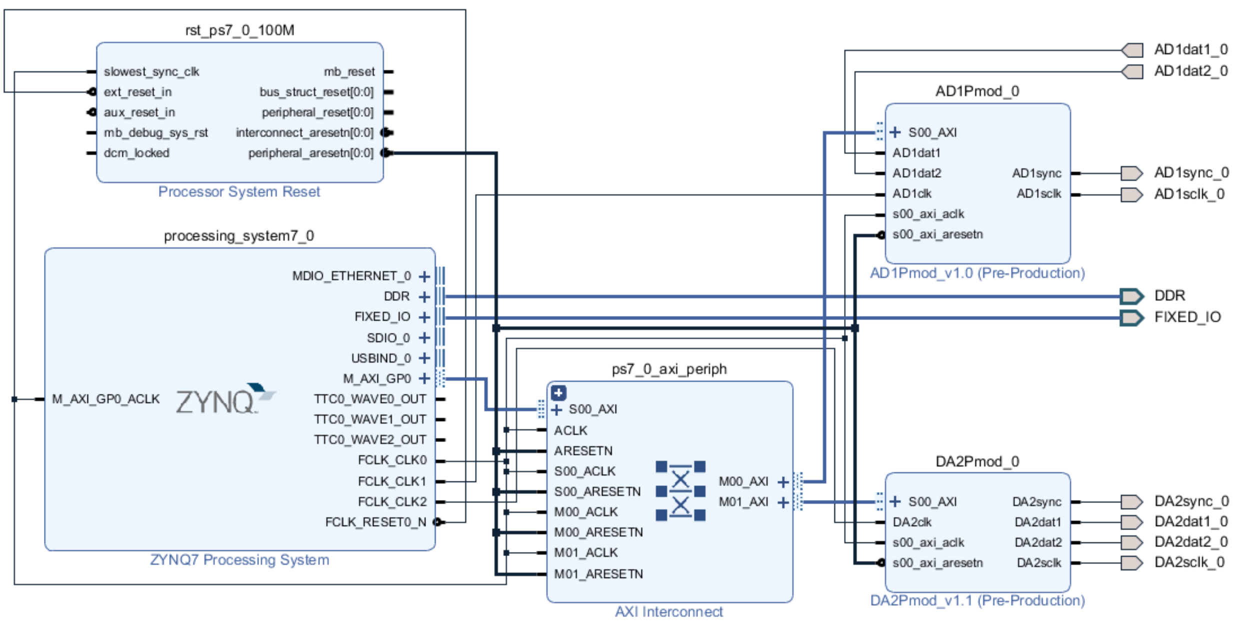
ECE3623 Embedded System Design Laboratory

Dennis Silage, PhD
silage@temple.edu

AD1Pmod and DA2Pmod in FreeRTOS v2

In this Lab 7 you will investigate utilizing a *queue* to send data between tasks in the FreeRTOS environment on the Zybo board. A Vivado hardware project that uses the PmodAD1 ADC and the PmodDA2 DAC is shown below. The *Zynq SPI Peripherals ADC and DAC* PPT posted on Canvas is the reference for the PmodAD1 ADC and PmodDA2 DAC. The PmodAD1 and PmodDA2 Vivado SDK example project *main.c* is shown below and is also presented in the lecture.



These AD1Pmod and the DA2Pmod IPs are not that provided by Digilent but use the controller-data path formulation for increased performance in their data throughput. The PmodAD1 ADC uses the FCLK_CLK1 external clock signal at nominally 30 MHz (actually 30.303 MHz) for a SPI clock frequency of approximately 15 MHz where the maximum is 20 MHz. The PmodDA2 DAC uses the FCLK_CLK2 external clock signal at 50 MHz for a SPI clock frequency of 25 MHz where the maximum is 30 MHz.

PL Fabric Clocks				
☑ FCLK_CLK0	IO PLL	100	100.000000	0.100000 : 250.000000
☑ FCLK_CLK1	IO PLL	30	30.303030	0.100000 : 250.000000
☑ FCLK_CLK2	IO PLL	50	50.000000	0.100000 : 250.000000
☐ FCLK_CLK3	IO PLL	50	10.000000	0.100000 : 250.000000

The constraint file *AD1DA2JE.xdc* on Canvas uses the JE Pmod connector on the Zybo board with the PmodAD1 on the upper 6 pins and the PmodDA2 on the lower 6 pins.

Diagram | Address Editor | **AD1DA2JE.xdc**

C:/zynq_Pmod_ED/AD1DA2PmodEx/AD1DA2PmodEx.srcs/constrs_1/imports/zy

```
#set_property PACKAGE_PIN V18 [get_ports {JD10}]
#set_property IOSTANDARD LVCMOS33 [get_ports {JD10}]

#Pmod Header JE
set_property PACKAGE_PIN V12 [get_ports {AD1sync_0}]
set_property IOSTANDARD LVCMOS33 [get_ports {AD1sync_0}]
set_property PACKAGE_PIN W16 [get_ports {AD1dat1_0}]
set_property IOSTANDARD LVCMOS33 [get_ports {AD1dat1_0}]
set_property PACKAGE_PIN J15 [get_ports {AD1dat2_0}]
set_property IOSTANDARD LVCMOS33 [get_ports {AD1dat2_0}]
set_property PACKAGE_PIN H15 [get_ports {AD1sclk_0}]
set_property IOSTANDARD LVCMOS33 [get_ports {AD1sclk_0}]
set_property PACKAGE_PIN V13 [get_ports {DA2sync_0}]
set_property IOSTANDARD LVCMOS33 [get_ports {DA2sync_0}]
set_property PACKAGE_PIN U17 [get_ports {DA2dat1_0}]
set_property IOSTANDARD LVCMOS33 [get_ports {DA2dat1_0}]
set_property PACKAGE_PIN T17 [get_ports {DA2dat2_0}]
set_property IOSTANDARD LVCMOS33 [get_ports {DA2dat2_0}]
set_property PACKAGE_PIN Y17 [get_ports {DA2sclk_0}]
set_property IOSTANDARD LVCMOS33 [get_ports {DA2sclk_0}]

#USB-OTG overcurrent detect pin
#set_property PACKAGE_PIN U13 [get_ports otg_oc]
#set_property IOSTANDARD LVCMOS33 [get_ports otg_oc]
```

The *main.c* program in SDK is given below. This is a single task, standalone operating system application and not FreeRTOS. The application is a *straight-through* ADC to DAC system using the control and status signals for the ADC and DAC. The Address Editor for Vivado provides that start address for the PmodAD1 and PmodDA2 IP blocks.

Diagram | **Address Editor**

Cell	Slave Interface	Base Name	Offset Address	Range	High Address
processing_system7_0					
Data (32 address bits : 0x40000000 [1G])					
AD1Pmod_0	S00_AXI	S00_AXI_reg	0x43C0_0000	64K	0x43C0_FFFF
DA2Pmod_0	S00_AXI	S00_AXI_reg	0x43C1_0000	64K	0x43C1_FFFF

The ADC input and DAC output signals are shown in the Digilent *Waveforms* display. The *straight-through* ADC to DAC application has a nominal gain of 1 but there is a difference in the fixed offsets between the ADC and DAC. You should verify the performance of the template program. This is a two channel application but the ADC and DAC SPI interface can convert two channels simultaneously and thus provide no additional burden if two channels are required. The tasks for this Lab 7 are as follows:

1. Describe in detail the single task, standalone operating system template application. What are the various control and status signals? Using the Waveforms oscilloscope,

measure the sampled data throughput rate beginning at an initial ADC data conversion (ADC SS high to active low) to the beginning of the next ADC data conversion (ADC SS high to active low).

Show the measurements and report the result as samples/second *fs*. Measure the time for each of the components of this data acquisition sequence. That is, the measured time for ADC conversion (ADC SS active low to high), DAC conversion (DAC SS active low to high) and the remaining time as overhead of the single task application.

2. The second Lab 7 task is to configure this single task, standalone operating system application as two tasks, *AD1task* and *DA2task*, within FreeRTOS. You are to use a *queue* function to transfer data for *straight-through* operation between the two tasks in sequence AD1task > DACtask. The *AD1task* and *DA2task* have equal priority and the single queue size is large enough at 10 elements so that the tasks are not blocked. Data transfer is to be set by sending data in *AD1task* and waiting for receiving data from the queue in *DA2task*.

Repeat the measurements of the sampled data throughput rate as given in Task 1. Comment on the performance difference between Task 1 and Task 2.

3. The third Lab 7 task is to insert a third task *SQRtask* which squares the input signal from the ADC and outputs the result to the DAC. The task sequence then is *AD1task > SQRtask > DACtask*.

Note that since the input signal is squared and has an offset it is then possible to saturate the DAC output. You should comment on any offset precautions taken to avoid saturation of the DAC output.

The *AD1task, SQRtask* and *DA2task* have equal priority and the two queue sizes are large enough at 10 elements so that the tasks are not blocked. Data transfer is to be set by sending data in *AD1task* to the first queue, waiting for receiving data from the first queue in *SQRtask* and after squaring send the data to the second queue, and waiting for receiving data from the second queue in *DA2task*.

Repeat the measurements of the sampled data throughput rate as given in Task 1. Comment on the performance difference between Task 2 and Task 3.

You are to use the *Project Report Format* posted on *Canvas*. You are to upload your *Report* to Canvas for time and date stamping to avoid a late penalty. This Laboratory is for the week starting March 16th and due no later than 11:59 PM March 23rd.

main.c

```c
//AD1DA2PmodPT Example  ECE3622  c2019 Dennis Silage

#include "xparameters.h"
#include "xil_io.h"
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
#include "sleep.h"

//AD1Pmod from Address Editor in Vivado, first IP
#define AD1acq		0x43C00000	//AD1 acquisition    - output
#define AD1dav		0x43C00004	//AD1 data available - input
#define AD1dat1		0x43C00008	//AD1 channel 1 data - input
#define AD1dat2		0x43C0000C	//AD1 channel 2 data - input

//DAC2Pmod from Address Editor in Vivado, second IP
#define DA2acq		0x43C10000	//DA2 acquisition    - output
#define DA2dav		0x43C10004	//DA2 data available - input
#define DA2dat1		0x43C10008	//DA2 channel 1 data - output
#define DA2dat2		0x43C1000C	//DA2 channel 2 data - output

int main(void)
{
	int adcdav;			//ADC data available
	int adcdata1;			//ADC channel 1 data
	int	adcdata2;		//ADC channel 2 data

	int dacdata1;			//DAC channel 1 data
	int dacdata2;			//DAC channel 2 data
	int dacdav;			//DAC data available

	xil_printf("\n\rStarting AD1-DA2 Pmod demo test...\n");
	Xil_Out32(AD1acq,0);		//ADC stop acquire
	adcdav=Xil_In32(AD1dav);	//ADC available?
	while(adcdav==1)
		adcdav=Xil_In32(AD1dav);
	Xil_Out32(DA2acq,0);		//DAC stop acquire
	dacdav=Xil_In32(DA2dav);	//DAC available?
	while(dacdav==1)
		dacdav=Xil_In32(DA2dav);

	while (1)
	{
		//ADC
		Xil_Out32(AD1acq,1);	//ADC acquire
		while (adcdav==0)	//ADC data available?
			adcdav=Xil_In32(AD1dav);
		Xil_Out32(AD1acq,0);	//ADC stop acquire
		adcdata1=Xil_In32(AD1dat1);	//input ADC data
		adcdata2=Xil_In32(AD1dat2);
		while (adcdav==1)		//wait for reset
			adcdav=Xil_In32(AD1dav);
```

```
		dacdata1=adcdata1;		//ADC -> DAC pass through
		dacdata2=adcdata2;

		//DAC
		Xil_Out32(DA2dat1, dacdata1);	//output DAC data
		Xil_Out32(DA2dat2, dacdata2);
		Xil_Out32(DA2acq,1);		//DAC acquire
		while (dacdav==0)		/DAC data output?
			dacdav=Xil_In32(DA2dav);
		Xil_Out32(DA2acq,0);		//stop DAC acquire
		while(dacdav==1)		//wait for reset
			dacdav=Xil_In32(DA2dav);
	}
}
```