**TELKOMNIKA**, Vol.15, No.2, June 2017, pp. 646~653
ISSN: 1693-6930, accredited **A** by DIKTI, Decree No: 58/DIKTI/Kep/2013
**DOI:** 10.12928/TELKOMNIKA.v15i2.3612


# Design of Digital to Analog Voice Data Packet Conversion from Ethernet Protocol using FPGA

**Barlian Henryranu Prasetio[1*], Dahnial Syauqy[2]**
[1, 2]Computer System and Robotics Lab, Faculty of Computer Science, University of Brawijaya, Malang, Indonesia
*Corresponding author, e-mail: barlian@ub.ac.id*[1], dahnial87@ub.ac.id[2]

***Abstract***

*This paper describes a system that is designed to be able to receive packet voice signal using the Ethernet protocol in local networks using FPGA which was programmed decode the data packets. The digital data packets are then converted back into analog data that will be used to control another system. The design was implemented as four components consisting of frame starter unit, address matching unit, buffer unit and DAC processing unit. The system was designed on Xilinx development board using ISE design suite and simulated on ISIM. The test results showed that the system response was less than 40 ms. The result also showed that our proposed design only occupies 11% of number of slices and it also requires 5% of total IOBs on Xilinx Spartan 3-E.*

***Keywords:*** *digital design, voice, FPGA, ethernet protocol*



## 1. Introduction

The effort to replace the traditional user interface, such as RS-232, RS-485 and others has made a revolution to the Internet protocol (IP) packet-based networks. In the field of communication system, the efforts are approached through the implementation of different algorithm used in voice based compression encoding technology. Voice protocol is related to the process of sending a voice in the form of data packets over a computer network [1]. When there is an incoming voice call on a shared network, in some cases, it causes quite a big delay of data reception and processing [2]. The voice protocol should be given high priority when carrying data [3]. This makes data transmission management becomes a major thing in the control network. Voice control is widely used because it can reduce the use of manual effort [4]. The voice protocol converts our voice which is in form of analog signal into digital signal and transmits it over the Internet. Unlike an old phone which transmits voice in electrical signals through wires, the voice protocol can be used directly through the computer [5].

The process of sending and receiving voice data packets requires multiple units or sub-devices to perform any function on voice protocol. In the current research, we proposed a system to receive voice data packet which was previously sent by sender unit through Ethernet protocol. An alternative device that can be used to design the voice data packets reception is an FPGA [6]. The FPGA allows functions to be executed in a single configuration process where certain functions and its timing can be controlled [7-9]. The FPGA is also able to work in real time system [10] which is suitable for digital to analog voice processing.

As previously stated, the purpose of this research is to design a digital decoder from voice data packets through Ethernet communication so that it can be converted as analog signal using FPGA. The research is expected to be able to demonstrate the principles of data communication and explain an understanding of software and hardware that were implemented as data communication on embedded system applications and integrating on a single chip. The network uses 802.3 Ethernet which allows many hosts to send and receive data over twisted pair network [11].

*Received January 5, 2017; Revised March 18, 2017; Accepted April 6, 2017*

## 2. Research Method

The overall system consisting sender and receiver units is shown in Figure 1. The sender system uses a personal computer (PC) connected to a microphone which is used as a voice input. The receiver system uses FPGA board connected to a speaker as the voice output. The FPGA is connected to the PC by using UTP cable through a switch. A software application embedded on PC serves as communication controller between the PC and the FPGA. The communication controller application is called the Sender Software. After the voice data packet has been sent, the FPGA will produce the same sound as the voice input to the PC microphone. In the current research we proposes the design of the receiver unit which will obtain the voice data packet, decode the data, and finally convert it into analog signal for the speaker as the output of the system.

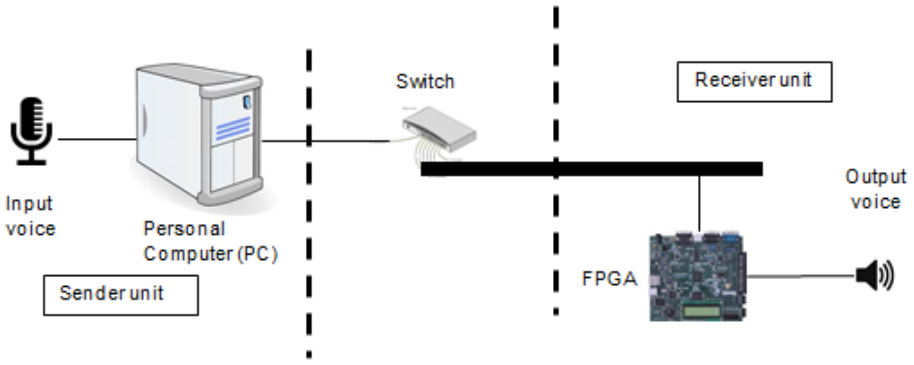



Figure 1. The system block diagram

In the receiver unit, the main process is divided into four sub processes as FS (Frame Starter), AM (Address Matching), Buffer, and DAC (Digital to Analog Converter). As the base of FPGA design, we use Xilinx Spartan 3-E starter board which provides an SPI-compatible, four-channel, serial Digital-to-Analog Converter (DAC) with 12 bit unsigned resolution [12]. The four outputs from the DAC appear on the J5 header of the board. The J5 interface signals can be seen in Table 1.

Table 1. The J5 interface signals

| Signal | Description |
|---|---|
| DAC_OUTA | Analog Output 0 |
| DAC_OUTB | Analog Output 1 |
| DAC_OUTC | Analog Output 2 |
| DAC_OUTD | Analog Output 3 |
| GND | Ground |
| VCC3V3 | 3.3V |

The information in the data packets includes RX_CLK which refers to recovered clock rate of the received packet, RX_DV which refers to the validity of the received signal, and RX_D which refers to 4 bits of data. When the data packets have been received through the Ethernet, the data will be checked for the validity. If it is valid, the data will be entered into Frame Starter (FS). Then, the data will be checked to make sure that the MAC address, IP and port are correct in Address Matching (AM). After the data has been checked that it really came from the particular application sender, then the data will be stored in the buffer (BUFF). Once the data are stored, it will be converted into a serial data in Digital to Analog Conversion (DAC) process. Finally, the serial data will be sent to the DAC unit. The overall process is shown in Figure 2.

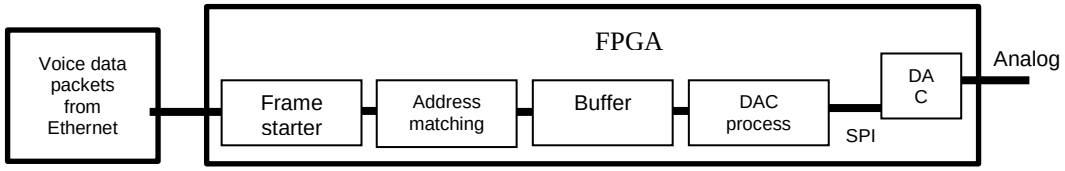



Figure 2. Proposed block diagram

The Frame starter (FS) provides the function to calculate the amount of the data received. If the data are received as "1010", the counter will count up to 15 (1111). However, when the received data are "1011" and the counter is already 15, then AM will be enabled and the FS will be reset. Otherwise, the counter will be reset to 0. The FS algorithm flowchart is shown in Figure 3.

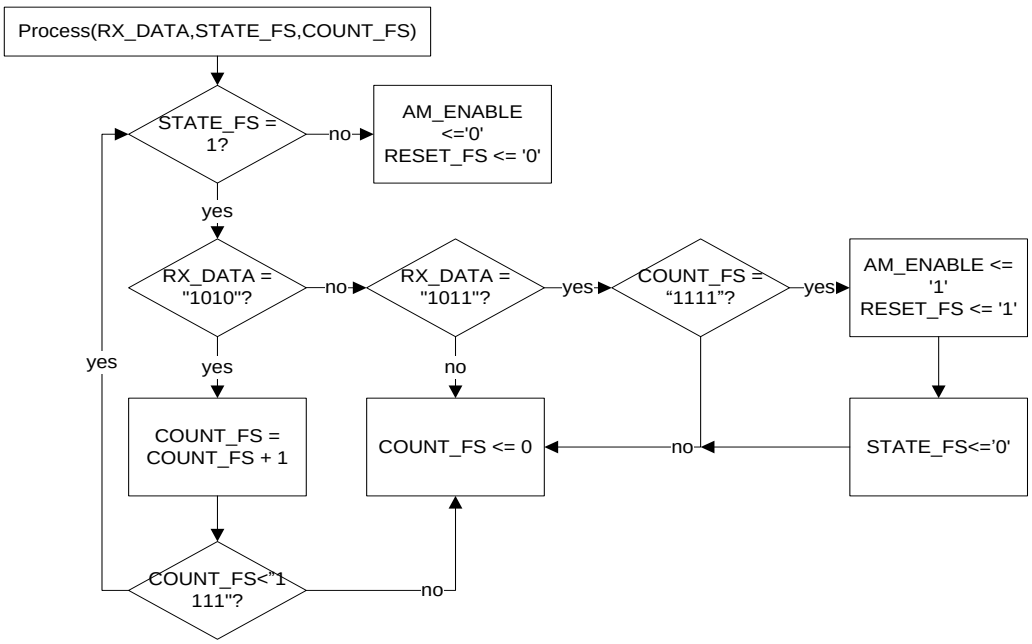



Figure 3. The FS Algorithm Flowchart

The Address Matching (AM) unit serves to ensure that the data packets are originating from the application sender in the sender unit. If the received data packet information is matched, then the AM enables the Buffer. In order to ensure that the data were derived from the application sender, the data packet information including MAC, Ethernet, protocol, IP and PORT will be compared [13]. The AM unit checks whether the address is matched to that one that has been assigned. If one of the addresses does not match then the data packet will not be forwarded. The Value of Address Matching component is shown in Table 2.

Table 2. The Value of Address Matching Component

| Address Matching Component | Component Value |
|---|---|
| MAC | FF.FF.FF.FF.FF.FF |
| ETH | 0800h (IP) |
| PRO | 11 |
| IP | 255.255.255.255 |
| PORT | 3435 |

When all the data packet information is matched, the buffer (BUFF) will be enabled. The AM unit algorithm flowchart is shown in Figure 4. After the buffer process is activated, the data will be stored in the address block between 00h to FFh. If the address in that block is full, the data can be stored in the next address blocks [14]. After the buffer process completes, the Digital-to-Analog Conversion (DAC) process will be enabled. The Buffer process flowchart is shown in Figure 5.

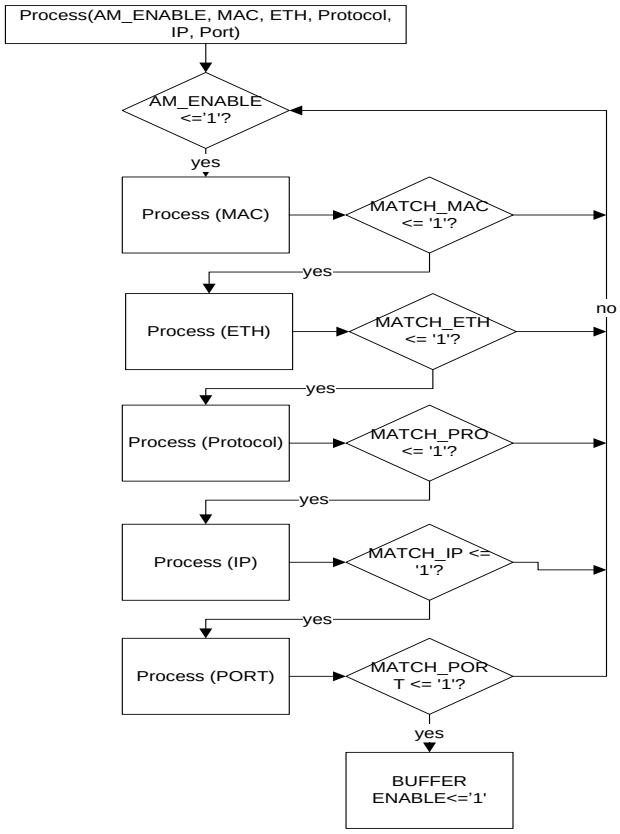



Figure 4. The AM Algorithm Flowchart

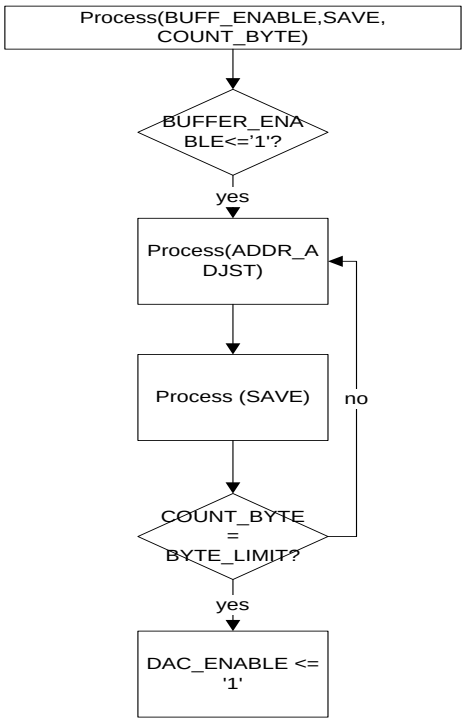



Figure 5. The Buffer Flowchart

When the DAC processing module receives the information containing data to be written in its RAM, the DAC unit requests buffer to send 1 byte of the data [15]. Those data will be processed by Digital Signal Processing (DSP) sub unit in the DAC module to be transformed into 12 bits. The 12 bits of data will be added by 8 bit don't care (DC), 4 bits command, 4 bits output address at the MSB and 4 bits don't care at LSB, which makes the whole size become 32 bits. After the 32 bits of data have been stored in the latch, the digital-to-analog process will be started.

This process will continue until all set of bytes of the digital data are converted to analog. At that time, the counter should be decremented. The process goes on until the counter reaches zero. The output of the DAC process is a serial data consisting of SPI_MOSI and SPI_SCK which will be sent to the DAC unit provided on the board. Figure 6 shows the DAC algorithm flowchart.

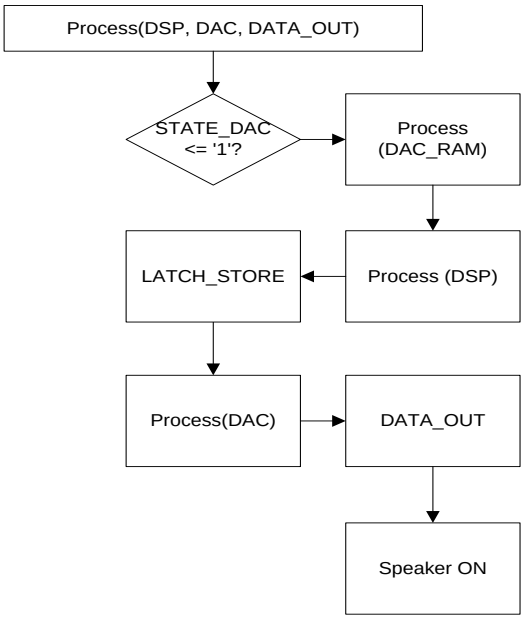



Figure 6. The DAC Algorithm flowchart

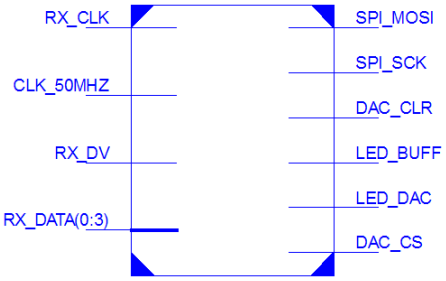
## 3. Results and Discussion

The receiver unit design was implemented in VHDL with Xilinx Spartan 3E started board using ISE Design Suite 13.4 and simulated in test bench using Isim [16]. The RTL Schematics Diagram and the description of the component which consist of four modules (FS, AM, BUFF, DAC) are shown in Figure 7 and Table 3, respectively.

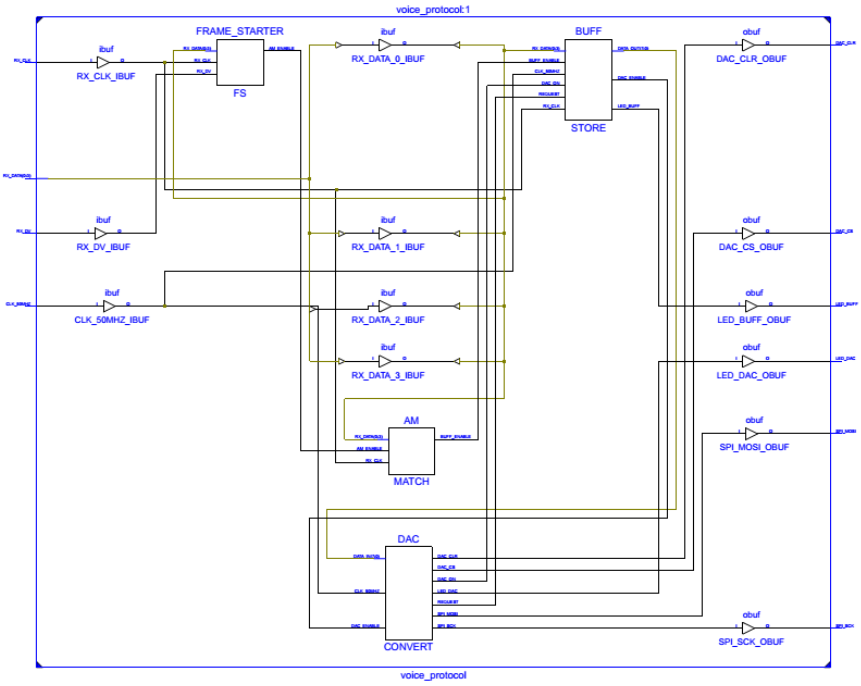
Table 3. The RTL Schematics Description

| Component | Input | Output |
|---|---|---|
| FRAME_STARTER (FS) | • RX_DATA(0:3)<br>• RX_CLK<br>• RX_DV | • AM_ENABLE |
| AM | • RX_DATA(0:3)<br>• AM_ENABLE<br>• RX_CLK | • BUFF_ENABLE |
| BUFF | • RX_DATA(0:3)<br>• BUFF_ENABLE<br>• CLK_50MHZ<br>• DAC_ON<br>• REQUEST<br>• RX_CLK | • DATA_OUT(7:0)<br>• DAC_ENABLE<br>• LED_BUFF |
| DAC | • DATA_IN(7:0)<br>• CLK_50MHZ<br>• DAC_ENABLE | • DAC_CLR<br>• DAC_CS<br>• DAC_ON<br>• LED_DAC<br>• REQUEST<br>• SPI_MOSI<br>• SPI_SCK |

As a comparison for the FPGA Xilinx implementation, we also implement the design in NI MyRIO FPGA board. Both device Utilization Summary are shown in Table 4. These summaries are calculated and generated by the design suite software during the compilation process. It can be seen that the proposed method only occupies 11% of number of slices on Spartan 3-E. We also notice that it also requires 5% of total IOBs.




(a)




(b)

Figure 7. The System RTL Schematics Diagram which is illustrated in (a) overall (b) detailed

Table 4. The Device Utilization Summary

| | Xilinx Spartan 3E Board | | | NI MyRio FPGA | | |
|---|---|---|---|---|---|---|
| Logic Utilization | Used | Available | Utilization | Used | Available | Utilization |
| Number of Slice | 549 | 4656 | 11% | 594 | 4400 | 13.5% |
| Number of Slice Flip Flops | 485 | 9312 | 5% | 485 | 9312 | 5% |
| Number of 4 input LUTs | 683 | 9312 | 7% | - | - | - |
| Total Number of 4 input LUTs | 947 | 9312 | 10% | 947 | 9312 | 10% |
| Number of bonded IOBs | 13 | 232 | 5% | 13 | 232 | 5% |
| Number of RAMB16s | 16 | 20 | 80% | 16 | 20 | 80% |
| Number of BUFGMUXs | 4 | 24 | 16% | 4 | 24 | 16% |

Finally, the simulation of our design was done using Test Bench in Isim. The simulation was performed to observe how fast the designed system responds to the input stimulus. In the simulation test bench, we provide virtual input stimulus to the designed system. The virtual input stimulus data were voice data packet from Ethernet protocol which constructed from MAC address information, Ethernet type, IP header, IP protocol, port, and the voice data itself. The observation was done via the timing diagram generated by Isim. Figure 8 shows the timing

diagram of the proposed design using virtual input stimulus. It can be observed that the last data from voice data packet which is represented as SPI_MOSI signal in the timing diagram were appearing after about 40 ms (39752.4 $u$s).

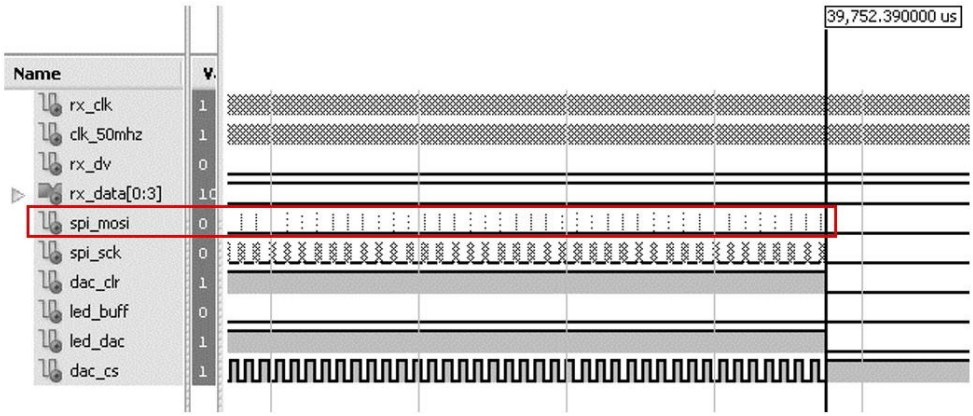



Figure 8. The System Timing Diagram

## 4. Conclusions

The system of Digital to Analog Voice Data Packet Conversion has been implemented on FPGA. The designed system consists of four sub unit: FS (Frame Starter), AM (Address Matching), Buffer, and DAC (Digital to Analog Converter). The test results based on ISIM simulation showed that the system provides the output response in less than 40 ms. According to the percentage of utilization of the device, it can be seen that our proposed design only occupies 11% of number of slices and it also requires 5% of total IOBs on Xilinx Spartan 3-E. Our next research will focus on audio compression to reduce the memory usage and the use of other signal processing techniques to improve the quality of the reproduced sound.

## References

[1] Rakesh Arora, Voice over IP : Protocols and Standards, http://www.cse.wustl.edu/~jain/cis788-99/ftp/voip_protocols.pdf, 1999.
[2] Karie Gonia, Latency and QoS for Voice over IP, SANS Institute InfoSec Reading Room. 2004.
[3] cisco, Quality of Service for Voice over IP, http://www.cisco.com/c/en/us/td/docs/ios/solutions_docs/qos_solutions/QoSVoIP/QoSVoIP.html, 2001.
[4] R. Puviarsi. Self-Assistive Technology for Disabled People – Voice Controlled Wheel Chair and Home Automation System. *International Journal of Robotics and Automation (IJRA).* 2014; 3(1): 30~38.
[5] Tomar. GS and George. ML, Hardware Implementation of Pulse Code Modulation Speech Compression Algorithm. *Asia-pacific Journal of Multimedia Services Convergence with Art, Humanities and Sociology*. 2012; 2(1): 19-26.
[6] Perkins C, RTP Audio and Video for the Internet, Addison Wesley. 2003.
[7] Izeboudjen N, A New Design Reuse Approach For Voip Implementation Into Fpsocs And Asics. *International Journal on Soft Computing (IJSC).* 2013; 4(4).
[8] Francisca O. Oladipo, Re-Engineering Campus-Wide Internet Telephony Using Voice over Internet Protocol. *International Journal of Networks and Communications.* 2015; 5(2): 23-30.
[9] Lamjed Touil. Towards Hardware implementation of video applications in new telecommunications devices. *Journal of Telecommunications*. 2010; 2(1).
[10] Yuansheng L. Research on the Key Technique of SPC Exchange Equipment Based on FPGA. *TELKOMNIKA Telecommunication Computing Electronics and Control*. 2013; 11(7): 3809~3820.
[11] Edet Bijoy K. An Approach to Sparse Reconfigurable Hardware for LPC of Speech for VoIP. *International Journal of Computer Applications, National conference on VSLI and Embedded systems*. 2013.
[12] Xilinx, Spartan-3E FPGA Starter Kit Board User Guide, http://www.eng.utah.edu/~cs3710/xilinx-docs/ug230.pdf, 2008

[13] Cisco, Online Help for Cisco IOS Release 12.3(02):JA Home: Configuring Filters, http://www.cisco.com/web/techdoc/wireless/access_points/online_help/eag/123-02.JA/1400BR/h_ap_howto_3.html, 2004
[14] Marc Defossez, Parallel LVDS High-Speed DAC Interface, Application Note: 7 Series FPGAs, xilinx, 2012
[15] X.Li, An FPGA implemented 24-bit audio DAC with 1-bit sigma-delta modulator, Circuits and Systems (APCCAS), 2010 IEEE Asia Pacific Conference, 2010
[16] Xilinx, ISE Simulator (ISim) 13.4 Quick Reference, www.xilinx.com/tools/feature/13_4_isim_quick_reference_guide.pdf, 2012