Spring 2024

EE 382N-4: Advanced Micro-Controller Systems

Lab Assignment #2

DUE FRIDAY MAR 1ST, 2024

Lab Goals:

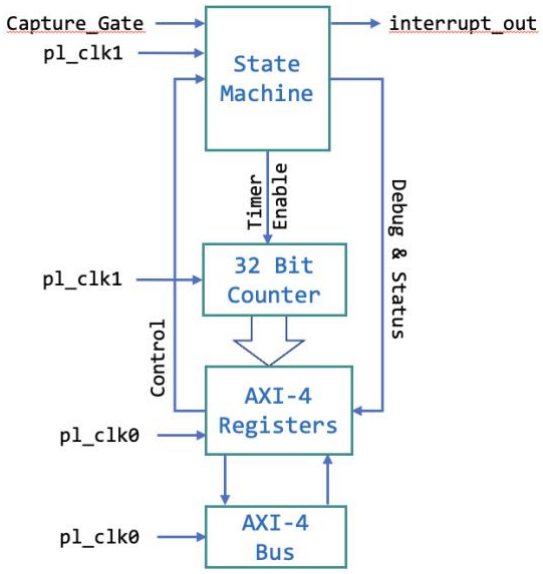
This lab focuses on Interrupt Service Routines (ISR). The student is expected to:

1. Learn how to use the Central Direct Memory Access (CDMA) AXI-4 peripheral. The CDMA will replace the BURST MASTER from Lab 1 to move data to and from the OCM and BRAM. The latency analysis will be done at the maximum PS/PL clock frequencies. There are 2 "switches" that will influence the performance of the CDMA unit. The student will explore the effects of these capabilities in the CDMA and generate a report that explains what each capability does to the transaction times.
2. Build a Timer-Capture AXI-4 unit that measures the time it takes to do a CDMA operation.
3. Augment the Timer-Capture AXI-4 unit to generate an interrupt to the PS. This will be used to measure maximum and minimum interrupt response times of the PS.

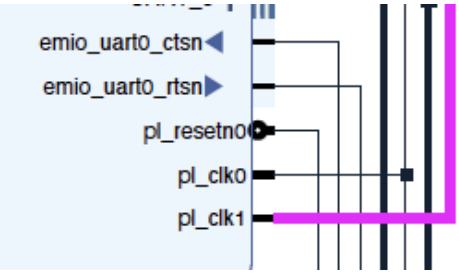
Initial Setup:

Generate a Timer-Capture AXI peripheral:

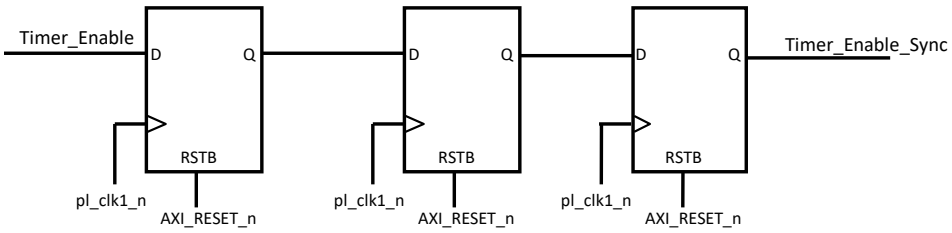
The block diagram is shown to the right. The timer-capture contains a counter that is gated by the CDMA Interrupt signal (Capture Gate). The counter is started when the CDMA transaction starts and then stops when the CDMA asserts the Capture Gate signal which is also an interrupt to the PS. The counter is read in the CDMA ISR and sent to the main routine for processing.



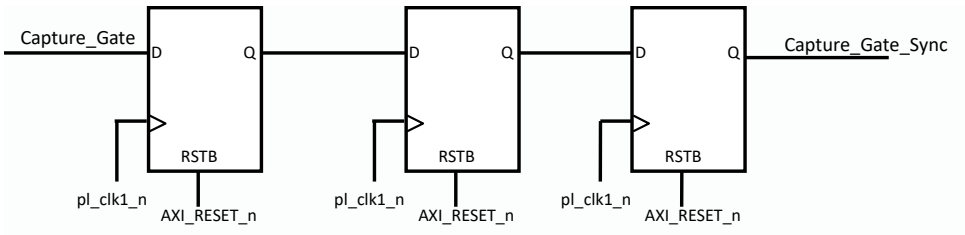
NOTE: You will need to configure the ZYNQ-MP to output a new clock (pl_clk1) that runs at 250 MHz. The pl_clk1 will be used to run the state machine and the counter.



There is a risk that the “Timer_Enable” control signal from the registers will cause the counter state machine to fail. To remedy this situation, add three flip flops that are clocked by pl_clk1_n to synchronize the “Timer Enable” control signal to pl_clk1_n:



There is also a risk that the “Capture_Gate” control signal from the CDMA unit will cause the counter state machine to fail.



You will notice that the synchronizers are clocked on the negative edge. This allows the “Sync” signal time to setup to the State Machine which is clocked on the rising edge of the clock.

The **CAPTURE_TIMER** block will be implemented in Verilog. Here are the register assignments that will be used for this Lab.

Reading register port `slv_reg0[31:0]` will return the following

```
slv_reg0[0]     = capture_gate;          // CDMA interrupt out signal
                                         // to the GIC. Used to halt counter.
slv_reg0[1]     = capture_complete;      // Flag to indicate that the
                                         // capture is complete
slv_reg0[31:2]  = {16'hBEAD, 14'h0};     // Canary
```

Writing to register port `slv_reg1[1:0]` is used to enable counting and the interrupt_out signal

```
assign interrupt_out = slv_reg1[0];        // Enable interrupt
assign timer_enable  = slv_reg1[1];        // Active high enables
                                           // capture timer to count.
```

Reading register `slv_reg1[31:0]` will return the following:

```
slv_reg1[0]      = interrupt_out;
slv_reg1[1]      = timer_enable;
slv_reg1[31:2]   = 30'h0;                  // Use these register bits for
                                           // observing signals
```

The state assignment for a timer counter will be:

```
parameter     RESET  = 3'b111;
parameter     COUNT  = 3'b010;
parameter     WAIT   = 3'b011;
parameter     IDLE   = 3'b100;
```

Reading register port `slv_reg2[31:0]` reads the following:

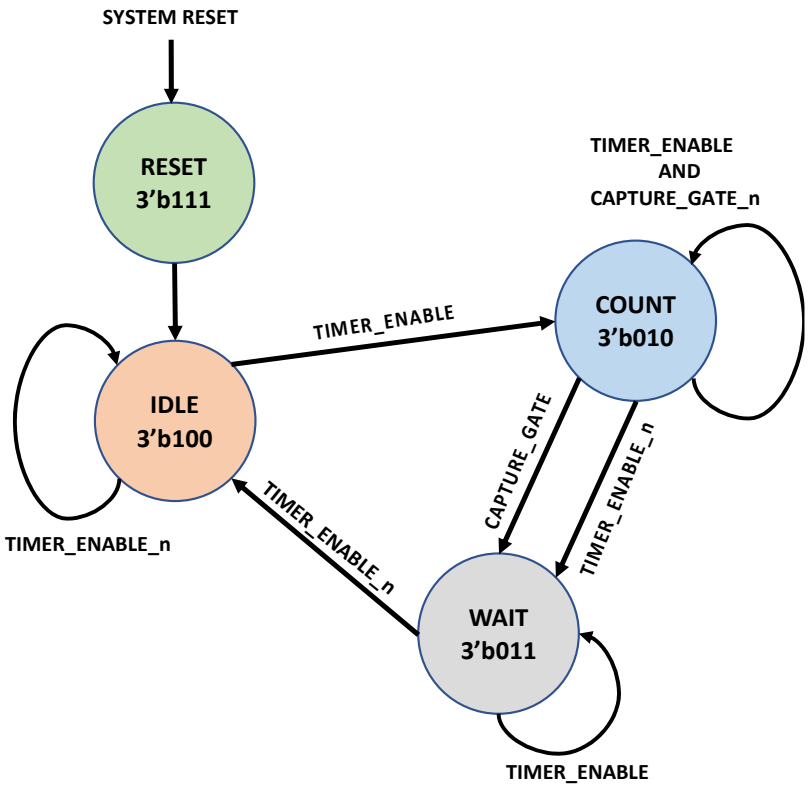
```
slv_reg3[2:0]    = state[2:0];             // This is the current state of
                                           // the timer counter state machine.

slv_reg3[31:3]   = TBD;                    // Use these register bits for
                                           // observing signals
```

Reading register port `slv_reg3[31:0]` reads the capture timer counter value.

```
slv_reg2[31:0] = Cap_Timer_Out[31:0];
```

The state diagram for the state machine would look like this:



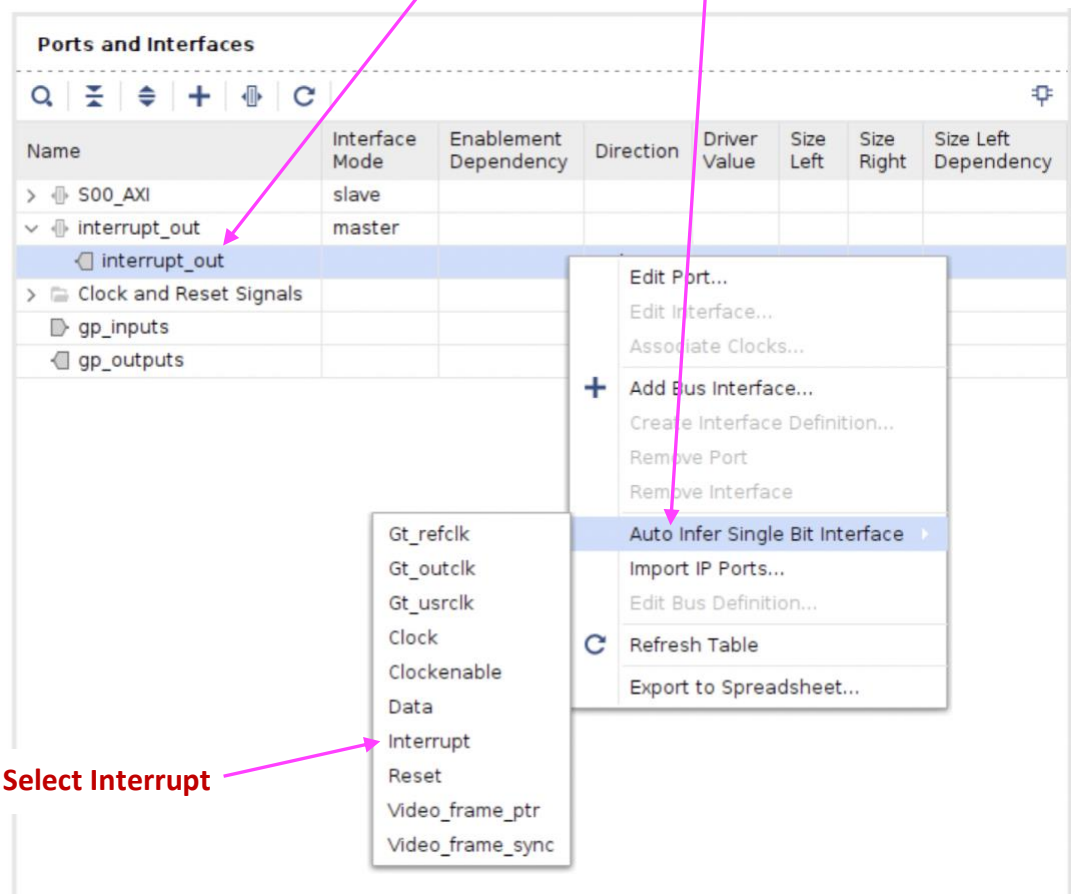
This timer is used to measure transaction times on the AXI-4 bus between the BRAM and OCM. It is also used to measure the latency of the ISRs. When the timer is enabled, it starts counting immediately. It continues to count until CAPTURE_GATE_n or TIMER_ENABLE are negated. The state machine stays in the WAIT state until the timer is disabled.

NOTE: The **interrupt_out** pin must be configured as an interrupt pin during editing in the IP Packager so that the DTB assigns an interrupt number to the pin.

Edit the **CAPTURE_TIMER** module in the IP Packager.

Go to *Ports and Interfaces* and select **interrupt_out** pin.

Using the right mouse key select the *Auto Infer Single Bit Interface*

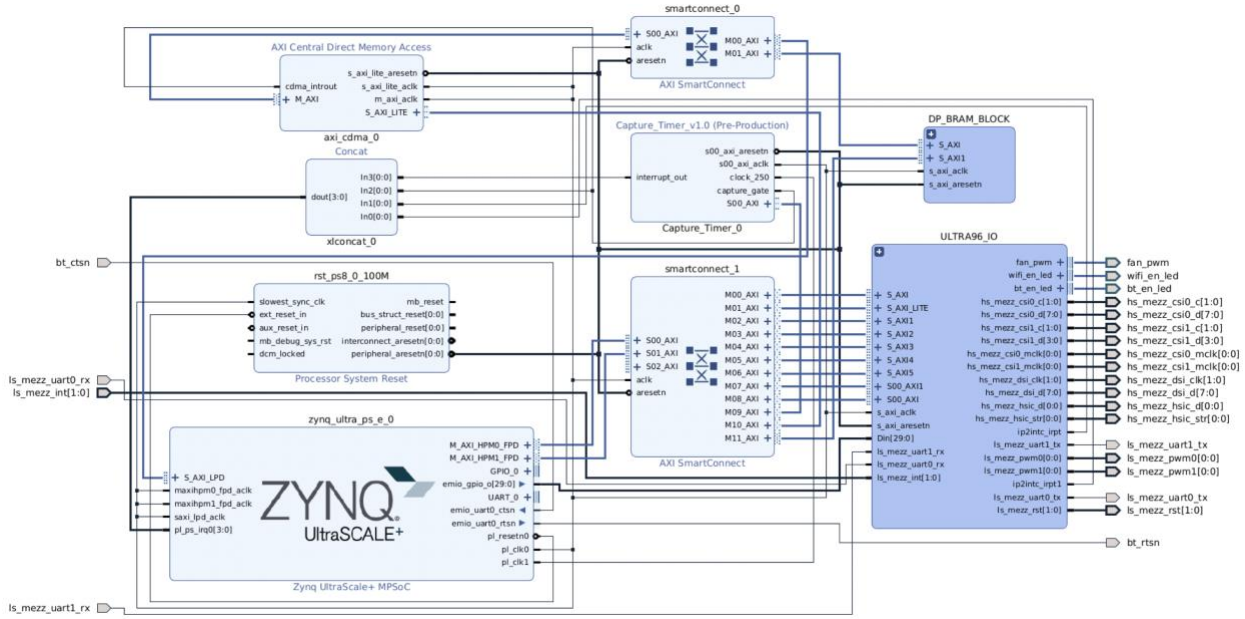


Select Interrupt

Schematic design procedure:

Referring to the block diagram below:

- Use the Vivado schematic from Lab 1. Remove the ***BURST_MASTER*** module and insert the ***AXI CDMA*** unit from the Xilinx Library
- Insert the ***CAPTURE_TIMER*** unit.
- Hook up the two new blocks.
- Modify the ***CONCAT*** module to add two additional interrupt input pins. Connect the **interrupt_out** signal from ***CAPTURE_TIMER*** module to the ***CONCAT*** module. Connect the **cdma_introut** signal from the **AXI CDMA** unit to the ***CONCAT*** module and to the Capture Gate signal on the ***CAPTURE_TIMER.***
-



The next step is to generate a bit file that can be dynamically downloaded into the FPGA. Refer to this guide on how to generate the FPGA bit file: [BIT File Generation Flow]
The PL will be synthesized at 300 MHz. The PS will be configured for 1500 MHz.

The updated Address Map must look like the table below to use the system.dtb that will be downloaded for this lab. Any changes will cause access timeout failures.

Name	Interface	Slave Segment	Master Base Address	Range	Master High Address
axi_cdma_0					
/zynq_ultra_ps_e_0/SAXIGP6	S_AXI_LPD	LPD_DDR_LOW	0x00_0000_0000	2G	0x00_7FFF_FFFF
/DP_BRAM_BLOCK/axi_bram_ctrl_CDMA/S_AXI	S_AXI	Mem0	0x00_C000_0000	8K	0x00_C000_1FFF
/zynq_ultra_ps_e_0/SAXIGP6	S_AXI_LPD	LPD_LPS_OCM	0x00_FF00_0000	16M	0x00_FFFF_FFFF
zynq_ultra_ps_e_0					
/ULTRA96_IO/Low_Speed_MEZZ/axi_uart16550_0/S_AXI	S_AXI	Reg	0x00_A000_0000	64K	0x00_A000_FFFF
/ULTRA96_IO/Low_Speed_MEZZ/axi_uart16550_1/S_AXI	S_AXI	Reg	0x00_A001_0000	64K	0x00_A001_FFFF
/ULTRA96_IO/BD_CTL_GPIO/axi_gpio_0/S_AXI	S_AXI	Reg	0x00_A002_0000	4K	0x00_A002_0FFF
/ULTRA96_IO/BD_CTL_GPIO/axi_gpio_1/S_AXI	S_AXI	Reg	0x00_A002_1000	4K	0x00_A002_1FFF
/ULTRA96_IO/Low_Speed_MEZZ/axi_gpio_2/S_AXI	S_AXI	Reg	0x00_A002_2000	4K	0x00_A002_2FFF
/ULTRA96_IO/SYS_MGMT/axi_gpio_3/S_AXI	S_AXI	Reg	0x00_A002_5000	4K	0x00_A002_5FFF
/ULTRA96_IO/SYS_MGMT/system_management_wiz_0/S_AXI_LITE	S_AXI_LITE	Reg	0x00_A002_6000	8K	0x00_A002_7FFF
/DP_BRAM_BLOCK/axi_bram_ctrl_PS/S_AXI	S_AXI	Mem0	0x00_A002_8000	8K	0x00_A002_9FFF
/ULTRA96_IO/Low_Speed_MEZZ/PWM_w_Int_0/S00_AXI	S00_AXI	S00_AXI_reg	0x00_A003_0000	64K	0x00_A003_FFFF
/ULTRA96_IO/Low_Speed_MEZZ/PWM_w_Int_1/S00_AXI	S00_AXI	S00_AXI_reg	0x00_A004_0000	64K	0x00_A004_FFFF
/ULTRA96_IO/Low_Speed_MEZZ/PWM_w_Int_1/S00_AXI	S00_AXI	S00_AXI_reg	0x00_A004_0000	64K	0x00_A004_FFFF
/axi_cdma_0/S_AXI_LITE	S_AXI_LITE	Reg	0x00_B000_0000	4K	0x00_B000_0FFF
/Capture_Timer_0/S00_AXI	S00_AXI	S00_AXI_reg	0x00_B000_2000	4K	0x00_B000_2FFF

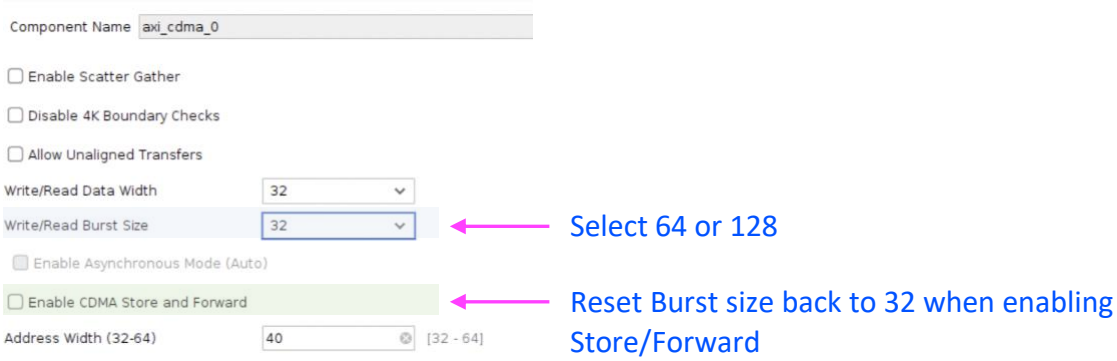
The system.dtb for this lab is located here:
https://projects.ece.utexas.edu/courses/spring_24/ee382n4-17365/arch/labs/SP24_LAB2/system.dtb

Lab procedure:

In this Lab you will need to write two different test routines each comprising of the application code and the corresponding kernel module. The capture timer block will be used to measure DMA transfer time and interrupt latency time using a counter clocked by the 250MHz clock as shown above. The second test will measure the interrupt latency.

TEST #1: Refer to CDMA Transfer Measurement Procedure outlined above for this test. The memory will be tested using the following sequence:

1. Fill a memory page (i.e., 4K bytes) in the OCM with a 1024 random 32-bit data values using the Linux `srand(time(0))` and `rand()` routines. Use 0xFFFC_0000 as the starting address of this page.
2. Transfer the random data in the OCM (0xFFFC_0000) to the BRAM (0xC000_0000) using the CDMA unit.
3. Measure the number of cycles it took to do the CDMA transfer using the Capture-Timer unit as described above.
4. Transfer the random data in the BRAM (0xC000_0000) back to the OCM at a different address (0xFFFC_2000) using the CDMA unit.
5. Measure the number of cycles it took to do CDMA transfer using the Capture-Timer unit as described above.
6. Once the steps above are complete, the OCM data at address 0xFFFC_0000 is compared to the OCM data at address 0xFFFC_2000. This confirms that DMA traffic to/from the OCM & BRAM works.
7. Change the highlighted features below in the CDMA and rerun the above tests. The features are changed using the “Customize Block” command in Vivado. Capture latency data for the different modes and generate a report that explains what each capability does to the transaction times.



TEST #2: Refer to the Interrupt Latency Measurement Procedure outlined above to measure interrupt latency in the kernel.

1. Measure the interrupt latency for a statically large number of interrupts using all 9 frequency combinations.
2. Generate a plot like what is shown below.

CDMA Data Transfer Measurement Procedure:

Use the DMA c-code that is in the /root/code/ULTRA directory on the ULTRA96 as starter code. You will need to generate the application code and the kernel module as shown in class.

The ***CAPTURE_TIMER*** must be configured in capture mode. The timer is enabled when the DMA transfer starts. The timer is halted when the `cdma_introut` signal from the **axi_cdma** module is asserted. The value in the `cap_timer_out[31:0]` register contains the number of `250 MHz` clock cycles that DMA transfer took. The `cdma_introut` signal is connected to the GIC interrupt pin on the PS. The interrupt handler will acknowledge the interrupt and set the `det_int` flag for the test program.

The test program will report the following information:

```
Test status  ---  # loops and # words transferred
Minimum Latency
Maximum Latency
Average Latency
Standard Deviation
```

Here is an example output of the CDMA transfer latency showing debug information

```
************************************************************
Memory test passed --- 500 loops and 1000 words
Minimum DMA Latency: 2689
Maximum DMA Latency: 9067
Average DMA Latency: 3166.000000
Standard Deviation: 372.000000

Number of samples: 500
Number of interrupts detected = 500
Interrupt #53: 9400000 GICv2 125 Edge cdma-controller
************************************************************
```

```
Memory test passed --- 500 loops and 1000 words
Minimum DMA Latency: 2653
Maximum DMA Latency: 6765
Average DMA Latency: 3224.000000
Standard Deviation: 433.000000
Number of samples: 500
Number of interrupts detected = 500
Interrupt 53: 9400500 GICv2 125 Edge cdma-controller
************************************************************
```

These values are for the kernel when it is not busy doing other tasks. You will need to open additional terminal windows and start other tasks to see what the impact is on the "Maximum Latency". An interesting task is to do a continuous recursive directory listing of the flash drive to stress the OS and the AMBA bus:

```
while (true) do ls -algR; done
```

The maximum latency and standard deviation will both increase:

```
************************************************************
Memory test passed --- 500 loops and 1000 words
Minimum DMA Latency: 2663
Maximum DMA Latency: 170834
Average DMA Latency: 4496.000000
Standard Deviation: 8762.000000

Number of samples: 500
Number of interrupts detected = 500
53: 9420000 GICv2 125 Edge cdma-controller
************************************************************
Memory test passed --- 500 loops and 1000 words
Minimum DMA Latency: 2578
Maximum DMA Latency: 1312215
Average DMA Latency: 7919.000000
Standard Deviation: 61545.000000

Number of samples: 500
Number of interrupts detected = 500
53: 9420500 GICv2 125 Edge cdma-controller
************************************************************
Note that the maximum latency has increased 194 times.
```

Interrupt Latency Measurement Procedure.

Generate new application code and the kernel module for this test. Merging it with the CDMA code will create interesting debug problems.

The steps to follow for the Interrupt Latency Measurement test are:

1) Assert the "`interrupt_out`" pin (`slv_reg1[0]`) on Capture-Timer module. This pin is connected to the GIC interrupt input on the PS.
2) At the same time assert the `slv_reg1[1]` pin. This will start the timer counter.
3) Wait for interrupt to be detected and handled in the kernel. Read the timer counter value.
4) Negate the interrupt and disable the timer counter.

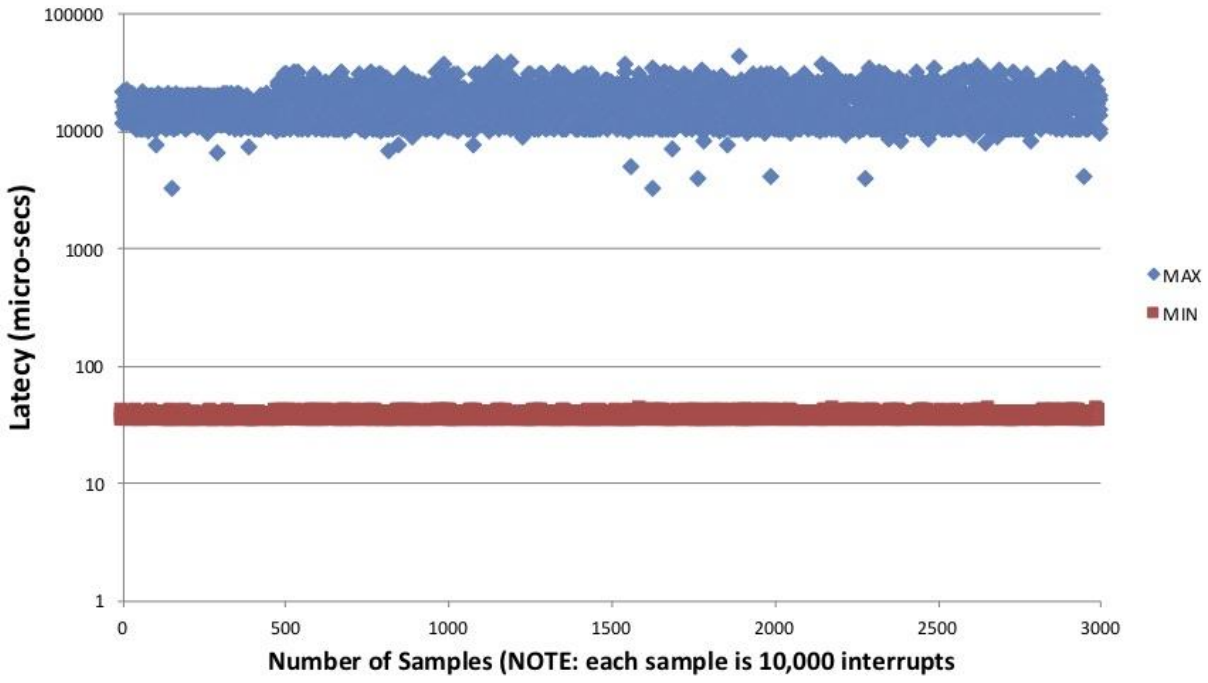
5) The value in the timer is read by the application program and the following data needs to be displayed:

```
Minimum Latency
Maximum Latency
Average Latency
Standard Deviation
Number of Samples
Number of Interrupts registered in the Kernel using /proc/interrupts
```

6) Here is an example output:

```
**************************************************************
Minimum Latency: 17
Maximum Latency: 2893
Average Latency: 25.540000
Standard Deviation: 49.710000
Number of samples: 10000
70: 430000
**************************************************************
Minimum Latency: 17
Maximum Latency: 1071
Average Latency: 27.470000
Standard Deviation: 11.150000
Number of samples: 10000
70: 440000
**************************************************************
```

7) The number of samples per test needs to be programmable. The maximum number of samples is approximately 10,000. Here is what you should see if you ran around 30 million interrupts:


100000
10000
1000
100
10
1
Latecy (micro-secs)
0
500
1000
1500
2000
2500
3000
Number of Samples (NOTE: each sample is 10,000 interrupts
MAX
MIN