ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Design and Development of AFDX Transmitter Schedular

Sahithi Priya
M.Tech Student,
TKR College of Engineering & Technology,
JNTU Hyderabad.

B Swapna Rani
Associate professor,
TKR College of Engineering & Technology,
JNTU Hyderabad.

ABSTRACT:

Avionic Full-Duplex Switched Ethernet (AFDX, is a specification for a deterministic aircraft data network bus for aeronautical, railway and military systems. The network is based on standard IEEE 802.3 Ethernet technology. AFDX extends the Ethernet standard by adding Quality of Service (QoS) and deterministic behaviour with a guaranteed dedicated bandwidth. An AFDX network consists of so called End Systems and switches. An End System is a component connected to the AFDX network and capable of handling all AFDX related protocol operations. Usually, an End System is part of an avionic or aircraft subsystem, which needs to send or receive data over the AFDX network. One or more switches, depending on the network hierarchy, are located on the data path between two End Systems.The point-to-point and point-to-multipoint connections are represented by virtual links (VL). It is the responsibility of the End System to regulate all outbound traffic according to the allocated time budgets of each virtual link. The mapping of the VL into the global switched network realizes a partitioning of the topology; each VL has an associated time fragmentation called BAG (Bandwidth Allocation Gap) and a maximum frame size. Therefore transmission for a given VL is blocked until the BAG has elapsed. The aim of the project is to develop Transmitter scheduling Mechanism in verilog and implement on Xilinx FPGA (VIRTEX-5) using chipScope Pro Analyser software.

I.INTRODUCTION:

Avionics Full Duplex Switched Ethernet, or AFDX, is becoming more popular in avionics markets worldwide. So much so, that it is no longer indigenous to avionics markets, and has now grown into some areas outside the field as well. This is because it offers the advantages of Ethernet while overcoming many of its disadvantages. AFDX networks have been extensively used in commercial as well as military based avionics technologies. These include AIRBUS A380, Boeing 787, Sukhoi Superjet 100, and many more.

It is based on the well-known network technology Ethernet (IEEE 802.3) and it uses entirely commercial off-the-shelf components (COTS). It is standardized as ARINC Specification 664 Part 7. The primary segments of AFDX are Redundancy, Full Duplex Technology, High Speed, Switched and Profiled Networks.AFDX adopted concepts (token bucket) from the telecom standard,Asynchronous Transfer Mode (ATM), to fix the shortcomings of IEEE 802.3 Ethernet. By adding key elements from Asynchronous Transfer Mode (ATM) to those already found in Ethernet, and constraining the specification of various options, a highly reliable Full-Duplex deterministic network is created providing guaranteed bandwidth and Quality of Service. Through the use of Full-Duplex Ethernet, the possibility of transmission collisions is eliminated.

The network is designed in such a way that all critical traffic is prioritized using QoS policies so delivery, latency, and jitter are all guaranteed to be within set parameters. A highly intelligent switch, common to the AFDX network, is able to buffer transmission and reception packets.Through the use of twisted pair or fibre optic cables, Full-Duplex Ethernet uses two separate pairs or strands for transmit and receiving data. AFDX extends standard Ethernet to provide high data integrity and deterministic timing. Further a redundant pair of networks is used to improve the system integrity.The main elements of an AFDX network are:

- AFDX End Systems
- AFDX Switches
- AFDX Links

II. RELATED WORK:

A.ALOHA:

To fully understand the scope of the project, its origins in Ethernet are considered. In 1970, the University of Hawaii deployed a packet radio system called “ALOHA network” to provide wireless communications between islands.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Since there was no centralized control among the stations, there was high chance for collision between signals that were transmitted by two or more different stations at the same time. This protocol dictated that if transmitting system had a message to send, it should send the message and if the signal collides with another, try resending the message after arbitrary period of time. The problems that were faced by this system were that there was no coordination between the systems and that the collisions caused by this lack of coordination caused unpredictable behaviour.

B.HALF DUFLEX ETHERNET:

In 1972, Robert Metcalfe and David Boggs at Xerox Palo Alto Research Center built upon the ALOHA network idea and used a coaxial cable as the communication medium and invented Ethernet. Ethernet is similar to the ALOHA protocol in the sense that there is no centralized control and transmissions from different stations (hosts) could collide. The Ethernet communication protocol is referred to as “CSMA / CD” (Carrier Sense, Multiple Access, and Collision Detection).

Half-duplex Mode Ethernet, where each host selects a random transmission time from an interval for retransmitting the data. If a collision is again detected, the hosts selects another random time for transmission from an interval that is twice the size of the previous one, and so on. This is often referred to as the binary exponential backoff strategy. Since there is no central control in Ethernet and in spite of the random elements in the binary exponential backoff strategy, it is theoretically possible for the packets to repeatedly collide.

C. AVIONICS FULL DUPLEX SWITCHED ETHERNET:

Avionics full duplex Ethernet, also known as AFDX is a system that was developed specifically for operations that reduced the drawbacks of ARINC 429 specifically with the requirements of Avionics systems in mind. It is a system proprietary to AIRBUS. It has 100 Mbit/sec, 10 Mbit/sec and 1Gbit/sec implementations available and it is built around commercial Ethernet including the standards with MAC, IP, UDP, SNMP protocols and deterministic behavior.The general structure of an AFDX network is shown in the figure 3

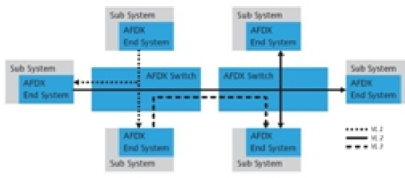


Figure 1: AFDX Network [18]

It is built using Copper and Fibre Optics and is divided into three main sub parts namely, End Systems, Switches and Links (Virtual Links). It is defined as a High Speed Commercial Ethernet with provisions for guaranteed deterministic timing and redundancy required for Avionics Applications.As seen, AFDX is a network with mainly two types of devices: the switch and the end system. The switch performs traffic policing and filtering and forwards packets to their destination End Systems. And End System is a device whose applications access the network components to send and receive data from the network. End Systems are used as an extension to the sub systems to allow them to connect to the network.

As one of the most advanced data transmission technologies in avionics field, AFDX features high bandwidth, low cost, extensive system integration, real-time property, reliability, and shows great potential in on-board avionics system. [14]The AFDX Network architecture is registered as ARINC 664. It is based on IEEE 802.3 as well as ARINC 429 Ethernet protocol standards. AFDX technology incorporates concepts such as BAG, or Bandwidth Allocation Gap, Virtual Links, and store and forward Switches to ensure reliable and fast transmission of data in a network where it is vital for data to be transferred as fast and as reliably as possible. Its working depends on two devices, the End System and the Switch. Both these devices are connected by Virtual Links. All communications in the system are Full Duplex and without a dedicated Bus for communications between switches. The network ensures reliability by using a redundancy management system.

As previously mentioned, the Full Duplex Ethernet had some drawbacks that meant it could not be used for a real time avionics network. One of those drawbacks was that a twisted pair must link every single device twice. This meant a lot of wire runs and weight; sometimes too much to be fit within the size and weight constraints of a missile or an airplane.

This was considered in AFDX and resolved my ensuring that each signal is connected to the switch only once, no matter how many subsystems were connected.Additionally, while an ARINC 429 transmitter can fan out to only 20 receivers, this constraint was removed in ARINC 664.

REDUNDANCY MANAGEMENT:

An AFDX network, for purposes of Redundancy Management, contains two sets of independent switched networks, called Network A and Network B. The function of the AFDX redundancy management is merely to eliminate frames that are redundant copies of frames that it has already passed on to the partition.This is represented in the figure as shown below.

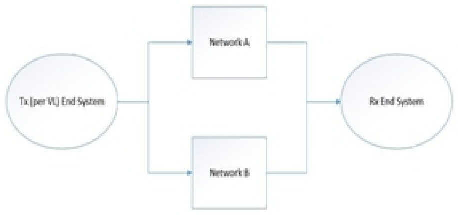


Figure 2:Redundancy Management system

If a certain packet is mistimed in Network A the correct timing according to BAG can be realized through Network B. If a certain packet is realized to be missing in Network A when comparing it to Network B, the receiving end system can send a resend request by giving a negative acknowledgement or it can discard th packet received through Network A and use the packet sent by network B. This can be the case in case of failed checksum or mistaken bit too. It is at the discrepancy of the programmer to decide the course of action in case of an erroneous packet sent.However, End systems require a method to identify the replicas that arrive on the A and B networks. So all packets sent over VL are given a 1-byte sequence number field. The sequence number field appears just before the FCS field in the Ethernet frame. An Ethernet frame with a sequence number is referred to as an AFDX frame.[9].

AFDX Frame:

The AFDX frame is as represented below. It is a total from size can be between 64 and 1518 bits. This includes a 14-bit long Ethernet Header, a 20 bit IP Header, an 8 bit UDP Header, a one bit sequence number (for Redundancy Management) and a 4 bit FCS.

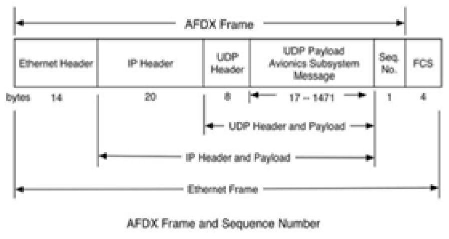


Figure 3: AFDX Frame and Sequence Number

II.PROPOSED WORK

A. SCHEDULING:

Scheduling methodologies are used to allocate resources between parties that request them simultaneously and asynchronously. These techniques are used in routers, to handle packets as well as in operating systems, to share CPU time among threads and processes. Other applications include disk drives (I/O scheduling), printers (print spooler), most embedded systems, etc. Scheduling deals with the problem of deciding which of the outstanding requests is to be allocated resources. [10]In general, (job) scheduling is performed in three stages: short-, medium-, and long-term.

The activity frequency of these stages is implied by their names. Long-term (job) scheduling is done when a new process is created. It initiates processes and so controls the degree of multi-programming (number of processes in memory).Medium-term scheduling involves suspending or resuming processes by swapping (rolling) them out of or into memory. Short-term (process or CPU) scheduling occurs most frequently and decides which process to execute next. [10]

Weighted Round Robin:

Weighted Round Robin, or WRR, was developed by Katevenis, Sidiropoulos and Courcoubetis in 1991, for their paper titled “Scheduling in ATM networks using fixed size packets (cells)”. The concept states that each process will be assigned a weight, which means that each process will come along with a certain requirement of processor time which will allow the scheduler to know exactly how much time to assign to the process. For example if one process takes 10ms and another takes 40ms, the scheduler will be informed beforehand of the requirement.

Instead of a fixed time slice or quantum, the scheduler will assign only the required amount of time for the respective processes. In this case, 10ms is for the first and 40ms is for the second. This method requires the transmitter to be able to provide a decent estimate of the time required in the processor.

Weighted Round Robin in AFDX:

A concept similar to Round Robin and Weighted Round Robin [3] has been used to design the scheduler. Since the scheduler to be designed does not have anything to do directly with required processor time, it has been adapted to the requirements of the system. A valid estimate of the required time to send the signal is given by the BAG value which is calculated by the Transmitter itself. The signals are then segregated by their BAG value such that the network will know how long a frame will take to be transmitted beforehand.

B. SCHEDULER USING RR-ALGORITHM (TOP MODULE):

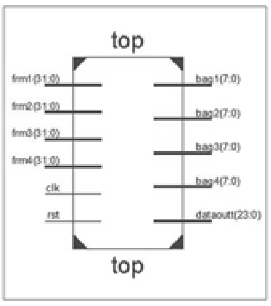


Figure 4 : RTL Schematic of Top Module

The overall operation of the scheduler depicted in the diagram below. The mechanism of this particular scheduler algorithm is designed to work specifically with a transmitter system of an AFDX end system. A total of 8 inputs (frm1, frm2, frm3, frm4, frm5, frm6, frm7, frm8) are given to the input ports of the scheduler.Frm is a 32'b input from corresponding incoming virtual link.The scheduler separates the VL IDs from their BAG values and segregates them according to this property. It then forwards each of these 24 bit values (divided according to BAG) to the redundancy management system. This will in turn send it onto the processor as requests. Bag1, Bag2, Bag3, Bag4, Bag5, Bag6, Bag7, Bag8 – Corresponding bag values from each of the inputs frm1 to frm8 sent to the concerned module. Data outt – 24'b output signal in serial form that is forwarded.

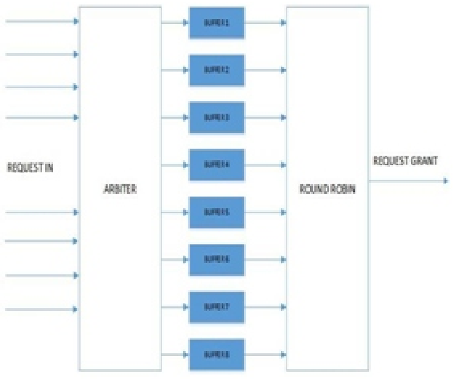


Figure 5: Block diagram of designed Schedule

C.SIGNAL FLOW OF ARBITER MODULE :

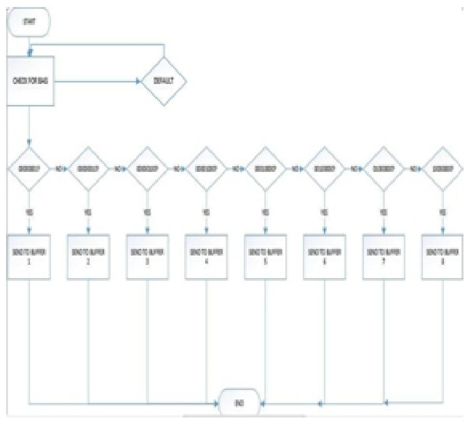


Figure 6.: Signal Flow chart of Arbiter

The arbiter module has been coded using case statements. Each case corresponds to a separate bag value. There is a round robin based algorithm running within the arbiter module that checks each incoming signal one after the other by cycling through each one. If an incoming signal is detected, then the arbiter receives the signal, separates the frame into two i.e. BAG frame and request frame.The bag frame is 8 bits long and the request frame is 24 bits long. BAG value can be anything from 20 to 28 in binary form. The BAG frame is forwarded to another system that receives will tell the processor which output signal corresponds to which BAG. The request frame is passed onto the buffer module.The round robin algorithm uses a counter to cycle between the eight incoming ports to check if a signal has been received. Once a signal is found on any one of the ports, a wait signal is sent to the transmitter that stays high until the signal is safely stored in the buffer. This ensures no data loss at the point of reception.

Once data is received, a write flag is sent to the buffer that corresponds to the BAG value of the received message. This write flag is a single bit that is sent to the buffer prior to the message signal.Out1, Out2, Out3, Out4, Out5, Out6, Out7, Out8 – Wires that connect as inputs to each of the buffersWrite1, Write2, Write3, Write4, Write5, Write6, Write7, Write8 – Write flags to activate write operation on the buffers

D.SIGNAL FLOW OF BUFFER MODULE:

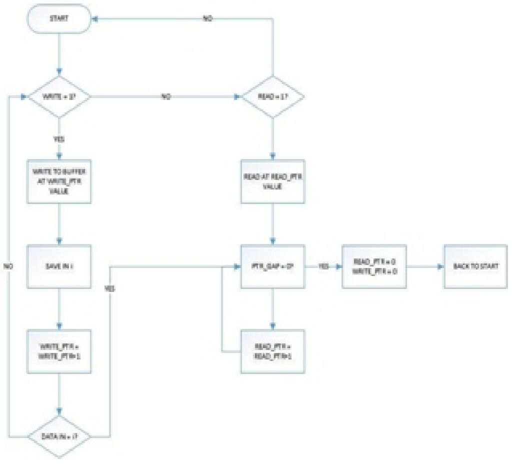


Figure 7:Signal Flow chart of Buffer

As shown in the flow chart, the start of the buffer is signalled by an incoming write signal from the arbiter module. The buffer has two different pointers – one for write (write_ptr) and one for read (read_ptr). The write pointer indicates the place in the buffer where the data will be written. Sometimes the same data is erroneously sent more than once or the buffer will write one piece of data more than once. To avoid this, an additional register, namely, 'i' is added. All the data is written in to the register as well as the buffer. Before new data is written onto the buffer, it is first referenced with the data in 'i'. If the same data is present in 'i', that means duplicate data might be written into the buffer and it is dropped.The buffer is designed to hold 8 total requests. This is due to the possibility of the scheduler not accepting the data before the second wave of signals comes into the arbiter and is passed on. This will cause data loss if the buffer does not have some extra space to hold this additional data.Once the data is stored in the buffer, it waits for the scheduler module to request it by sending a read flag. The read flag, like the write flag is a single bit, that when high, will allow the buffer to send forward any packets of data stored within it.

Once the read flag is high, the read pointer starts rising from the bottom. The buffer checks if there is any data stored at the address that is being pointed at by the read pointer and if the value is not 0, it sends the message forward. The read pointer runs until the difference between the write and read pointers (ptr_gap) is 0. This means that the read pointer has reached the last input message.This operation of the buffer is based on FIFO, or first in first out concept. Since the data sent to the buffer first will be sent forward first to the scheduler module. There are a total of 8 buffers operating within the TOP module. Each one corresponds to a separate BAG value. This sort of operation sorts all incoming packets according to their BAG and since from a particular buffer all the data will go forward at the same time this means that all the data with one particular BAG value that comes in within one cycle of the scheduler will be sent forward as requests at the same time. Additional spaces for messages can be added to the buffers at any time.

First In First Out *concept has been used the buffer design due to its simple advantages. The disadvantages are overcome by combining FIFO concept with Round Robin scheduler. The disadvantages of both are minimized.
Datain – connected to 'out' from arbiter. Each buffer has one datain and named datain1 to datain8 for the 8 buffers
Write – Flag sent by the arbiter to activate write operation for a particular buffer
Read – Flag sent by the scheduler to activate read operation for a particular buffer
Dataout – connected to input of scheduler module. Each buffer has one dataout named dataout1 to dataout8 for the 8 buffers.
'i' – This register stores the 24'b input that is input and is cross referenced before writing to the buffer every time.

E.ROUND ROBIN MODULE:

The round robin scheduler consists of a two level counter – one that counts from 0 to 255 (ctr) and the other, depending on the top level counter, changes from 000 to 111. This is named ptr. It operates in such a way that 000 corresponds to buffer 1 (1ms BAG), 001 to buffer 2 (2ms BAG) and so on to 111, which points to buffer 8 (128ms BAG).

The pointer increments by 1 for each of the 31 bits that the counter (ctr) points to. For each buffer that the pointer (ptr) points to, the buffer simultaneously sends the value forward to the redundancy management system which then sends it forward as a request to the processor as a request.

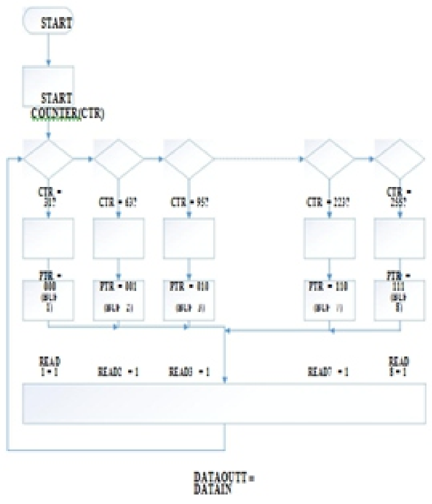


Figure 8: Signal Flow Chart of Scheduler

datain1 to datain8 – These are connected to dataout1 to dataout8 that are outputs from the buffer respectively
Dataoutt – output for the overall system that serially sends the received message packets in the required format.
Read1 to read8 – Read signals that correspond to each buffer from 1 to 8 and activate the read operation in the buffer when received

The serial number of the buffer that activates corresponds to the serial number of the read flag that is changed to high by the round robin.

III.RESULTS

The code was written in Verilog and highlighted in XILINX ISE (14.4 version) and verified using random environment conditions.Each module was verified to be accepting all expected real world conditions and they were input using the ISE's own testing capabilities. The standard input was only inclusive of the clock which was taken to be 10ns or 1 GHz for all cases. This is the maximum speed of the AFDX based networks.

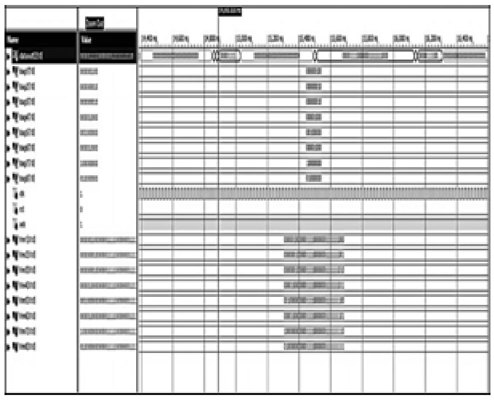


Figure 9: Screenshot of TOP module Operation

The TOP module takes 8 VL requests and their respective BAG values as input. It then processes the request and gives the grant based on the least BAG value and according to the round- robin scheduling algorithm. The TOP module has three blocks namely arbiter, FIFO buffer and the scheduler.

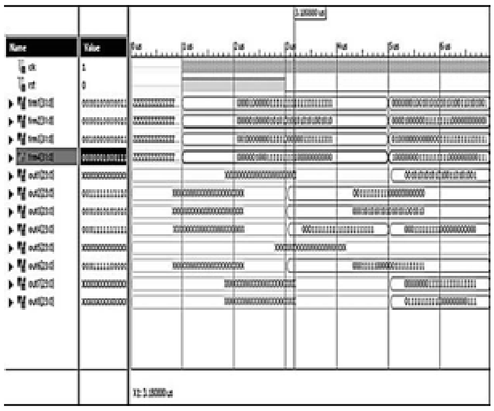


Figure 10: Screenshot of Arbiter Operation

As seen in the above screenshot, when the reset condition is set to zero, the arbiter accepts the incoming signals (frm1, frm2, frm3, frm4, frm5, frm6, frm7, frm8) which consist of 32 bit frames. It then proceeds to segregate them into 8 bit and 24 bit frames.The 24 bit frames are then sent on any output wire from out1 to out8 based on their corresponding 8 bit frames (BAG Values). Two waves of signals are shown to be received here and both are sent forward in an order that is determined by first come first serve as well as the round robin.to zero, the arbiter accepts the incoming signals (frm1, frm2, frm3, frm4, frm5, frm6, frm7, frm8) which consist of 32 bit frames. It then proceeds to segregate them into 8 bit and 24 bit frames.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

The 24 bit frames are then sent on any output wire from out1 to out8 based on their corresponding 8 bit frames (BAG Values). Two waves of signals are shown to be received here and both are sent forward in an order that is determined by first come first serve as well as the round robin.

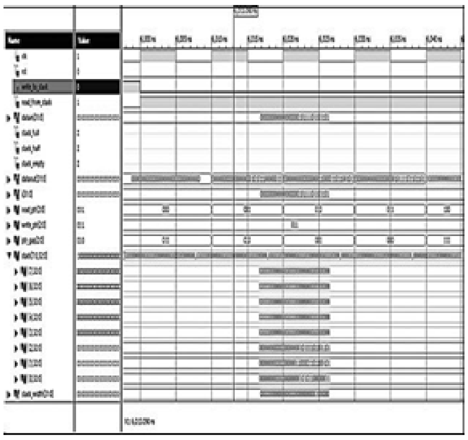


Figure 11: Screenshot of Buffer Operation

The buffer operates according to the First In First Out principle.The buffer has a total of eight addresses where data can be stored. The screenshot shows the incrimination of write_ptr to the fourth place in the buffer to write the values.To perform read operation, the read_ptr increments independently from the zero to the fourth position, all the while the buffer is forwarding the saved data at read_ptr position through dataout to the round robin module.

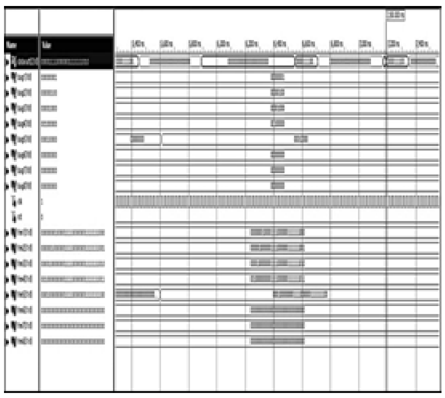
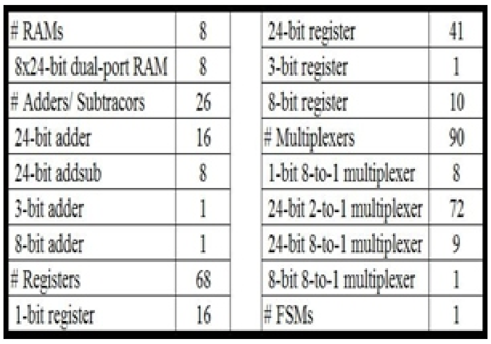


Figure 12: Screenshot of Scheduler Operation

A simple operation of the scheduler algorithm is as shown above. It receives the message through rrdatain1 to rrdatain8 from the 8 buffers and forwards them in a serial stream of data.

The counter counts from 0 to 255 continuously and the pointer increments by 3'b001, starting from 000 and ending at 3'b111. At each incrimination of the pointer, the scheduler sends read signal to the required buffer. The scheduler is designed using round-robin algorithm. Equal intervals of the counter are allocated for each FIFO buffer so that each request gets a fair chance to get the grant.

SYNTHESIS REPORT:
TARGET DEVICE: XC5VLX330T



# RAMs	8	24-bit register	41
8x24-bit dual-port RAM	8	3-bit register	1
# Adders/ Subtracors	26	8-bit register	10
24-bit adder	16	# Multiplexers	90
24-bit addsub	8	1-bit 8-to-1 multiplexer	8
3-bit adder	1	24-bit 2-to-1 multiplexer	72
8-bit adder	1	24-bit 8-to-1 multiplexer	9
# Registers	68	8-bit 8-to-1 multiplexer	1
1-bit register	16	# FSMs	1

Table 1 :Synthesis Report of Scheduler using Round-Robin Algorithm

The designed Round Robin Scheduler is integrated with the other blocks in the transmitter such as CPU Interface, state machine, DPRAM and the MAC to implement it on the Virtex-5 FPGA board and using the ChipScope Pro Analyser software.

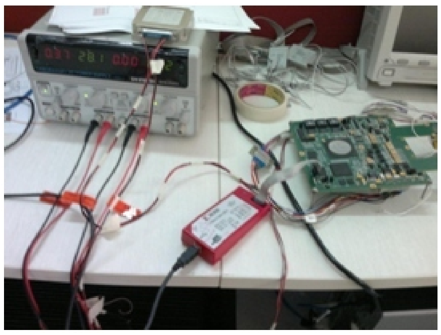


Figure 13 : Final Output.

V. CONCLUSION:

A scheduler that is optimized for the purpose of real time internal communication was successfully designed using Xilinx ISE tools and implemented in Vertex-5.

Building on the existing concepts already used within AFDX, and optimizing the pre-existing scheduler, a new scheduler design using both FIFO and Weighted Round Robin techniques was developed. The design concentrates on the advantages and strengths of both algorithms while minimizing data loss or delays from the weaknesses of both algorithms.A scheduler using FCFS algorithm is designed according to the BAG concept. The transmitter module is implemented on Virtex-5 using ChipScope Pro Analyser software.Other than avionics, this scheduler can be used for any generalized purposes that require operations in real time environments.

REFERENCES:

[1]System-Level Scheduling of Mixed-Criticality Traffics in Avionics Networks, International Conference on Parallel and Distributed Systems, 2013.

[2]Arbiter design using verilog for switching to communicate in between multiple resources, International Journal of Innovative Technology and Exploring Engineering (IJITEE), Volume 03, August, 2013.

[3]Speed efficient implementation of round-robin arbiter design using verilog, International Journal of Enhanced Research in Science Technology and Engineering, Volume 02, September, 2013.

[4]Scheduling Heterogeneous Flows with Delay-Aware Deduplication for Avionics Applications, IEEE transactions on parallel anddistributed systems, vol. 23, no. 9,September, 2012.

[5]Optimal scheduling policy for AFDX end-systems with virtual linksof identical bandwidth allocation gap, 25th IEEE Canadian conference on Electrical And Computer Engineering, 2012.

[6]A tight end-to-end delay bounding and scheduling optimization of an avionics AFDX Network, IEEE 2011.

[7]Scheduling design and analysis for end-to-end heterogeneous flows in avionics network, Conference Paper, University of Nebraska-Lincoln, Lincoln, NE, USA, 2010.

[8]AFDX wireless scheduler and free bandwidth managing in AFDX network, IEEE 2011.

[9]Yu Hua; Xue Liu; "Scheduling Design and Analysis for End-to-End Heterogeneous Flows in an Avionics Network", IEEE INFOCOM 2011.

[10]H.S. Behera; Simpi Patel; Bijayalakshmi Panda, "A New Dynamic Round Robin and SRTN Algorithm with Variable Original Time Slice and Intelligent Time Slice for Soft Real Time Systems", International Journal of Computer Applications (0975 – 8887) Volume 16– No.1, February 2011

[11]Modelling and performance analysis of AFDX based on petri net, IEEE 2010.

[12]The research of AFDX system simulation model, IEEE 2010.

[13]AEEC; Aeronautical Radio Inc.; "Aircraft Data Network Part 7 Avionics Full Duplex Switched Ethernet Network (ARINC SPECIFICATION 664 P7-1)", September 23, 2009

[14]Xin Chen; Xudong Xiang; Jianxiong Wan, "A Software Implementation of AFDX End System", 2009 International Conference on New Trends in Information and Service Science

[15]SWITCHED ETHERNET TESTING FOR AVIONICS APPLICATIONS, Ken Bisson, Troy Troshynski, 2007

[16]Communications for integrated modular avionics, IEEEAC Paper #1230, Version 1.3, December, 2006.

[17]A. Mifdaoui, F. Frances, C. Fraboul, "Real-Time Communication over Switched Ethernet for Military Applications" In Proceedings of the conference on Emerging network experiment and technology, pp. 45-56, ACM, 2005.

[18] Condor Engineering; "AFDX Protocol Tutorial", May 2005

[19]Round-robin arbiter design and generation, ISSS, October 2002.

[20]M. Shreedhar and G. Varghese. Efficient fair queuing using deficit round-robin. IEEE/ACM Transactions on Networking (TON), 4(3):385,1996.

[21]Manolis Katevenis; Stefanos Sidiropoulos; Costas Courcoubetis, "Weighted round-robin cell multiplexing in a general-purpose ATM switch chip", IEEE Journal on Selected Areas in Communications, (Vol. 9, Issue:8), October 1991

[22]Arbiters: Design Ideas and Coding Styles,Matt Weber, SiliconLogic Engineering, Inc.

[23] Jonathan Woodruff, "Deficit Round Robin Input Arbiter for NetFPGA"