Spring 2018

EE 382N-4: Advanced Micro-Controller Systems

Lab Assignment #3

Due April 17th, 2018

Overview:

This lab combines Labs #1 and #2 together while dynamically modulating the clock frequency of the Processor System (PS). The expected outcome is that the memory test and the interrupt latency measure test does not fail while the PLL and the Clock Divider control logic is dynamically (and randomly) modified. Refer to the Zynq Technical Reference Manual for clock control details.

Schematic Generation Procedure:

1) Use the Vivado schematic from Lab2.

Clock Control Procedures

There are two clock control modes which must be exercised in this lab: 1) PLL mode where the feedback divider value is modified and 2) where the 6-bit clock divider value is modified.

Fig 1. below shows a basic PLL implementation. On the Zed Board, the input frequency is 33MHz. The frequency divider is set to 40 which results in an output frequency of 1.32 GHz. The clock divider circuit is set to 2 which results in a CPU frequency of 666 MHz.

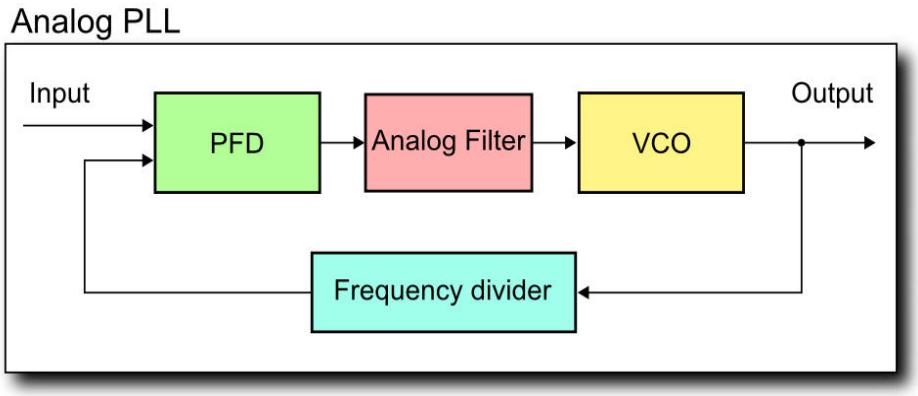

Figure 1. Basic PLL Block Diagram

The frequency divider is controlled by the ARM_PLL_CTRL register:

Absolute Address	0xF8000100
Width	32 bits
Access Type	rw
Reset Value	0x0001A008
Description	ARM PLL Control

Register ARM_PLL_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:19	rw	0x0	Reserved. Writes are ignored, read data is zero.
PLL_FDIV	18:12	rw	0x1A	Provide the feedback divisor for the PLL. Note: Before changing this value, the PLL must first be bypassed and then put into reset mode. Refer to the Zynq-7000 TRM, UG585, Clocks chapter for CP/RES/CNT values for the PLL.
reserved	11:5	rw	0x0	Reserved. Writes are ignored, read data is zero.
PLL_BYPASS_FORCE	4	rw	0x0	ARM PLL Bypass override control: PLL_BYPASS_QUAL = 0: 0: enabled, not bypassed. 1: bypassed. PLL_BYPASS_QUAL = 1 (QUAL bit default value): 0: PLL mode is set based on pin strap setting. 1: PLL bypassed regardless of the pin strapping.
PLL_BYPASS_QUAL	3	rw	0x1	Select the source for the ARM PLL Bypass Control: 0: controlled by the PLL_BYPASS_FORCE bit, bit 4. 1: controlled by the value of the sampled BOOT_MODE pin strapping resistor PLL_BYPASS. This can be read using the BOOT_MODE[4] bit.
reserved	2	rw	0x0	Reserved. Writes are ignored, read data is zero.
PLL_PWRDWN	1	rw	0x0	PLL Power-down control: 0: PLL powered up 1: PLL powered down
PLL_RESET	0	rw	0x0	PLL reset control: 0: de-assert (PLL operating) 1: assert (PLL held in reset)

The software sequence to update the feedback divider values is:

1. Program the **ARM_PLL CTRL[PLL_FDIV]** value and the PLL configuration register, **ARM_PLL_CFG[LOCK_CNT, PLL_CP, PLL_RESET]**, to their required values
2. Force the PLL into bypass mode by writing a 1 to **ARM_PLL_CTRL**

[PLL_BYPASS_FORCE<4>] and setting the **ARM_PLL_CTRL [PLL_BYPASS_QUAL<3>]** bit to a 0

3. Assert and de-assert the PLL reset by writing a 1 and then a 0 to **ARM_PLL_CTRL [PLL_RESET<0>]**
4. Verify that the PLL is locked by reading **PLL_STATUS [ARM_PLL_LOCK<3>]**
5. Disable the PLL bypass mode by writing a 0 to **ARM_PLL_CTRL [PLL_BYPASS_FORCE<4>]**

Figure 2. below shows the block diagram of the CPU generation logic.

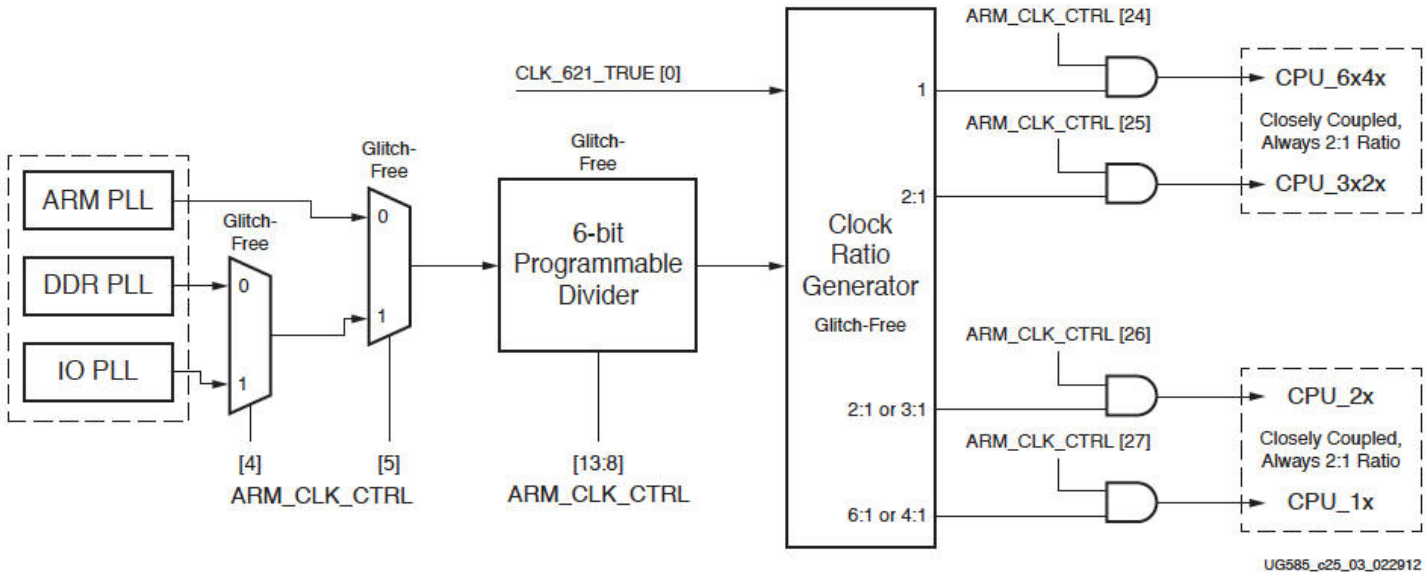

Figure 2.: CPU Clock Control Block Diagram

The Programmable Divider logic is controlled by the ARM_CLK_CTRL register:

Name	ARM_CLK_CTRL
Relative Address	0x00000120
Absolute Address	0xF8000120
Width	32 bits
Access Type	rw
Reset Value	0x1F000400
Description	CPU Clock Control

Register ARM_CLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:29	rw	0x0	Reserved. Writes are ignored, read data is zero.
CPU_PERI_CLKACT	28	rw	0x1	Clock active: 0: Clock is disabled 1: Clock is enabled

CPU_1XCLKACT	27	rw	0x1	CPU_1x Clock control: 0: disable, 1: enable
CPU_2XCLKACT	26	rw	0x1	CPU_2x Clock control: 0: disable, 1: enable
CPU_3OR2XCLKACT	25	rw	0x1	CPU_3x2x Clock control: 0: disable, 1: enable
CPU_6OR4XCLKACT	24	rw	0x1	CPU_6x4x Clock control: 0: disable, 1: enable
reserved	23:14	rw	0x0	Reserved. Writes are ignored, read data is zero.
DIVISOR	13:8	rw	0x4	Frequency divisor for the CPU clock source. (When PLL is being used, 1&3 are illegal values)
reserved	7:6	rw	0x0	Reserved. Writes are ignored, read data is zero.
SRCSEL	5:4	rw	0x0	Select the source used to generate the CPU clock: 0x: ARM PLL 10: DDR PLL 11: IO PLL This field is reset by POR only.
reserved	3:0	rw	0x0	Reserved. Writes are ignored, read data is zero.

NOTE: The Programmable Divider can be dynamically changed with any special SW sequencing.

Implementation Details

1) Combine the SW components from Labs #1 and #2. You should not have to generate a new .bit file for this lab depending on how you implemented Lab #2.
2) The combined code should run the BRAM memory test and the interrupt latency test at the same time.
3) Once the combined code runs both tests, implement the frequency modulation scheme. The matrix of tests to run are:

Test #	PLL Divider	Clock divider	Notes
1	40	2	666 MHz
2	44	2	726 MHz --- A little over clocking :-)
3	20	12	55 MHz
4	48	3	528 MHz
5	2	2	33 MHz
6	34	20	56.1 MHz

7	1	2	16.66 MHz
8	48	2	792 MHz --- Pushing it for sure!!

All of the tests should run for random amounts of time in non-stop mode (unless test #8 fails and then you should scale it back to where the test will run). Use the LFSR to generate the random time values. You will notice that the latency numbers remain the same regardless of the frequency. Speculate why this is the case?

Deliverables:

1) Download your bit file to the Zed Board and demonstrate your code running on the Zed Board.

2) Upload your C-code and Vivado directories to Canvas.

3) Write a README summarizing the design and testing process

Tutorials:

Zynq Technical Reference Manual
An FPGA Tutorial using the ZedBoard
Zynq Design from Scratch
Zynq Development
Free Electrons
Zynq Training

Environmental Details:

1) Vivado is available on the Linux machines. Refer to this web page ECE-Linux-Machines for details on accessing the Linux machines remotely. NOTE: VNC is no longer supported.

2) All software will be compiled on the ZedBoard using the GNU tool suite. **Do not use the Vivado SW development tools. These are for bare-metal implementations**.