The Programmable Interval Timer

Luke Hsiao & Jeff Ravert

Brigham Young University

6 November 2014

Register Descriptions

The Programmable Interval Timer (PIT) contains two software accessible registers: the Control Register and the Delay Value Register. Both of these registers are 32-bits, and can be accessed using our Driver API. The Control Register is used to programmably change the mode of operation of the PIT. The Delay Value Register contains the programmable value that the PIT should count down from, allowing the time between interrupts to be changed during runtime.

The software is able to access these registers by addressing an offset of the PIT's base address. In this implementation of the hardware, the base address and high address of the PIT are defined as:

```
#define XPAR_PIT_TIMER_0_BASEADDR 0x70E00000
#define XPAR_PIT_TIMER_0_HIGHADDR 0x70E0FFFF
```

Table 1 shows the offset that must be added to the base address of the PIT to access the register. For convenience, and API has been created and described below that provides a simple macro for writing to these registers.

Register Name	Offset from Base Address	Function
Control Register	0x4	Writing to the bottom three bits of this register can change the mode of operation of the PIT.
Delay Value Register	0x0	Write a 32-bit number that will be interpreted as an unsigned integer to this register to set the time to wait between interrupts.

Table 1: General Register Descriptions

Control Register

The PIT's behavior is controlled by writing values to software accessible registers. The 32-bit delay-value register contains the value to be loaded into the PIT counter. The 32-bit Control Register determines which mode of operation the PIT will function in. There are three control bits located as the three least significant bits of the Control Register as shown below in Figure 1. All of the other bits (31 through 4) are ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 1: 32-bit Control Register Layout

These control bits affect the mode of operation of the PIT as shown in Table 2.

Bit Number	Name	Function
0	Counter Enable	'1' = Allow the counter to decrement '0' = Hold the counter at it's current value
1	Interrupt Enable	'1' = Send a 1 clock cycle interrupt signal when the counter reaches zero '0' = Disable Interrupts from being sent
2	Reload Enable	'1' = Reload the counter with the delay-value when it reaches zero '0' = Do not reload the counter after it reaches zero
3	Force Load	'1' = While asserted, load the value of the Delay Register into the counter. '0' = Deassert for normal operation.

Table 2: Descriptions of Control Bits

Delay Value Register

The Delay Value Register simply provides the value that is loaded into the PIT counter when it is reloaded. This 32-bit value is read and interpreted as a 32-bit unsigned integer. Thus, the maximum value that can be loaded is 4,294,967,295. This allows the user application to change the amount of time that passes between interrupts during runtime. Note that when the PIT is reset, a default value of 0xFFFFFFFF is loaded into the counter register.

Timing Diagrams

The waveforms below illustrate several examples of the behavior of the PIT given the control signals discussed above.

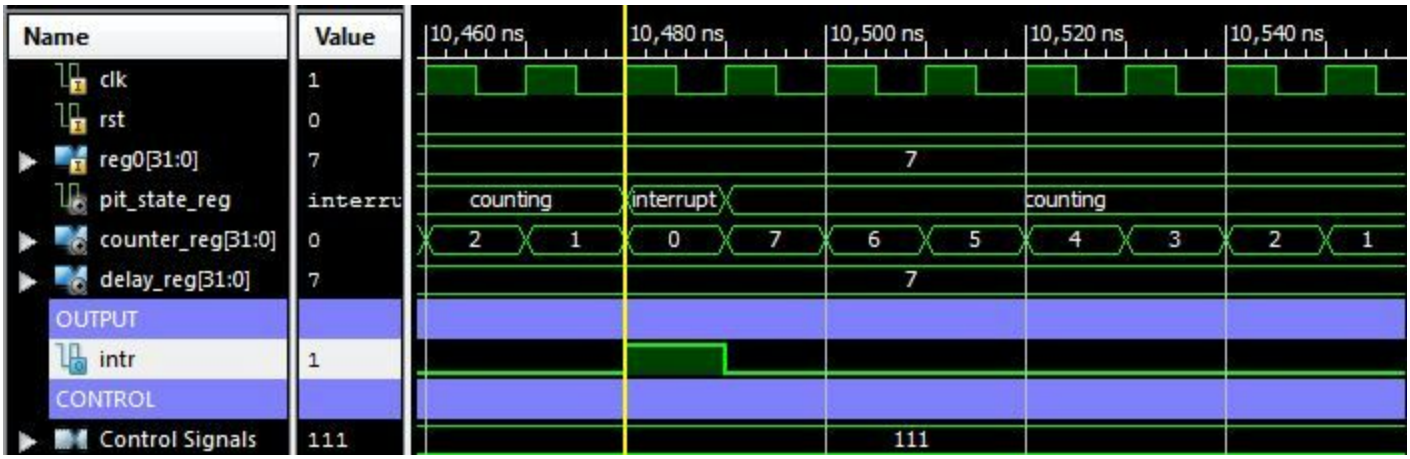

Figure 2: Normal Countdown Operation

Figure 2 shows the timing diagram of the PIT if the Delay Value Register contained a value of 7, and the control signals were all set HIGH. This is the normal countdown operation of the PIT and gives a one clock cycle interrupt signal each time the counter hits zero. This is the behavior that matches most closely with the Xilinx Fixed Interval Timer (FIT) behavior.

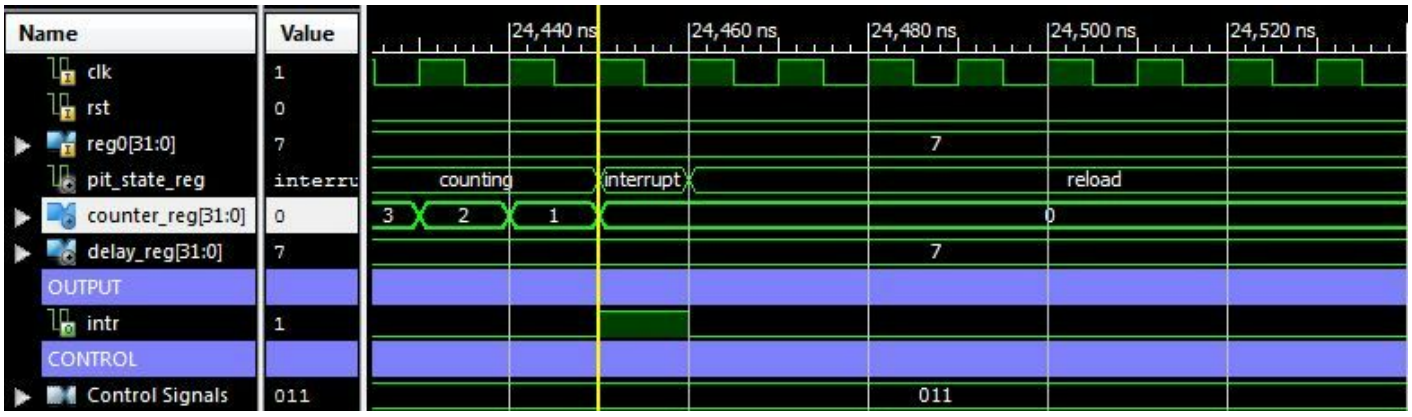

Figure 3: Reload Disabled Operation

Figure 3 shows the timing diagram when control bit 2, the reload enable signal, is set LOW. As illustrated, once the timer counts down to zero, it remains zero indefinitely until the reload enable is set HIGH.

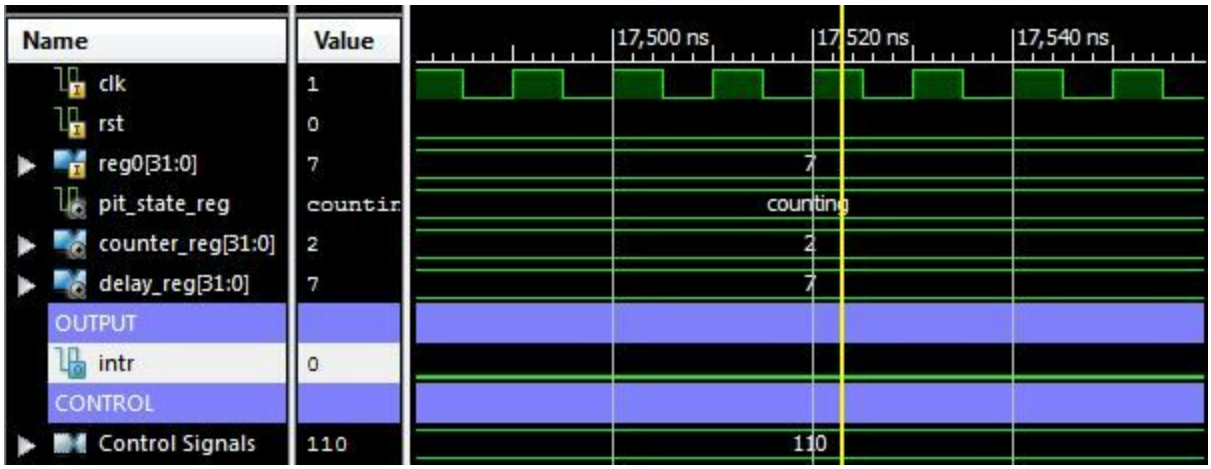

Figure 4: Count Disabled Operation

Figure 4 shows the behavior of the PIT when the Count Enable control signal (bit 0) is set LOW. As shown, the counter no longer decrements, and holds the value that was present in the register when the Count Enable signal was set LOW.

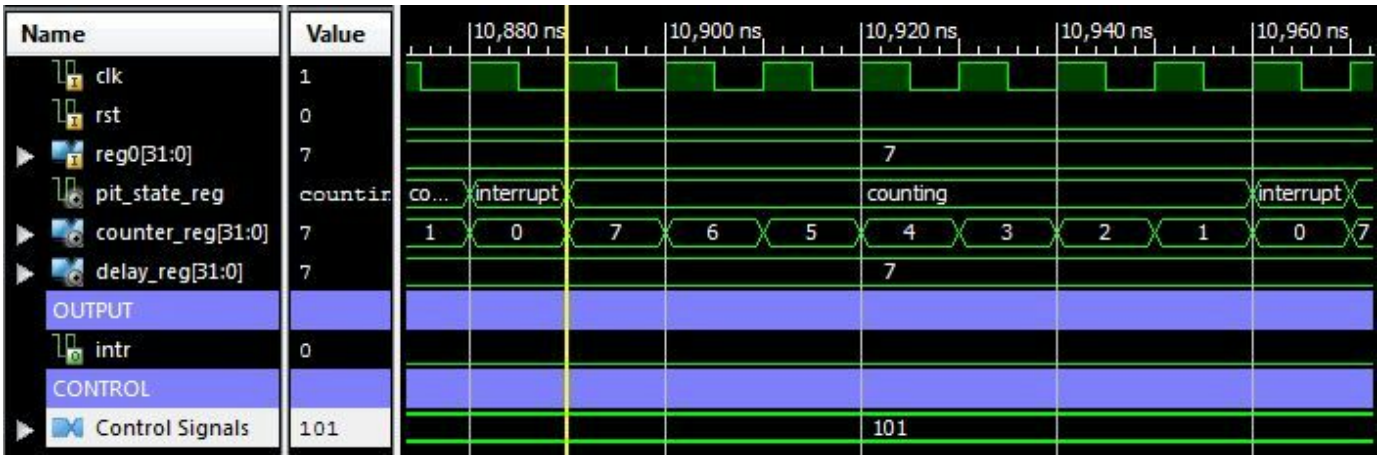

Figure 5: Interrupts Disabled Operation

Figure 5 shows that when the Interrupt Enable signal (bit 1) is set LOW, the counter's interrupt output is not asserted, even when the counter reaches zero.

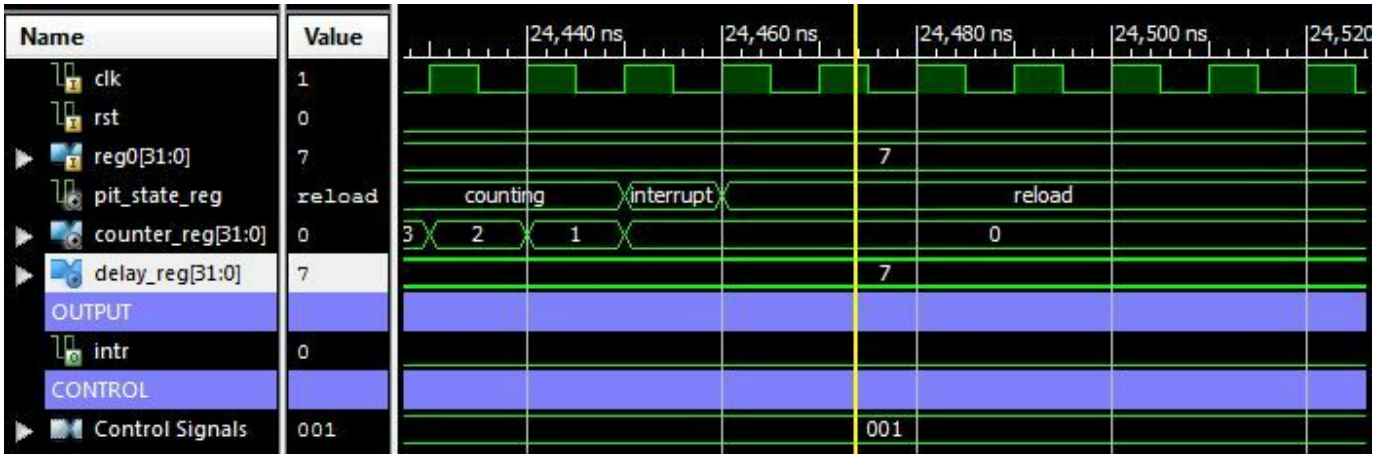

Figure 6: Interrupts Disabled and No Reload Operation

These control signals can be used in any combination. For example, Figure 6 shows the behavior when both Interrupts and Reload are disabled. As shown, no interrupt output is asserted, and the counter's value is never reloaded.

Driver API

The Application Programming Interface defined to interact with the Programmable Interval Timer is outlined below. Although Xilinx's XPS generated additional macros, these are the only ones important to the operation of the PIT. These macro definitions can be found in pit_timer.h.

PIT_TIMER_SET_DELAY(delayValue)

This macro writes the 32-bit 'delayValue' to the PIT's Delay Value Register. The 32-bit value in this register will be interpreted as an unsigned 32-bit integer, and loaded into the PIT's counter. Loading a larger number lengthens the time between interrupts. Loading a smaller number decreases the time between interrupts.

PIT_TIMER_READ_DELAY

This macro returns the value currently in the Delay Value Register as type 'Xuint32'. This can be used to verify that the value written to the Delay Value Register is correct.

PIT_TIMER_WRITE_CONTROL(val)

This macro writes the 32-bit value 'val' to the PIT's Control Register. Bits 31 through 4 are ignored. Bits 4 through 0 can change the mode of operation of the PIT as outlined in the 'Control Register' section above.

PIT_TIMER_READ_CONTROL

This macro returns the value currently in the Control Register as type 'Xuint32'. This can be used to verify that the value written to the Control Register is correct.

PIT_TIMER_RESET

This macro resets the PIT via software. When reset, the counter register in the PIT contains the value 0xFFFFFFFF and does not decrement. The delay-value register also is set to contain zero, and interrupts are disabled.

An example Initialization and use of the API in application code is shown below:

```
#define COUNTER_ENABLE       0x00000001
#define INTR_ENABLE          0x00000002
#define RELOAD_ENABLE 0x00000004
#define FORCE_LOAD           0x00000008

...
// Unmask the PIT interrupts with the Interrupt Controller
...

// Initialize the PIT
PIT_TIMER_RESET;
PIT_TIMER_SET_DELAY(0xFFFF);
PIT_TIMER_WRITE_CONTROL(FORCE_LOAD);        // Force Load the Delay Value

// Run the Timer with interrupts, counter, and reload enabled
PIT_TIMER_WRITE_CONTROL(INTR_ENABLE | COUNTER_ENABLE | RELOAD_ENABLE);
```

If using an interrupt controller, the application programmer will need to ensure that the PIT is connected to the interrupt controller properly, and that the PITs interrupts are enabled with the controller.

Bug Report

BUG	SYMPTOM	SOLUTION
Clock Cycle Delay on Reload	Because we used a hardware state machine to implement our PIT, we initially noticed that it would take one extra clock cycle to reload the value because our state machine needed to transition to an extra state.	We were able to maintain three states and a Moore interrupt output by implementing logic that could, in essence, look ahead and see if the value would need to be reloaded, and perform the reload one clock cycle early. This fix allowed us to get the ideal behavior shown in the timing diagrams.
Xilinx Tools Workspace Issues	After exporting to the SDK the xgpio.h wasn't found. The push buttons were not in the XPS	We rebuilt our XPS project from the base project provided in Lab 4, then added our user peripheral again. This ensured that no other people's settings or configurations messed up our hardware.