IMPLEMENTATION OF VARIOUS COMMUNICATION PROTOCOLS ON FPGA

Dissertation submitted in partial fulfillment of the requirement for the

Degree Of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

BY

HARSHIT JAIN-151053

SHAURYA MEHROTRA-151081

UNDER THE GUIDANCE OF

DR. HARSH SOHAL

Department of Electronics and Communication Engineering

TABLE OF CONTENTS

DECLARATION BY THE SCHOLAR

I hereby declare that the work reported in the B-Tech thesis entitled **“IMPLEMENTATION OF VARIOUS COMMUNICATION PROTOCOLS ON FPGA”** submitted at **Jaypee University of Information Technology, Solan, India,** is an authentic record of my work carried out under the supervision of **DR. HARSH SOHAL**. I have not submitted this work elsewhere for any other degree or diploma.

HARSHIT JAIN

SHAURYA MEHROTRA

Department of Electronics and Communication Engineering

Jaypee University of Information Technology, Waknaghat, India

Date: 13 MAY, 2019.

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

(Established by H.P. State Legislative vide Act No. 14 of 2002)
P.O. Waknaghat, Teh. Kandaghat, Distt. Solan - 173234 (H.P.) INDIA
Website: www.juit.ac.in
Phone No. (91) 01792-257999
Fax: +91-01792-245362

CERTIFICATE

This is to certify that the work reported in the B.Tech project report entitled **"IMPLEMENTATION OF VARIOUS COMMUNICATION PROTOCOLS ON FPGA"** which is being submitted by **HARSHIT JAIN(151053) & SHAURYA MEHROTRA(151081)** in fulfillment for the award of Bachelor of Technology in Electronics and Communication Engineering by the Jaypee University of Information Technology, is the record of candidate's own work carried out by him/her under my supervision. This work is original and has not been submitted partially or fully anywhere else for any other degree or diploma.

DR. HARSH SOHAL

Assistant Professor (Senior Grade)
Department of Electronics & Communication Engineering
Jaypee University of Information Technology, Waknaghat.

ACKNOWLEDGMENT

We take this opportunity to express our gratitude to our supervisor **DR. HARSH SOHAL,** for his insightful advice, motivating suggestions, invaluable guidance, help and support in successful completion of this project and also for his constant encouragement and advice throughout our Bachelor's program.

The in-house facilities provided by the department throughout the Bachelor’s program are also equally acknowledgeable. We would like to convey our thanks to the teaching and non-teaching staff of the Department of Electronics and Communication Engineering for their invaluable help and support.

HARSHIT JAIN	SHAURYA MEHROTRA
151053	151081

LIST OF FIGURES

ABSTRACT

This project presents the implementation of various communication protocols on three different fpga chips (zynq-7000, kintex-7 & artix-7) in order to know the performance and to evaluate the power requirements of the protocols on these FPGA chips. These protocols are I2C, SPI & UART which widely serves the industrial purpose. The comparative analysis of these protocols helps us to understand the advantages of one over other and to make the specific requirements in the application areas. Our project will give a better understanding of these protocols in an analytical way.

CHAPTER 1

INTRODUCTION

Communication in technical terms is a process by which a transmitter transmit a message signal whether analog or digital which passes through a channel which is a physical environment can be wireless or wired , which is further received by a receiver again it can be analog or digital further decoded into readable format . The message when transmitted has certain faults or areas to look after associated with it, like the correct transmission, correct reception . All this is ensured by the sender and the receiver through there error free synchronization . So the success of any communication is measured by its results at the receivers end. To make sure this reliability in communication certain protocols are made . these protocols utilize error detection and correction techniques which we will see in near future in there detailed explanation.

These protocols are:

1. SPI (Serial Peripheral Interface)
2. UART (Universal Asynchronous Receiver Transmitter)
3. I2C (Inter Integrated Circuit)

1.1FPGA

1.2.1 Zynq-7000

It is a programmable system integrated fpga embedded with A-9 processing unit, can be use to program artix-7 and kintex-7 based SoCs. It is empowered with 28nm design chip with upto 6.6milloin small cellular units offering a speed of 6.25 gigabytes per sec. to 12.5 gigabytes per sec. It offers wide variety of software application installation including multi-camera with helping framework.

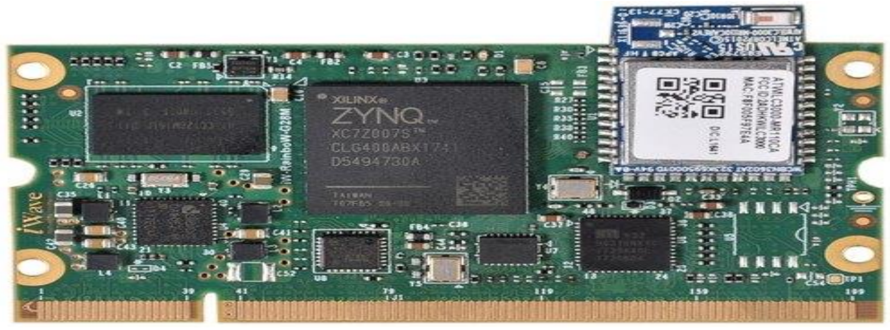


Figure 1.1

1.2.2 KINTEX-7

Programmable System Integration is upto to 478K logic cells.Increased System Performance to 34Mb BRAM, DDR3-1866. Cost is reduce to half the price of similar density to 40nm device. Power is reduced to 50% than previous generation 40nm devices & has accelerated design productivity optimized architecture.

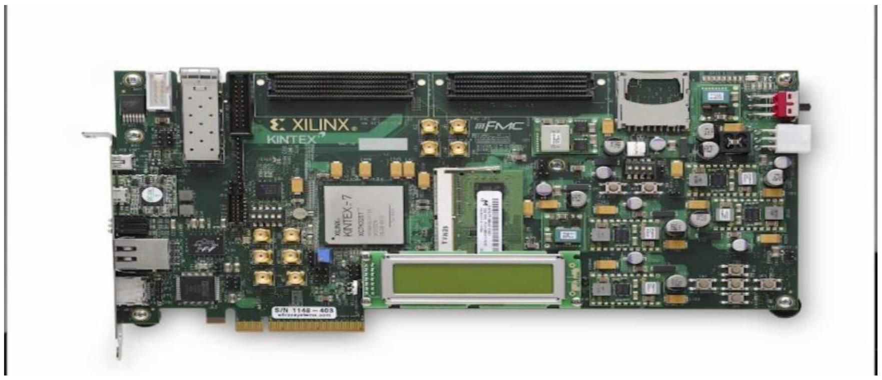


Figure 1.2

1.2.3 ARTIX -7

Programmable System Integration is upto to 215K logic cells. Increased System Performance to 13Mb BRAM, DDR3-1066. Has a small wire bond packaging with upto $5 analog component cost saving. Has 65% lower static and 50% lower power than 45nm generation devices, scalable optimized architecture.

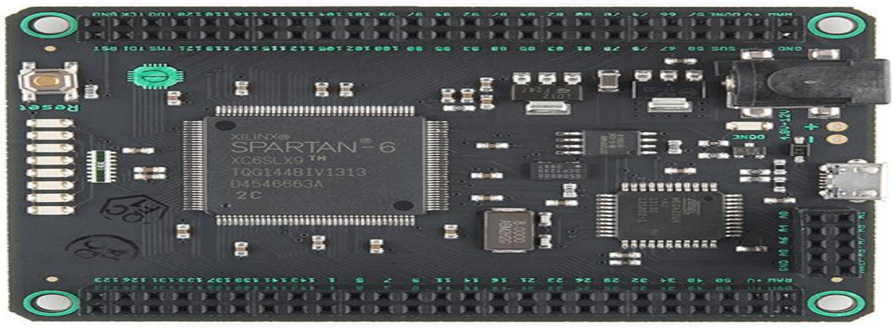


Figure 1.3

1.2LITERATURE SURVEY

In [1] the authors has done performance analysis of the FIR filter using four FPGA on Vivado design suite-Xilinx. These FPGAs are zynq-7000, Artix-7, kintex-7 & kintex ultrascale. On these fpga chips performance analysis is subjected to power evaluation, analysis of delay, latency and interval of 3 FPGA and 1 SOC. From [1] we have learned how to do power evaluations on various fpga which is the main abstract of our project.

[1] has helped us to get a better understanding of how these fpga work and how there timing diagrams can be evaluated to know about there performance and about various methodology to improve the efficiency of these communication protocols through these fpga chips.

CHAPTER -2

PROTOCOLS

2.1 I2C (INTER-INTEGRATED CIRCUIT)

2.1.1 INTRODUCTION

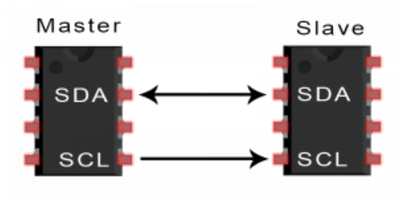


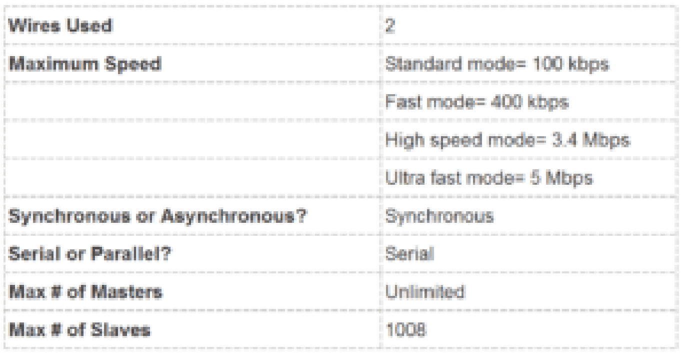
Figure 2.1

[2]I2C is a synchronous protocol like SPI and have a feature of multiple masters and multiple slaves like SPI. It requires only two wires like UART for transmission and reception of data. These two wires are:

SDA- It is serial data line which is use to send and receive data.

SCL- It is serial clock line which carries the clock, require for communication between masters and slaves.

It is a serial communication protocol therefore bit by bit transmission takes place.



Wires Used	2
Maximum Speed	Standard mode= 100 kbps
	Fast mode= 400 kbps
	High speed mode= 3.4 Mbps
	Ultra fast mode= 5 Mbps
Synchronous or Asynchronous?	Synchronous
Serial or Parallel?	Serial
Max # of Masters	Unlimited
Max # of Slaves	1008

FIGURE2.2

2.1.2 WORKING

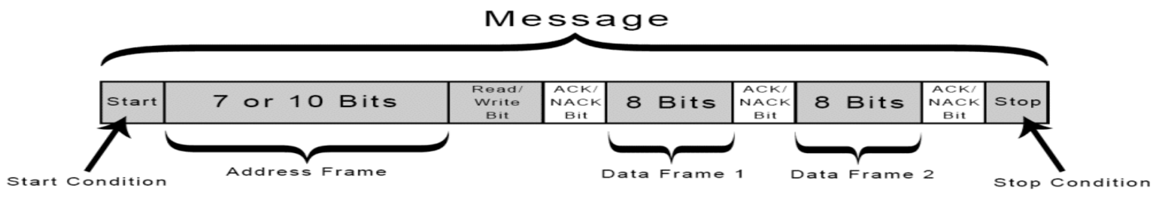


Figure 2.3

<u>START CONDITION</u>: The serial data line moves from a volt. level'1' to a volt. level'0', before the serial clock line moves from voltage level '1' to voltage level '0'.

<u>STOP CONDITION</u>: The serial data line moves from a volt. level '0' to a volt. level'1', after the SCL line moves from '0' to '1'.

<u>ADDRESS FRAME</u>: The size of this frame is about 7 up to 10 bit sequence unique to each slave which identifies the slave when the master wants to talk to that particular slave.

<u>R/W BIT</u>: One bit is used to specify whether the master wants to send data to the slave that is low voltage level or wants to receive data from it that is high voltage level.

ACK/NACK BIT: ACK- Acknowledgment, it is attached with each frame by the receiver if it successfully receive the data frame. NACK-Negative Acknowledgement, it is attached with each frame by the receiver if it do not receive the data frame.

ADDRESSING: The address frame is the first frame after the start bit in a new data frame. Each slave receives the address send by the master to whom it wants to communicate. If the address matches with the slaves own address than it send ACK bit to master showing that is ready to communicate.

READ/WRITE BIT: At the end of address frame there is a read/write bit. If master wants to send data then R/W bit is set to '0'else it set to '1'.

THE DATA FRAME: It is a 8 bit long sequence of binary numbers, sent in MSB first sequence, then ACK/NACK bit at its end to verify its correct transmission.

STEPS OF I2C DATA TRANSMISSION

1. The SDA line made from '0' to '1' before SCL line made from '1' to' 0' so that master begins the start condition to each connected slave.

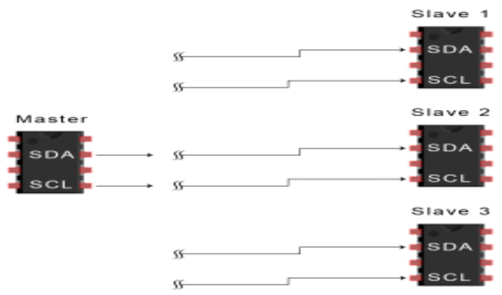


Figure 2.4

2. Now 7 to 10 bit of slave address is send by master to each and every slave and R/W bit is also send.

3. Every slave compares its address to the slave address send by the master, if this address matches with the slaves own address then it send an ACK bit.

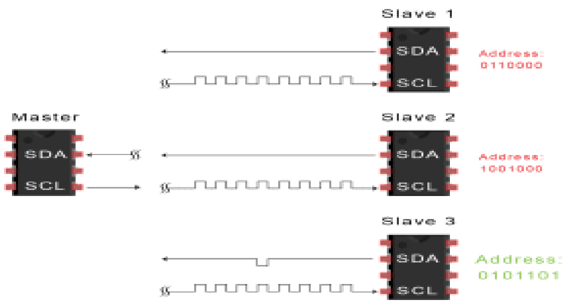


Figure 2.5

4. Now the data frame is send by master to the slave.

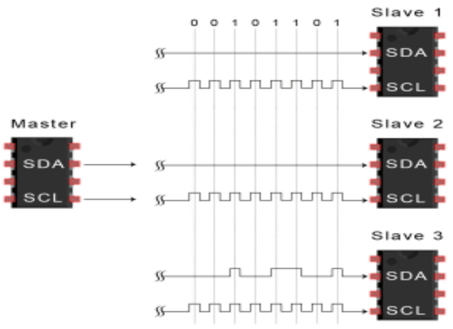


Figure 2.6

5. To admit the successful reception of the data frame the receiver send back the ACK bit to the sender.

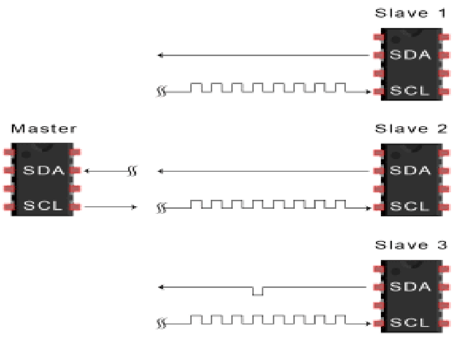


Figure 2.7

6. Now SCL is made ‘1’ before making SDA ‘1’ in order to ensure the stop condition by the master.

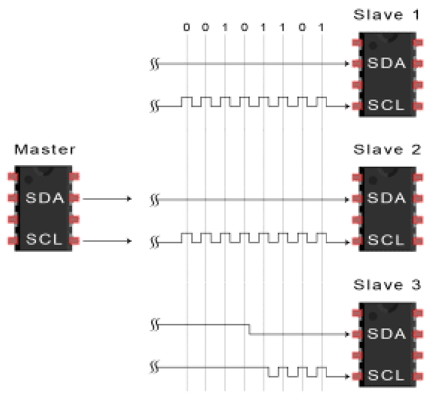


Figure 2.8

2.1.3 ADVANTAGES AND DISADVANTAGES

I2C seems to bit complicated to implement as compared to other protocols but it has certain added advantages in comparison to its counterpart:

ADVANTAGES

- Only 2 wires SDA & SCL are required.
- Facility of multi master-multi slave is provided.
- Correct transmission can be ensured through ACK /NACK bit.
- Simplicity in hardware implementation in comparison to its counterparts.
- Popularly known protocol.

DISADVANTAGES

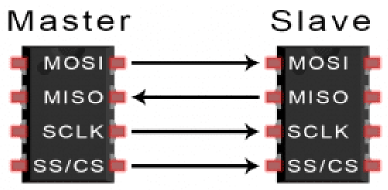
- Data rate is slow and hardware is more complex than SPI.
- Limitation in terms of data frame size that is limited to only 8 bit.

2.2 SPI(SERIAL PERIPHERAL INTERFACE)

2.2.1 INTRODUCTION

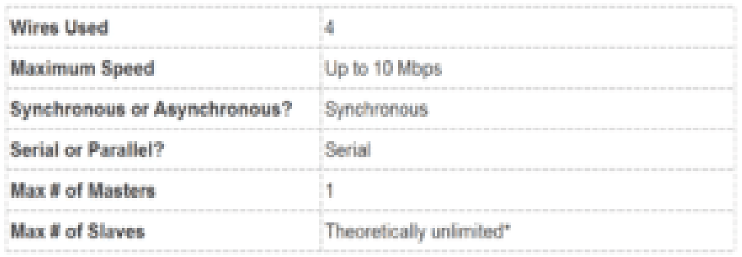
SPI is a three wire full duplex (transmitter and receiver works simultaneously). Basically operated at very high speed .Master generates the clock pulse .Master and slaves generate the data addresses are stored in peripheral devices two wire buses are used by many devices.

It is asynchronous type like I2C that is a clock signal is required to drive the data transmission process. It is generally use to transfer data at higher rate than I2C and UART. The simplicity in its hardware configuration makes it easy to use and its master –slave integration is one of the region for its wide use.



.

Figure 2.9



Wires Used	4
Maximum Speed	Up to 10 Mbps
Synchronous or Asynchronous?	Synchronous
Serial or Parallel?	Serial
Max # of Masters	1
Max # of Slaves	Theoretically unlimited*

Figure 2.10

2.2.2 WORKING

THE CLOCK: The clock signal has a specific purpous in this protocol, in order to synchronize master with its slaves it plays amajor role in it. The frequency of clock determines the speed of data transmission in SPI.

SLAVE SELECT: The slave which needs to be communicated is selected by the master through making chip select line to volt. Level ‘0’. In general or in free State the chip select line remain at volt. Level ‘1’.

MULTIPLE SLAVES: SPI has two mode of operation one is in single master-slave configuration and other is in single master-multi slave configuration.
The later one depends on the number of chip select lines available with the master IC.

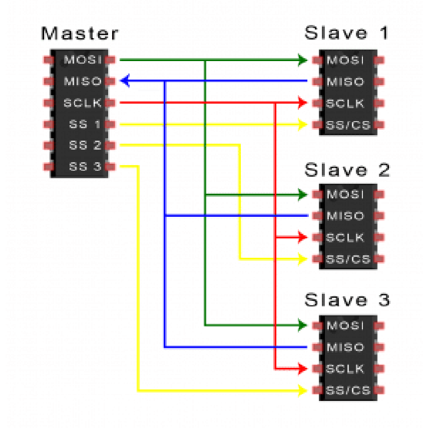


Figure 2.11

On availability of only 1 select line , only 1 slave device can be use to communicate.

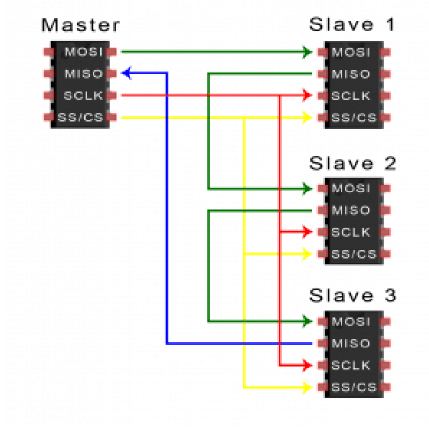


Figure 2.12

<u>MOSI & MISO</u>

The data can be sent back to master from slave through the MISO. The data is sent in a LSB format by the slave to the master.

STEPS OF SPI DATA TRANSMISSION

1.Firstly the clock signal is send by the master.

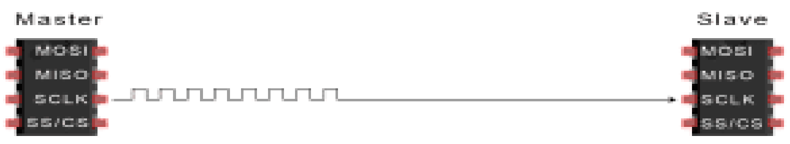


Figure 2.13

2. In order to activate the salve, the master makes the chip select line to active low that is ‘0’.

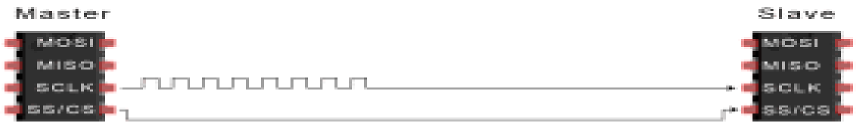


Figure 2.14

3. 1 bit at a time is send to the slave device, that is it ensures the serial communication of data.

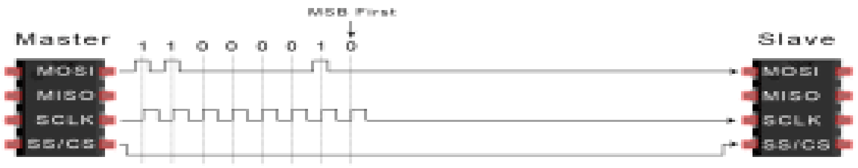


Figure 2.15

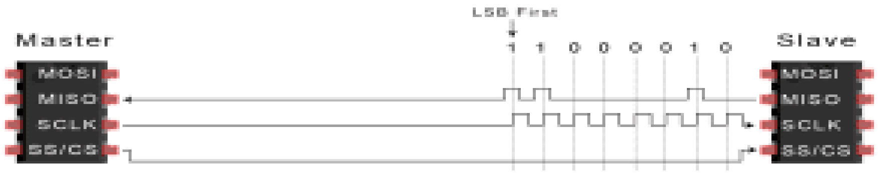


Figure 2.16

2.2.3 ADVANTAGES & DISADVANTAGES

There are certain disadvantages associated with SPI but its overwhelming advantages cover the region of doubt on his protocol.

ADVANTAGES

- Start bit & stop bit are not present, in order to stream data continuously.
- Unlike I2C complications are little less.

- Higher data rate than I2C , around 2 times of former.
- Transmission and reception of the data is done simultaneously.

DISADVANTAGES

- More no. of wires are required that is 4 as compared to I2C.
- No ACK/NACK bit is used .
- Less reliability as compare to its counterparts.
- Only single master is allowed.

2.3 UART (Universal Asynchronous Receiver Transmitter)

2.3.1 INTRODUCTION

UART is based on single master single slave concept, although it has no concept of master and slave, simplex transmission takes place. It is an asynchronous protocol unlike its counterparts which are synchronous, hence no clock is required. It converts parallel data received from CPU into serial form and then transmits it to receiver UART. Like I2C only 2 wires are needed for transmission & reception of data.

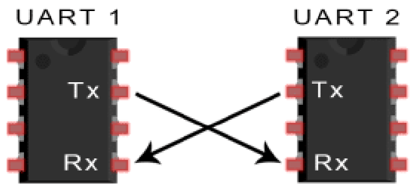
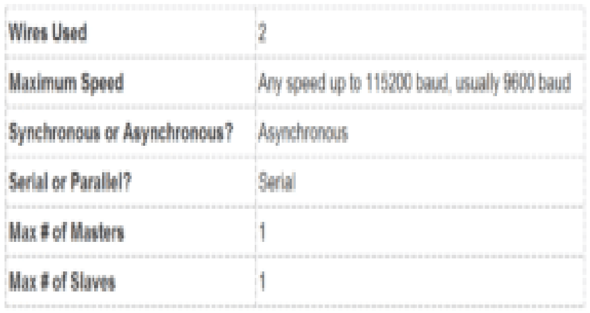


Figure 2.17



Wires Used	2
Maximum Speed	Any speed up to 115200 baud, usually 9600 baud
Synchronous or Asynchronous?	Asynchronous
Serial or Parallel?	Serial
Max # of Masters	1
Max # of Slaves	1

FIGURE 2.18

2.3.2 WORKING

The transmitting data is arrange in the form of packets. Each and every packet contain a start bit , five to nine bits of transmitting data , a parity bit which is optional and one or two stop bit.

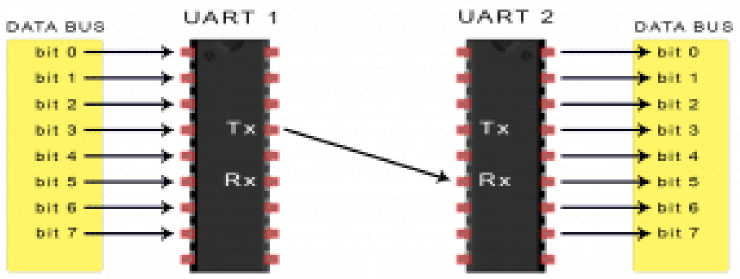


Figure 2.19

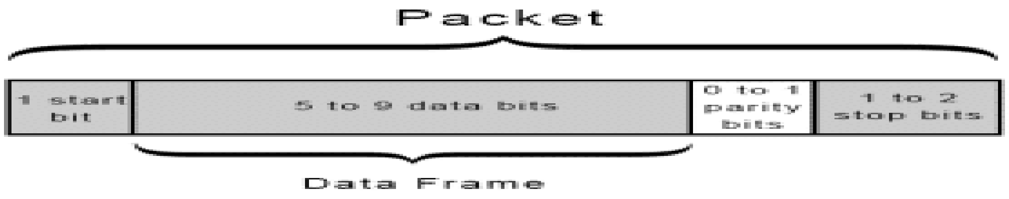


Figure 2.20

START BIT: When there is no transmission of data, the data line is on active high. Data transmission starts by moving transmission line from active high to active low.

DATA FRAME: The data frame is of 5 to 9 bit long contains the data which need to be transferred. If parity bit is used then data is 8 bit long else 9 nit long.

STOP BITS: To stop the transmission of data a stop bit is send by the sender UART to the receiver UART.

STEPS OF UART TRANSMISSION

1. The data bus sends data in parallel form to the transmitting UART

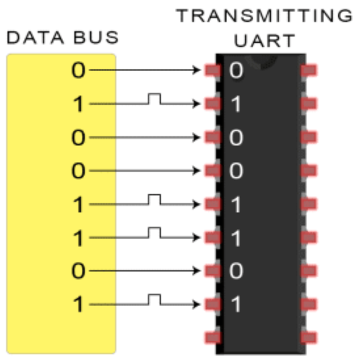


Figure 2.21

2. The start bit, parity bit, and the stop bit(s)are added by the transmitting UART to the data frame:

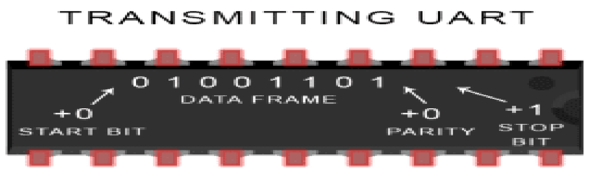


Figure 2.22

3. This transmission is done at specified baud rate

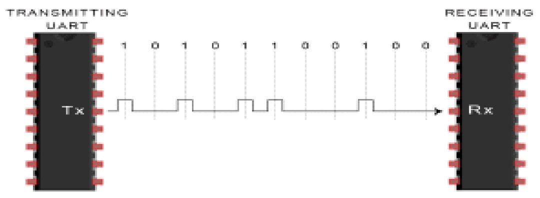


Figure 2.23

4. The start bit is discarded by receiving UART, data frame discards parity bit and stop bit:

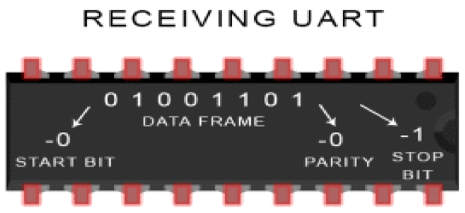


Figure 2.24

5. The serial data back is converted into parallel by receiving UART and transfers it to the data bus.

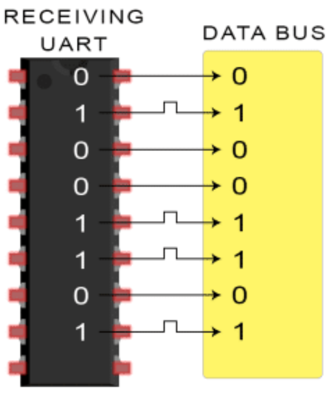


Figure 2.25

2.3.3 ADVANTAGES AND DISADVANTAGES

ADVANTAGES

- It uses only 2 wires
- Clock signal is not necessary
- Error checking is done with UAR
- Widely used method and well documented

DISADVANTAGES

- Data frame size is limited only to 9 bits
- Multi. Master-slave systems are not supported
- The baud rates of each communicating UART must be under 10%.

CHAPTER - 3

VIVADO DESIGN SUIT-XILINX

3.1 SPECIFICATION

Vivado design suit is used for desigining FPGA based applications. It is used for speeding purpose such as software design which define IP . High Level Synthesis also for block based .It is used for accelerating verifications such as Vivado logic simulation, accelerated verification with languages like C,C++ or System C & also for IP purpose . It is used for accelerating implementation such as 4X fast implementation upto 3 speed grade performance advantage for the low end and range vary upto 35% of power benefit in high end also as 20% better design denity**.**

3.2APPLCATIONS

- Aerospace & Defence
- Automative
- Broadcast
- Industrial
- Medical

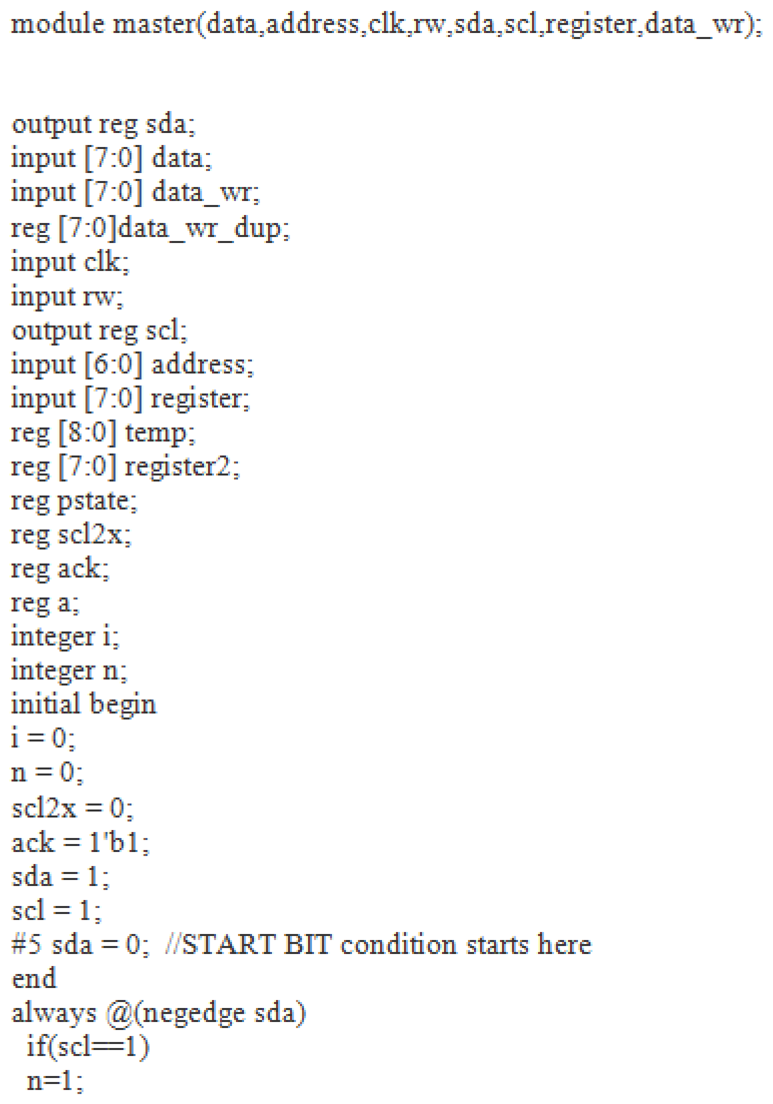
CHAPTER - 4

RESULTS

4.1 CODES & SIMULATION

4.1.1 I2C

Master module and its inputs:



```
module master(data,address,clk,rw,sda,scl,register,data_wr);


output reg sda;
input [7:0] data;
input [7:0] data_wr;
reg [7:0]data_wr_dup;
input clk;
input rw;
output reg scl;
input [6:0] address;
input [7:0] register;
reg [8:0] temp;
reg [7:0] register2;
reg pstate;
reg scl2x;
reg ack;
reg a;
integer i;
integer n;
initial begin
i = 0;
n = 0;
scl2x = 0;
ack = 1'b1;
sda = 1;
scl = 1;
#5 sda = 0;  //START BIT condition starts here
end
always @(negedge sda)
  if(scl==1)
  n=1;
```

```
always @(posedge clk)begin
ack = 0;
temp = {address,rw,ack};
register2 = register;
data_wr_dup = data_wr;
if(n==1 && rw==1)
repeat(50)begin
#2 scl <= !scl;n=0;
#1 scl2x <= !scl2x;n=0;
end
else if(n==1 && rw==0)
repeat(64) begin
#2 scl = !scl;
#1 scl2x = !scl2x;n=0;
end
end
always @(posedge clk)begin
if(i==25 && rw==1)
repeat(2)
#1 scl2x = !scl2x;
else if(i==32 && rw==0)
repeat(2)
#1 scl2x = !scl2x;end
always @(posedge clk)begin
if(i==25 && rw==1)
repeat(2)
#1 scl2x = !scl2x;
else if(i==32 && rw==0)
repeat(2)
```

```
#1 scl2x = !scl2x;
else if(i==32 && rw==0)
repeat(2)
#1 scl2x = !scl2x;end
always @(posedge clk)begin
if(i==25 && rw==1)
repeat(2)
#1 scl2x = !scl2x;
else if(i==32 && rw==0)
repeat(2)
#1 scl2x = !scl2x;end
always @(posedge scl2x)begin
if(i<=9)begin
sda = temp[8];
temp = temp<<1;
end
else if(i==12 || i==13)
sda = 1'b0;
else if(i>=14)begin
sda = register2[7];
register2 = register2<<1;
end
if(rw==0 && i>=23)begin
sda = data_wr_dup[7];
data_wr_dup = data_wr_dup<<1;
end
i = i + 1;


if(i>32 && rw ==0)
sda= 1;
else if(i>25 && rw==1)
sda = 1;
end
slave slv(data,sda,scl);
endmodule
```

Slave Module and its inputs-:

```
 module slave(out,sda,scl);
 input sda;
 input scl;
 output reg [7:0]out;
 integer j = 0;
 reg [6:0]temp;
 reg [7:0]add;
 reg rw;
 reg [7:0]register_address;
 reg bitin;
 reg [7:0]storage[0:38];
 initial
 storage[37]=16;
 parameter address = 7'b1101001;
 always @(posedge scl)begin
 //if({sda,scl}==2'b01)begin
 bitin = sda;
 if(j<8)
 temp = {temp,bitin};
 if(j==8)
 if(bitin==0)
  rw = 0;
 else
  rw = 1;
 j =j +1 ;
 if(temp==address && (j>15 && j<24) && rw==1)begin
  add = {add,bitin};
 end
 if(temp==address && rw == 0 && j>15 && j!=24 && j<33)begin
 add = {add,bitin};
 end
else if(j==24)
 register_address = add;
if(j==33 && rw==0)
storage[register_address]=add;
out = storage[add];
end
endmodule
```

Simulation

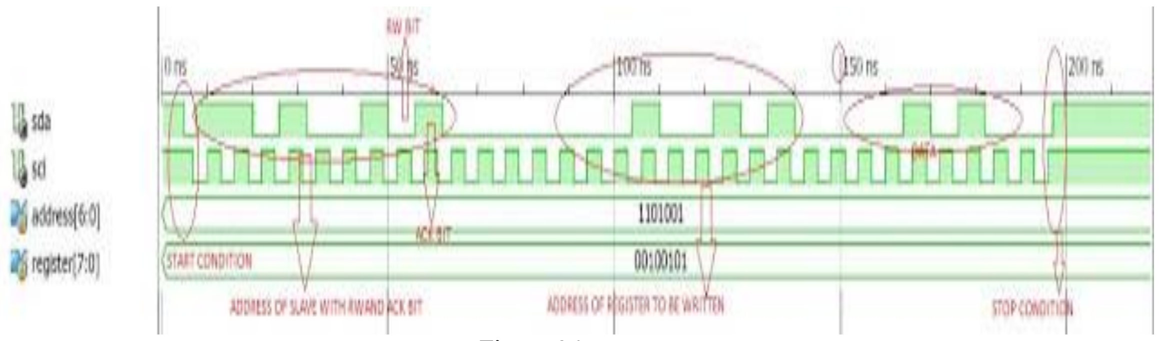


Figure 4.1

4.1.2 SPI

Master module and its input:

```
always@ (negedge clk or negedge rstb) begin
if(rstb==0)
cur<=finish;
else
cur<=nxt;
end
//FSM I/O
always @ (start or cur or nbit) begin
nxt=cur;
clr=0;
shift=0;//ss=0;
case (cur)
idle:begin
if (start==1)
begin
case (cdiv)
2'b00: mid=2;
2'b01: mid=4;
2'b10: mid=8;
2'b11: mid=16;
```

```
endcase
shift=1;
done=1'b0;
nxt=send;
end
end //idle
send:begin
ss=0;
if (nbit!=8)
begin shift=1; end
else begin
rdata=rreg;
done=1'b1;
nxt=finish;
end
end//send
finish:begin
shift=0;
ss=1;
clr=1;
nxt=idle;
end
default: nxt=finish;
endcase
end//always
```

CLOCK GENERATOR BLOCK:

```
always@(negedge clk or posedge clr) begin
if(clr==1)
begin cnt=0; sck=1; end
else begin
if(shift==1) begin
cnt=cnt+1;
if(cnt==mid) begin
sck=~sck;
cnt=0;
end //mid
end //shift
end //rst
end
```

Tx BLOCK:

```
always@ (negedge sck or posedge clr) begin
if(clr==1) begin
treg=8'hFF;
 dout=1;
```

```
end
else begin
if(nbit==0) begin //load data into TREG
treg=tdat; dout=mlb?treg[7]:treg[0];
end //nbit_if
else begin
if(mlb==0) //LSB first, shift right
begin treg={1'b1,treg[7:1]}; dout=treg[0]; end
else//MSB first shift LEFT
begin treg={treg[6:0],1'b1}; dout=treg[7]; end
end
end //rst
end
```

Rx BLOCK:

```
always@(posedge sck or posedge clr ) begin // or negedge rstb
if(clr==1)  begin
nbit=0; rreg=8'hFF;  end
else begin
if(mlb==0) //LSB first, din@msb -> right shift
begin  rreg={din,rreg[7:1]};  end
else  //MSB first, din@lsb -> left shift
begin  rreg={rreg[6:0],din};  end
nbit=nbit+1;
end //rst
end
```

Slave module & its input:

```
always @(posedge sck or negedge rstb)
begin
if (rstb==0)
begin rreg = 8'h00;  rdata = 8'h00; done = 0; nb = 0; end   //
else if (!ss) begin
if(mlb==0)
begin rreg ={sdin,rreg[7:1]}; end
else
begin rreg ={rreg[6:0],sdin}; end
nb=nb+1;
if (nb! =8) done=0;
else begin rdata=rreg; done=1; nb=0; end
end
end

//send to sdout
always @(negedge sck or negedge rstb)
begin
if (rstb==0)
```

```
begin treg = 8'hFF; end
else begin
if (!ss) begin
if(nb==0) treg=tdata;
else begin
if(mlb==0)
begin treg = {1'b1,treg[7:1]}; end
else
begin treg = {treg[6:0],1'b1}; end
end
end //!ss
end //rstb
end //always
```

Simulation

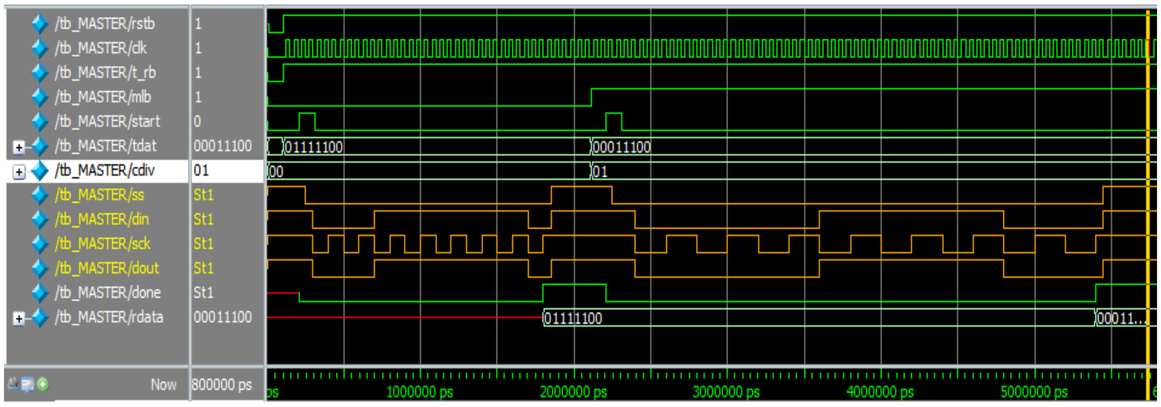


Figure 4.2

4.1.3 UART

UART RX Logic

```
always @ (posedge rxclk or posedge reset)
if (reset) begin
rx_reg <= 0;
rx_data <= 0;
rx_sample_cnt <= 0;
rx_cnt <= 0;
rx_frame_err <= 0;
rx_over_run <= 0;
rx_empty <= 1;
rx_d1 <= 1;
rx_d2 <= 1;
rx_busy <= 0;
end else begin
// Synchronize the asynch signal
rx_d1 <= rx_in;
rx_d2 <= rx_d1;
// Uload the rx data
if (uld_rx_data) begin
rx_data <= rx_reg;
rx_empty <= 1;
end
// Receive data only when rx is enabled
if (rx_enable) begin
// Check if just received start of frame
if (!rx_busy && !rx_d2) begin
rx_busy <= 1;
rx_sample_cnt <= 1;
rx_cnt <= 0;
end



// Start of frame detected, Proceed with rest of data
if (rx_busy) begin
rx_sample_cnt <= rx_sample_cnt + 1;
// Logic to sample at middle of data
if (rx_sample_cnt == 7) begin
if ((rx_d2 == 1) && (rx_cnt == 0)) begin
rx_busy <= 0;
end else begin
rx_cnt <= rx_cnt + 1;
// Start storing the rx data
```

```
if (rx_cnt > 0 && rx_cnt < 9) begin
rx_reg[rx_cnt - 1] <= rx_d2;
end
if (rx_cnt == 9) begin
rx_busy <= 0;
// Check if End of frame received correctly
if (rx_d2 == 0) begin
rx_frame_err <= 1;
end else begin
rx_empty <= 0;
rx_frame_err <= 0;
// Check if last rx data was not unloaded,
rx_over_run <= (rx_empty) ? 0 : 1;
end
end
end
end
end
end
if (!rx_enable) begin
rx_busy <= 0;
end
end
```

UART TX Logic

```
always @ (posedge txclk or posedge reset)
if (reset) begin
tx_reg <= 0;
tx_empty <= 1;
tx_over_run <= 0;
tx_out <= 1;
tx_cnt <= 0;
end else begin
if (ld_tx_data) begin
if (!tx_empty) begin
tx_over_run <= 0;
end else begin
tx_reg <= tx_data;
tx_empty <= 0;
end
end
if (tx_enable && !tx_empty) begin
tx_cnt <= tx_cnt + 1;
if (tx_cnt == 0) begin
tx_out <= 0;
end
```

```
if (tx_cnt > 0 && tx_cnt < 9) begin
tx_out <= tx_reg[tx_cnt -1];
end
if (tx_cnt == 9) begin
tx_out <= 1;
tx_cnt <= 0;
tx_empty <= 1;
end
end
if (!tx_enable) begin
tx_cnt <= 0;
end
end
endmodule
```

Simulation

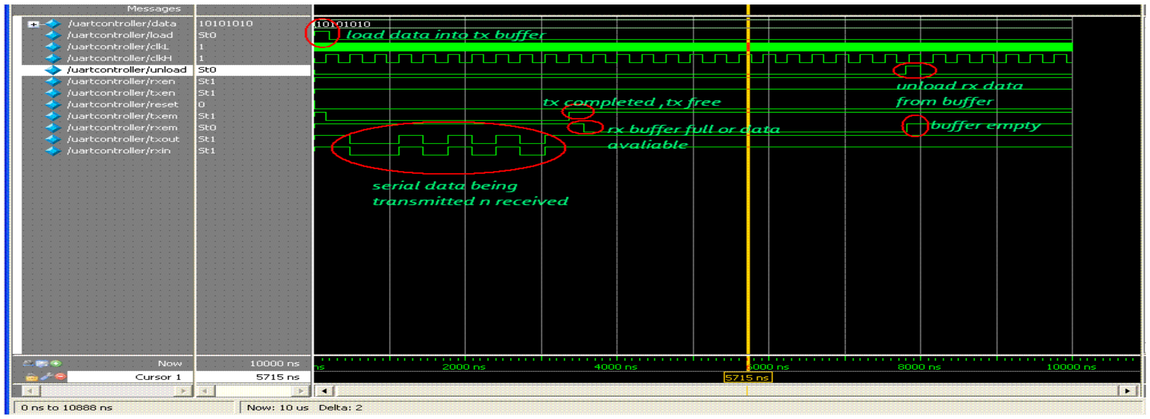


Figure 4.3

4.2 IMPLEMENTATION ON FPGA

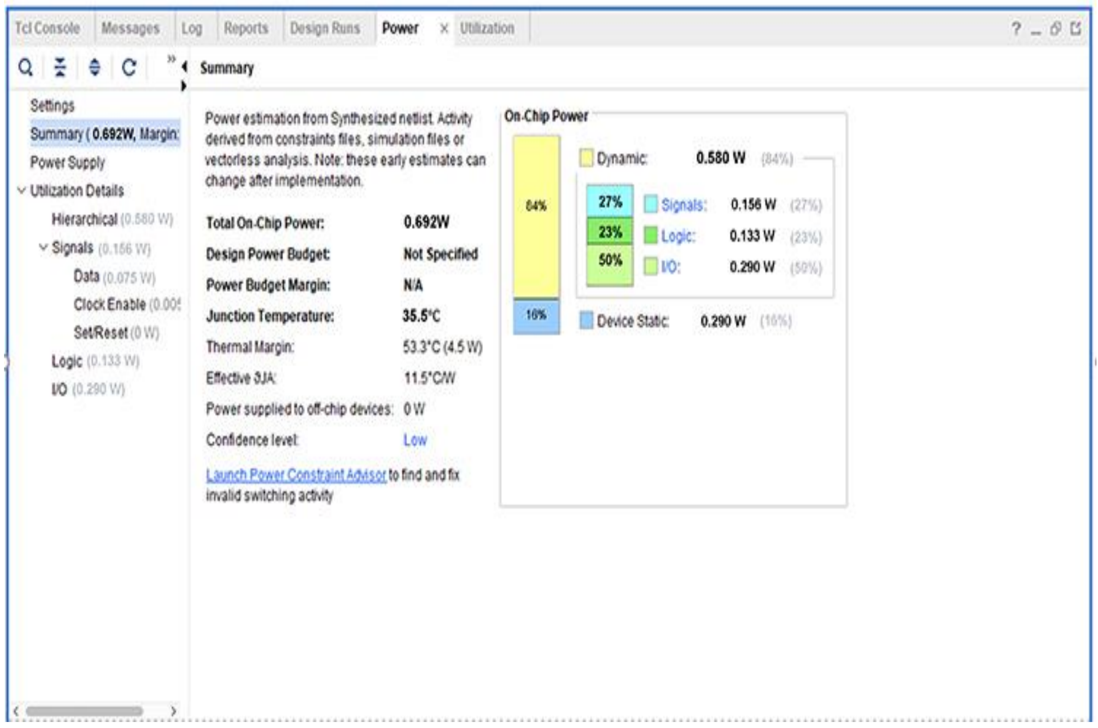
Now these protocols are implemented on three FPGAs, and there power analysis can be seen in the below mentioned figures. These FPGAs zynq-7000, kintex-7 & artix-7 .

4.2.1. ZYNQ-7000

Computing the performance on I2C, SPI, UART on ZYNQ-7000.

I2C

Fig. 4.4 depicts the power performance of I2C on zynq-7000.



Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.692W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**35.5°C**
Thermal Margin:	53.3°C (4.5 W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

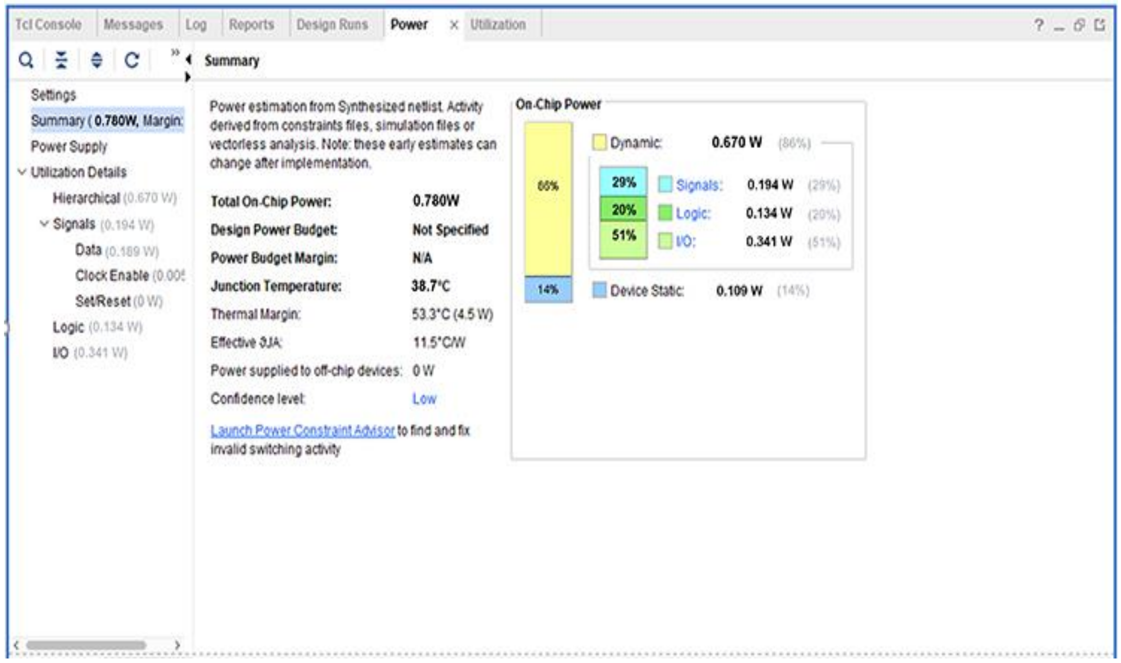
Launch Power Constraint Advisor to find and fix invalid switching activity

Dynamic:	0.580 W	(84%)
Signals:	0.156 W	(27%)
Logic:	0.133 W	(23%)
I/O:	0.290 W	(50%)
Device Static:	0.290 W	(16%)

Figure 4.4

SPI

Fig. 4.5 depicts the power performance of SPI on zynq-7000



Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.780W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**38.7°C**
Thermal Margin:	53.3°C (4.5 W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

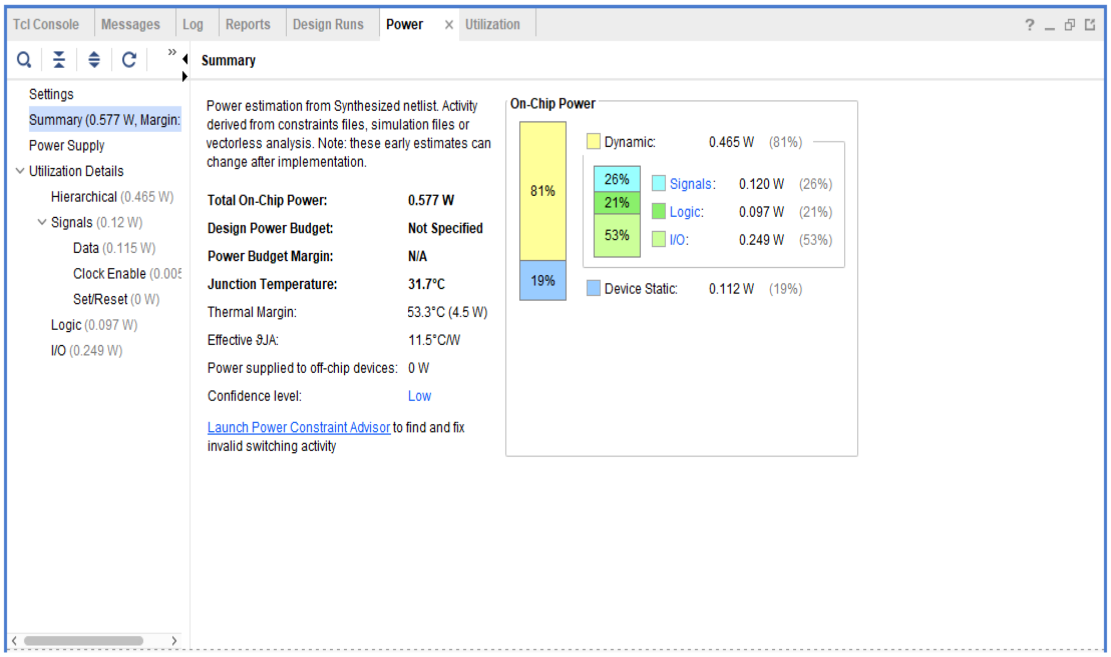
Launch Power Constraint Advisor to find and fix invalid switching activity

Dynamic:	**0.670 W**	(86%)
Signals:	**0.194 W**	(29%)
Logic:	**0.134 W**	(20%)
I/O:	**0.341 W**	(51%)
Device Static:	**0.109 W**	(14%)

Figure 4.5

UART

Fig. 4.6 depicts the power performance of UART on zynq-7000



Settings
Summary (0.577 W, Margin:
Power Supply
Utilization Details
Hierarchical (0.465 W)
Signals (0.12 W)
Data (0.115 W)
Clock Enable (0.005
Set/Reset (0 W)
Logic (0.097 W)
I/O (0.249 W)

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.577 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**31.7°C**
Thermal Margin:	53.3°C (4.5 W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

Dynamic:	0.465 W	(81%)
Signals:	0.120 W	(26%)
Logic:	0.097 W	(21%)
I/O:	0.249 W	(53%)
Device Static:	0.112 W	(19%)

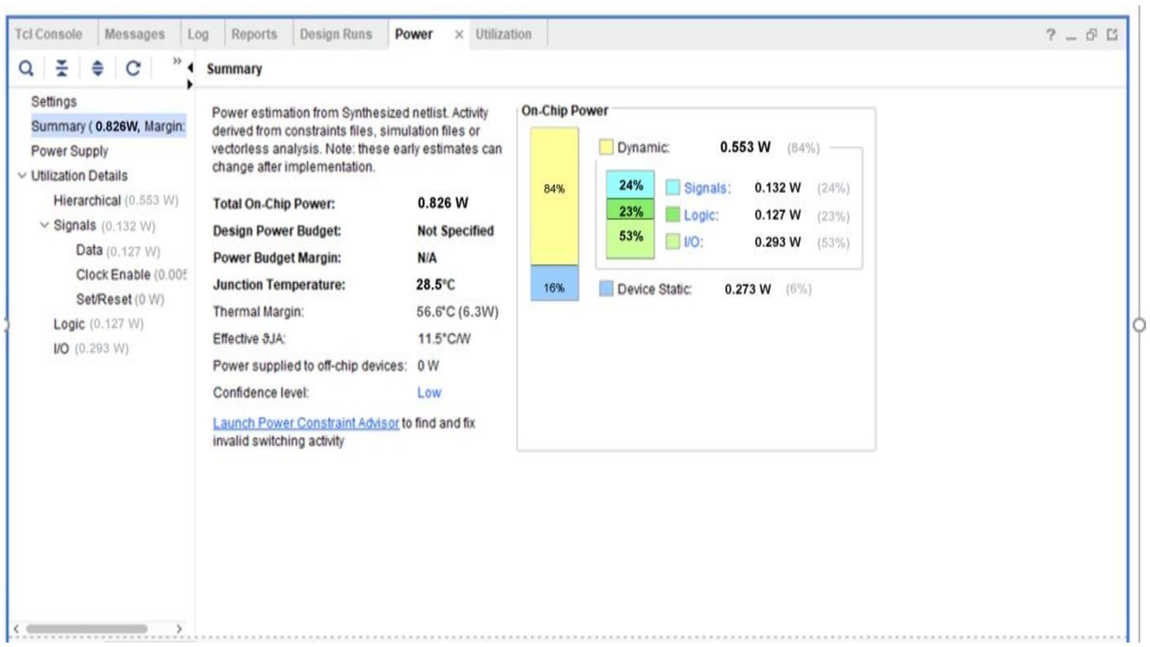
Figure4.6

4.2.2. KINTEX-7

Computing the performance on I2C, SPI, UART on KINTEX-7.

I2C

Fig. 4.7 depicts the power performance of I2C on kintex-7.



Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.826 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**28.5°C**
Thermal Margin:	56.6°C (6.3W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

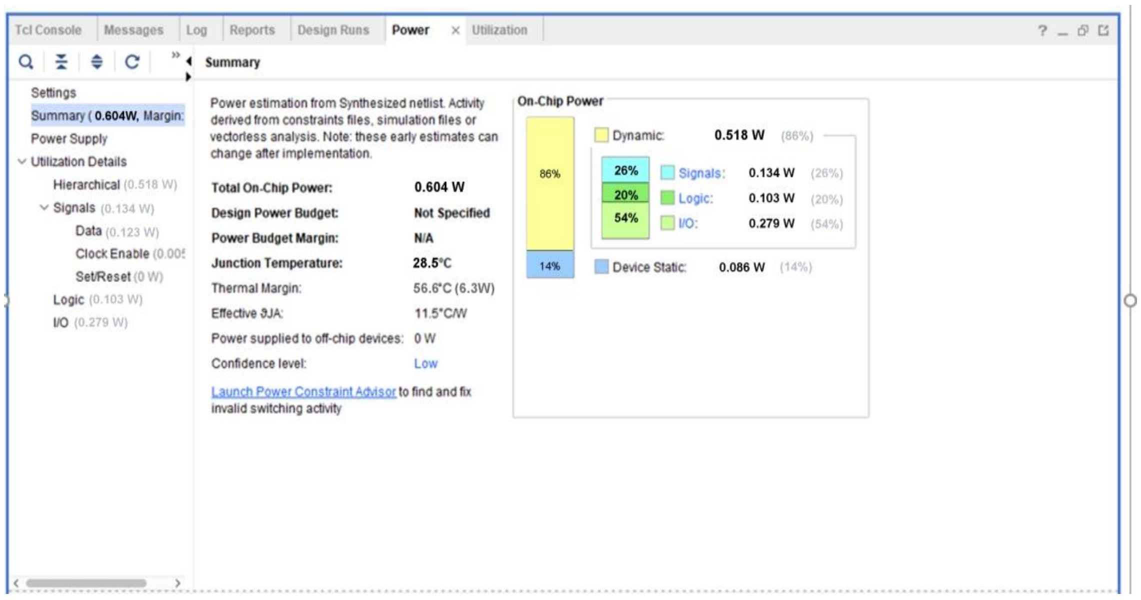
Launch Power Constraint Advisor to find and fix invalid switching activity

Dynamic:	**0.553 W**	(84%)
Signals:	0.132 W	(24%)
Logic:	0.127 W	(23%)
I/O:	0.293 W	(53%)
Device Static:	0.273 W	(6%)

Figure 4.7

SPI

Fig. 4.8 depicts the power performance of I2C on kintex-7.



Settings
Summary (**0.604W,** Margin:
Power Supply
Utilization Details
Hierarchical (0.518 W)
Signals (0.134 W)
Data (0.123 W)
Clock Enable (0.005
Set/Reset (0 W)
Logic (0.103 W)
I/O (0.279 W)

Summary

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.604 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**28.5°C**
Thermal Margin:	56.6°C (6.3W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

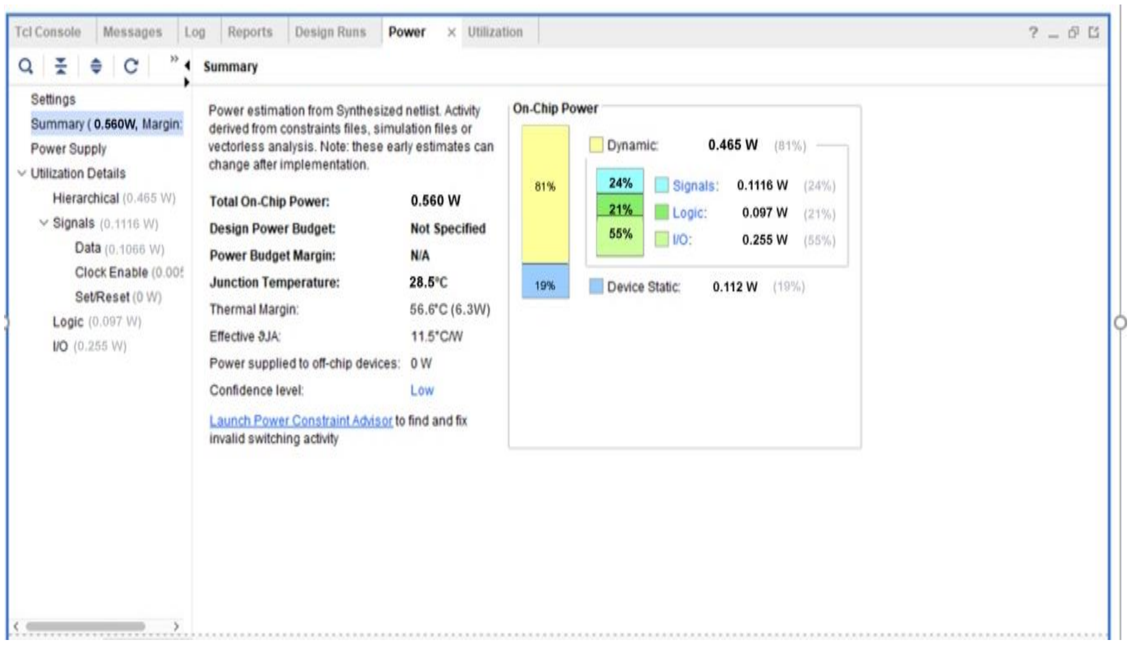
On-Chip Power

86%	Dynamic:	**0.518 W**	(86%)
26%	Signals:	**0.134 W**	(26%)
20%	Logic:	**0.103 W**	(20%)
54%	I/O:	**0.279 W**	(54%)
14%	Device Static:	**0.086 W**	(14%)

Figure 4.8

UART

Fig. 4.9 depicts the power performance of UART on kintex-7.



Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.560 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**28.5°C**
Thermal Margin:	56.6°C (6.3W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

Dynamic:	**0.465 W**	(81%)
Signals:	0.1116 W	(24%)
Logic:	0.097 W	(21%)
I/O:	0.255 W	(55%)
Device Static:	0.112 W	(19%)

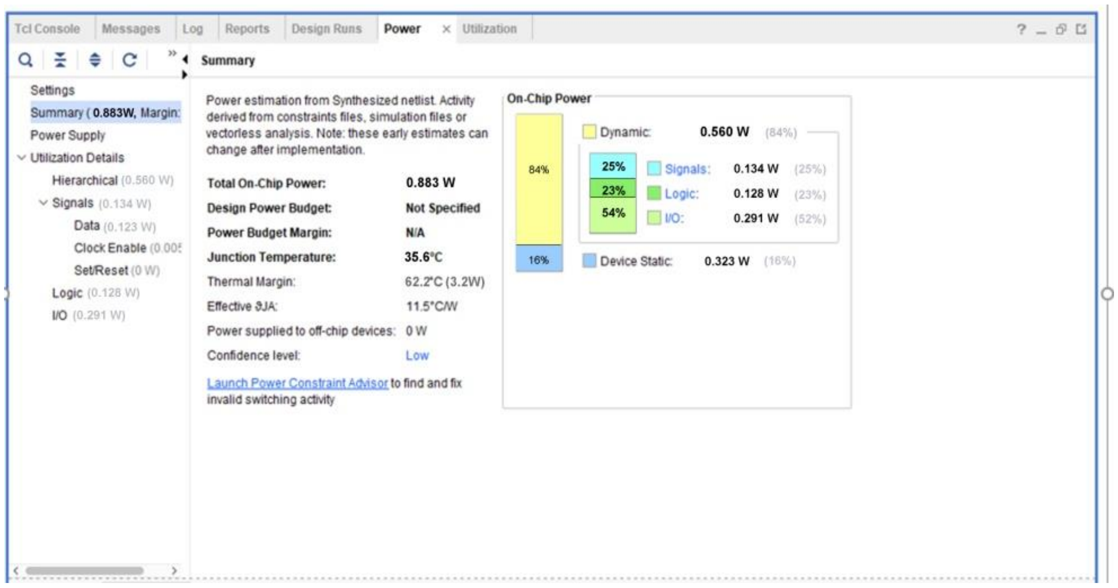
Figure 4.9

4.2.3. ARTEX-7

Computing the performance on I2C, SPI, UART on artix-7.

I2C

Fig. 4.10 depicts the power performance of I2C on kintex-7.



Settings
Summary (0.883W, Margin:
Power Supply
Utilization Details
Hierarchical (0.560 W)
Signals (0.134 W)
Data (0.123 W)
Clock Enable (0.005
Set/Reset (0 W)
Logic (0.128 W)
I/O (0.291 W)

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.883 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**35.6°C**
Thermal Margin:	62.2°C (3.2W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

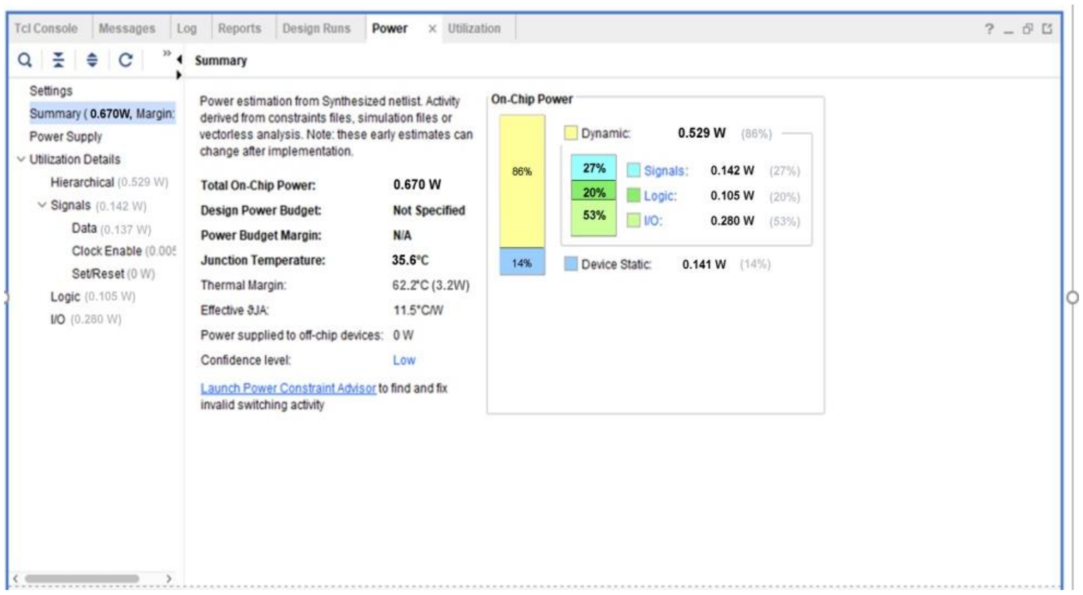
Launch Power Constraint Advisor to find and fix invalid switching activity

Dynamic:	**0.560 W**	(84%)
Signals:	0.134 W	(25%)
Logic:	0.128 W	(23%)
I/O:	0.291 W	(52%)
Device Static:	0.323 W	(16%)

Figure 4.10

SPI

Fig. 4.11 depicts the power performance of SPI on kintex-7.



Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.670 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**35.6°C**
Thermal Margin:	62.2°C (3.2W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

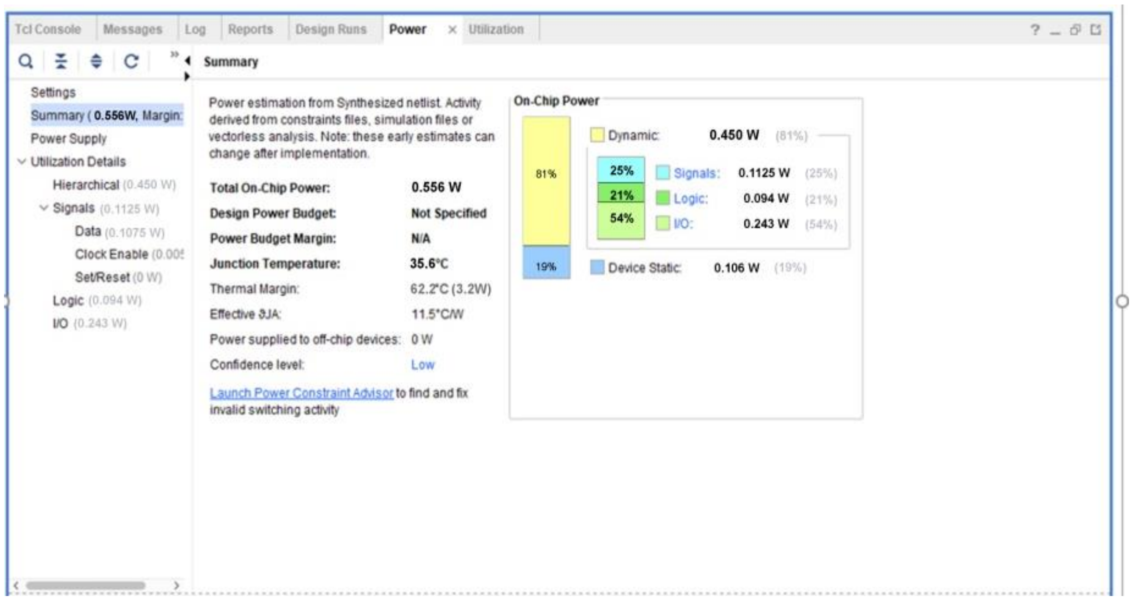
On-Chip Power

Dynamic:	**0.529 W**	(86%)
Signals:	0.142 W	(27%)
Logic:	0.105 W	(20%)
I/O:	0.280 W	(53%)
Device Static:	0.141 W	(14%)

Figure 4.11

UART

Fig. 4.12 depicts the power performance of UART on artix-7.



Settings
Summary (0.556W, Margin:
Power Supply
Utilization Details
Hierarchical (0.450 W)
Signals (0.1125 W)
Data (0.1075 W)
Clock Enable (0.00
Set/Reset (0 W)
Logic (0.094 W)
I/O (0.243 W)

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	0.556 W
Design Power Budget:	Not Specified
Power Budget Margin:	N/A
Junction Temperature:	35.6°C
Thermal Margin:	62.2°C (3.2W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:	0.450 W	(81%)
Signals:	0.1125 W	(25%)
Logic:	0.094 W	(21%)
I/O:	0.243 W	(54%)
Device Static:	0.106 W	(19%)

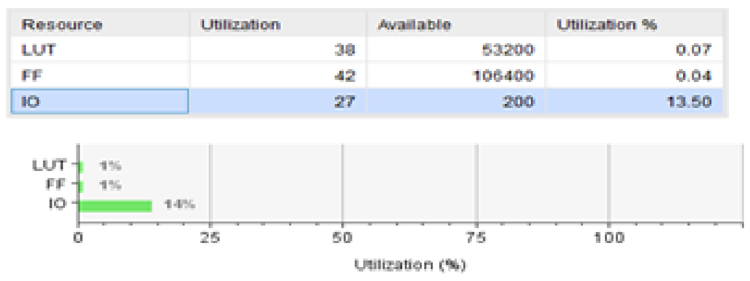
Figure 4.12

Comparative analysis of these FPGA:

Comparative analysis shows how these 3 FPGAs behave on similar look up table that is how they are utilizing the available flip-flops.

<u>ZYNQ 7000</u>

The Fig 4.13 depicts the behavior & utilization percentage of zynq-7000 having look up table
(LUT) value 38.

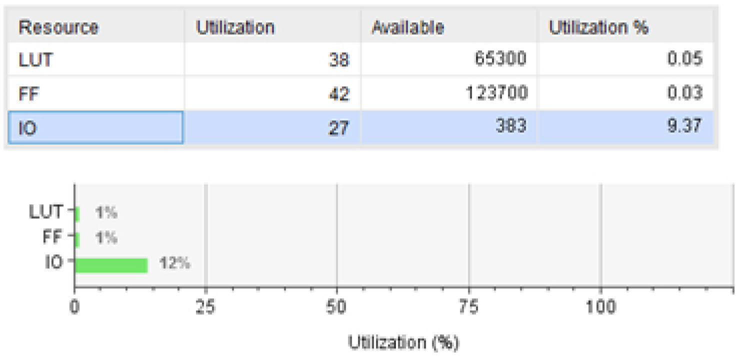


Resource	Utilization	Available	Utilization %
LUT	38	53200	0.07
FF	42	106400	0.04
IO	27	200	13.50

Figure 4.13

Kintex-7

The Fig 4.14 depicts the behavior & utilization percentage of kintex-7 having look up table (LUT) value 38.

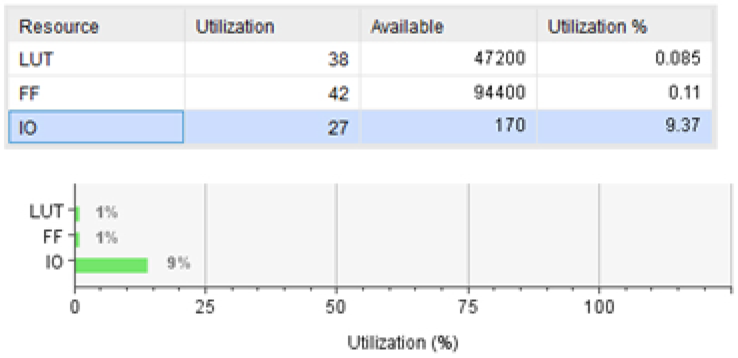


Resource	Utilization	Available	Utilization %
LUT	38	65300	0.05
FF	42	123700	0.03
IO	27	383	9.37

Figure 4.14

Artix 7

The Fig 4.15 depicts the behavior & utilization percentage of artix-7 having look up table (LUT) value 38.



Resource	Utilization	Available	Utilization %
LUT	38	47200	0.085
FF	42	94400	0.11
IO	27	170	9.37

Figure 4.15

CHAPTER-5

CONCLUSION & FUTURE SCOPE

6.1 CONCLUSION

I2C,SPI and UART are the sequential correspondence conventions for both bury and intra–chip low or medium bandwith information exchanges. This undertaking analyzes programming execution of the three conventions through various ongoing XILINX's FPGA families, for example, zynq-7000, Kintex-7 and Artix-7,showing up which highlight is in charge of additional power utilization in these conventions. This important data help originators to comprehend which convention can be use for a specific application reason. For a thorough similar examination, the majority of the three conventions are executed as universally useful IP arrangements, joining every single vital element required by current ASIC/SoC applications as indicated by an ongoing business sector examination of a significant number of business I2C , SPI and UART gadgets.

6.2 FUTURE SCOPE

In this project we have implemented these protocols of single master and single slave which has the further scope of implementation on multiple master and multiple slave in case of I2C protocol and single master and multiple slave in case of SPI protocol. Energy efficient techniques are not used in our project before power evaluation of these protocols.

In future capacitance scaling, thermal scaling, voltage scaling can be used to make these protocols more energy efficient.

Further IP configuration of the improvements done in these protocols can be implemented on different SOC configurable devices. The improvised version of these protocols can be further used for industrial purpose .

REFERENCES

[1] Performance evaluations of FIR filter after implementation on different FPGA & SOC and its utilization in communication and network By Bishwajeet Pandey, Bhagwan Das, Amanpreet Kaur, Tanesh Kumar, Abdul Moid Khan, D. M. Akbar Hussain & Geetam Singh Tomar

[2] Shiva Mehotra, Nisha Charaya., Journal on Design and Implementation of I2C single master on FPGA using Verilog, PISER 18, Vol.3,2006, ISSN 2347-6680(E).

[3]Philips Semiconductor “I2C Bus Specification”, April 1995.

[4]M.Morris Mano, “Digital Design” EBSCO publishing. Inc., 2002.

[5]Philips Semiconductors, “The I2C Bus Specification”, version 2.1, January 2000.

[6] Philips Semiconductors, “I2C Bus Manual”, AN10216-01, March 24 2003.

[7] I2C Tutorial “Using the I2C Bus”, http:// www.robotelectronics.co.uk/acatalog/ I2C_Tutorial.html.

[8]I2C tutorials at https://learn.sparkfun.com.

[9] https://www.skyfilabs.com/online-courses/smart-water-monitoring-using-iot?v1