XAPP1266 (v1.0) September 18, 2015

# Using Quality of Service (QoS) Capabilities in Zynq-7000 AP SoC Devices

Authors: Mrinal J. Sarmah, Naveen Kumar Gadddipati

## Summary

In the Zynq®-7000 All Programmable (AP) SoC device, it is possible to implement the following types of quality of service (QoS):

- *Basic QoS*, which is a per-transaction priority based on the AXI QoS signals You can use the Basic QoS signals of Interconnect AXI HP ports by configuring the priorities, and controlling the read and write transactions. Additionally, you can use Port DDR QoS signals in DDR slave ports to control traffic from and to AXI HP master ports during video playback.
- *Advanced QoS*, which is based on QOS-301 block, and has additional capability to control peak rate, bursting, and so forth. You can use the Advanced QoS signals of CPU by using the peak rate, burst rate, and average rate of the CPU along with enabling the regulation for read or write transactions to avoid the latency issues during video playback.

The application uses the AXI traffic generator IP (ATG), test pattern generator (TPG) and also the Sobel filter.

- ATG IP is used to generate the traffic on one of the AXI HP ports and the ACP port.
- TPG IP is used to generate the test pattern video source on one of the AXI HP ports.
- Sobel filter IP is used to filter the video source given by TPG on one of the AXI HP ports.
- The Linux application running on the Zynq device controls the ATG traffic by enabling the QoS for the CPU, AXI HP ports and also on the DDR ports.

***Note:*** If there is no latency or bandwidth issue in DDR ports, while writing the data using AXI HP or ACP ports, there is no requirement for QoS.

The performance of the ATG, TPG and Sobel filter are measured using the AXI performance monitor (APM) IP. This application note demonstrates the operation of both basic QoS and advanced QoS in Zynq-7000 devices, and describes the following:

1. The hardware and software design
2. A web-server based GUI interface used to enable or disable the video
3. A Sobel filter
4. An AXI traffic generator (ATG)
5. The QoS services

## Prerequisites

The following are the pre-requisites for running this reference design:

- Vivado Design Suite (version 2015.2 or higher (this is an IP integrator design).
- Petalinux Tools version 2015.2 or higher (one that includes XSDK) [Ref 9].

Also, see the following documents for more information:

- *SDK System Performance Guide* (UG1145) [Ref 7]
- *System Performance Analysis of an All Programmable SoC Application Note* (XAPP1219) [Ref 8]

***Note:*** If there is no latency or bandwidth issue in the system, there is no need for QoS.

## Introduction

The Zynq-7000 family is based on the Xilinx All Programmable (AP) SoC architecture. These products integrate a feature-rich dual-core ARM® Cortex™-A9 MPCore™ based processing system (PS) and Xilinx programmable logic (PL) in a single device. The ARM Cortex-A9 MPCore CPUs are central to the PS, and the PS also includes on-chip memory, external memory interfaces, and a rich set of I/O peripherals.

The communication between PS and PL happens using dedicated PS-PL interface ports. In a system-level design, there are multiple sources of data generators in the PS and PL that include the Cortex-A9 CPU, PS DMA, PL masters, and so on.

The PL masters can be attached to the PS-PL interface slave ports. Congestion in the Interconnect can be caused when all the masters try to access a shared resource (for example, DDR memory). This becomes critical for latency-sensitive applications, like video, because not ensuring the required latency might cause jitter in the output video.

The AXI interface offers basic QoS on a per-transaction basis using the AXI QoS signals that have a 4-bit QoS configuration field. The value in this field determines the priority of the AXI transaction.

You can implement QoS in two ways in the Zynq-7000 device, using the following:

- Dynamic QoS configuration that requires the QoS value to be driven in a PL master per transaction basis. Dynamic QoS is managed with AXI-compliant AxQoS signals.
- Static QoS setting of the AXI Interconnect and the DDR controller ports. Static QoS setting fixes the priority of the port in AXI Interconnect irrespective of the value of the AxQoS signal.

Figure 1, page 3 shows the placement of the QoS blocks in PS of the Zynq-7000 device.

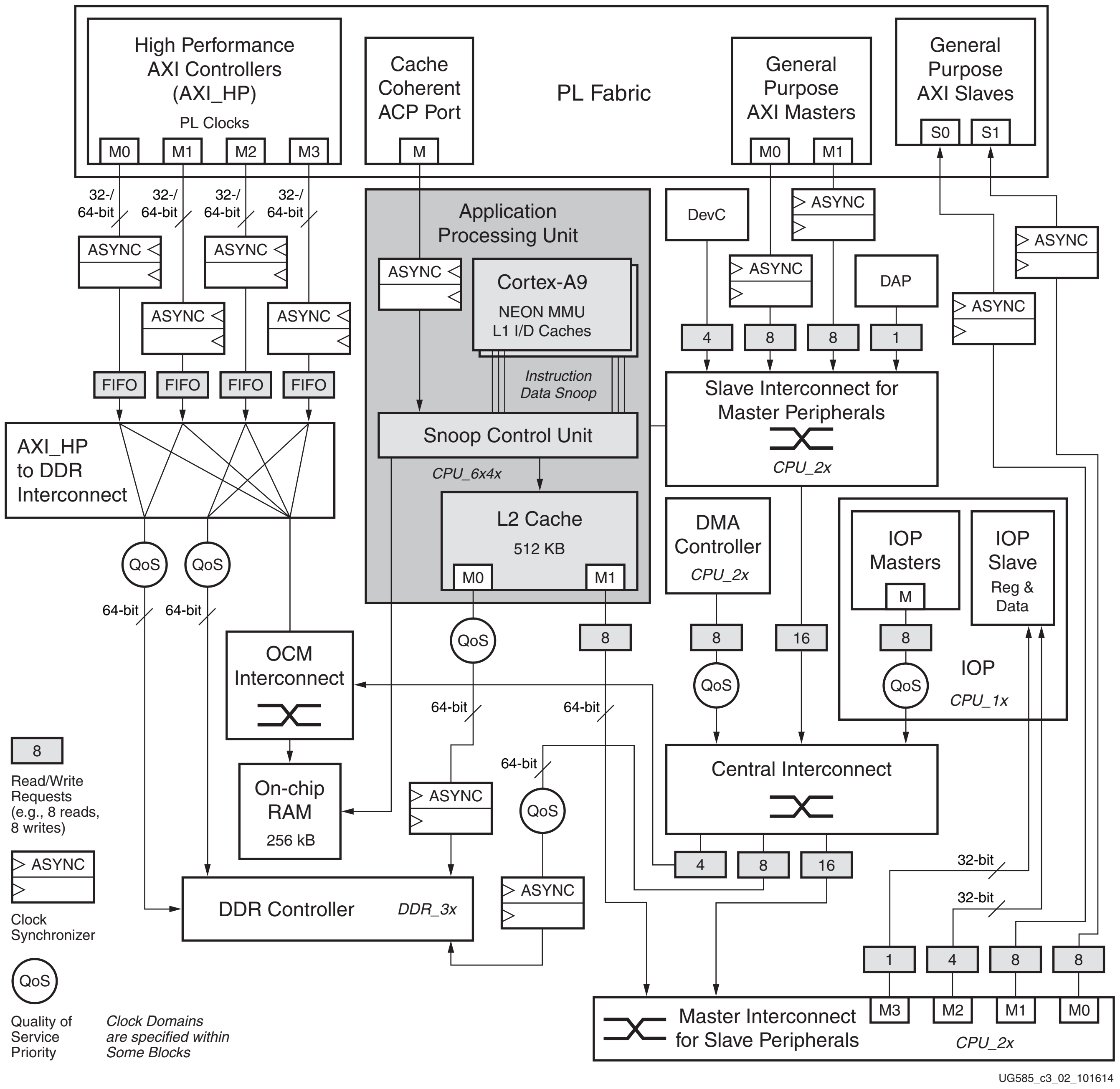


*Figure 1:* **System View Diagram**

The QoS blocks can be configured using register programming interface.

In Zynq-7000 AP SoC devices, it is possible to enable the advanced QoS option using the QoS-301 block that enables rate control, based on number of bursts in the data traffic.

Each Interconnect in the PS (central, master, slave, memory) uses a two-level arbitration scheme to resolve traffic contention.

- The first-level arbitration is based on the priority indicated by the `AW/RQOS` signals from the master or programmable registers. The highest QoS value has the highest priority.
- The second-level arbitration is based on a least recently granted (LRG) scheme and is used when multiple requests are pending with the same QoS signal value.

In addition to the basic arbitration, the AXI Interconnect in the PS provides an advanced QoS control mechanism using the QoS-301 block. This programmable mechanism influences arbitration within the Interconnect for requests from these masters:

- CPUs and ACP requests to DDR
- DMA controller requests to DDR and OCM (through the central Interconnect)
- AMBA master requests to DDR and OCM (through the central Interconnect)

In the PS, advanced QoS signals exist on the following paths:

- Path from L2 cache to DDR
- Path from DMA controller to the central interconnect
- Path from AHB masters to the central interconnect

The QoS-301 signals in the ARM processor provides facilities to regulate transactions as follows:

- Maximum number of outstanding transactions
- Peak rates
- Average rates
- Burst rates

For more information, see Section “B.20 Interconnect QoS (QoS-301)” in the *Zynq-7000 Technical Reference Manual* (UG585) [Ref 1].

You need to perform QoS arbitration for all slave interfaces with careful deliberation, because fixed priority arbitration leads to starvation issues if not used properly. By default, all ports have equal priority so starvation is not an issue.

This reference design describes an approach that demonstrates how you can use QoS when the memory subsystem is stressed.

A designer is expected to create “well behaved” masters in the PL, which sufficiently throttle their rate of command issuance, or use the `AXI_HP` issuance capability settings. However, traffic from CPUs (through L2 cache), the DMA controller, and the IOP masters can interfere with traffic from the PL. The QoS modules allow throttling these PS masters to ensure expected and consistent throughput and latency for the user design in the PL or in specific PS masters.

The PS Interconnect uses all four QoS signals except where it attaches to the DDR memory controller, which takes only the most significant QoS signal. A three-input MUX selects among the QoS signals, another signal from the `SLCR` register, and a `DDRARB` signal directly from the PL to determine if a request is urgent.

This application note describes a QoS example that is an embedded video processing application designed to showcase the feature and capabilities of the Z-7020 device QoS signals. This design consists of the following elements:

- The Zynq-7000 AP SoC device processing system (PS)
- A test pattern generator (TPG) connected to one PS AXI HP port
- A video processing pipeline in programmable logic (PL)
- ATG is used to measure performance in the HP and ACP interfaces.

You can control the ATG using the EMIO pins to start and stop the traffic generation, and change the traffic in the ATG by changing the burst size of the traffic in the ATG IP.

The AXI performance monitor (APM) monitors and measures the memory bandwidth used on different AXI HP ports.

This application note demonstrates how a user application can control the traffic using QoS signals on AXI HP ports and DDR ports when the data is congested during high traffic conditions.

## Hardware Design Overview

The hardware design is based on the *Zynq-7000 All Programmable SoC ZC702 Base Targeted Reference Design* (Vivado Design Suite 2015.2) (UG925) [Ref 6].

The hardware design is intended to provide the connectivity between the TPG block, the Sobel image filter, the ATG block in the PL, and the Processing System-7, as follows:

- The PL uses the TPG block to generate the test pattern of different colors in vertical blocks by writing into DDR.
- The Sobel image filter block filters the image by reading and writing into the DDR.
- The ATG block generates the traffic by writing the data into pre-defined address range of the DDR.
- The APM is used to monitor the data coming out of the different AXI HPI ports.

See the *ZC702 Video and Imaging Kit Reference Design* [Ref 6] for more details about the hardware design.

The ATG IP is configured to generate high-level data traffic. The burst size is configured between four to 16 bursts to allow generation of an average AXI burst length of eight. The inter-frame gap is set to five cycles so that the ATG master does not create a starvation scenario.

The start and stop signals are mapped to EMIO and the signals can be controlled using software running in PS block.

The following describes the hardware data flow in the design:

1. The TPG block generates video traffic of resolution 1080p or 760p, based on the software configuration.
2. The VDMA interfacing to the TPG block writes the streaming video frames to the DDR of the PS.
3. The VDMA interfacing to the Sobel IP fetches the raw video frames from memory and passes the frames to Sobel filter for image edge detection
4. The PS DDR memory stores the Sobel-processed frames.
5. The display controller fetches the processed frames from memory and displays the video frames using HDMI interface to display device
6. The AXI traffic generator (ATG) blocks are enabled or disabled, based on user selection, to create contention and show jitter on output video.
7. The QoS option is enabled to prioritize the video traffic and demonstrate jitter removal.

The following diagram shows the hardware block diagram.

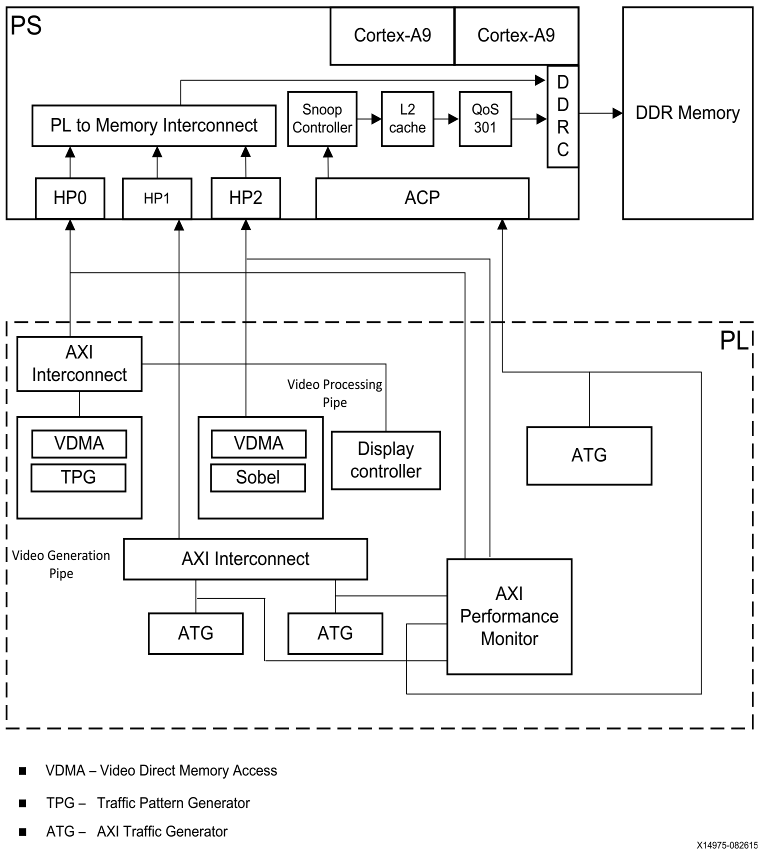


*Figure 2:* **Hardware Block Diagram**

The design is created using the Vivado® IP integrator tool flow. After you create the block design, the IP integrator generates a hardware wrapper that instantiates the IP-specific wrapper files. The following table lists the Xilinx IP in the hardware design.

*Table 1-1:* **IP in Hardware Design**

| IP Name | Description | Configuration |
|---|---|---|
| Processing System-7 | Generates a wrapper that instantiates Processing System-7 hard block. | Configured with ZC702 default values that are preset from the Processing System-7 IP wizard. |
| Image Filter IP | Used to filter the outline of the input video source. | |
| AXI Interconnect IP | AXI Interconnect IP that sends the AXI4 transactions from AXI4 to DDR of processing system. | Configured for 1 master and 1 slave interface. |
| ATG IP | Used to generate the traffic. | Configured to start and stop the traffic using EMIO pins. Configured to generate high level traffic. |
| Proc Sys Reset | Applies reset to peripherals and AXI Interconnect IP. | Configured to apply reset to AXI Interconnect and ATG IP. |
| AXI Performance Monitor | Used to monitor the traffic at different AXI HP interfaces. | Configured for different APM slots for each HP Interface. |

## Software Architecture

Linux provides the following facilities to the user space application:

- Character driver interface for receiving and controlling the information.
- Media control and V4L2 framework is the standard framework used to stream video using TPG as input source

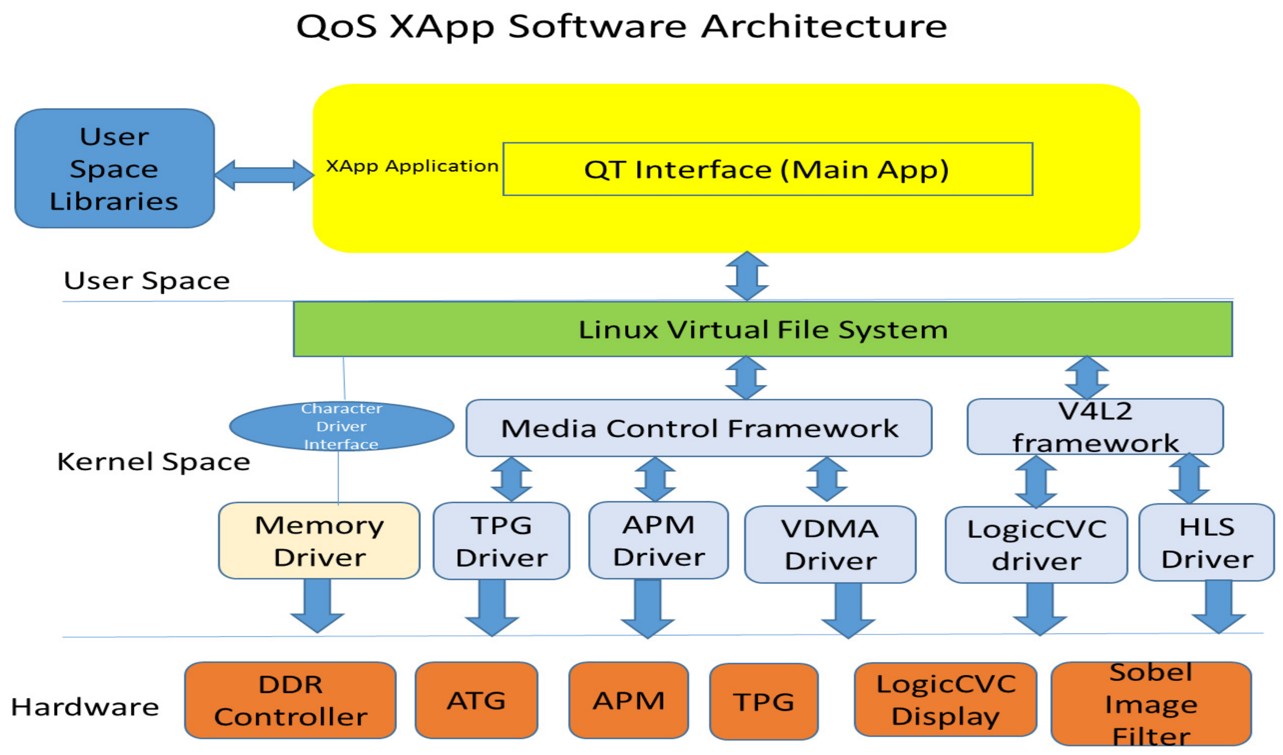
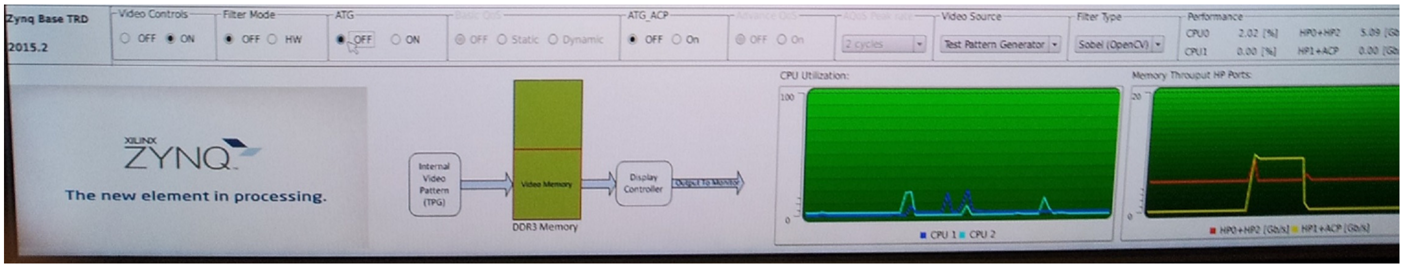


*Figure 3:* **Software Architecture Block Diagram**

The software application implemented for this application note can be divided into four logical sections:

1. Video playback IP:
   a. In this application, the TPG IP provides for video input, which copies the data into DDR using AXI `HP0` interface.
   b. The AXI `HP0` by the LogicCVC controller reads the input video data to display the frames on a connected monitor using HDMI.
   c. The Sobel image filter reads that same data to apply the filtering for getting the outline of the image and writing into the output into the DDR using AXI `HP2` interface.
   d. The LogicCVC reads the output of the Sobel image filter concurrently for display on the monitor.
2. The AXI traffic generation IP (ATG):
   a. Is configured to high-level traffic mode.
   b. Writes the data into a specific region of DDR using AXI `HP1`, which creates the congestion on the DDR `S2` port.
   c. EMIO pins control the start and stop signals of the ATG core.

3. QoS signals:
   a. Basic QoS are the AXI interface signals that the master generates.
   b. Enable the DDR urgent arbitration signals for DDR `S2` and `S3` ports.
   c. Advance QoS configures the QoS301 block for regulating the transactions using peak rate and the number of burst transactions.
4. QT interface: The following figure shows the QT interface:


Zynq Base TRD
2015.2

Video Controls: ○ OFF ● ON
Filter Mode: ● OFF ○ HW
ATG: ● OFF ○ ON
Basic QoS: ◉ OFF ○ Static ○ Dynamic
ATG_ACP: ● OFF ○ On
Advance QoS: ◉ OFF ○ On
AQoS Peak rate: 2 cycles
Video Source: Test Pattern Generator
Filter Type: Sobel (OpenCV)
Performance:
```
CPU0   2.02 [%]   HP0+HP2   5.09 [Gb
CPU1   0.00 [%]   HP1+ACP   0.00 [Gb
```

*Figure 4:* **QT User Interface**

The application normally runs as a demon process in the background on the Zynq-7000 system. The following setting are included in the QT interface:

a. **Video Control:** This control is used to switch on and off the video on the QT interface, which is supplied by TPG and display using LogicCVC IP.

b. **Filter Mode:** This control is used to switch between on and off the Hardware Filtering of the video on the QT interface, which is supplied by TPG.

   When you switch on the Hardware Filter mode, this control used to get the outline of the video source and display using LogicCVC IP. When we switch the filter off, normal TPG video source displays using LogicCVC IP.

c. **ATG**: This control used to start and stop the ATG controllers using EMIO pins on AXI HP ports in basic QoS. When you switch on the ATG along with hardware filter on, it will create congestion while writing the data into DDR ports. This congestion affects the video display and causes flickering on the screen based on the available bandwidths. When you switch off the ATG, you will not see any flickering on the video.

d. **Basic QoS:** This controls enables the QoS signals for static and dynamic configurations of AXI HP ports and DDR ports. It switches off the QoS signals used for the AXI HP ports and DDR ports. By applying the static or dynamic configuration, you can limit the bandwidth for the required AXI HP ports and make the smooth display by decreasing the priority on HP1 and increasing the priority on `HP0` and `HP2` interfaces. In addition to this configuration, you need to enable the DDR QoS signals in the DDR Urgent Selection Register.

e. **ATG ACP:** This control starts and stops the ATG controllers using EMIO pins on ACP port for advanced QoS along with the Hardware Filter on. When you enable the ATG control, ATG makes the congestion while writing the data into DDR ports. This congestion affects the video display and causes flickering on the screen based on the available bandwidths. When you switch off the ATG, you do not see any flickering on the video.

f. **Advanced QoS**: This controls enabling and disabling configurations of CPU write transactions. By applying the QoS-301 configuration to enable the regulation on write rate and outstanding transactions, you can avoid the congestion and make the smooth display by decreasing the peak rate and number of transfers on CPU.

g. **AQoS Peak rate**: This control lets you select the number of cycles required for each transfer, which affects the peak rate graph for various cycles. The following figure shows the peak rate graph.

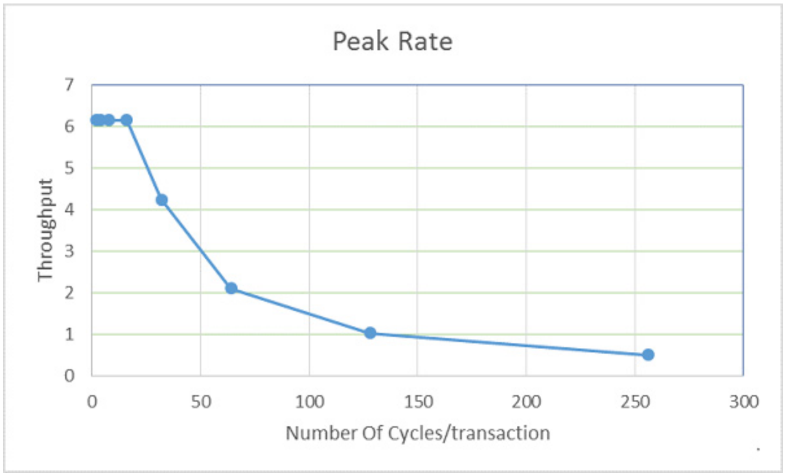


*Figure 5:* **Peak Rate Graph Results**

h. **CPU Utilization**: This plots the graph of the utilization of both CPUs during different use case scenarios.

i. **Performance:** This control used to plot the graph of the performance combination of HP0 and HP2 throughput measured using APM controller slots during usage of the different use case scenarios with and without enabling the filter mode. This control also plot the graph of the performance of HP1 during basic QoS mode and ACP performance during QoS advance mode measured using APM controller slots during usage of the different use case scenarios.

## Application Code

The application code has the following components:

1. For video playback, use the Base TRD source code with QoS hardware design file along with the modified Linux application from reference design zip file. For the build procedure, see the Base TRD URL.

2. To control the ATG start and stop, these are the steps for Basic and Advanced QoS.

   a. Set the respective ATG EMIO pin to start and stop by writing the data into a specific region of the DDR port using `atg_on.sh` and `atg_acp.sh` scripts.

      i. Read and write the GPIO direction register for particular GPIO

      ii. Read and write the GPIO output enable register for particular GPIO

iii. Read and write the GPIO data register for particular GPIO

iv. Repeat the above three steps to start each ATG controller with the respective GPIO pin.

2. To enable the Basic QoS signals, the following code base in `qos_en.sh` has been used:

   a. Configure the priorities on AXI HP interface and DDR ports statically for all transactions as follows:

      i. Assert the write priority for each HP interface

      ii. Assert the read priority for each HP interface

      iii. Assert the DDR S2 and S3 ports QoS signals for static configuration

```
#echo Write priority asset for each HP port
devmem 0xF800801C 32 0xF
devmem 0xF800901C 32 0x1
devmem 0xF800A01C 32 0xF

#echo Read priority asset for each HP port
devmem 0xF8008008 32 0xF
devmem 0xF8009008 32 0x1
devmem 0xF800A008 32 0xF

#SLCR reg for unlock
devmem 0xf8000008 16 0xDF0D
#set the DDR to select ARQoS & AWQoS
devmem 0xf800061C 32 0x5050
#SLCR reg for lock
devmem 0xf8000004 16 0x767B
```

   b. Configure the priorities on AXI HP Ports and DDR ports dynamically for all transactions source code as follows using qos_en.sh

      i. Assert the write priority for each HP interface

      ii. Assert the read priority for each HP interface

```
Assert the DDR S2 and S3 ports QoS signals for Dynamic configuration
 #echo write channel control using QoS for HP0 and HP2
devmem 0xF8008014 32 0xF0A
devmem 0xF800A014 32 0xF0A

#echo read channel control using QoS for HP0 and HP2
devmem 0xF8008000 32 0xA
devmem 0xF800A000 32 0xA


       #SLCR reg for unlock
devmem 0xf8000008 16 0xDF0D
#set the DDR to select ARQoS & AWQoS
devmem 0xf800061C 32 0xA0A0
#SLCR reg for lock
devmem 0xf8000004 16 0x767B
```

4. To enable the Advanced QoS signals, the following code base used in `aqos_en.sh`:

   a. Configure the ACP write transactions as follows:

      i. Enable QoS regulation for AW rate and outstanding write transactions.

ii. Set the maximum number of write outstanding transactions.

iii. Set the number of cycles required for each transaction.

iv. Set the number of transfers required:

```
#Enable QoS regulation for AW rate & outstanding write transactions
devmem 0xF894610C 32 0x21
# max no of aw_ot transactions
devmem 0xF8946110 32 0xF00
#Number of cycles for every transaction selected by user
devmem 0xF8946118 32 <peak rate value>
#channel burstiness set to 1 for write transfers
devmem 0xF894611C 32 0x1
       #SLCR reg for unlock
devmem 0xf8000008 16 0xDF0D
#set the DDR to select ARQoS & AWQoS
devmem 0xf800061C 32 0xA0A2
#SLCR reg for lock
devmem 0xf8000004 16 0x767B
```

## Procedure to use the QT interface

1. Select the Video control to switch on the Video; this enables the Filter mode, ATG, and ATG_ACP control settings.
2. Select the Hardware Filter mode to filter the video source generated by TPG.
3. Select the ATG control; this enables the Basic QoS control selection. Observe the flickering on the screen.
4. Select the Basic QoS either in static or dynamic configuration to avoid the flickering on the display screen.
5. Select the QoS off mode or ATG off mode to enable the ATG_ACP control setting.
6. Select the ATG_ACP control to switch on the ATG on ACP, which enables the Advance QoS control setting.

   Observe the flickering on the screen.
7. Set the Advanced QoS control on by selecting various peak rate cycles from AQoS peak rate interface, which decreases the ATG write transactions while increasing the number of cycles.

## Conclusion

This application note demonstrates the use of Basic and Advanced QOS features to counter traffic congestion issues for video applications.

**IMPORTANT:** *If there is no latency or bandwidth issue, there is no requirement for QoS.*

You can use the Basic QoS signals of Interconnect AXI HP ports by configuring the priorities and controlling the read and write transactions channels along with the DDR QoS signal configuration for DDR slave ports controlling for AXI HP master ports during video playback.

You can use the Advanced QoS signals of CPU by using the peak rate, burst rate, and average rate of the CPU along with enabling the regulation for read or write transactions to avoid the latency issues during video playback.

***Note:*** Follow the instructions provided in the Base TRD for building hardware and software code provided in the `xapp1266-zynq-7000-qos.zip` file.

## Reference Design

You can download the Reference Design Files for this application note from the Xilinx website.

The following table shows the reference design matrix.

*Table 1-2:* **Reference Design Matrix**

| Parameter | Description |
|---|---|
| General | |
| Developer Name | Xilinx |
| Target Devices | Zynq-7000 AP SOC |
| Source code provided | Yes |
| Source code format | C |
| Design uses code and IP from existing Xilinx application note and reference designs, CORE™ Generator software, or third-party. | Yes |
| Simulation | |
| Functional Simulation Performed | No |
| Timing simulation performed | No |
| Test bench used for functional and timing Simulations | No |
| Test bench format | -NA- |
| Simulator software/version used | -NA- |
| SPICE/IBIS simulations | -NA- |
| Implementation | |
| Synthesis software tools/version used | Vivado Design Suite 2015.2 |
| Implementation software tools/versions used | Vivado Design Suite 2015.2 |
| Static timing analysis performed | Vivado Design Suite 2015.2 |
| Hardware Verification | |
| Hardware verified | Yes |
| Hardware platform used for verification | ZC702 |

## References

1. *Zynq-7000 All Programmable SoC Technical Reference Manual* (UG585)
2. *Zynq-7000 All Programmable SoC Software Developer User Guide* (UG821)
3. *LogicCORE IP AXI Traffic Generator Product Guide* (PG125)
4. *LogicCORE IP AXI Performance Monitor Product Guide* (PG037)
5. *Vivado Design Suite AXI Reference Guide* (UG1037)
6. *Zynq-7000 All Programmable SoC ZC702 Base Targeted Reference Design (Vivado Design Suite 2015.2 User Guide)* (UG925)
7. *SDK System Performance Guide* (UG1145)
8. *System Performance Analysis of an All Programmable SoC Application Note* (XAPP1219)
9. PetaLinux Tools

**VIDEO:** *Zynq-7000* Quick Take Videos

## Revision History

The following table shows the revision history for this document.

| Date | Version | Revision |
|---|---|---|
| 09/18/2015 | 1.0 | Initial Xilinx release. |

## Please Read: Important Legal Notices