DWC Ethernet

GMAC Universal core

AVB Software Driver User Guide

AVB Software Driver User Guide

Release 3.60a

Version 0.2

Sept 23, 2009

Revision History

Version	Revision History	Author	Date
0.1	Prelim draft.	Synopsys	June 07, 2009
0.2	Enabled to support Average Bits Per Slot reporting	Synopsys	Sept 23, 2009

Contents

1 Introduction

The document outlines the software functional (architectural) specifications for DW Gmac Ethernet Universal core to support IEEE 802.1Qav Audio Video Bridging.

The IEEE P802.1Qat/D2.1 standard allows network resources to be reserved for specific traffic streams traversing a bridged local area network. The AV traffic support can be enabled by selecting coreKit parameter “Select AV Feature”. When this feature is enabled the top-level block diagram of the Ethernet MAC core is as given in Fig 1for the GMAC-AHB configuration.

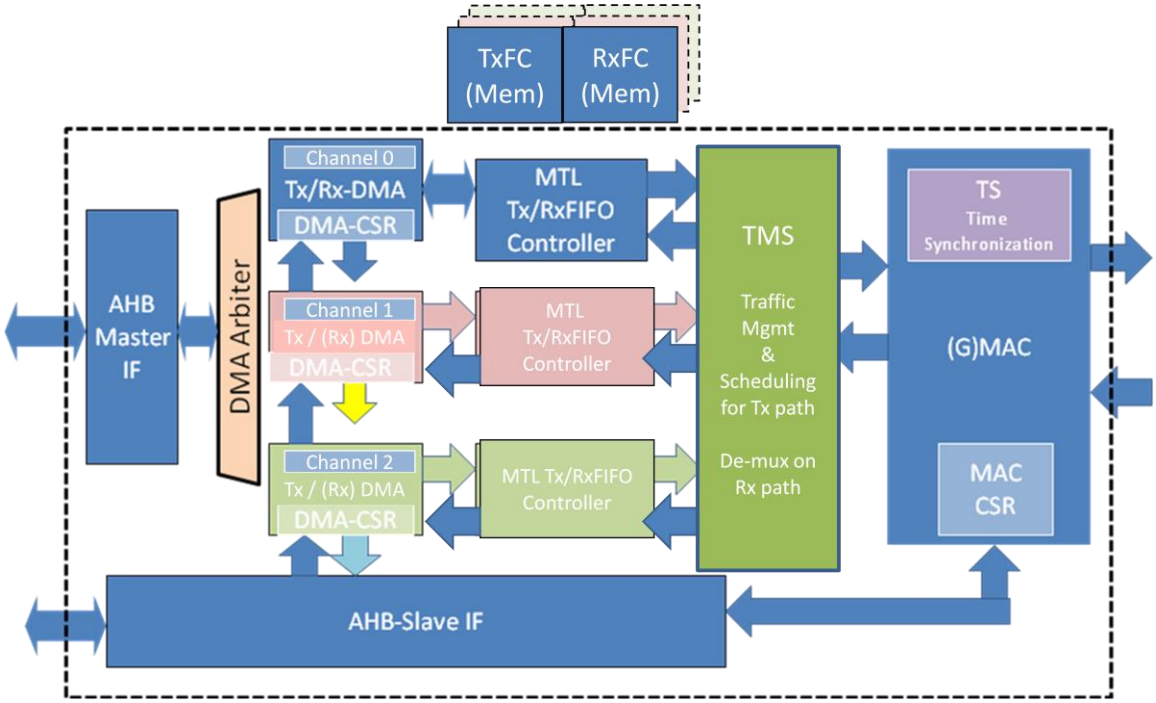


Fig 1 Top level block diagram with only 2 additonal Tx channels

2 High-Level View of the Linux Network Stack

Figure 2 provides a high-level view of the Linux network stack. The shared boxes indicates the Synopsys components.

Between the protocols layer and driver is a agnostic interface layer, shown as a circle. This agnostic interface layer connects protocols to a variety of hardware device drivers with varying capabilities. It's a general practice that Device driver registers only one "interface" for every hardware it is intended to control. Since AVB demands multiple (more than one) payload from different sources, driver needs to register multiple interfaces to underlying hardware. Since the application for AVB is not well defined and stack is not yet developed, we follow a different approach to handle multiple payload streams of AVB. The approach is intended to

1. validate the hardware and
2. Collecting various statistics.

Once the Network protocol stack is enhanced and applications for AVB are defined, driver can be modified to meet the requirements with minimal effort.

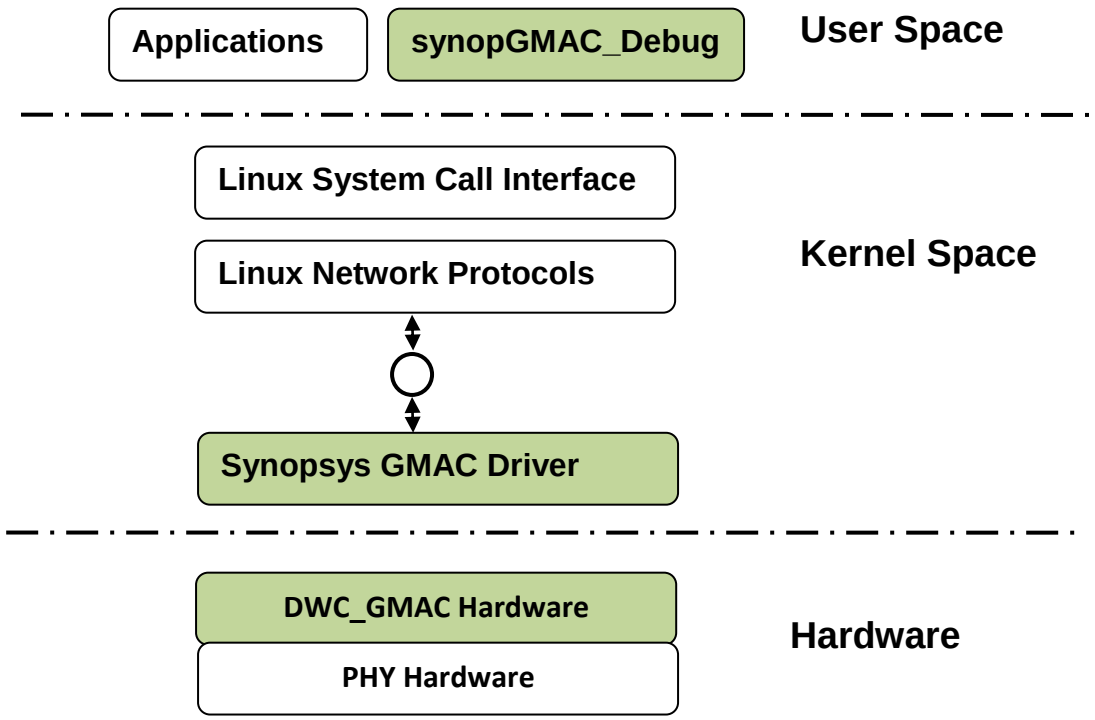


Fig 2: GMAC Driver architecture

3 AVB support in GMAC driver

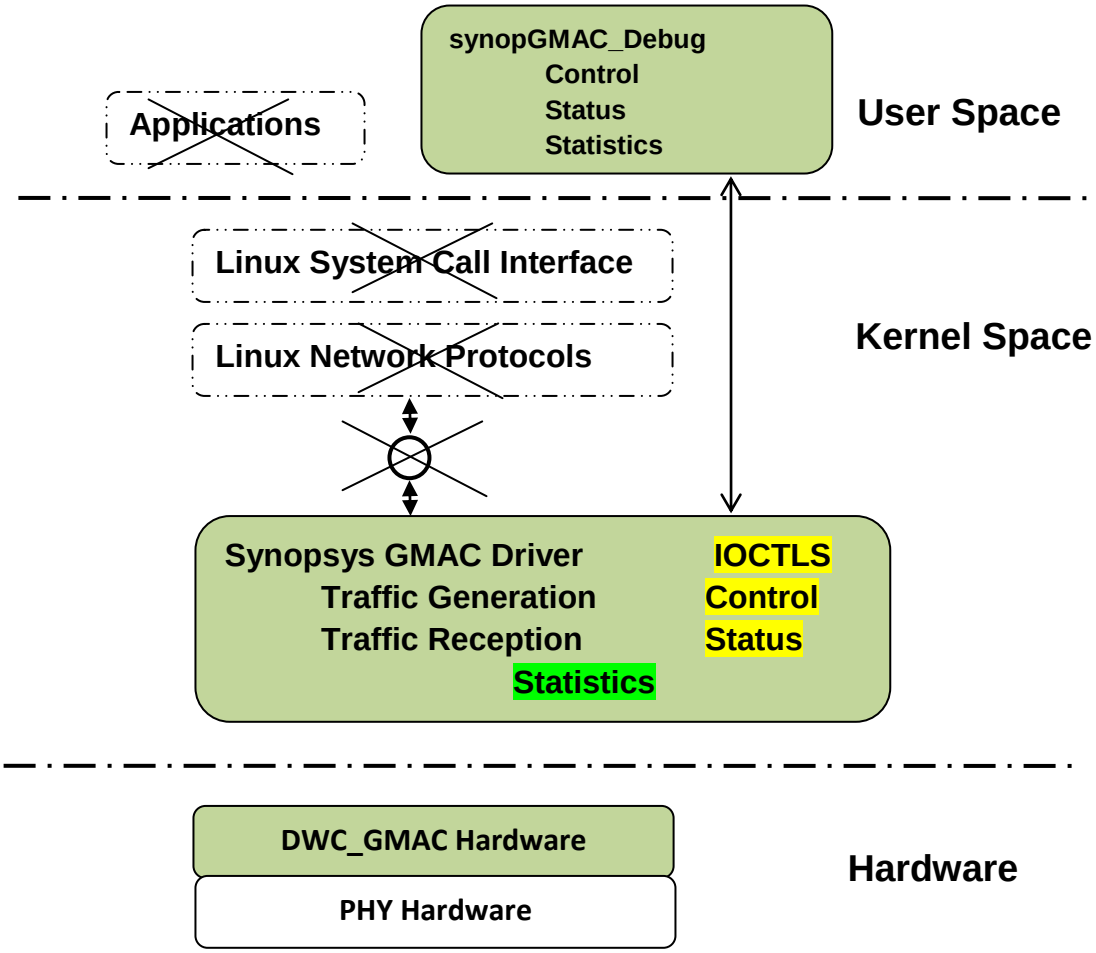


Fig 3: GMAC Driver architecture for AVB

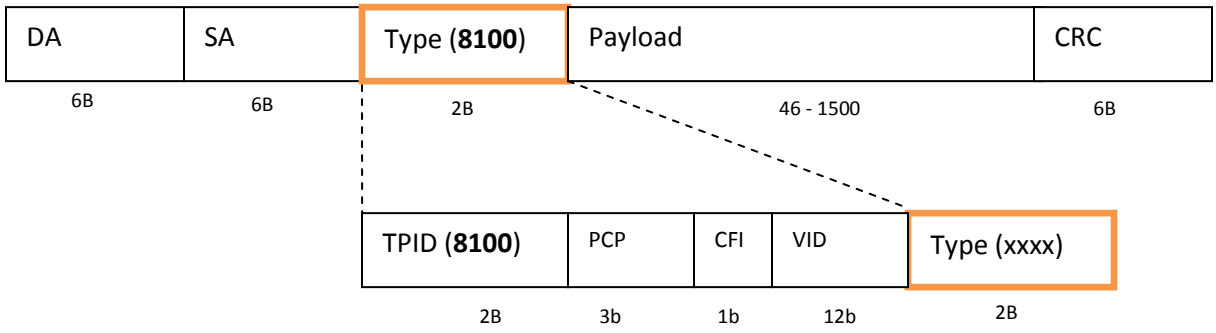
The AVB support in GMAC driver comes with compiling the driver software with the AVB_SUPPORT switch. When the driver is compiled with the switch "AVB_SUPPORT", driver does not service the normal traffic. AVB packets (Both Class-A and Class-B) and the Best Effort Ethernet packets are generated by the driver and transmitted. The driver does not communicate any packets/frames with the Operating system and hence the networkings stack.

Note: When AVB evaluation is going on the normal traffic is interrupted.

4 AVB frames used for Evaluation

The AVB frames used for evaluation are VLAN tagged frames.

Example AVB Data/Control Frames:



TPID : Tag Protocol ID (TPID)

PCP : Priority Code Point (PCP) range is from 0 through 7

CFI : Canonical Format Indicator (CFI)

VID : VLAN Identifier (VID)

The Driver assumes the following for a valid AVB frame

TPID : 0x8100 indicating it is a 802.1Q frame

PCP : 0 through 7 (we can set two values to identify Class A and Class B traffic)

4 => Class A

5 => Class B

CFI : 0 for AVB

VID : 0 for AVB

Type : TBD (Programmable at receiver in AVTYPE field of AV MAC Control Register)

Talker:

A sample AVB packet with fixed frame length and a known payload is populated inside the driver. This frame is a VLAN tagged frame with TPID=8100, PCP=4,CFI=0,VID=0 and Type=xxxx. This frame acts as a sample Class-A AVB frame.

Create a similar frame (can change the frame length and payload) and populate with AVB attribute similar to Class-A AVB frame except PCP=5. This frame serves as a Class-B AVB frame.

Generate a untagged frame with a known frame length and known payload which acts as a best effort Ethernet frame.

Listener:

The software does not differentiate the transmitter (talker) and Listener. Frames/packets are identified based on the Ethernet Mac address and GMAC hardware handles this (MAC frame filter configuration).

Note: the type field in the AV frame and the AVTYPE in the AV MAC Control registers are programmed with the same value (xxxx).

Changing the Transmit Frame Structure

The frames for each channel are identified based on the type of its header. Channel1 and Channel2 frames are Tagged VLAN frames, where as Channel 0 frames are untagged frames. The Ethernet header for respective channels is hard coded in "synopGMAC_network_interface.c" file.

```
static int frame_header_ch2[] = {
        0x73560D00, //   DST: 00:0D:56:73:D0:F3
        0x5500F3D0, //   SRC: 00:55:7B:B5:7D:F7
        0xF77DB57B,

        0x00A00081  //   Eth type: 0x8100 VID = 0x0000 CFI=0 PCP =5
                    //   AVTYPE is read from the #define and used
                         while creating a frame
};

static int frame_header_ch1[] = {
        0x73560D00, //   DST: 00:0D:56:73:D0:F3
        0x5500F3D0, //   SRC: 00:55:7B:B5:7D:F7
        0xF77DB57B,

        0x00800081  //   Eth type: 0x8100 VID = 0x0000 CFI=0 PCP =4
                    //   AVTYPE is read from the #define and used
while creating a frame
};

static int frame_header_ch0[] = { //BEST EFFORT ETHERNET HEADER
        0x73560D00, //   DST: 00:0D:56:73:D0:F3
        0x5500F3D0, //   SRC: 00:55:7B:B5:7D:F7
        0xF77DB57B,

        0x00000008  //   Eth type: 0x0800
};
```

The first three long words (12 bytes) of the frame_header indicates the Destination address and Source address. For channel2, with Destination and Source address being 00:0D:56:73:D0:F3 and 00:55:7B:B5:7D:F7 respectively, The twelve bytes are as shown below. The numbers in red indicate the destination address and numbers in blue indicate the Source address. These three long words remain same for all the three channels

0x73560D00
0x5500F3D0
0xF77DB57B

For Channel1 and Channel2, the fourth long word indicates the VLAN tag. Considering channel2, the VLANG tag contents of the frame is decoded as shown below.

Channel2
0x00A00081
0081 ETH_TYPE is 0x8100, indicating this is a VLAN frame.
00A0 VID=H'000 CFI=B'0 and PCP=B'101

Channel1
0x00800081
0081 ETH_TYPE is 0x8100, indicating this is a VLAN frame.
0080 VID=H'000 CFI=B'0 and PCP=B'100

Channel0 is a untagged frame. The fourth long word of the Channel0 header is decoded as

Channel0
0x00000008
0008 Eth type is 0x0800, indicating it is untagged frame

The data pattern transmitted on a particular channel is fixed to a known value. This helps in identifying the tx channel at the receiving end. The patterns are hard coded to the values given below in "synopGMAC_network_interface.c" file.

```
#define FRAME_PATTERN_CH2    0xAAAAAAAA
#define FRAME_PATTERN_CH1    0x55555555
#define FRAME_PATTERN_CH0    0x11223344
```

Transmit Path

Transmitter can be configured to use at maximum three channels (one for Best Effor Ethernet and two for prioritized AV traffic). Each of the Channels is assigned its own descriptors. Respective descriptors hold the buffer address from a corresponding data buffer. Descriptors are populated with the frames for transmission. On every transmit complete interrupt, descriptor is assigned with a data buffer address and a Frames_Counter_Channel# counter is incremented. This Frames_Counter_Channel# is used to compute the percentage of bandwidth utilization.

This collected Frames_Counter_Channel# is checked against the Channel # Average Bits Per Slot accumulated over the duration of the experiment. The numbers are verified against the counters maintained at the listener.

Receive Path

Only one receive path is enabled immaterial of number of channels used in the transmitter/transmit path. For the validation purpose, we assume AV packets are always VLAN tagged frames. Following information identifies the different packets received.

1. Best Effort Ethernet packets are Non-Vlan-Tagged frames.
2. PCP = 4 set for Channel#1
3. PCP = 5 set for channel#2

Once received, the packets are identified by checking the Rx descriptor Extended status VLAN Tag Priority Value. Respective counters are incremented which are tallied against the transmitter to validate the proper functioning.

6 Software for AVB Testing

The driver should be compiled with the ENH_DESC, ENH_DESC_8W and AVB_SUPPORT compilation switches to enable the AV testing. If driver is compiled without the above mentioned switches, functionality is undefined and could possibly lead to a system hang*.

```
EXTRA_CFLAGS += -DENH_DESC -DENH_DESC_8W -DAVB_SUPPORT
```

The driver software is accompanied with the supporting application software synopGMAC_Debug.c. This software should be run in the user domain in contrast to the driver software which runs in the kernel domain. The application software is compiled as shown below

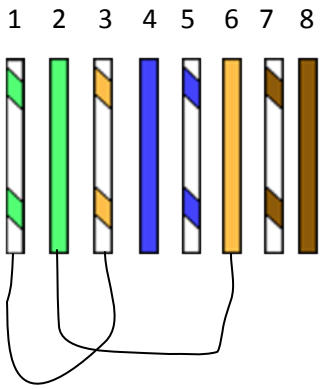
```
gcc synopGMAC_Debug.c –o synopGMAC_Debug
```

To minimize the variables in the experiment and to get the experimental results to a greater precision, only one GMAC hardware is incorporated in the AV testing. The hardware setup and programming steps are explained in the next section

*- The Driver supports only RING_MODE of Descriptor structure. Don’t use the current version of the driver with CHAIN_MODE of Descriptor structure.

7 Hardware Setup for AVB Testing

The testing makes use of one hardware board for AV validation. The setup incorporates a “Ethernet Loop Back cable” to enable the testing of transmit path function and receive path functions from the same system. The cable modifications are as shown below.



Pin#1	Transmit+
Pin#2	Transmit-
Pin#3	Receive+
Pin#4	Unused
Pin#5	Unused
Pin#6	Receive-
Pin#7	Unused
Pin#8	Unused

The altered (as shown above) cable is plugged in to the Ethernet plug on the board. After successful loading of the driver, the PHY is forced to work in 100Mbps Fullduplex mode by writing the Phy control register using the debug application

```
insmod synop_GMAC_ether.ko

ifconfig eth0 up

./synopGMAC_Debug eth0 write mii 0x0 0xA100
```

Please refer the data book for the Ethernet PHY (88E1011S-Marvel-DS) for details.

8 AV Testing

After proper insertion of the driver module to the kernel, invoke the debug application as shown below. The application should be launched with two arguments

1. The name of the interface (eth0 in this case)

2. avbtest, to launch the ABV test application.

```
[root@matrix Gmac-driver-3.6]# ./synopGMAC_Debug eth0 avbtest
*******************************************
 *  synopGMAC debug Utility for Linux       *
 *  Copyright(c) 2007-2007 Synopsys, Inc    *
 *******************************************
Successfully Configured AVB parameters
Successfully Configured The Hardware for AVB
You are about to start AVB Testing....
```

The following menu is displayed which will be used to carry out AV testing

```
-------------------------------------------------
Select the Options Below:
-------------------------------------------------
(01) Show/Set the Avb Parameters
(02) Send Parameters to Driver
(03) Configure HW for AVB Test
(04) Run AVB test
(05) Show AVB Test Reports
(06) Enter Debug Mode
(07) Print AVB Structure
(99) To quit this menu
Your Option here::1
```

AV test can be run by selecting the option (04). This runs the test with default AV parameters. The default parameters for AV testing are shown in below dump. The option (01) displays the submenu shown below. This menu facilitates the changing the AV test parameters.

```
-------------------------------------------------
Select the Options Below:
-------------------------------------------------
(01) Duration to run the Experiment (seconds)  : 10

(02) DMA Channel Enable                        : 07

(11) Ch2 Frame size in bytes (Only Payload)    : 1000
(12) Ch2 Bandwidth Allocation                  : 50%
(13) CH2 Enable Slot Number Check              : 0
(14) CH2 Advanced slot interval data fetch     : 0
(15) Ch2 Slot Counter for AVG Bits Reporting   : 16
(16) Ch2 Tx and Rx Priority Scheme  (1/0)      : 0
(17) Ch2 Tx has High priority over Rx (1/0)    : 1
(18) Ch2 Tx and Rx Priority Ratio              : 1
(19) Ch2 Uses Credit Shaped Algorith (1/0)     : 1
     Ch2 Uses Credit Control Algorith (1/0)    : 0
(20) Ch2 Tx Desc Starting Slot No.   <0 .. 15> : 0
(21) Ch2 Tx Desc Slot No. Skip count <0 .. 15> : 0
(22) Ch2 Arbitration Weight          <1,2,3,4> : 1

(31) Ch1 Frame size in bytes (Only Payload)    : 1000
(32) Ch1 Bandwidth Allocation                  : 25%
(33) CH1 Enable Slot Number Check              : 0
(34) CH1 Advanced slot interval data fetch     : 0
(35) Ch1 Slot Counter for AVG Bits Reporting   : 16
(36) Ch1 Tx and Rx Priority Scheme  (1/0)      : 0
(37) Ch1 Tx has High priority over Rx (1/0)    : 1
(38) Ch1 Tx and Rx Priority Ratio              : 1
(39) Ch1 Uses Credit Shaped Algorith (1/0)     : 1
     Ch1 Uses Credit Control Algorith (1/0)    : 0
(40) Ch1 Tx Desc Starting Slot No.   <0 .. 15> : 0
(41) Ch1 Tx Desc Slot No. Skip count <0 .. 15> : 0
(42) Ch1 Arbitration Weight          <1,2,3,4> : 1

(51) Ch0 Frame size in bytes (Only Payload)    : 1000
(52) Ch0 Tx and Rx Priority Scheme  (1/0)      : 0
(53) Ch0 Tx has High priority over Rx (1/0)    : 1
(54) Ch0 Tx and Rx Priority Ratio              : 1
(55) Ch0 Arbitration Weight          <1,2,3,4> : 1

(99) To quit this menu

Your Option here::99
```

One should select the number in the brackets to modify the parameters.

For example, "**Duration to run the Experiment (seconds) : 10**" indicates the experiment will be run for 10 seconds. If uses select the option (1) or (01), another sub-menu is displayed where user can select a different value for this parameter.

The parameters select can be sent to the driver module (kernel space) by selecting the option (02) from the main menu. The dump for the same is shown below

```
-------------------------------------------------

Select the Options Below:

-------------------------------------------------

(01) Show/Set the Avb Parameters

(02) Send Parameters to Driver

(03) Configure HW for AVB Test

(04) Run AVB test

(05) Show AVB Test Reports

(06) Enter Debug Mode

(07) Print AVB Structure

(99) To quit this menu


Your Option here::2

Successfully Configured AVB parameters
```

From these user set parameters, actual register configuration are derived inside the driver software. Hardware configuration is done through the option (03). This programs the GMAC/DMA registers with the deriver configuration values.

The transmit descriptors buffer pointers are populated with the valid data and the length field of the descriptor is modified to the actual length of the packet.

The option 03 is provided to facilitate the debugging. One can have the register dumps after the option 03, to ensure the registers are configured with the correct values before executing the test.

```
------------------------------------------------

Select the Options Below:

------------------------------------------------

(01) Show/Set the Avb Parameters

(02) Send Parameters to Driver

(03) Configure HW for AVB Test

(04) Run AVB test

(05) Show AVB Test Reports

(06) Enter Debug Mode

(07) Print AVB Structure

(99) To quit this menu


Your Option here::3

Successfully Configured The Hardware for AVB
```

AV testing will start with the option 04. When user invokes the "**(04) Run AVB test**",

1. A timer called "AV timer" is started with timer expiry period programmed to the parameter "Duration of the Experiment" parameter sent to the driver
2. The DMA channels, as indicated in "**DMA Channel Enable" parameter** are started for transmission
3. Receiver has only one channel CH0 and all the packets are received on this channel.
4. The non-Vlan tagged frames are put under Channel0, AV-tagged frames with PCP value 5 are put under Channel2 and AV-tagged frames with PCP value 4 are put under Channel1.
5. When the "AV timer" expires, all the DMA's are stopped

After the successful completion of the AV testing, user is informed with a message as shown below. This message gets logged in to /var/log/messages. One way to look for the message is to run the command in one of the terminals, and look for the message after running the AV test.

```
tail –f /var/log/messages
```

```
AVB Experiment is Complete... You can Retrieve the Report
```

AV test report can be retrieved by making use of option (05). The report retrieved for a sample test is shown below

```
Your Option here::5
Successfully Retrieved AVB Results
-------------------------------------------------------------------------
Duration Of The Experiment (Seconds)       : 10
DMA Channel Enabled                        : 07
-------------------------------------------------------------------------
                                   CH2            CH1            CH0
CH Prio Weights                    1              1              1
Frame Size                         1000           1000           1000
Band Width Allocation              050(%)         025(%)         025(%)
Slot Number Check Enabled?         0              0              -
Adv Slot Int Data Fetch?           0              0              -
Slot Counter For Avg Bit Report    08             16             -
Tx-Rx Prio Scheme(0=>RR 1=>SP)     0              0              0
Tx Has High Prio Over Rx           1              1              1
Tx and Rx Prio Ratio (For RR )     1              1              1
Ch Uses Credit Shape Algo?         1              1              -
Ch Uses Credit Control Algo?       0              0              -
Ch Idle Slope                      00000800       00000400       -
Ch Send Slope                      00000800       00000c00       -
Ch Hi Credit                       00100000       00080000       -
Ch Lo Credit                       fffffc00       fffffa00       -

Ch Transmit Frame Count            0000ee60       00007731       00007730
Ch Receive Frame Count             0000ee60       00007731       00007730
Ch Bw Util Achieved                49.40(%)       24.70(%)       24.70(%)  *
(Preamble+Ether-Frame+FCS-IPG )
Ch Avg Bits Per Slot               000017b8       00000bdc       -
Ch Avg Bits Per Slot Accumulated(Hi)00000000      00000000       -
Ch Avg Bits Per Slot Accumulated(Lo)03ae362d      00eb87b2       -
Ch Avg Bits Interrupts (total)     000026f6       0000137b       -
Channel BW from Avg Bits           49.53(%)       24.76(%)       -         *
-------------------------------------------------------------------------
Enter 99 return to main menu::
```

*****- In the report “Channel BW from Avg Bits” reports a slightly higher bandwidth with respect to the “Ch Bw Util Achieved”. The duration of experiment, in the above experiment it is 10 seconds, is controlled by the Software timer facilitated by Linux Operating system. This timer may not give accurate interrupts. Higher accuracy can be achieved by incorporating hardware timer (GMAC Time stamp interrupt).

The following snapshot gives the AV Report when Channel 2 is disabled by writing 0x03 to “DMA Channel Enable”. Note that Channel 1 still continues with the same allocated BW of 25 percent. All remaining BW is allocated to Channel 0, resulting 75% of allocation to Channel 0.

```
Your Option here::5
Successfully Retrieved AVB Results
-------------------------------------------------------------------------
Duration Of The Experiment (Seconds)       : 10
DMA Channel Enabled                        : 03
-------------------------------------------------------------------------
                                    CH2            CH1            CH0
CH Prio Weights                     1              1              1
Frame Size                          1000           1000           1000
Band Width Allocation               050(%)         025(%)         025(%)
Slot Number Check Enabled?          0              0              -
Adv Slot Int Data Fetch?            0              0              -
Slot Counter For Avg Bit Report     08             16             -
Tx-Rx Prio Scheme(0=>RR 1=>SP)      0              0              0
Tx Has High Prio Over Rx            1              1              1
Tx and Rx Prio Ratio (For RR )      1              1              1
Ch Uses Credit Shape Algo?          1              1              -
Ch Uses Credit Control Algo?        0              0              -
Ch Idle Slope                       00000800       00000400       -
Ch Send Slope                       00000800       00000c00       -
Ch Hi Credit                        00100000       00080000       -
Ch Lo Credit                        fffffc00       fffffa00       -

Ch Transmit Frame Count             00000000       00007733       00016592
Ch Receive Frame Count              00000000       00007733       00016592
Ch Bw Util Achieved                 0.00(%)       24.70(%)       74.10(%)
(Preamble+Ether-Frame+FCS-IPG )
Ch Avg Bits Per Slot                00000000       00000c63       -
Ch Avg Bits Per Slot Accumulated(Hi)00000000       00000000       -
Ch Avg Bits Per Slot Accumulated(Lo)00000000       00eb938d       -
Ch Avg Bits Interrupts (total)      00000000       0000137c       -
Channel BW from Avg Bits            NAN(%)       24.76(%)       -
-------------------------------------------------------------------------
Enter 99 return to main menu::
```

```
Your Option here::5
Successfully Retrieved AVB Results
--------------------------------------------------------------------------
Duration Of The Experiment (Seconds)       : 02
DMA Channel Enabled                        : 01
--------------------------------------------------------------------------
                                    CH2            CH1            CH0
CH Prio Weights                     1              1              1
Frame Size                          1000           1000           1000
Band Width Allocation               050(%)         025(%)         025(%)
Slot Number Check Enabled?          0              0              -
Adv Slot Int Data Fetch?            0              0              -
Slot Counter For Avg Bit Report     08             16             -
Tx-Rx Prio Scheme(0=>RR 1=>SP)      0              0              0
Tx Has High Prio Over Rx            1              1              1
Tx and Rx Prio Ratio (For RR )      1              1              1
Ch Uses Credit Shape Algo?          1              1              -
Ch Uses Credit Control Algo?        0              0              -
Ch Idle Slope                       00000800       00000400       -
Ch Send Slope                       00000800       00000c00       -
Ch Hi Credit                        00100000       00080000       -
Ch Lo Credit                        fffffc00       fffffa00       -

Ch Transmit Frame Count             00000000       00000000       00005f5b
Ch Receive Frame Count              00000000       00000000       00005f5b
Ch Bw Util Achieved                 0.00(%)       0.00(%)       98.81(%)
(Preamble+Ether-Frame+FCS-IPG )
Ch Avg Bits Per Slot                00000000       00000000       -
Ch Avg Bits Per Slot Accumulated(Hi)00000000       00000000       -
Ch Avg Bits Per Slot Accumulated(Lo)00000000       00000000       -
Ch Avg Bits Interrupts (total)      00000000       00000000       -
Channel BW from Avg Bits            NAN(%)       NAN(%)       -
--------------------------------------------------------------------------
Enter 99 return to main menu::
```

1. Synopsys GMAC Driver software
2. IEEE P802.1Qav/D4.0 Virtual Bridged Local Area Networks - Amendment XX: Forwarding and Queuing Enhancements for Time - Sensitive Streams Specification - January 2009 (http://standards.ieee.org)
3. IEEE P802.1Qat/D2.1 **Virtual Bridged Local Area Networks -Amendment 9: Stream Reservation Protocol (SRP)** Specification - February 2009 (http://standards.ieee.org)
4. IEEE P802.1AS/D5.0 Timing and Synchronization for Time -Sensitive Applications in Bridged Local Area Networks Specification - February 2009 (http://standards.ieee.org)
5. DWC Ethernet GMAC Universal data book version 3.50a
6. IEEE P1722/D1.3 Draft Standard for Layer 2 Transport Protocol for Time Sensitive Applications in Bridged Local Area Networks – February 2009 (http://standards.ieee.org)
7. GMAC_UNIV Release 3.60a Functional Specification Version 0.41 – March 2009