e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:10/October-2023 Impact Factor- 7.868 www.irjmets.com

REAL-TIME IMPLEMENTATION OF A DMA IP ON FPGA

A. Charan Reddy*1

*1M.Tech, Vlsisd, Ece, Jawaharlal Nehru Technological University, Anantapuram, India.

DOI : https://www.doi.org/10.56726/IRJMETS45347

ABSTRACT

On field implementation of System on Chip (SoC) is necessary since its behavior changes when it is taken away from the laboratory or simulated in a computer. Sometimes it might not even work at all in the field. DMA IP Parameters may vary and its performance will decrease when it is implemented in the field. The Direct Memory Access (DMA) is a logical IP core of ANANTH fabless SoC designed and developed by JNTUA, is transformed into a hardware chip. Subsequently this transformed chip is implemented on FPGA. The real-time implementation of DMA IP on FPGA is performed in two branches: Hardware development and Software development. Under hardware development, we create a Vivado project with necessary IP blocks such as the A20 processor, AXI DMA core, etc., then generate a bitstream and export it as the hardware platform. As for as the software development concerned, we use Vitis to create a platform project using the exported hardware and then create an application project in C to run it on the hardware device, making ANANTH SoC fabricated.

Keywords: DMA Soft IP, Hardware Chip Design, Implementation, Latency, Throughput, Performance, Resource Utilization, Power Dissipation, Fabricate.

I. INTRODUCTION

"ANANTH" is a fabless SoC (System on Chip) designed and developed by ECE Department, JNTUA College of engineering. Anantapur. This SoC compromises of Power Core A20 Processor which uses Advanced eXtensible Interface-4 (AXI-4) as an on-chip bus so that the processor can communicate with other sub-master and sub-slave (peripherals interfaced) devices such as DMA, SPI, 12C, FLASH NAND, FLASH NOR, DDR3, ETHERNET and PCIe.

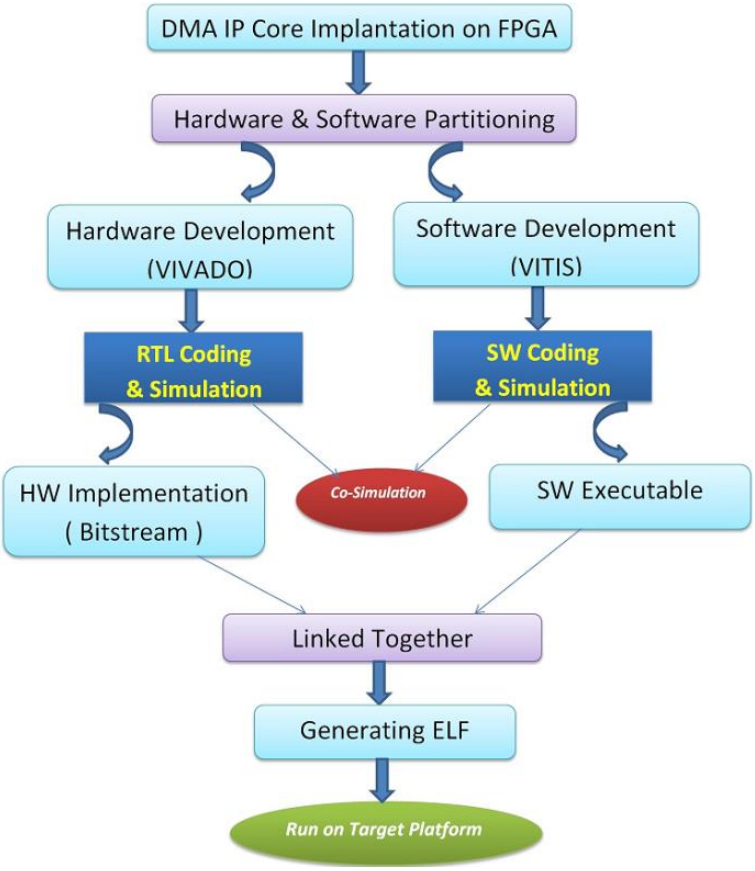

Fig. 1 Project Flow

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:10/October-2023 Impact Factor- 7.868 www.irjmets.com

The Implementation of a DMA IP on FPGA is performed in two parts:

i. Hardware Development and

ii. Software Development.

Under hardware development, we create a Vivado project sourcing DMA IP logical files, create XDC timing constraints, perform synthesis, Implement and then finally generate a bitstream and export the hardware platform. As for the software development, we use Vitis to create a platform project using the exported hardware and then create an application project in C to drive the FPGA logic. The partitioning of the Hardware / Software is the mechanism that separates the functions into a part of the hardware and software. The partitioning phase gathers the information from the profiling task to make decisions about the instances to map the software and the hardware. Once the instance of hardware and software creation has been established, and the interfaces have been established between them, the next step is the coding & simulation. In this stage, specs are refined, where autonomous system specifications are translated into HW and SW specifications.

When the coding & simulation action is over, the subsequent phase is authenticated by the design flow, which collectively simulates both the hardware/software. This is called the co-simulation process. The co-simulation process verifies whether or not the design objective has been achieved. When the specification is appropriate, the co-simulation ends. The design cycle is repeated until the satisfactory design output is reached; if the design is not satisfactory, the design returns to the HW/SW partitioning step.

If the results of co-simulation are acceptable, then the next task is the simulation-level implementation of both hardware and software parts. Once the partitioning task is finished, the executable program and the bitstream files are combined to create the .elf file and to run it on the hardware platform. The performance of these parameters relies on the level of design constraints and the reduction

II. HARDWARE DEVELOPMENT

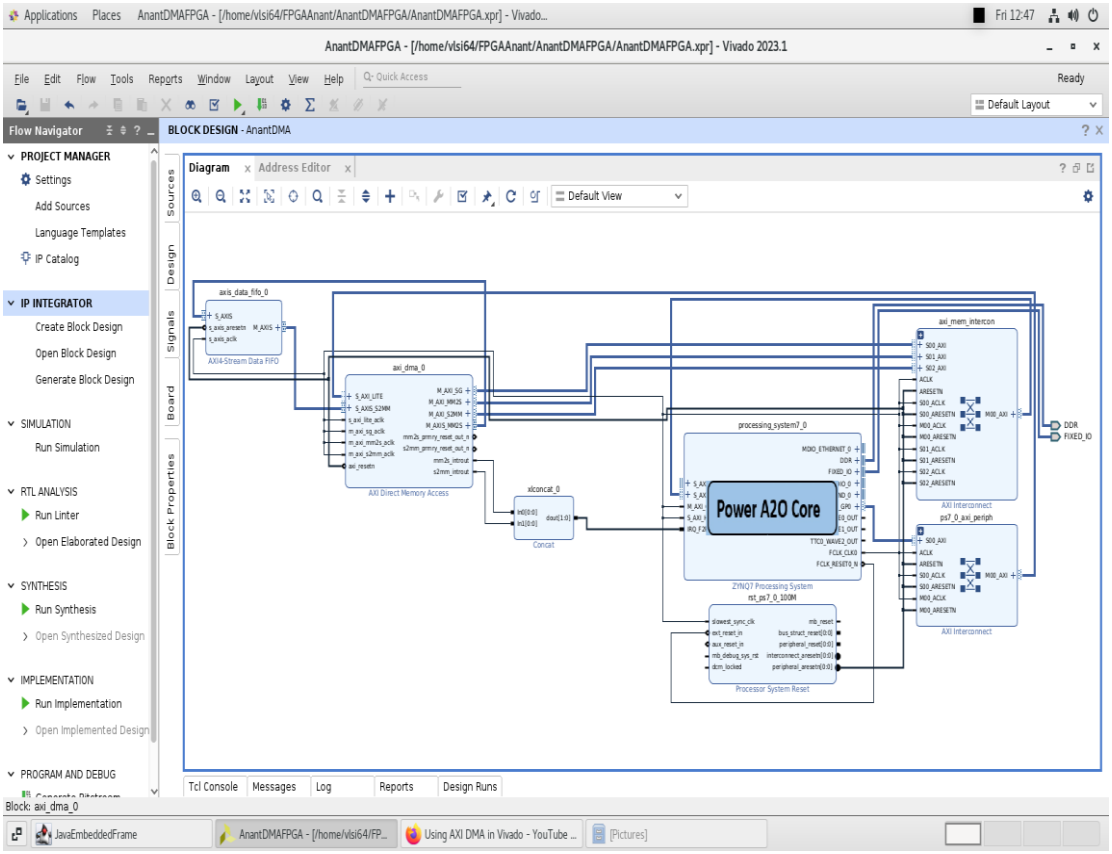

Fig. 2 DMA IP Block Design

In order to create a file that can be used to program the target board, each stage of the "compilation pipeline" needs to be run:

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:10/October-2023 Impact Factor- 7.868 www.irjmets.com

• Create a Vivado project sourcing Verilog HDL modules/files and targeting a specific FPGA device located on the board.

• Before you compile the Verilog code, you need to provide timing constraints for the design. You'll create an Xilinx Design Constraint (XDC) file that contains commands to let the Vivado software know how to close timing on the design. Without it, you will get warning messages in the compile flow because the Vivado software has no idea how to close timing on the design.

What is an XDC file, and why do I need one?

XDC stands for Xilinx Design Constraints and is an industry standard format which defines the timing constraints for a hardware (silicon) design such as the target frequency of the device, and the timing to external peripherals. The XDC file provides a way for Vivado to verify that the system generated meets its timing requirements.

• During synthesis of your FPGA, Vivado reads the timing constraints files, calculates the timing of the internal FPGA signals, and compare these timings to the timing requirements specified by the XDC files. A report is created which verifies timing is met and / or identifies signals which fail to meet timing and require optimization.

• The Analysis process checks design files for syntax and semantic errors, then performs netlist extraction to build a database which integrates all the design files.

• Synthesis creates a description of the logic gates and connections between them required to perform the functionality described by the HDL files, given the constraints included in XDC files. The output of Synthesis is then passed to Implementation.

• Implementation has several steps. The steps that are always run are:

Opt Design (Optimize the design to fit on the target FPGA),

Place Design (Lay out the design in the target FPGA fabric) and Route Design (Route signals through the fabric).

This output is then passed on to the Bitstream Generator.

• The Bitstream Generator generates the final output file needed for programming the FPGA. The generator will create a '.bit' file.

Depending on the complexity of the design, the board used, and the strength of your computer, the process of building the project can take between 5 and 60 minutes.

The “write_bitstream complete” status message can be seen in the top right corner of the window, indicating that the demo is ready to be deployed to your board.

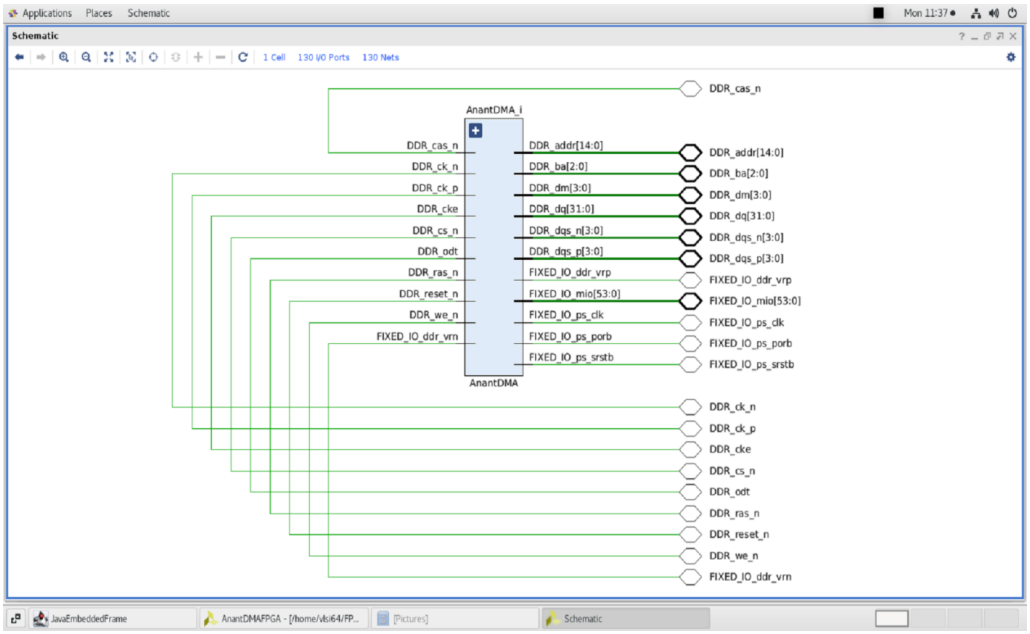

Fig. 3 DMA Chip Schematic

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:10/October-2023 Impact Factor- 7.868 www.irjmets.com

• Program and Run the Design

The final step is to program the bit file onto the FPGA. In Vivado from the top toolbar, select Xilinx Tools, then Program FPGA.

Congratulations!

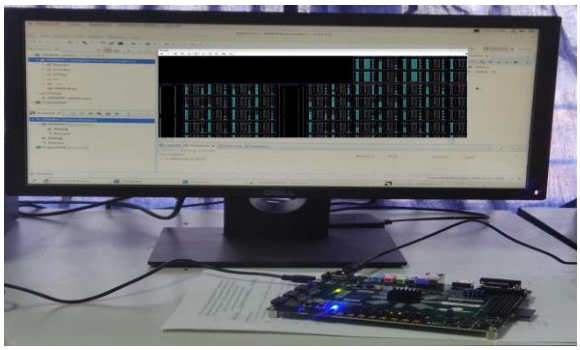

Fig. 4 Program FPGA Board

III. SOFTWARE DEVELOPMENT

As for the software development, we use Vitis to create a platform project using the exported hardware and then create an application project in C to drive the FPGA logic.

• Once the bitstream generation is completed, what we should do is export hardware. In Vivado Tool we select the platform type as Fixed and we will include the bitstream. Click Finish to save the hardware description (.xsa) file.

Vitis Platform Project

The next step in the workflow is to create a Vitis platform project to create an application.

In Vivado project select Tools -> Launch Vitis IDE. Then select a preferred workspace and launch the IDE. Then select the hardware description file you exported previously in Vivado and click Finish.

• Once the platform project is created, we can start creating the application project where we write the driver to control the FPGA hardware.

Vitis Application Project

• In Vitis, select File -> New -> Application Project.

• Then Select the platform that you previously created.

• Specify a name for the application project and click Next.

• Select standalone domain and click Next.

• Select Empty application from available templates and click Finish.

• After the project is created, we need to import sources.

To do this right-click the source directory and click Import Sources.

Then browse to the directory dma_platform, select .h and .c files and click Finish.

• Build and Run

We have set up everything required to run the example now.

• Connect the Zedboard to the host computer using USB-A to micro USB cable.

• Connect to the serial console using any serial port communication program.

• I usually use Teraterm for serial communication.

• Build the project and then click Run As -> Launch on Hardware.

• You will notice the serial terminal will be populated with numbers generated using the program executable.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:10/October-2023 Impact Factor- 7.868 www.irjmets.com

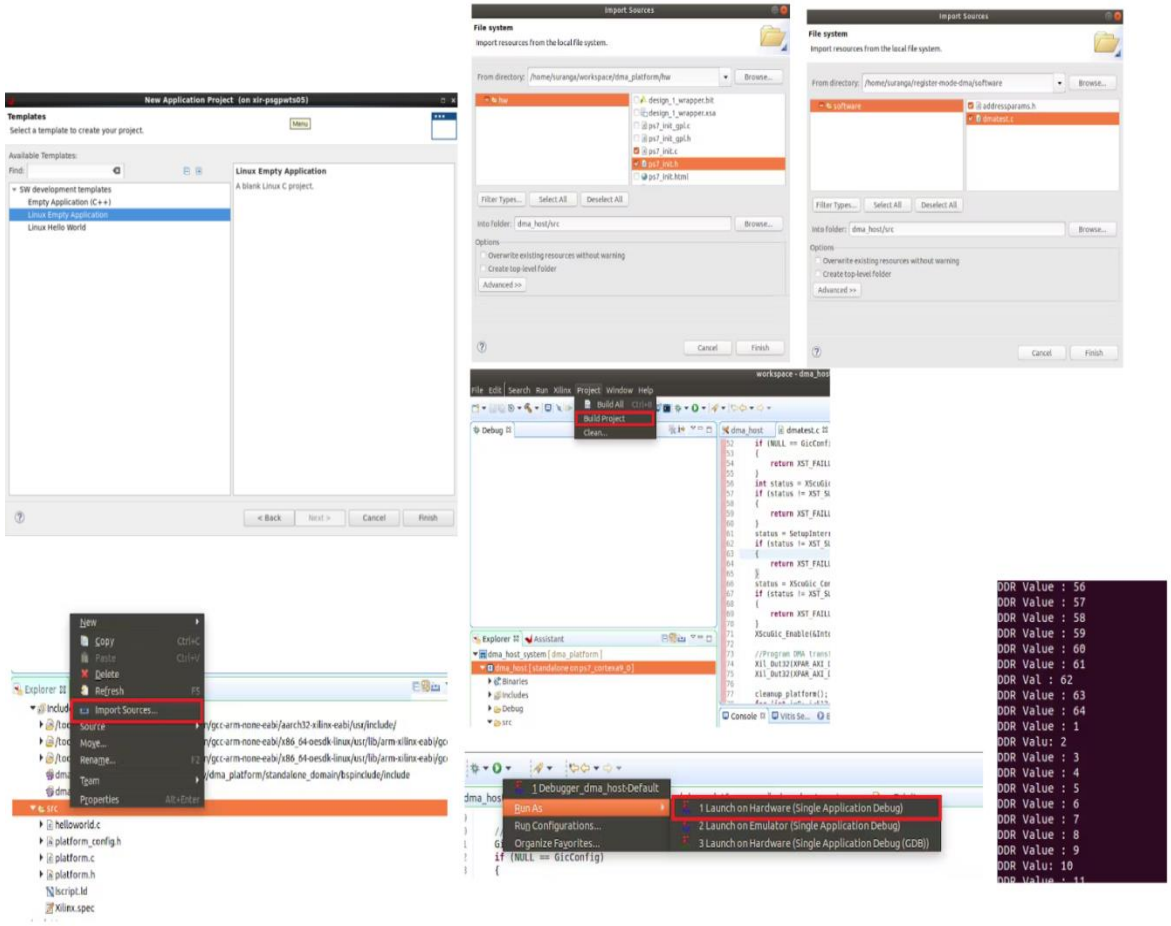

Fig. 5 Vitis Project execution Flow

IV. RESULTS & ANALYSIS

DMA Memory-to-Memory Transfer

MICRO CODE PROGRAM:

DMAMOV CCR, SB4 SS64

DB4 DS64

DMAMOV SAR, 0x1000

DMAMOV DAR, 0x4000

DMALP 16

DMALD

DMAST

DMALPEND

DMAEND

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:10/October-2023 Impact Factor- 7.868 www.irjmets.com

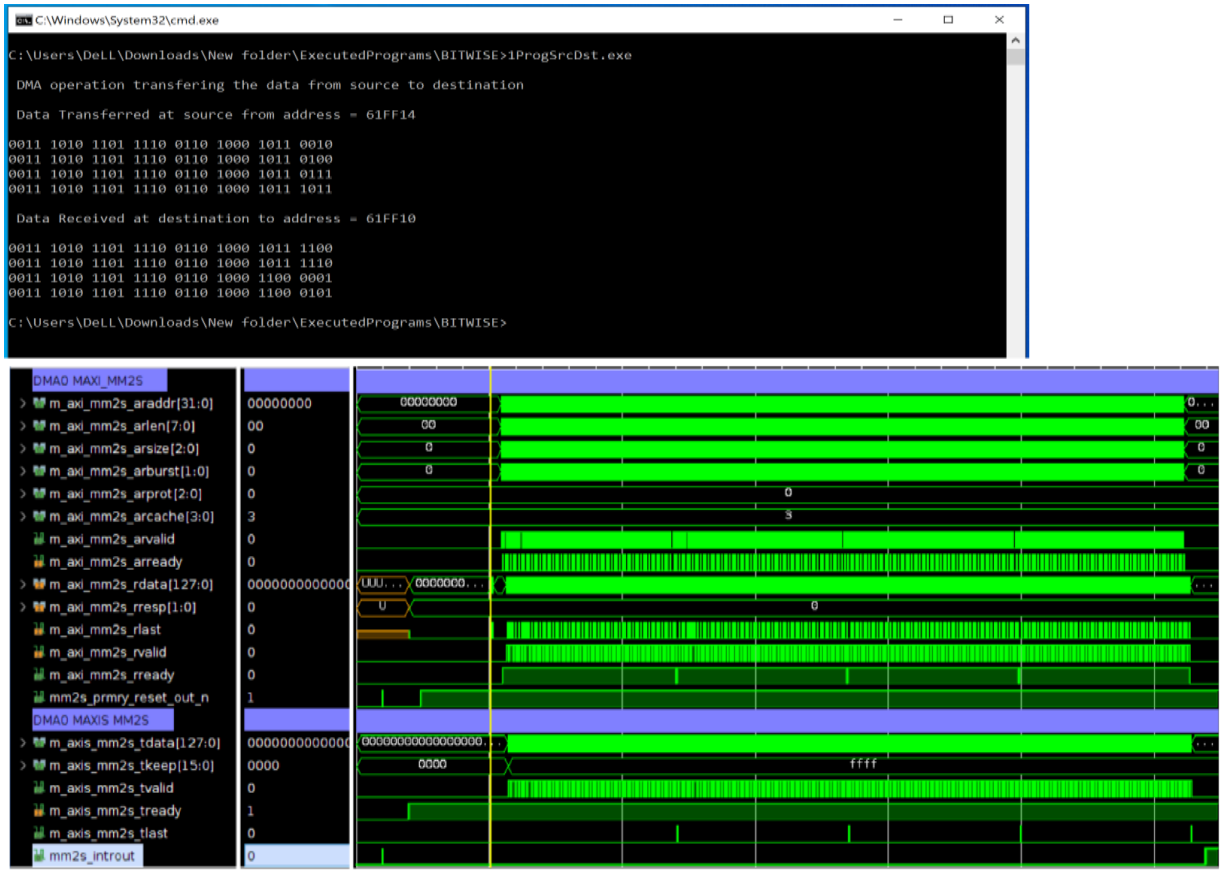

Fig .6 DMA Memory to Memory Transfer Output & Graph

DMA LATENCY:

The latency in the IP cores can vary on an interface-to-interface basis, depending on how the IP cores are configured. The latency is calculated in clocked cycles and is measured as the time that it takes from the assertion of the slave interface TVALID signal to the first assertion of the master interface TVALID signal.

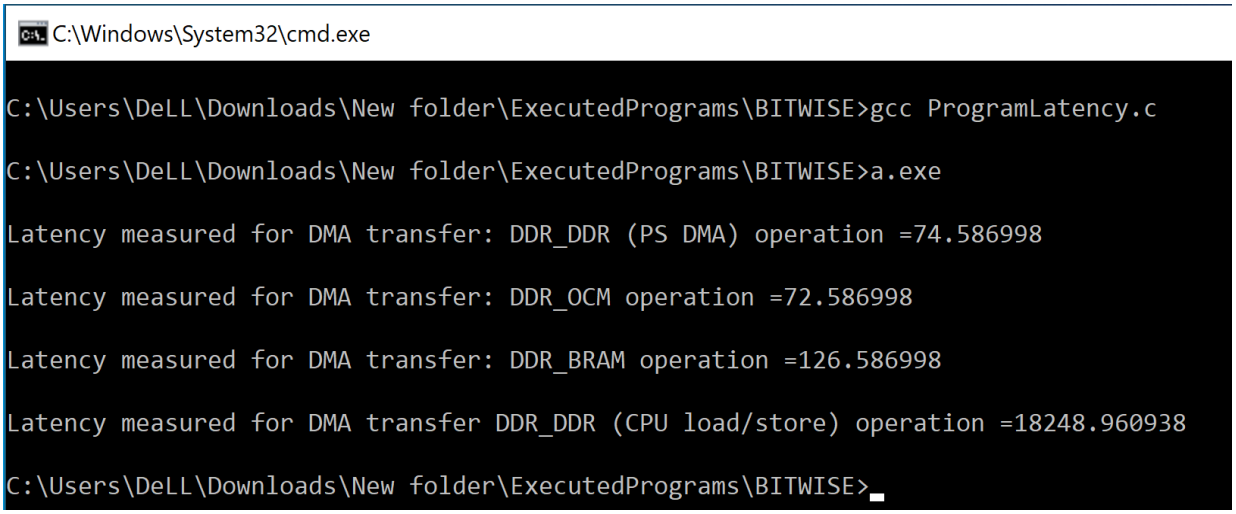

Fig. 7 DMA Latency in Nano Seconds

DMA THROUGHPUT

The throughput of a data path through each AXI4-Stream Infrastructure IP is calculated as

TDATA width x clock frequency of each of the paths determined by the SI interface and MI interface. The minimum throughput of an individual path in a system for which the transfer will traverse determines the overall throughput of the data path.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:10/October-2023 Impact Factor- 7.868 www.irjmets.com

```
C:\Windows\System32\cmd.exe

C:\Users\DeLL\Downloads\New folder\ExecutedPrograms\BITWISE>gcc ProgramTP.c

C:\Users\DeLL\Downloads\New folder\ExecutedPrograms\BITWISE>a.exe
 DMA Transfer from RAM to RAM
 DMA ThrougPut for Transfer Size =1Byte is 26 MB/S
 DMA ThrougPut for Transfer Size =1Byte is 94 MB/S
 DMA ThrougPut for Transfer Size =1Byte is 126 MB/S
 DMA ThrougPut for Transfer Size =1Byte is 182 MB/S
 DMA Transfer from ROM to RAM
 DMA ThrougPut for Transfer Size =1Byte is 18 MB/S
 DMA ThrougPut for Transfer Size =1Byte is 74 MB/S
 DMA ThrougPut for Transfer Size =1Byte is 166 MB/S
 DMA ThrougPut for Transfer Size =1Byte is 186 MB/S

C:\Users\DeLL\Downloads\New folder\ExecutedPrograms\BITWISE>
```

Fig. 8 DMA Through-Put

DMA RESOURCE UTILISATION

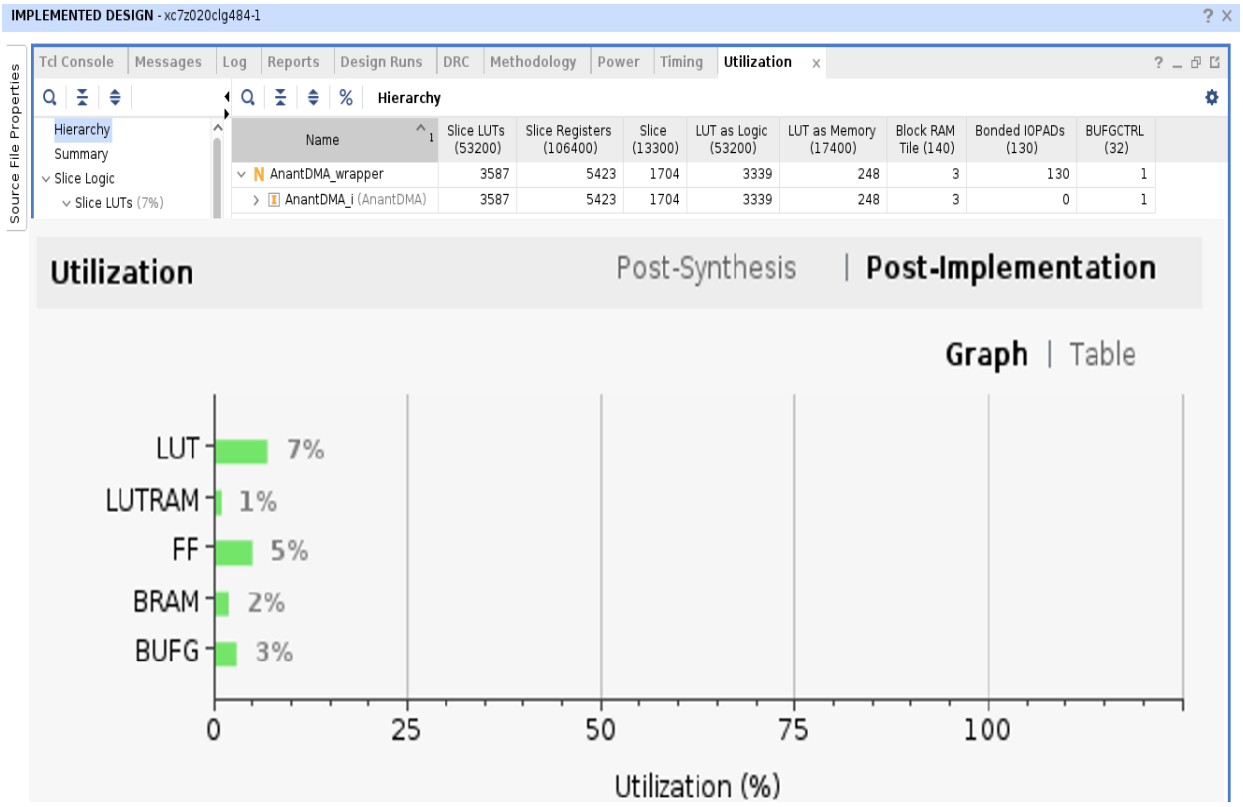

Chart 1: DMA Resource utilization

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:10/October-2023 Impact Factor- 7.868 www.irjmets.com

DMA POWER ANALYSIS

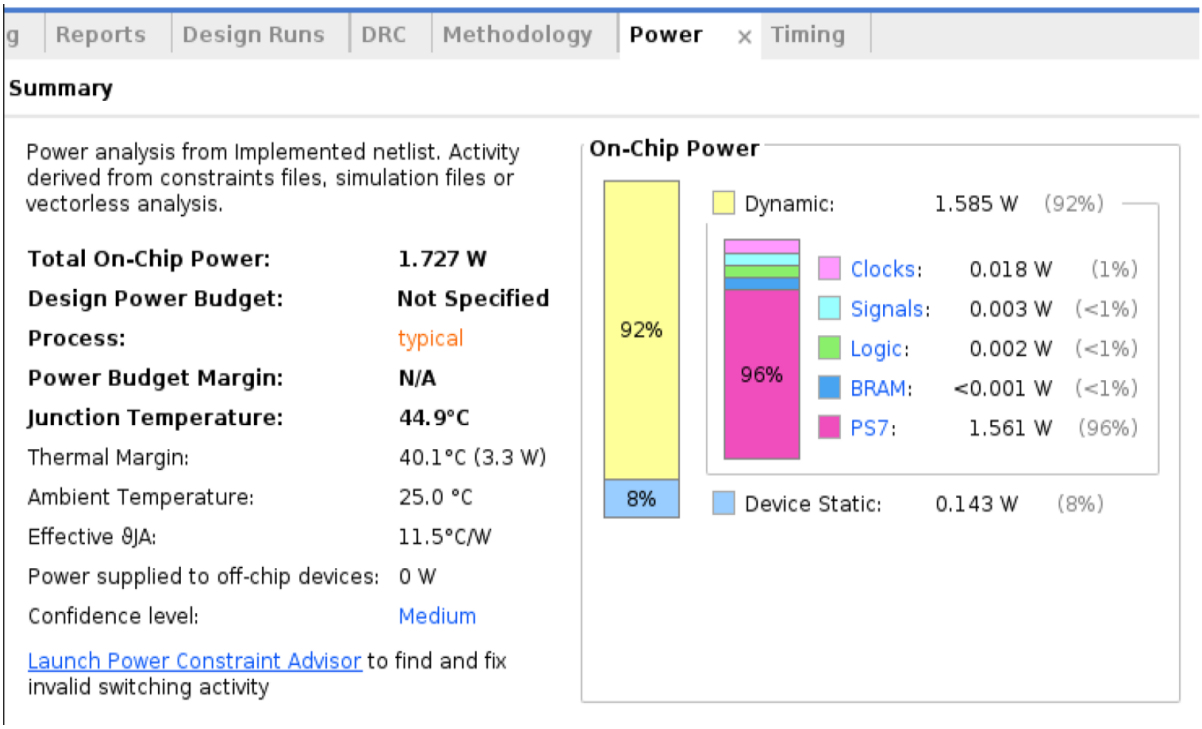

Chart. 2: Power analysis

V. CONCLUSION

I have successfully implemented DMA IP logical core of ANANTH SoC on hardware Zedboard FPGA in real-time, which was simulated on PC. The results are based on the implementation of the hardware design using the Vivado design tools on Zebboard FPGA. The DMA operation modes: Register direct more and Scatter Gather mode are accurately measured. Its latency and throughput parameters are interpreted realistically. DMA Perormance and Resouce utilization values have met the desired values. These results concludes ANANTH SoC is no more fabless, its fabricated. Congratulations!

VI. REFERENCES

[1] https://drive.google.com/drive/mobile/folders/1n8XPYOVdvJLlUxWthme3NTFSzFHjOgGQ
[2] https://docs.xilinx.com/v/u/en-US/wp459-data-mover-IP-zynq
[3] https://github.com/Xilinx/embeddedsw/blob/master/XilinxProcessorIPLib/drivers/axidma/examples/xaxidma_example_sg_intr.c
[4] https://zipcpu.com/blog/2021/08/14/axiperf.html