DESIGN OF A COMPATIBLE USB 3.1 UNSCRAMBLER AND 132B/128B DE-CODER

LUIS ENRIQUE RUEDA DUARTE

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FISICO-MECÁNICAS
ESCUELA DE INGENIERIA ELECTRICA, ELECTRONICA Y TELECOMUNICACIONES
BUCARAMANGA
2016

DESIGN OF A COMPATIBLE USB 3.1 UNSCRAMBLER AND 132B/128B DE-CODER

LUIS ENRIQUE RUEDA DUARTE

Trabajo de grado para optar al título de Ingeniero Electrónico

Director
ELKIM FELIPE ROA FUENTES
Ingeniero Electricista, Ph.D

Co-Director
CKRISTIAN RICARDO ESTEBAN DURAN BLANCO
Ingeniero Electrónico

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FISICO-MECÁNICAS
ESCUELA DE INGENIERIA ELECTRICA, ELECTRONICA Y TELECOMUNICACIONES
BUCARAMANGA
2016

AGRADECIMIENTOS

A Dios, por darme la salud y sabiduría para lograr culminar mi carrera.

A mis padres, Luis y Claudia, por apoyarme incondicionalmente durante mis estudios y formarme como persona.

A mis hermanas, Slendy, Silvia, Angelica y Maria, por su ayuda y motivación en los tiempos difíciles.

A mis amigos y compañeros, que siempre me han prestado ayuda y consejo cuando los necesito.

A mis profesores, que con sus enseñanzas y consejos me formaron como profesional y me han ayudado a seguir adelante.

Luis Enrique Rueda Duarte

CONTENTS

Page.

LIST OF FIGURES

Page.

LIST OF TABLES

Page.

RESUMEN

Título Design of a compatible USB 3.1 Unscrambler and 132b/128b Decoder[*].

Autores Luis Enrique Rueda Duarte.[**]

Palabras clave Microelectrónica, USB 3.1, microcontrolador, protocolo, unscrambler, decoder, 132b/128b decoder.

DESCRIPCIÓN

Este trabajo presenta el diseño totalmente sintetizable en tecnologías CMOS 65nm y 130nm de los módulos 132b/128b decoder y unscrambler de la capa física del estándar USB 3.1, los módulos unscrambler y decoder 132b/128b son diseñados para su futura implementación en la segunda versión de la tarjeta de desarrollo ONCHIP. En este trabajo se muestra el funcionamiento general de la capa física de USB 3.1 comprendiendo la importancia de cada módulo y comprendiendo a totalidad el funcionamiento de los módulos unscrambler y decoder 132b/128b, para comprender el funcionamiento se muestra la lógica, siguiendo del datapath implementado y la máquina de estados que hace cumplir el estándar USB 3.1. El decoder implementado es usado para la alineación de datos y la detección de errores, lógica que es mostrada en el presente trabajo. El módulo unscrambler usa registros de desplazamiento con retroalimentación lineal (LFSR por sus siglas en ingles) para crear un polinomio de orden 23, además se muestra un patrón para la verificación de este tipo de LFSR y unscrambler total. Finalmente, en el trabajo son reportados los resultados possíntesis en 130nm and 65nm con frecuencias para el decoder en caso típico de 1.21GHz y 1.47GHz respectivamente y para el unscrambler frecuencias de 1.27GHz y 1.57GHZ.

[*] Trabajo de grado.

[**] Facultad de ingenierías Físico-mecánicas. Escuela de Ingeniería Electrica, Electronica y Telecomunicaciones. Director: Elkim Felipe Roa Fuentes. Co-Director: Ckristian Ricardo Esteban Duran Blanco.

ABSTRACT

TITLE Design of a compatible USB 3.1 Unscrambler and 132b/128b Decoder[*].

Authors Luis Enrique Rueda Duarte.[**]

Keywords Microelectronics, USB 3.1, microcontroller, protocol, unscrambler, decoder, 132b/128b decoder.

DESCRIPTION

This document presents the complete designs of fully synthesized in 65 and 130nm CMOS technology the 132b/128b decoder and unscrambler modules of the USB 3.1 standard physical layer. The unscramble and 132b/128b decoder are designed for their implementation in the second version of ONCHIP development target. This work shows the general functionality of the USB 3.1 standard physical layer, understanding the importance of each module and focused in the unscramble and 132b/128b decoder. The work shows de timing logic, datapath and state machine which manages to meet the USB 3.1 standard. The implemented decoder is used for data aligner and error detection in the block header of transferred data, its logic is shown in the present work. The unscramble module uses linear feedback shift register LFSR to generate an order 23 polynomial, also, this work shows a check pattern for LFSR and Unscrambler. Finally, this work reports the possynthesis results in 130nm and 65nm CMOS technology, the synthesis is done according parameter of USB 3.1 standard as maxim frequency and clock uncertainty (jitter). The synthesis results show to up frequency for decoder of 1.21GHz 130nm and 1.47GHz in 65nm for typical case and the unscramble 1.27GHz 130nm and 1.57GHz in 65nm for typical case.

[*] Bachelor degree

[**] Faculty of Physico-Mechanical Engineering. School of Electronics and Electrical engineering and telecommunications. Advisor: Elkim Felipe Roa Fuentes. Co-Advisor: Ckristian Ricardo Esteban Duran Blanco.

INTRODUCTION

USB 3.1 is the state of the art of USB standards versions, standard 3.1 is a performance enhancement to SuperSpeed USB 3.0 presented by USB as a solution to increase the bandwidth in application as larger and faster storage, higher resolution video, broader use of USB as an external expansion, and development target providing more power and more than double the bandwidth for devices compatibles with standard 3.1, also USB 3.1 is fully backward compatible with 2.0 and 3.0 versions with a double bus architecture implemented in the standard. USB 3.1 change to more efficient data encoding which delivers more than twice the effective data [1], [2].

USB uses a 132b/128b decoder to provide the enough changes for data and clock circuit recovery. The codification allows an error detection in the block header and correction of single-bit error, decoding also is used in the data alignment of the receptor. Standard 3.1 scramble the data using a linearfeedback shift register (LFSR), this scramble uses a 23 order polynomial. The data may be scrambled or not depending upon if are data or control type [1].

This report presents a description of the 132b/128b decoder and unscrambler behavior according to the USB 3.1 standard for gen2 of the receptor. In addition, the circuits of both modules are proposed and designed, their area, power consumption and timing reports are shown in the synthesis results. Finally, this report shows the development target in which will be implemented the fully USB 3.1.

1. SPECIFICATION FOR THE PHYSICAL LAYER OF THE GEN2 FOR USB 3.1 RECEPTOR

The USB 3.1 is structured by four different layers: physical layer, Link layer, protocol layer, and device or host layer where each layer has a special architecture. The physical layer architecture is shown in **¡Error! No se encuentra el origen de la referencia.** and its back end is shown in **¡Error! No se encuentra el origen de la referencia.**. The First step in the process of the physical layer is, serial bits are received by a differential receiver and an equalization data, then these bits are sent

Figure 1. Physical Layer of USB 3.1 [1].

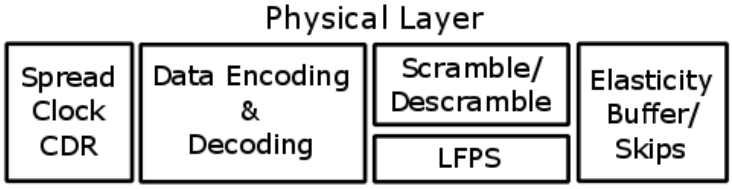
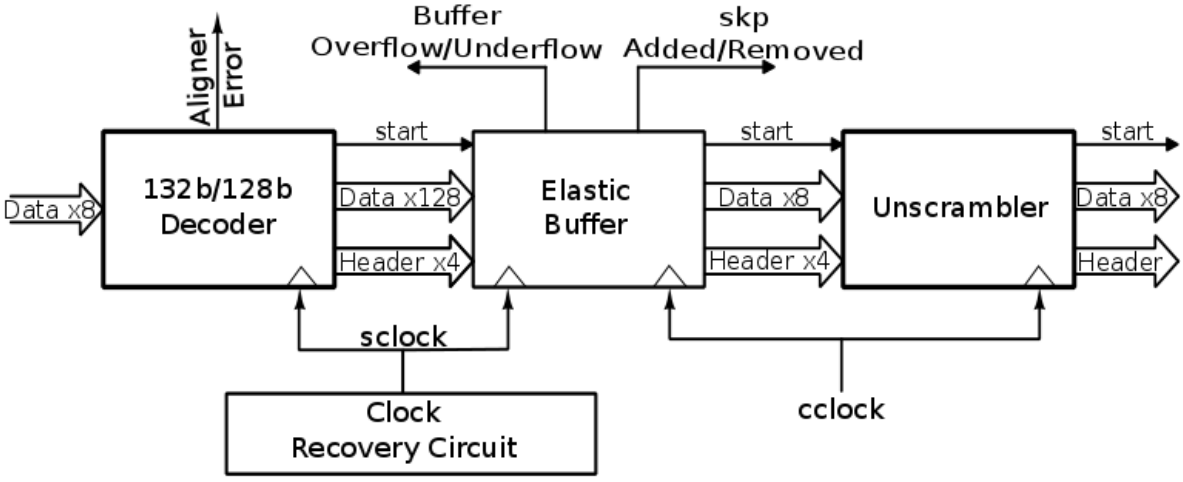


Figure 2. Back End Receiver Block Diagram [1].



to data recovery circuit which processes these data and corrects the damage occurred in the transmission. After of recovery data, the serial bits are delivered to a serial-parallel circuit which has an 8-bit output signal that is the input of the 132b/128b block aligner also called 132b/128b decoder. This decoder takes 8-bit data and delivery a 4-bit block header, data block of 128-bit and control signal of 1-bit then it is processed by elastic buffer and finally by unscrambler which recovery of original data.

This paper is focused on the 132b/128b decoder and Unscrambler blocks, the functionality of each block is explained below.

A. 132b/128b Decoder

The encoding 128b/132b is used in USB 3.1 for more efficient data encoding, the encoder delivery more than twice the effective data. In addition, encoding provide the enough changes for data and clock circuit recovery and add a block header of 4-bit to indicate the data type, 1100 for control and 0011 for data. Another advantage that provides this encoding is a simple logic which facilities to the data rate of 1.2 Mbps required by USB 3.1 standard.

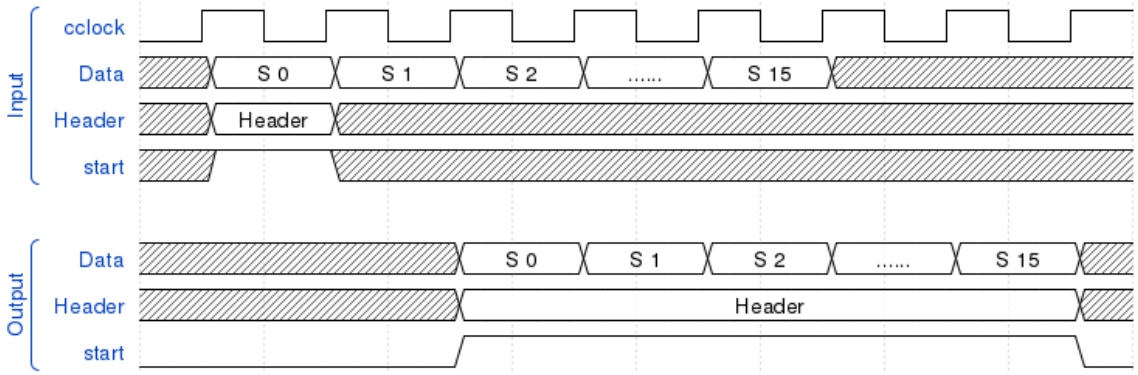
In the receptor, the decoder has two important functions, the first is data aligner, the decoder takes 8-bit data delivered by serial to parallel block and decoder delivery a 128-bit data, a 4-bit block header and single bit control signal [1], [3] as shown in **¡Error! No se encuentra el origen de la referencia.**. The second function is the error detection in the block header, when decoder detected a single-bit error, the decoder correct and report the error but the error is more than one bit, the decoder report the error and bypass the block header.

Table 1. An 8-bit value of 0 repeatedly encoded with the lfsr

	0	1	2	3	4	5	6	7
00	B8	92	0D	75	BB	BC	F5	69
08	A5	84	42	8A	AD	E6	9E	B3
10	42	7C	BC	B9	5E	65	4D	8B
18	A7	AA	F8	93	EC	9A	94	7E
20	D2	66	D6	9B	C0	27	A4	C9
28	E0	8E	93	1B	38	52	42	A5
30	07	8A	5C	6C	1B	57	AF	93
38	F2	83	D9	4A	A1	F8	E8	2C
40	36	A7	5E	ED	5A	83	D1	4B
48	BC	18	D2	D4	B4	FA	62	A3
50	FC	7B	41	7B	B2	FB	8B	50
58	73	8C	0E	19	15	49	01	52
60	58	D8	E8	D2	42	BD	2A	D0
68	F5	95	CD	77	9C	BD	D7	24
70	0E	48	A3	5A	50	SC	6C	DA
78	91	CE	26	FF	17	EC	30	F6

The operation of the decoder is shown in the **¡Error! No se encuentra el origen de la referencia.**, the SCKL is a recovered symbol clock, when appearing a positive edge in SCLK serial to parallel module delivery a symbol of 8-bit, this symbols must be accumulated until the decoder has a first 136-bit. The first least significant 132 bit are the block header and data output of the decoder, then the four remaining bit are the block header of the next package delivery by the decoder, and the data output are the following 16 input symbols. The decoder uses a signal of start to indicate that a complete data can be used by elastic buffer.

Figure 3. Input and output Behavior of Decoder.



B. **Unscrambler**

The scramble is used to send out the data using spread spectrum clocking to lower EMI emissions [1] further assists to keep DC balance, this function is the result of xor operation the with 8-bit generate by the polynomial $X^{23} + X^{21} + X^{16} + X^8 + X^5 + X^2 + 1$ which is implemented using a free running Linear Feedback Shift Register (LFSR). The unscrambler has three functionality modes, these modes are shown in Table I. Each advance of the scrambler is the 8 serial shifts in the LFSR. The xor operation is the data with eight the most significant of LFSR with the least significant bit of data. Operation modes are defined by rules described in [1], the above rules depend on data type which could be data or control, when it is data, the scrambler advance for all symbols and another case the mode is defined by control block type that can be TS1, TS2, TSEQ, SYNC, SKP and SDS order set. Each control block has its sequence of modes for each symbol of the control block.

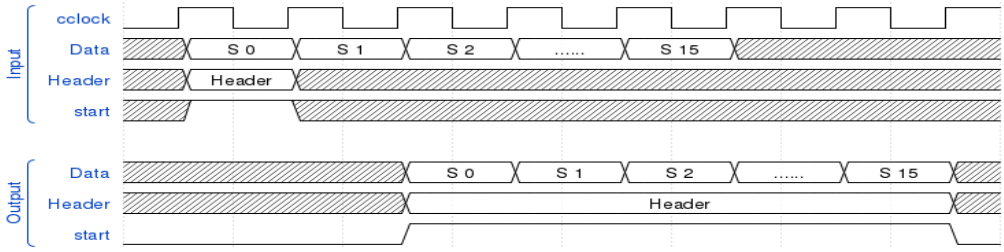
The operation of unscrambler must follow a pattern, this patter can verify generating an 8-bit value of 0 repeatedly and encoded with the LFSR, this pattern is shown in the Table. II. The timing diagram of the unscrambler is shown in the **¡Error! No se encuentra el origen de la referencia.**, the input signals appear, the unscrambler following the rules of USB 3.1 xored or bypass the data, the header is

a bypass, the start is high when the first output data is processed by the scrambler, this start change to low in the next cycle from last one output symbol appear.

Table 2. Functionality modes of unscrambler

Mode	Scrambler	Bypass/Xored
1	Advances	Xored
2	Advances	Bypass
3	No Advances	Bypass

Figure 4. Input and output Behavior of Unscrambler.



2. IMPLEMENTATION

According to USB 3.1 standard are proposed the 132b/128b decoder and unscrambler modules. The modules are described in verilog, follow is showing their architecture and methodology of each block to meet requirements of USB 3.1 standard.

A. 132b/128b Decoder

The datapath designed is shown in the **¡Error! No se encuentra el origen de la referencia.**. The datapath has a counter which registers the amount of the received bytes, then according to count FSM sends the control signal to the register 17x8, the counter, and the error detected. The datapath is controlled by a finite state machine (FSM) shown in the **¡Error! No se encuentra el origen de la referencia.**. The control signals of the FSM are reset, enable and end count. The FSM has three states, Wait, Count 17 and count 16. In state of Wait FSM disables all block of datapath, in Count 17 or Count 16, the FSM enables all blocks and the counter has a maximum count of 17 or 16. When the counter finishes the count, the FSM enables the output register 17X8 and the error detected.

Figure 5. Decoder's Datapath Architecture.

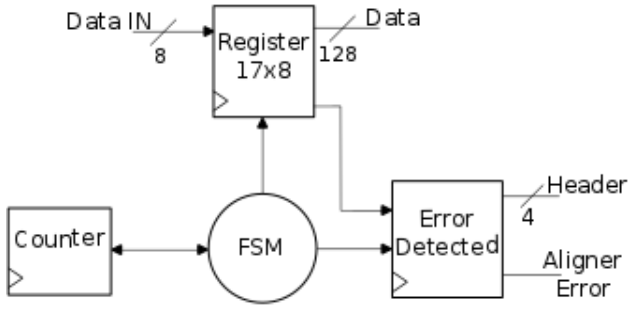
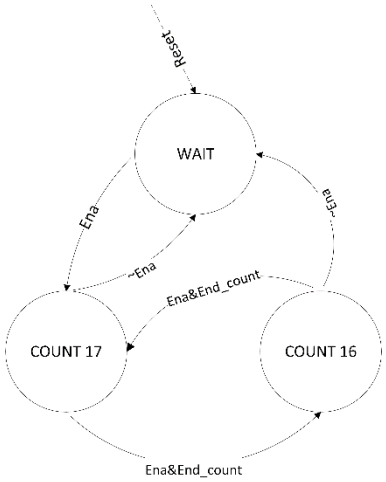


Figure 6. FSM Decoder



The register 17x8 shown in the **¡Error! No se encuentra el origen de la referencia.** organizes and accumulates the input bits which are organized following the deserialization of the USB 3.1 standard, the least significant bit of the first byte is the most significant in the 136 bits accumulated by bank registers. When the register receives the first byte, the register sends the first four least significant bits to the error detected. The first four bits are the block header, the format of block header is 1100 for control data and 0011 for data, this format allows single bit error in the header. Error detected follow the logic shown in the Table. III, when the error is the single bit, this block correct and active the aligner error for report the error to link layer, when the error is greater than a bit the header is bypass and the error is reported to link layer trough of aligner error signal. Thirty-three bytes have two complete 132-bit data, in the first 17 bytes, the decoder takes the first least significant 128 bits to build the first complete data and keep the last four data which are the block header of the second data, this process is repeated cyclically. Therefore, the counter has two maxim accounts, sixteen and seventeen which is selected by FSM.

Table 3. Output logic of the header error detect

Header Code IN	Bit Error	Header Code OUT
0000	2-bit	0000
0001	1-bit	0011
0010	1-bit	0011
0011	Non error	0011
0100	1-bit	1100
0101	2-bit	0101
0110	2-bit	0110
0111	1-bit	0011
1000	1-bit	1100
1001	2-bit	1001
1010	2-bit	1010
1011	1-bit	0011
1100	Non error	1100
1101	1-bit	1100
1110	1-bit	1100
1111	2-bit	1111

B. Unscrambler

The architecture of the unscrambler is shown in Fig 7. In this design, the unscrambler is divided into three parts, LFSR polynomial, Xored block and FSM. The linear feedback shift register (LFSR) polynomial used in the unscrambler is $X^{23} + X^{21} + X^{16} + X^{8} + X^{5} + X^{2} + 1$ [1], the schematic implemented with this polynomial is also shown in the specification of USB 3.1. Input data is xored or bypass depending on its type, this xor operation is decided by FSM and executed by XOR circuit.

The last block is the FSM which is shown in the **¡Error! No se encuentra el origen de la referencia.**, the FSM must enforce the rules of scrambler the USB 3.1 [1]. The last states of the FSM can return to WAIT, DATA, SKP, SDS, SYNC or TRAINING state depending on new input data. In the Table VI is described the operation mode of the scrambler in each state and the number of the symbol which is processes in that moment. The TRAINING DC, TRAINING ID, DATA, SKP, SDS and LAST SYNC state can skip to WAIT, DATA, SKP, SDC, TRAINING or SYNC state, the skip is according to the next package.

Figure 7. Decoder's Datapath Architecture..

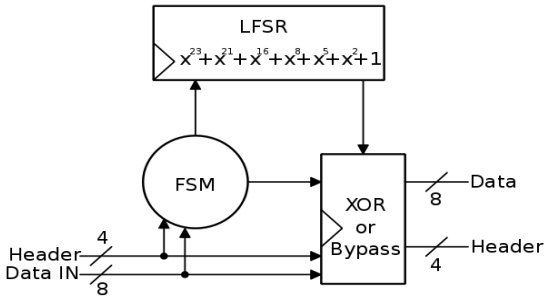


Figure 8. FSM Unscrambler.

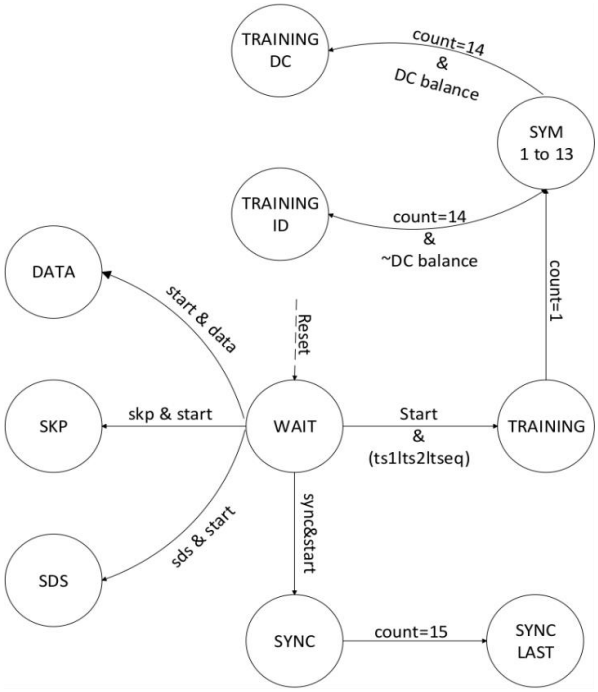


Table 4. States of unscrambler's fsm, in each state is shown the operation mode of unscrambler and the number of symbol input.

State	Advances	Xored	No. Symbol Input
WAIT	0	0	----
DATA	1	1	0 – 15
SKP	0	0	0 – 15
SDS	1	0	0 – 16
SYNC	1	0	0 – 14
SYNC LAST	0	0	15
TRAINING	1	1	0
SIM 1 TO 13	1	1	1 – 13
TRAINING ID	1	1	14 – 15
TRAINING DC	1	0	14 – 15

3. SYNTHESIS AND RESULTS

The unscrambler and decoder have been fully synthesized and verified in 130 and 65 nm CMOS technology. The synthesis in two different technologies allows to analyzing the maxim frequency for implementation of the USB. For this reason, Synthesis process is performed with generic use standard cells to three different corners available in both technologies, the results of synthesis process corresponding to the exposed in Table VII.

The synthesis setup is performed with 100 ps of jitter according to USB 3.1 standard, the worst path is calculated by RTL compiler. The selected worst path is the lesser of several iterations to different clocks.

Table 5. Worst path in 130 nm for worst case.

Pin	Type	Delay [ps]	Arrival [ps]
(clock symbclk)	launch		0
count reg[0]/CP			0
count reg[0]/Q	DFQD4	+288	288
g41076/A2		+0	288
g41076/ZN	NR2D4	+133	421
g40764/A2		+0	421
g40764/ZN	ND3D8	+131	552
g66/B		+0	552
g66/ZN	OAI21D1	+92	644
g40027/A2		+0	644
g40027/ZN	NR2XD0	+85	729
g39719/A2		+0	729
g39719/ZN	ND2D1	+87	816
g39326/A1		+0	816
g39326/ZN	IOA21D1	+133	949

Pin	Type	Delay [ps]	Arrival [ps]
data1321 reg[71]/D	DFQD4	+0	949
data1321 reg[71]/CP	setup	+97	1046
(clock symbclk)	capture		1196
	uncertainty	-100	1096
Cost Group : 'symbclk' (path group 'symbclk') Timing slack : 50ps Start-point : count reg[0]/CP End-point : data1321 reg[71]/D			

The results of unscrambler show the maxim frequency in the different corners, the frequency in worst case of 130nm is enough for an application which to be implemented the modules, the application is exposed in the next section. The results of the decoder show its maxim frequencies for the three different corners but only in the worst case no reach the frequency required in the USB 3.1 standard. Otherwise in 65nm the typical and best case exceed the required frequency.

In the Table IV is presented the worst path for 65 nm with delays for each cell, the more time in the worst path is the delay time of the register, the delay time is a 34.6% to a clock frequency of 1.2 GHz. The combinational cells also have a significant percentage, the NR2D4 and IOA21D1 have a percentage of 15.9 %. In another case of 65 nm also the register has the longest delay which is a 28.8% and the most significant standard cell is the 12 % as is shown in the Table V. The above results show that is necessary a synthesis with standard cells for high-speed applications. In the Table IV and V also is shown an uncertainty of clock, the value is according to jitter specified in the USB 3.1 standard which is 100 ps that meaning a 12% of the complete clock period. In addition to jitter, the synthesis process has a timing slack of 50 ps as error margin.

With the values additional the maxim period available of worst path to 1.2 GHz must be 683 ps but in 130 nm the delay of register now represent a 42% and the cell NR2D4 represent a 19% of maxim period clock, for this reason, and to improve the maxim frequency is proposed as future work the implementation of fastest standard cells or the implementation of the modules in smaller technologies with general purpose standard cells.

Table 6. Worst path in 65 nm for worst case.

Pin	Type	Delay [ps]	Arrival [ps]
(clock symbclk)	launch		0
count reg[4]/CK			0
count reg[4]/Q	DFFQ X4M A9TR	+240	240
g56116/A		+0	240
g56116/Y	INV X11M A9TR	+52	292
g20/A		+0	292
g20/Y	AND2 X11M A9TR	+102	394
fopt56051/A		+0	394
fopt56051/Y	INV X7P5M A9TR	+46	440
g55412/A		+0	440
g55412/Y	OR2 X4M A9TR	+100	539
g55237/B		+0	540
g55237/Y	NAND2 X6A A9TR	+83	622
g54895/A		+0	622
g54895/Y	NOR2XB X4M A9TR	+90	713
g54636/A0		+0	713
g54636/Y	OAI2XB1 X1M A9TR	+96	808
data1321 reg[67]/D	DFFQ X2M A9TR	+0	809
data1321 reg[67]/CK	setup	+68	877
(clock symbclk)	capture		1027
	uncertainty	-100	927
Cost Group : ’symbclk’ (path group ’symbclk’) Timing slack : 50ps Start-point : count reg[4]/CK End-point : data1321 reg[67]/D			

4. APPLICATION

The above modules are designed to be implemented in the development target of Onchip group, the first generation is shown in the **¡Error! No se encuentra el origen de la referencia.**, this generation target has a 32-bit microcontroller, its architecture is shown in the **¡Error! No se encuentra el origen de la referencia.**, the core has an architecture RISC-V, two buses, a principal AXI bus and a peripheral bus APB Bridge and six slaves, SPI APB, ADC, DAC, GPIO, RAM and SPI AXI. Development target actually is programmed using SPI, in the next version this microcontroller will be programmed using USB 3.1 and compatible IDE Arduino.

Figure 9. Diagram Block of actual microcontroller [4].

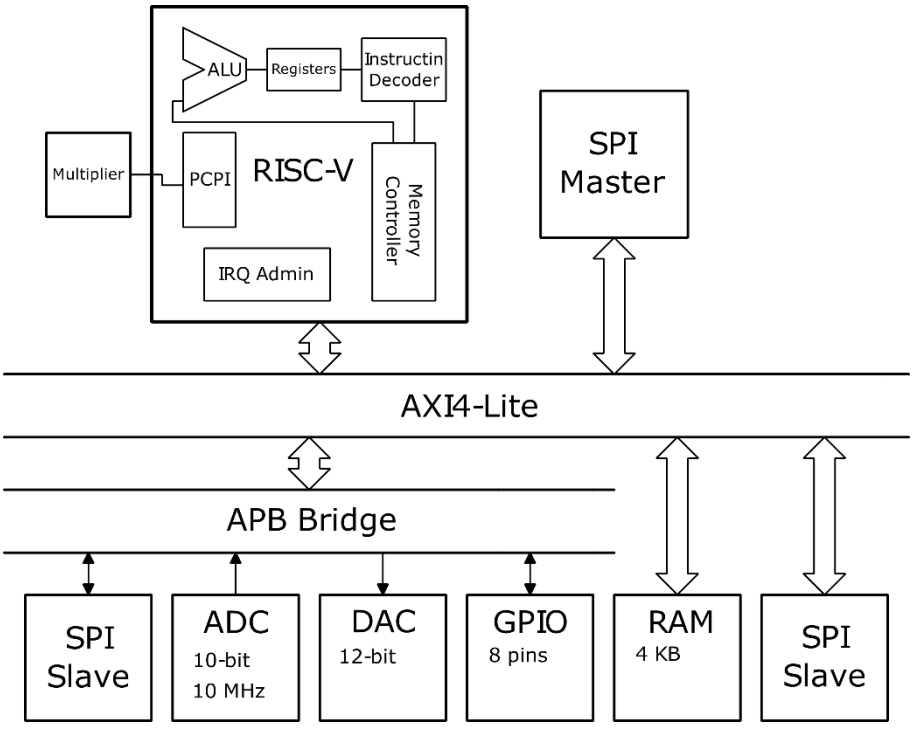
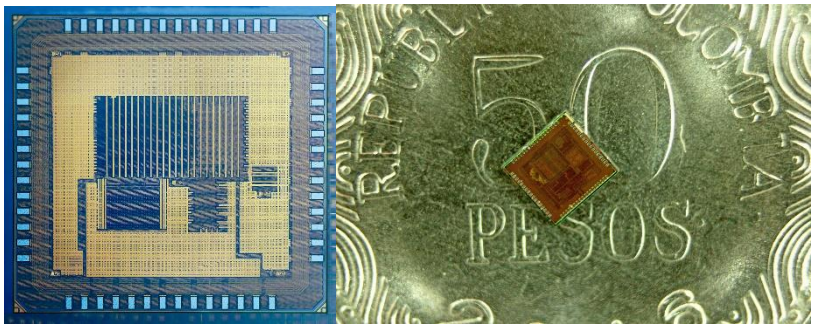
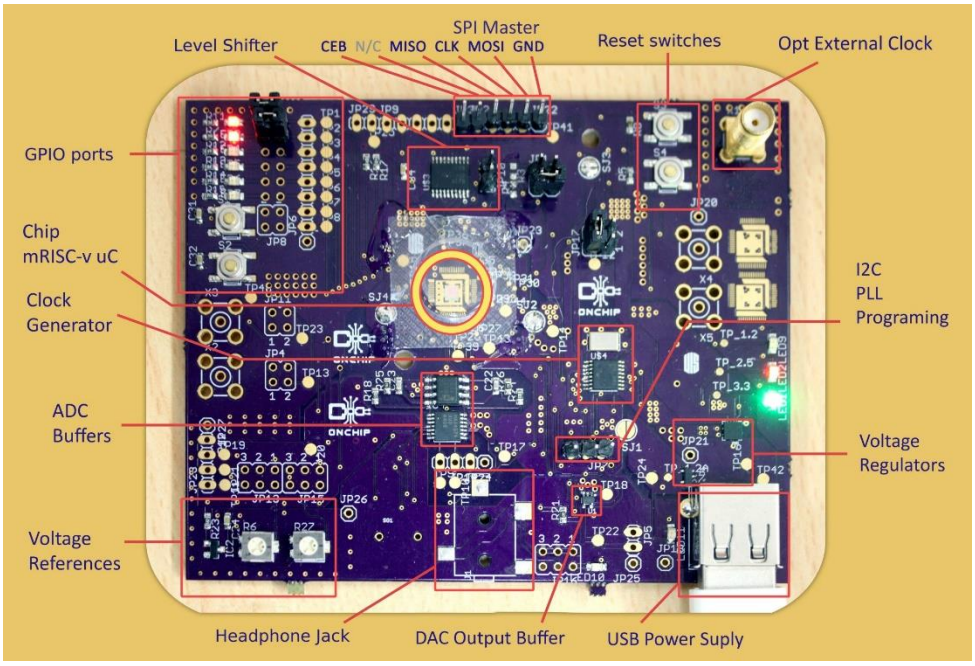


Figure 10. First generation of development target of Onchip group.



(a) (b)



(c)

5. CONCLUSIONS AND FUTURE WORK

The paper presented fully-synthesized the decoder and the unscrambler modules on 130nm and 65 nm CMOS technology following the USB 3.1 standard. The propose of both modules is to improve a development target which would be communicated using USB. The decoder and the unscrambler support the necessary clock frequency for the USB 3.1 application, in the bets case the decoder and the unscrambler modules support up 1.52 and 1.59 GHz respectively in 130nm. In 65nm the maxim frequency is 2.05 GHz to the decoder and 2.1 GHz to the unscrambler. The future work is the implementation of fastest standard cells to achieve a frequency of 1.2 GHz in the worst case, for a full security operation in the implementation of the decoder and unscrambler. Then of implementation of standard cells in the RTL compiler, the complete integration of USB 3.1 and its fully digital flow must be done.

REFERENCES

[1] U. S. Bus, “3.1 specification,” Hewlett-Packard Company, Intle Corporation, Microsoft Corporation, Renesas Corporation, ST-Ericsson, Texas Instruments, Revision, vol. 1.

[2] U. S. Bus, “USB. Org-Documents,” USB. org.

[3] Z. Qinglun, L. Chunyan, and W. Yong, “Hardware Implementation of 64B/66B Encoder/Decoder for 10-Gigabit Ethernet,” in 2006 International Conference on Communication Technology, pp. 1–4, IEEE, 2006.

[4] C. Duran, D. L. Rueda, G. Castillo, A. Agudelo, C. Rojas, L. Chaparro, H. Hurtado, J. Romero, W. Ramirez, H. Gomez, et al., “A 32-bit RISCV AXI4-lite Bus-Based Microcontroller with 10-bit SAR ADC,” in 2016 IEEE 7th Latin American Symposium on Circuits & Systems (LASCAS), pp. 315–318, IEEE, 2016.

BIBLIOGRAPHY

C. Duran, D. L. Rueda, G. Castillo, A. Agudelo, C. Rojas, L. Chaparro, H. Hurtado, J. Romero, W. Ramirez, H. Gomez, et al., “A 32-bit RISCV AXI4-lite Bus-Based Microcontroller with 10-bit SAR ADC,” in 2016 IEEE 7th Latin American Symposium on Circuits & Systems (LASCAS), pp. 315–318, IEEE, 2016.

U. S. Bus, “3.1 specification,” Hewlett-Packard Company, Intle Corporation, Microsoft Corporation, Renesas Corporation, ST-Ericsson, Texas Instruments, Revision, vol. 1.

U. S. Bus, “USB. Org-Documents,” USB. org.

Z. Qinglun, L. Chunyan, and W. Yong, “Hardware Implementation of 64B/66B Encoder/Decoder for 10-Gigabit Ethernet,” in 2006 International Conference on Communication Technology, pp. 1–4, IEEE, 2006.