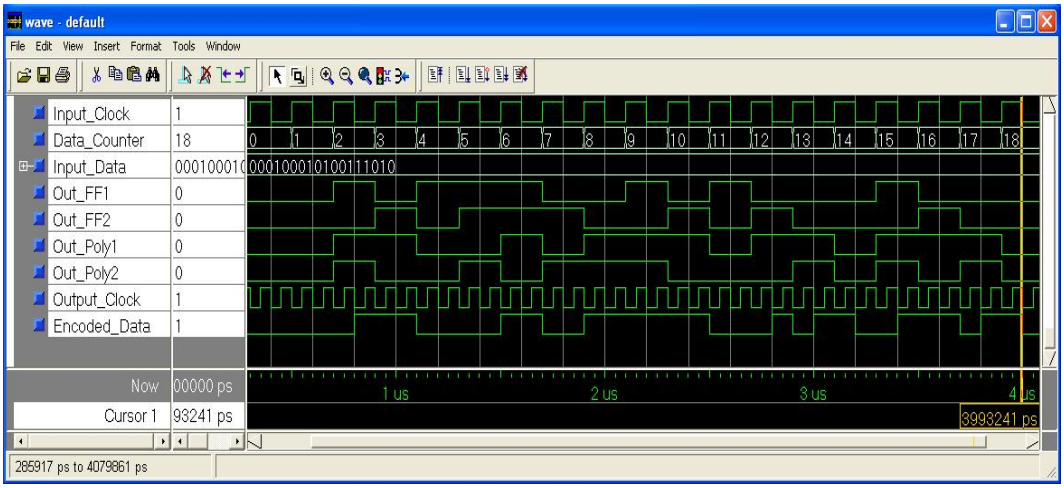


Figure (9) Timing Simulation of Convolution Encoder

4.2 FPGA Synthesis and Test

The utilization summary of Convolutional encoder and Viterbi decoder architectures are illustrated in table (1). The Viterbi decoder occupied more area than Convolutional encoder due to the complexity increasing of this design. The maximum operating frequency (last row in this table) depends on the target technology of the design. In the used Spartan 3E, the maximum operating frequency is equal to the clock frequency provided by the kit, that is 50 MHz.

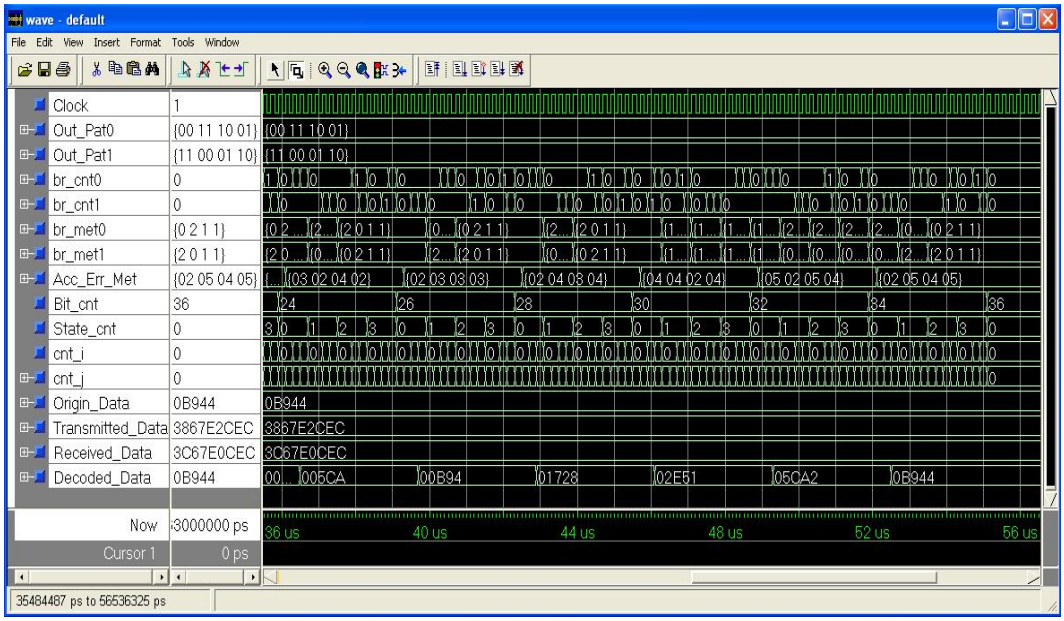


Figure (10) Timing Simulation of Viterbi Decoder

After the final implementation process on the FPGA, the bit files of the Convolutional encoder and Viterbi decoder are merged in one MCS file and downloaded through the serial link into the Intel Strata Flash memory (TE28F128) that located on the Spartan-3E FPGA kit.

By changing the multi- boot switch option, the FPGA can be configured to be as Convolutional encoder or Viterbi decoder as shown in figure (11-a and 11-b) respectively.

Table (1) The Convolution Encoder and Viterbi Decoder Synthesis Report Summary

Selected Device (3S500E FG320-5)	Convolution Encoder	Viterbi Decoder
Number of Slices	1 out of 4656 0%	383 out of 4656 8%
Number of Slice Flip Flops	2 out of 9312 0%	181 out of 9312 1%
Number of 4 input LUTs	1 out of 9312 0%	732 out of 9312 7%
Number of IOBs	5 out of 232 2%	19 out of 232 8%
Number of GCLKs	1 out of 24 4%	1 out of 24 4%
Maximum Frequency	654.686MHz	84.600MHz

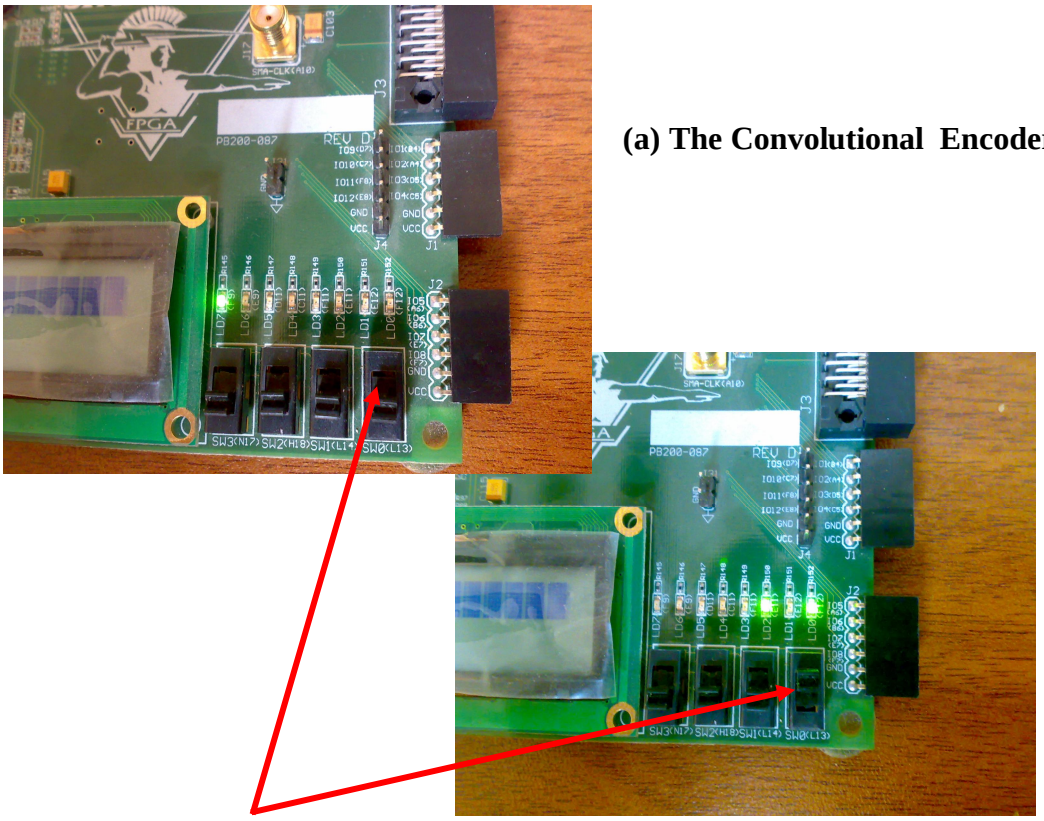


Figure (11) The Multiple Booting Technique

5. Conclusions

Applying the multiple booting technique can be done on any FPGA kit that supports the use of such technique like Spartan 3E FPGA Starter kit (supported with XC3S500E).

In this research, the architectures of the Convolutional encoder (2, 1, 3) and the Viterbi decoder were designed and implemented on XC3S500E FPGA chip built in Spartan 3E FPGA Starter kit using multiple booting technique.

The reuse of the same hardware for different applications is the main benefit of multiple booting technique which gives an efficient re-configurability to the chip. This scheme gives the flexibility to change the function of the chip between the Convolutional encoder and Viterbi decoder which is very useful in modern systems.

Multiple booting technique allows other types of encoders with different parameters and the corresponding decoders to be added. Consequently, the device can be operated in various modes as needed.

References

[1] I. Kang, A. N. W. Jr, "*Low-Power Viterbi Decoder for CDMA Mobile Terminals*", IEEE J. Solid-State Circuits, vol. 33, no. 3, Mar. 1998.

[2] B. W. Kim, J. H. Yang, C. M. Kyung, et al., "*MDSP: 16-bit DSP with Mobil Communication Accelerator*", in Proc. IEEE Custom Integrated Circuits Conference, pp. 2.1.1-2.1.4, 1998.

[3] Tuominen J. and Plosila J ., "*Asynchronous Viterbi Decoder in Action Systems*", TUCS Technical Report, No. 710, September 2005.

[4] Lin C. C., Shih Y. H., Chang H. C., and Lee C. Y.." *A Low Power Turbo/ Viterbi Decoder for 3GPP2 Applications*", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 14, No. 4, pp. 426-430, April 2006.

[5] Lin Y. Baron N., Lee Y, Mahlke S., and Mudge T., "*A Programmable Vector Coprocessor Architecture for Wireless Applications*", Advanced computer architecture laboratory, University of Michigan, 2004. http://www.eecs.umich.edu~tnmpaperswasp04.pdf

[6] Ken Chapman, "*NOR FLASH Programmer for Spartan-3E Starter Kit*", User Guide, Xilinx Ltd., March 2006. Available Via Internet at Web Site: http://www.xilinx.com/products/boards/s3estarter/ files/s3esk_picoblaze_nor_flash_programmer.pdf

[7] Jameel Hussein, "*Multiple-Boot with Platform Flash PROMs*", Application Note on Spartan-3E FPGA, Xilinx Company, April 2007. Available Via Internet at Web Site:

http://www.xilinx.com/support/documentation/application_notes/xapp483.pdf

[8] B. Sklar, "*Digital Communications Fundamentals and Applications*", Prentice Hall, Second Edition, 2001.

[9] Xilinx, Inc., "*Spartan-3E Starter Kit Board User Guide*", UG230 (v1.0) March 9, 2006. http://www.xilinx.com.

[10] Xilinx, Inc., San Jose Calif., "*Spartan-3E Field Programmable Gate Arrays*", 2006. http://www.xilinx.com.

The work was carried out at the college of Engg. University of Mosul