FPGA Control for an Integrated Photonics Laser Control System

We will be controlling a system similar to this preprint, but we'd like to control all channels simultaneously and synchronize all channels with the external control infrastructure.

"DC" control

- 32 channels
- 16 bit resolution
    - -12 bit resolution should be acceptable.
- PC controlled
- no speed requirements
- 0-3.3V voltage range

Pulsed control

- 32 channels
- amplitude:
    - basic goal G1 - on-off only (0, 3.3V)
    - stretch goal G2 - on-off with varying amplitudes - 14-bit resolution between 0, 3.3V
    - ideal goal G3 - full amplitude control at 10ns, 16-bit resolution
- experiments to be held in memory
- external trigger to execute on all 32 channels
- 10ns timing resolution in each channel and between channels
- 50ms maximum experiment time.

	Au 22	**Win 23**	**Sp 23**	**Su 23**	**Au 23**
Optics	Build setup	Characterize single channels	Characterize single channels	Multi-channel device characterization	Integration with ions
DC control	N/A	2+ channels 16-bit analog resolution	2+ channels 16-bit analog resolution	4+ channels 16-bit analog resolution	32 channels 16-bit analog resolution
Pulsed control	N/A	Not needed	2+ channels 16-bit analog amplitude resolution (G2)	4+ channels 16-bit analog amplitude resolution	32 channels 16-bit analog amplitude resolution

Options (from Geoff 10/12):
Updated 10/14/22

Potential Hardware :

'Eclypse' Z7 FPGA board with Xilinx Zynq 7020

The Z-7020 FPGA has a dual core ARM processor, 85K logic cells, 4.9 Mbits of RAM and 220 DSP slices (multipliers)

https://digilent.com/shop/eclypse-z7-zynq-7000-soc-development-board-with-syzygy-compatible-expansion/

The board has two Zmod connectors and two Pmod connectors.
Cost is $500, or $580 with 2x Zmod AWG boards.
There are cheaper FPGA boards but this board has the best mix of connectors for the analog add-on boards that we would need. The board has a UART interface that can be used to download commands, analog parameters and waveform tables. The on-FPGA ARM CPU can be programmed to boot up with a simple program to manage the UART interface. Expansion to 32 channels would require 8 copies of the hardware.

"Pulsed" channels

The Digilent Zmod AWG 1411, is an open-source hardware SYZYGY™ compatible pod containing a dual-channel 14-bit DAC and the associated front end. The Zmod AWG is intended to be used with any SYZYGY™ compatible carrier board having the required capabilities.
https://digilent.com/reference/zmod/awg/start Cost $99

"DC" channels

Eight 12-bit D2A outputs on a PMOD board. Serial control with a 32-bit word at 50MHz
https://digilent.com/shop/pmod-da4-e a ight-12-bit-d-a-outputs/
The board uses a 12-bit DAC (AD5628) but there is a pin-compatible 16-bit device (AD5668) that could be soldered onto the board to replace the original part.
Cost $25

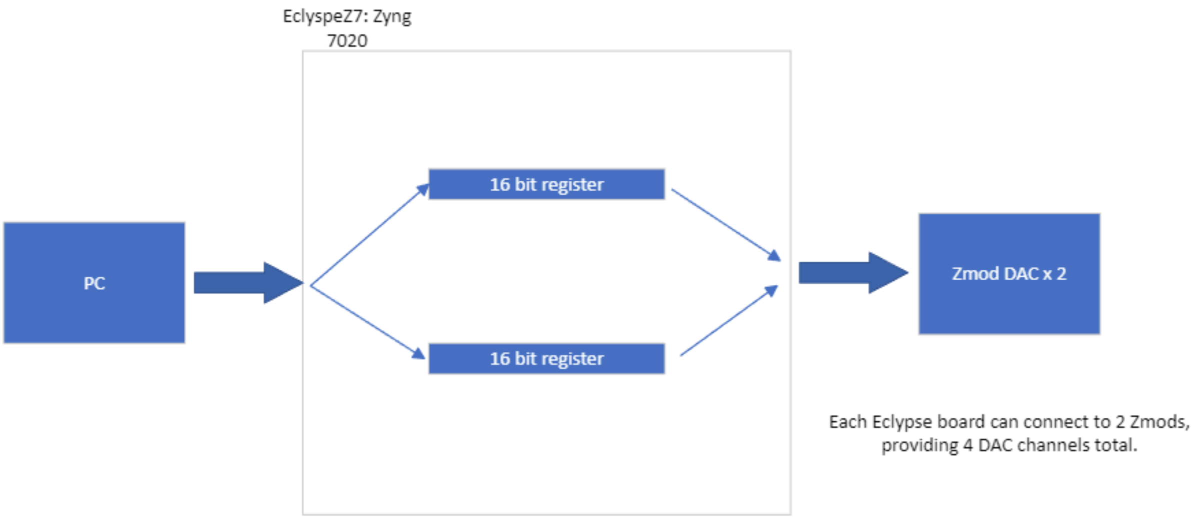
A system with an Eclypse board, two AWG1411 zmod boards and 1 PMOD-DA4 board would meet the requirements for a 4-channel DC and 4-channel pulsed system.
Total cost (579 + 25) - 15% discount = $515
The DC would initially be limited to 12-bit resolution and the Pulsed would be 14-bit instead of both 16-bit.

An enclosure can be purchased for $50.
https://digilent.com/shop/eclypse-z7-enclosure-kit/
Syzygy Breakout board $30
https://opalkelly.com/products/szg-brk-std/

FPGA Control for an Integrated Photonics Laser Control System

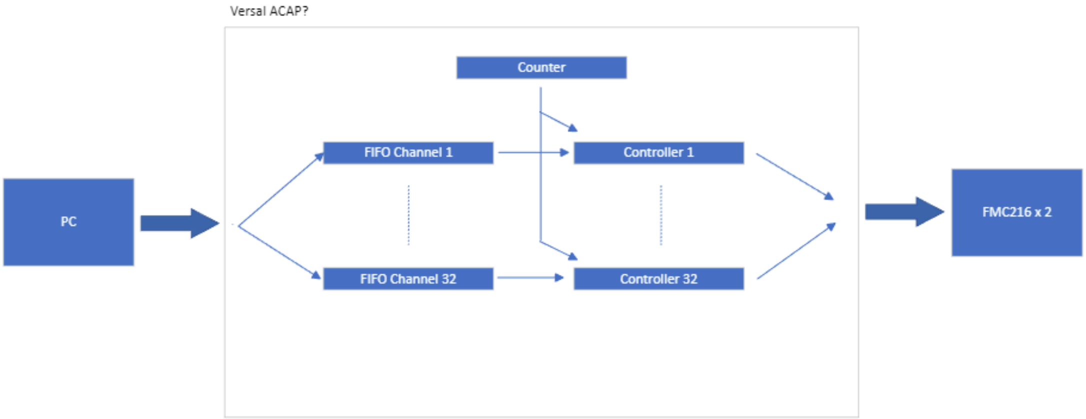
Block Diagram Draft: FPGA Block Diagram

Basic DC setup:



Each Eclypse board can connect to 2 Zmods, providing 4 DAC channels total.

Amplitude Control, 32 channel setup



Setup		Cost with no free parts	Goal 1	Goal 2	Goal 3
Eclypse Board + 2 Zmods + 1 Pmod	4 12-bit DC channels 4 14-bit Pulsed control channels	$515	$515	$515	$4,120
Versal + 1 FMC216 (16-bit resolution)	16 16-bit DC OR Pulsed control channels	$15,345	$15,345	$15,345	N/A
Versal + 2 FMC216 (16-bit resolution + 4 Pmods	32 16-bit DC and Pulsed control channels	$21,345	N/A	N/A	$21,345
		Cost assuming free parts from Xilinx/Altera/Digilent			
Eclypse Board + 2 Zmods + 1 Pmod		$0	$0		
Versal + 1 FMC216 (16-bit resolution)		$6,000	$6,000	$6,000	N/A
Versal + 2 FMC216 (16-bit resolution		$12,000	N/A	N/A	$12,000

Altera Options:
Intel Agilex Direct RF-Series Soc FPGA AGRM027:
https://www.intel.com/content/www/us/en/products/details/fpga/agilex/direct-rf-series.html

- 16 14-bit resolution DACs per board, all in one piece.
- We would need 4 boards total for the end goal
- Price unknown, but it's from Altera so it should be free.

Altera EP4CE10F17C8N FPGA Development Board

https://www.aliexpress.us/item/2251832570124283.html?gatewayAdapt=glo2usa4itemAdapt&_randl_shipto=US

- Very cheap $113 per board
- 2 DAC channels per board, only 12-bit precision
- Would need 32 boards for the end goal so that might be a bit too cumbersome.

Stage	Description
1	Eclypse board. Blink LED0 green. 50% duty cycle. 1 second on, 1 second off
2	Use 125 MHz clock. Generate internal 100 MHz clock. Use 'blink' block in 'qlaser_top' to drive LED0 green with 'flash' output and LED1 with stretched 'tick_sec'.
3	Add serial interface to 'qlaser_top.vhd'. Serial interface supports commands “W AAAA DDDDDDDD” (Write 32-bit data to 16-bit address) and “R AAAA XXXXXXXX” (Read data from address AAAA.) Add a 'debug' block to qlaser_top to generate a trigger signal driven from a register. Stretch trigger pulse to drive LED1 blue. Use a console terminal (TeraTerm) to send a message to set the trigger high. Observe LED1. Add a version number readback to the debug block. Verify version number with console. Write Python script to generate COM messages instead of using TeraTerm.
4	Add a block to set DC DAC levels on the PMOD boards. The DAC interface uses 4 PMOD boards. Each board drives 8 DAC values. Currently we only have 1?. Block should have 4 channels to support writing to 4 SPI buses at the same time. Block should decode register addresses to decide which interface to update. An update occurs whenever the corresponding register is written. If multiple registers in one channel are updated at the same time then updates for those DACs are queued. Perhaps SPI interface runs fast enough so that transfers complete faster than register writes can occur. Check for this. Simulate block. Use COM interface to write block registers to update DACs. Check the PMOD interface waveforms with a logic analyzer.
5	Similar to ‘4’ for Pulsed output. 32 channels with 24-bit ‘time’ values and 16-bit ‘voltage’ values. Each channel has a 32word x 40-bit table. Common 24-bit timer started by ‘trigger’. Each channel output voltage is updated when the next table time value equals timer count. Write testbench. Simulate In initial hardware just route MSB of pulsed DAC values to GPIO pins (on spare PMOD interfaces?)i.e. Bit 15. Observe with a logic analyzer. This maybe easier on the Zybo board since it has more PMOD output pins that can be used for GPIO. Otherwise we could look at using a Syzygy breakout board : https://opalkelly.com/products/szg-brk-std/ ($30)
6	What is the high speed DAC (ZMOD) interface? Replace two pulsed channels with full 14-bit outputs to AWG1411 boards. Look at Syzygy spec and other documentation https://syzygyfpga.io/specification/

	The ZMOD interface to the AWG1411 includes a SPI bus connection to the DAC for initialisation and an I2C bus to retrieve calibration tables. Digilent supplies controller IP for these interfaces: https://github.com/Digilent/vivado-library/tree/zmod/v2/2019.1-2/ip/Zmods/ZmodAWGController

The UART in the FPGA can receive two different commands:

- Write: “WAAAADDDDDDDD”

- Read: “RAAAADDDDDDDD”

DC Channels:

Simply select a channel using the address and write the data value for the DAC to a register. 3 bits of the data value can also be used to select one of 8 channels on a Pmod.

Pulsed Channels:

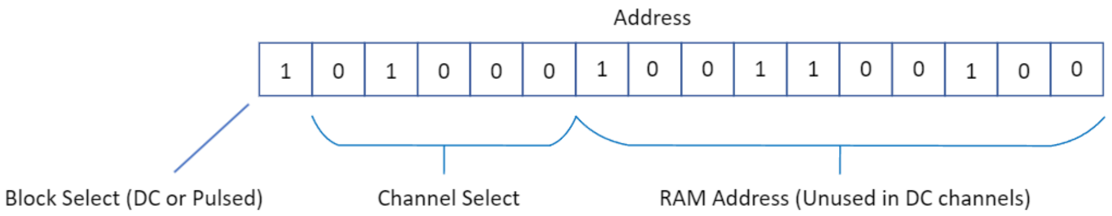
The user inputs a voltage, start time of a duration for a pulse.

Three writes to the FPGA are needed per pulse in an experiment.

1. Write #1 stores 14-bit DAC value to a temporary register.

2. Write #2 provides 24-bit pulse start time, FPGA combines the 14-bit DAC value with the 24-bit value and writes the 40 bits to the RAM with the address given from the UART message.

3. Write #3 provides 24-bit stop time and stores 14-bit “0” value with 24-bit stop time to the RAM.


Address
1
0
1
0
0
0
1
0
0
1
1
0
0
1
0
0
Block Select (DC or Pulsed)
Channel Select
RAM Address (Unused in DC channels)

The number of address bits allows for 40x1024 RAMs, capable of storing up to 512 pulses per channel in an experiment, but the UART in the FPGA can be modified to increase the number of Address bits in a message if necessary.

Pmod DAC controls:

Send 32 bits SPI message configured with below commands, address, and data

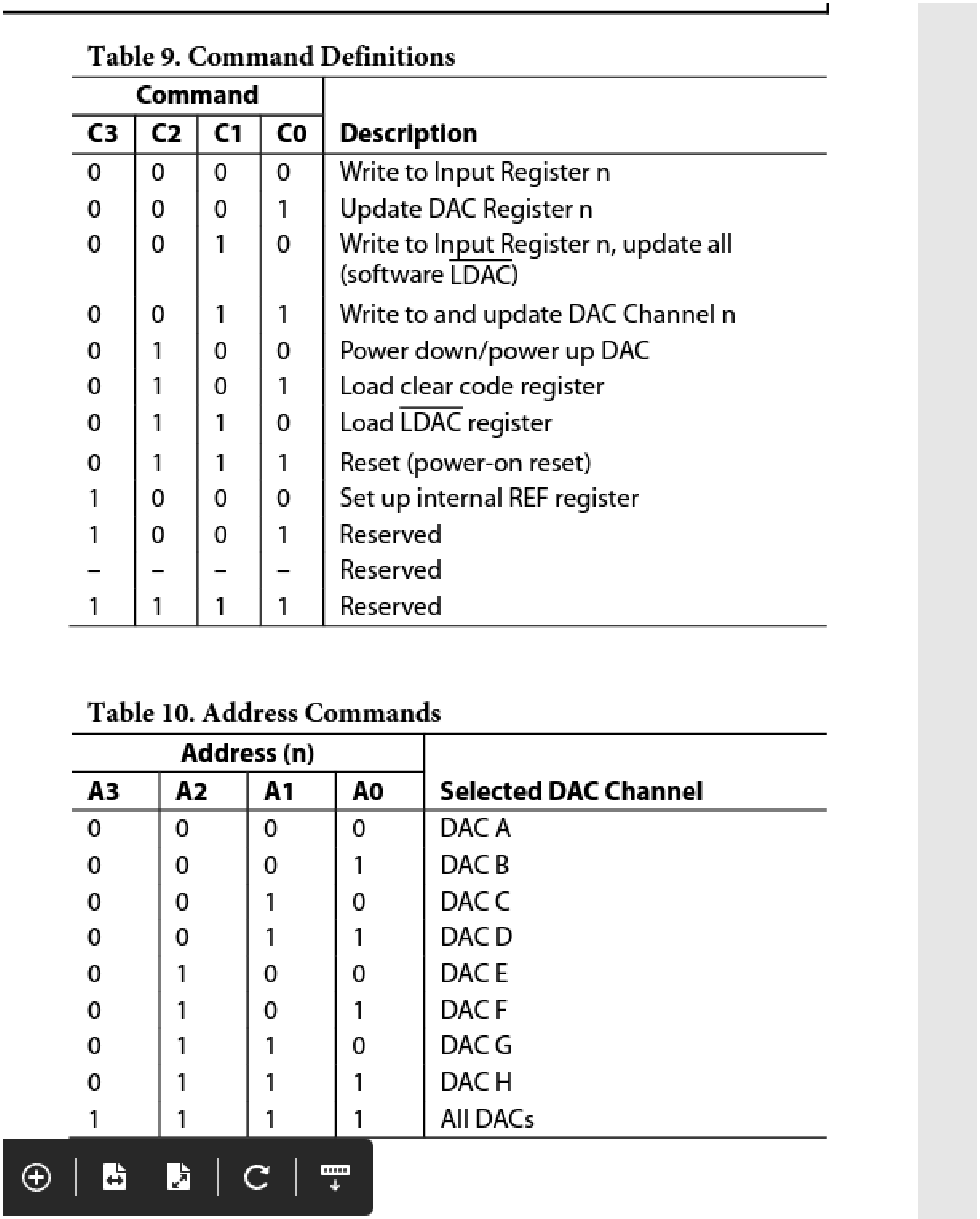
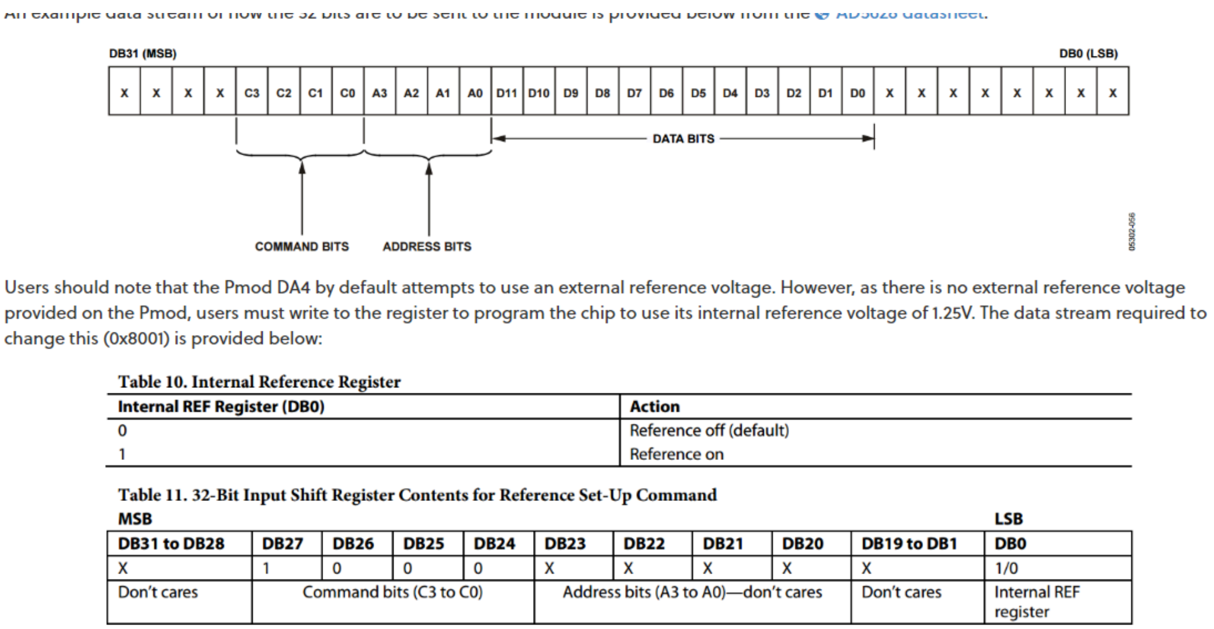


Table 9. Command Definitions

Command				
C3	C2	C1	C0	Description
0	0	0	0	Write to Input Register n
0	0	0	1	Update DAC Register n
0	0	1	0	Write to Input Register n, update all (software $\overline{\text{LDAC}}$)
0	0	1	1	Write to and update DAC Channel n
0	1	0	0	Power down/power up DAC
0	1	0	1	Load clear code register
0	1	1	0	Load $\overline{\text{LDAC}}$ register
0	1	1	1	Reset (power-on reset)
1	0	0	0	Set up internal REF register
1	0	0	1	Reserved
–	–	–	–	Reserved
1	1	1	1	Reserved

Table 10. Address Commands

Address (n)				
A3	A2	A1	A0	Selected DAC Channel
0	0	0	0	DAC A
0	0	0	1	DAC B
0	0	1	0	DAC C
0	0	1	1	DAC D
0	1	0	0	DAC E
0	1	0	1	DAC F
0	1	1	0	DAC G
0	1	1	1	DAC H
1	1	1	1	All DACs



An example data stream of how the 32 bits are to be sent to the module is provided below from the AD5628 datasheet.

DB31 (MSB) DB0 (LSB)

X	X	X	X	C3	C2	C1	C0	A3	A2	A1	A0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	X	X	X	X	X	X	X	X

Users should note that the Pmod DA4 by default attempts to use an external reference voltage. However, as there is no external reference voltage provided on the Pmod, users must write to the register to program the chip to use its internal reference voltage of 1.25V. The data stream required to change this (0x8001) is provided below:

Table 10. Internal Reference Register

Internal REF Register (DB0)	Action
0	Reference off (default)
1	Reference on

Table 11. 32-Bit Input Shift Register Contents for Reference Set-Up Command

MSB LSB

DB31 to DB28	DB27	DB26	DB25	DB24	DB23	DB22	DB21	DB20	DB19 to DB1	DB0
X	1	0	0	0	X	X	X	X	X	1/0
Don't cares	Command bits (C3 to C0)				Address bits (A3 to A0)—don't cares				Don't cares	Internal REF register