

DAC board control manual version 2

Dept of Physics & Astronomy, Seoul National University

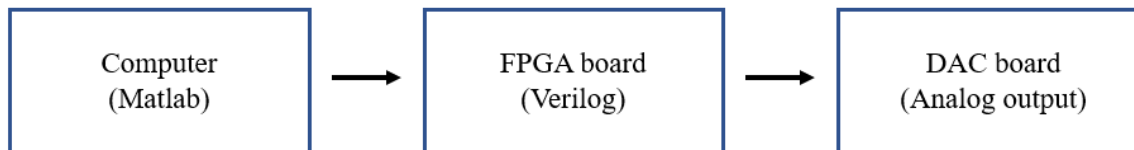
Center of nano liquid ion trap quantum computing group

Undergraduate internship Jiyong Yu

This manual is written for easy use of DAC board in CNL Ion trap group

1. Introduction

DAC(Digital to Analog Converter) board는 말 그대로 computer에서 내보내는 digital signal을 실제 실험에서 활용할 수 있는 analog signal로 바꾸어 주는 역할을 한다. 실제로 우리의 실험에서 DAC board의 output은 laser lock control, attenuator control 등 다양한 목적으로 활용된다. 이 때 computer에서 digital signal을 내보낸 뒤, DAC board에서 analog output으로 바뀌는 과정은 다음과 같은 순서로 이루어 진다.












위 그림에서 보이듯이 Computer에서 내보내진 digital signal이 DAC board에서 최종적으로 analog output으로 바뀌는 과정은 크게 3단계로 나뉜다. 먼저 computer에서 digital signal을 Matlab code를 통하여 FPGA board로 전달한다. 이 때 FPGA board는 이미 작성된 Verilog code에 의하여 programmed 된 상태이고, FPGA board는 Verilog code를 통하여 computer로부터 받은 digital signal을 output으로 변환한다. 마지막으로 FPGA board에서 발생한 output이 DAC board의 각 chip에 전달되고 원하는 analog output을 얻을 수 있다. 이 때 FPGA board와 DAC board는 pin을 통하여 전기적으로 연결되어 있다.

우리가 실제로 사용하고 있는 FPGA board는 arty-s7 이고 DAC board는 SKT에서 제작된 것이다. 자세한 manual은 첨부된 arty-s7_rm.pdf 와 dac8734.pdf 에서 참조할 수 있다.

2. Matlab instruction

Matlab code는 computer에서 FPGA board로 digital signal을 communicate 하는 용도로 활용된다. Experiment_Scripts\Serial_DAC_control_v2.0\Serial_DAC_control_v2.0_Matlab folder에서 해당하는 code들을 확인할 수 있으며 다음과 같다.

 Converter_hex.m	2019-02-13 오후 2...	MATLAB Code	1KB
 RangeChecker.m	2019-03-31 오후 9...	MATLAB Code	3KB
 sDAC_init.m	2019-02-13 오후 1...	MATLAB Code	2KB
 sDAC_monitor.m	2019-02-20 오후 1...	MATLAB Code	1KB
 sDAC_quit.m	2019-01-30 오후 2...	MATLAB Code	1KB
 sDAC_readout.m	2019-01-31 오후 4...	MATLAB Code	1KB
 sDAC_reset.m	2019-01-30 오후 2...	MATLAB Code	1KB
 sDAC_setV.m	2019-02-14 오후 3...	MATLAB Code	3KB
 sDAC_trig.m	2019-01-30 오후 2...	MATLAB Code	1KB

각 function의 기능은 다음과 같다.

DAC_object = sDAC_init(Port, Reference, Polarization): FPGA board와 computer 간의 communication을 행하는 객체를 생성하는 함수이다. 이 때 DAC_object는 cell 형태로 이루어져 있고 DAC_object{1}에 해당하는 것이 serial port object이다. 따라서 computer에서 digital signal이 발생하는 경우 serial에 해당하는 DAC_object{1}에 signal이 쓰이게 된다. 또한 DAC_object{2}, DAC_obejct{3}는 각각 DAC board의 reference voltage, polarization을 나타내는 array variable이다. 실제로 사용하는 경우 dac=sDAC_init('COM5', reference, polarization)와 같이 사용한다. Port의 경우 현재 사용중인 Port name에 의해 결정되며 check가 필요하다. Reference, polarization은 각 chip의 reference voltage, polarization을 나타내는 array variable이다. (Refer to comments in code)

sDAC_setV(DAC_object, Channel, Voltage, Trigger): Initialization을 통하여 생성된 serial object에 실제로 원하는 voltage를 전달하는 함수이다. 이 때 Channel은 DAC board의 pin을 나타내는 변수이다. DAC board에는 총 8개의 chip이 존재하고, 각 chip마다 4개의 output port가 존재한다. 따라서 Channel은 1 ~ 32이 가능하고 해당하는 DAC board에서의 address는 00, 01, 02, 03, 10, 11, 12, 13, ..., 70, 71, 72, 73 이 된다.

위와 같이 channel을 통하여 DAC board의 해당 pin을 알아낸 뒤에는 voltage를 변환하는 과정이 필요하다. 먼저 voltage가 올바른 range에 속하는지를 확인해야 한다. 예를 들어 reference voltage가 7.5V이고 bipolar인 경우 가능한 voltage range는 -15 ~ 15V이다 (Refer to

DAC board instruction). 따라서 RangeChecker 함수를 통하여 이 range를 벗어나는 voltage를 fix 해주는 과정이 필요하다. 예를 들어 16V를 입력하는 경우 output이 15V로 fix 된 뒤 “too large voltage!” 의 message가 발생하게 된다.

RangeChecker 함수를 통하여 올바른 voltage를 얻은 후에는 이를 DAC board의 register에 저장될 수 있는 값으로 변환하는 과정이 필요하며, 이 과정은 Converter_dac fuction에 의하여 수행된다. Input voltage가 register value로 변환되는 과정은 다음과 같다. (Refer to dac8734.pdf page 20)

TRANSFER FUNCTION FOR THE ANALOG OUTPUTS (V_{OUT-0} to V_{OUT-3})

For bipolar output:

$$V_{OUT} = \text{Gain} \times V_{REF} \times \left(\frac{\text{INPUT_CODE}}{65536} + \frac{\text{ZERO_CODE}}{8 \times 65536} \right) \times \left(1 + \frac{\text{GAIN_CODE}}{2 \times 65536} \right) \quad (1)$$

For unipolar output:

$$V_{OUT} = \text{Gain} \times V_{REF} \times \left(\frac{\text{INPUT_CODE}}{65536} + \frac{\text{ZERO_CODE}}{8 \times 65536} \right) \times \left(1 + \frac{\text{GAIN_CODE}}{65536} \right) \quad (2)$$

Where:

GAIN is the DAC gain, which can be set to x2 or x4 and is determined by the connection of pins R_{FB1-X} and R_{FB2-X} to V_{OUT-X} , and the GAIN bit in the Command Register.

INPUT_CODE is the decimal equivalent value of the code written into the DAC input register.

ZERO_CODE is the decimal equivalent value of the code written into the Zero Register.

GAIN_CODE is the decimal equivalent value of the code written into the Gain Register.

위의 식에서 input_code가 바로 DAC board의 register에 저장되는 값이다. 따라서 위 식을 역으로 써서 register에 저장되는 값을 다음과 같이 얻을 수 있다. Negative value의 경우는 2s complement를 사용하여 처리하면 된다.

```
function Data_Register = Converter_dac(Reference, Voltage)
% For converting rule, see dac8734 manual page 20
gain = 4; % gain is fixed for dac_board
if(Voltage < 0)
    Data_Register = 65536 + (65536*Voltage)/(gain*Reference); % 2's complement
else
    Data_Register = (65536*Voltage)/(gain*Reference);
end
end
```

마지막으로 위에서 얻은 register value를 hexadecimal number로 변환한다. 그리고 이 hexadecimal number를 Ascii code를 사용하여 한 번 더 변환해준다(97을 더하면 된다). 예를 들어 register value가 0001 인 경우 이를 aaab로 변환한다. 성공적으로 write가 완료되었다면 W01D aaab와 같은 message가 출력되며 이를 serial object에 전달한다. 이 때 01은 DAC

board의 address를 의미하며 D는 Data register를 나타낸다.

Trigger의 경우는 DAC board의 register를 바로 update 할 것인지를 결정하는 변수이다. Trigger가 1인 경우에는 DAC board의 register가 바로 update되며, 0인 경우에는 trigger가 되기 전까지 update되지 않는다. 이 경우 sDAC_trig function을 사용하여 trigger를 진행할 수 있다.

마지막으로 DAC board instruction에서 설명하겠지만 **DAC board address의 25, 26이 서로 뒤바뀌어 있다**(Chip 6's first and second pin). 이러한 문제로 인하여 다음과 같은 code가 추가되었다.

```
% Pin 25 and 26 are reversed, so we exchange them manually
if(Chip_number == 6)
    if(Pin_number == 0 || Pin_number == 1)
        Pin_number = 1 - Pin_number;
    end
end
```

sDAC_readout(DAC_object): DAC_board의 32개의 chip register(pin)에 현재 저장되어 있는 값을 모두 출력해주는 기능을 하는 function이다.

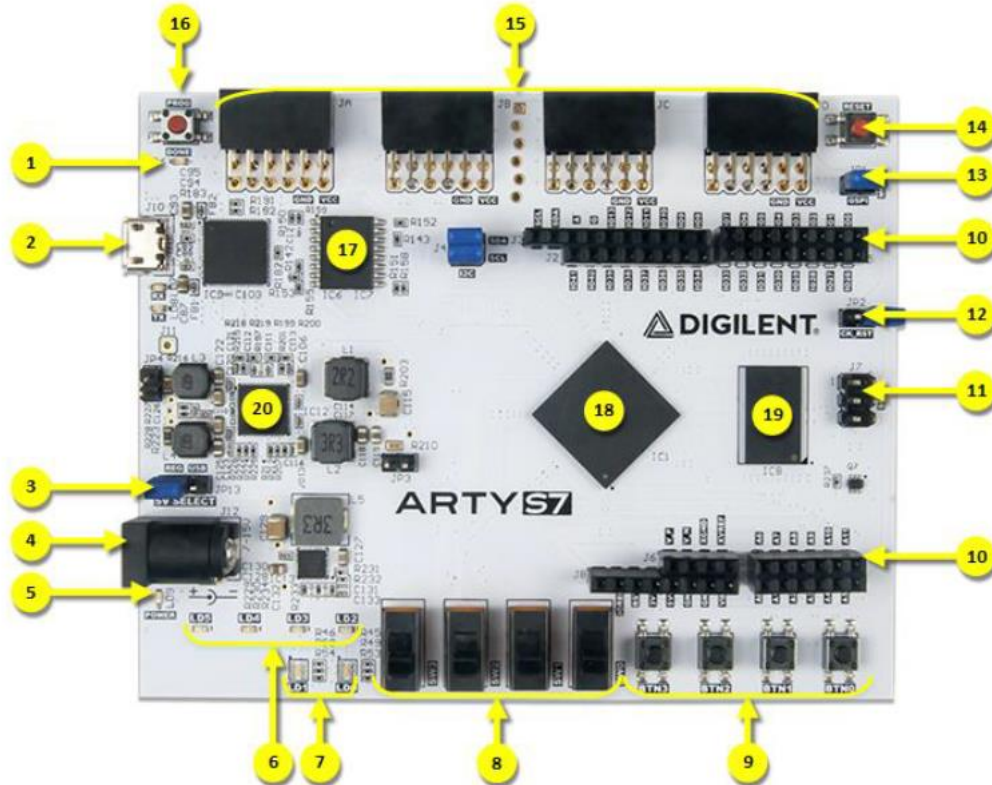
sDAC_monitor(DAC_object, Channel): 원하는 channel number(00, 01, ..., 72, 73)에 대하여 그 channel의 실제 output value를 monitor하기 위하여 사용되는 함수이다. 예를 들어 00 channel을 monitor하고 싶은 경우 M00D라는 data를 serial object를 통하여 FPGA board에 전달하게 된다. FPGA board는 해당하는 output signal을 DAC board로 전달하게 되고, DAC board의 monitor pin에서 해당 channel의 output value를 확인할 수 있게 된다. (Refer to DAC board instruction)

sDAC_reset(DAC_object): DAC board의 모든 register와 output을 초기화하는 function이다.

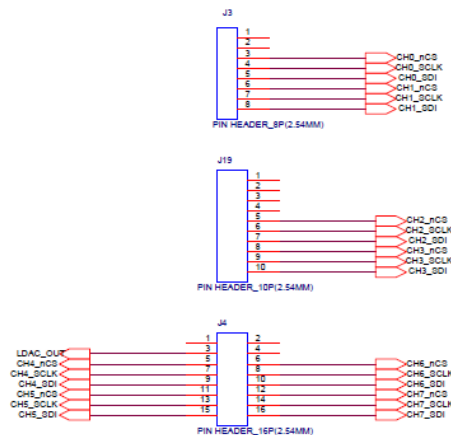
마지막으로 FPGA와의 connection이 제대로 이루어져 있는지 확인하려면 *IDN? 을 query 하면 된다. Connection이 정상적으로 이루어져 있다면 8CH DAC8732 CONTROLLER message가 변환된다. 이 과정은 Matlab의 tmttool(test and measurement tool)을 활용하면 편리하다.

3. FPGA board and Verilog instruction

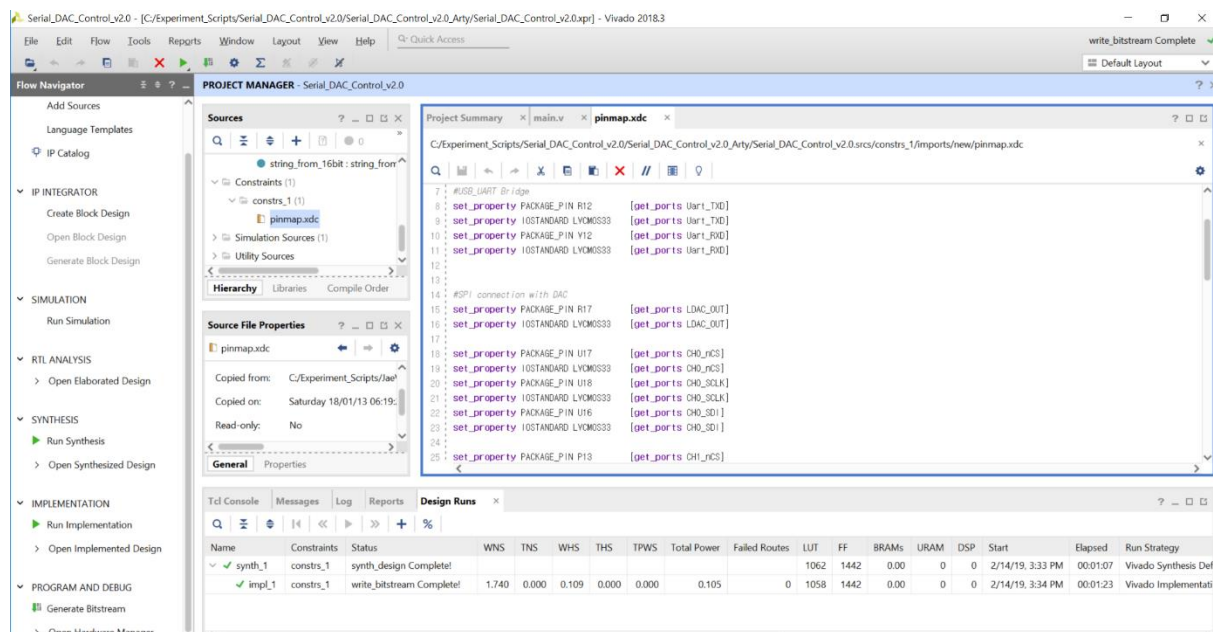
우리가 실험에서 활용하고 있는 FPGA board는 arty-s7이며 자세한 manual은 첨부된 자료에서 확인할 수 있다. FPGA board의 구조는 다음과 같다.



위 그림에서 2번 port는 computer와의 USB connection을 위하여 사용된다. 10번 pin들이 모두 DAC board와의 전기적인 connection을 이루는 pin들이다. 즉 10번 port에서 나오는 output signal (Chip select, SPI clock, Data)이 DAC board에 전달되어 원하는 analog output을 얻게 해준다. 해당되는 DAC board의 pin들은 다음과 같다. 이 때 J3, J19, J4의 pin들이 FPGA board의 10번 pin들과 전기적으로 연결되어 있다. (올바르게 연결되어 있는지는 필자가 확인하였다.)



다음으로 Verilog code를 살펴보자. Verilog의 대략적인 구조는 다음과 같다.



Main function을 살펴보기 앞서 Verilog code의 register와 실제 FPGA board의 pin을 mapping 해주는 file이 필요하며 이에 해당하는 것이 Constraints에서 볼 수 있는 pinmap.xdc이다. 위에서 볼 수 있듯이 각 pin들이 R17,U17 등으로 naming 되어있다. 우리가 사용하는 arty s7의 각 pin의 실제 naming은 <https://github.com/Digilent/digilent-xdc/blob/master/Arty-S7-25-Rev-E-Master.xdc> 에서 확인할 수 있다. (올바르게 naming되어있는지 역시 필자가 확인하였다.)

다음으로 main function을 간략히 살펴보자. Main function에서 가장 중요한 3가지 변수는 nCS, SCLK, SDI이다. nCS는 not chip select에 해당하는 것으로 nCS가 0인 경우에만 DAC board의 register에 data가 write된다 (즉, Enable과 동일한 역할을 한다). SCLK은 SPI clock에 해당하는 것이며 FPGA board와 DAC board 간의 communication의 기준이 되는 clock signal이다. FPGA board의 clock은 100MHZ이고 이를 이용하여 SPI clock의 speed를 임의로 조절할 수 있다. (현재는 FPGA clock보다 8배 느리게 되어있다.) 마지막으로 SDI는 DAC board의 register에 write 되는 data를 의미하며 이 value를 통하여 DAC board에서 analog output이 결정되게 된다. 아래 그림은 필자가 실제로 FPGA board에서 나오는 nCS, SCLK, SDI를 순서대로 oscilloscope를 통하여 본 것이다. nCS가 0일 때 SDI가 발생하고 전달되고 있음을 확인할 수 있다. 이 때 각 chip마다 해당하는 nCS, SCLK, SDI가 각각 존재한다.



또한 Main function은 크게 Idle state와 MSG_IN state를 가지고 있는 finite state machine으로 동작한다. MSG_IN state의 경우 크게 Monitor, Read, Write, Reset 로 나뉜다. 각 state 모두 DAC board의 data register에 output signal을 전달하게 되는데, 이를 이해하기 위해서는 DAC board의 register가 어떻게 구성되어 있는지 알아야한다. 먼저 SPI shift register의 구성은 다음장에 주어진 그림과 같다. (For more specific information, refer to dac8734.pdf) Data bit의 길이는 24 bit로 주어진다.

이 때 Data_bit[23] 은 R/W를 나타내며 0인 경우 Write를 나타낸다. 또한 Data_bit[22 : 20] 은 0이며 Data_bit[19 : 16] 은 address bit이다. 이 address bit에 의하여 data가 쓰이는 register의 종류가 결정되게 된다. 남은 bit인 Data_bit[15 : 0] 가 실제로 우리가 쓰고자 하는 data의 정보를 담고 있다.

SPI SHIFT REGISTER

The SPI Shift Register is 24 bits wide, as shown in [Table 2](#). By default, the SPI shift register resets to 000000h at power-on or after a reset.

Table 2. SPI Shift Register Format

MSB								
DB23	DB22	DB21	DB20	DB19	DB18	DB17	DB16	DB15:DB0
R/W	0	0	0	A3	A2	A1	A0	DATA

R/W—Indicates a read from or a write to the addressed register.

R/W = '0' sets a write operation and the data are written to the specified register.

R/W = '1' sets a read-back operation. For read operation, bits A3 to A0 select the register to be read. The remaining are *don't care* bits. During the next SPI operation, the data appearing on SDO pin are from the previously addressed register.

[A3:A0]—Address bits that specify which register is accessed.

DATA—16 data bits

All DAC8734 registers (command registers and data registers) are 16-bit. [Table 3](#) shows the register map.

Table 3. Register Map

ADDRESS BITS				DATA BITS												REGISTER
A3	A2	A1	A0	DB15	DB14	DB13	DB12	DB11	DB10	DB9	DB8	DB7	DB6	DB5: DB2	DB1: DB0	
0	0	0	0	A/B	LD	RST	PD-A	PD-B	Rsvd ⁽¹⁾	GPIO-1	GPIO-0	DSDO	NOP	GAIN Bits	Rsvd ⁽¹⁾	Command Register
0	0	0	1	MDAC-3	MDAC-2	MDAC-1	MDAC-0	AIN	Reserved ⁽¹⁾							Monitor Register
0	1	0	0	D15:D0												DAC-0
0	1	0	1	D15:D0												DAC-1
0	1	1	0	D15:D0												DAC-2
0	1	1	1	D15:D0												DAC-3
1	0	0	0	Reserved ⁽¹⁾							Z8:Z0					Zero Register-0
1	0	0	1	Reserved ⁽¹⁾							Z8:Z0					Zero Register-1
1	0	1	0	Reserved ⁽¹⁾							Z8:Z0					Zero Register-2
1	0	1	1	Reserved ⁽¹⁾							Z8:Z0					Zero Register-3
1	1	0	0	Reserved ⁽¹⁾							G7:G0					Gain Register-0
1	1	0	1	Reserved ⁽¹⁾							G7:G0					Gain Register-1
1	1	1	0	Reserved ⁽¹⁾							G7:G0					Gain Register-2
1	1	1	1	Reserved ⁽¹⁾							G7:G0					Gain Register-3
Others				Reserved ⁽¹⁾												—

(1) Writing to a reserved bit has no effect; reading the bit returns '0'.

예를 들어 Write하는 경우를 살펴보자. DAC board의 각 chip은 총 4개의 pin을 가지고 있고 위에서 address bit 0100, 0101, 0110, 0111이 각각 이 pin에 해당한다. 따라서 예를 들어 Chip0의 first pin에 write를 하는 경우 Data_bit[23 : 16] 은 0000 0100이 되어야 한다. 그리고 남은 bit에 실제 data가 쓰이게 된다. 이와 같은 과정은 다음 code로 구현되어 있다.

```

else if(RXString[9*8-1:8*8] == "W") begin
    if(RXString[8*8-1:5*8] == "000") begin
        DATA00 <= Reg_In;
        DAC_Data0[24-1:16] <= 8'b00000100;
        DAC_Data0[16-1:0] <= Reg_In;
        DAC_Ready[0] <= 1;
        TXLen <= 12;
        TXString <= "Data Fetched";
        state <= SPI_WAIT_BUSY_ANY;
    end
end

```


또 다른 예시로 Monitor 하는 경우를 살펴보자. Monitor process는 조금 더 복잡하므로 다음과 같이 register를 한번 더 보아야 한다.

Monitor Register. Default = 0000h.

The Monitor Register selects one of the four DAC outputs or the external signal AIN that is to be monitored through the V_{MON} pin. Only one bit can be set to '1' at a time. When all bits = '0', the monitor is disabled and V_{MON} is placed in a high-impedance state. The default value after power-on or reset is 0000h.

Table 5. Monitor Register

DB15	DB14	DB13	DB12	DB11	DB10:DB0	V_{MON} CONNECTS TO
0	0	0	0	1	Reserved ⁽¹⁾	AIN
0	0	0	1	0	Reserved ⁽¹⁾	DAC-0
0	0	1	0	0	Reserved ⁽¹⁾	DAC-1
0	1	0	0	0	Reserved ⁽¹⁾	DAC-2
1	0	0	0	0	Reserved ⁽¹⁾	DAC-3
0	0	0	0	0	Reserved ⁽¹⁾	Monitor disabled, Hi-Z (default)

(1) Writing to a reserved bit has no effect; reading the bit returns '0'.

위에서 Vmon 가 Hi-Z, 즉 high impedance인 경우는 monitor 상태가 아님을 의미한다. 이 경우 monitor pin은 아무런 기능을 하지 않는다. Monitor 기능이 사용되는 경우, DAC board의 monitor pin들을 각 chip에 대하여 연쇄적으로 이어져 있다. (For more detail, refer to DAC board instruction) 예를 들어 chip3을 monitor하고자 하는 경우 chip 3의 monitored value가 하위의 chip2, chip1, chip0의 monitor pin으로 모두 전달되어 4개의 monitor pin이 같은 값을 가져야 한다. 즉, chip2의 Vmon가 Ain에 연결되어 있다는 것은 chip2의 monitor pin이 chip3의 monitor pin을 그대로 반영한다는 뜻이다. 결론적으로 각 chip들의 monitor pin은 decreasing order로 연쇄적으로 연결되어 있다.

따라서 monitor를 하기 위해서는 다음과 같이 먼저 모든 pin의 Vmon을 Ain에 연결되도록 해주는 과정이 필요하다. 다음과 같은 code로 쉽게 구현 가능하다.

```
else if(RXString[4*8-1:3*8] == "M") begin
    DAC_Data0[24-1:16] <= 8'b00000001;
    DAC_Data0[16-1:11] <= 5'b00001;
    DAC_Data0[11-1:0] <= 10'b0000000000;
    DAC_Ready[0] <= 1;

    DAC_Data1[24-1:16] <= 8'b00000001;
    DAC_Data1[16-1:11] <= 5'b00001;
    DAC_Data1[11-1:0] <= 10'b0000000000;
    DAC_Ready[1] <= 1;

    DAC_Data2[24-1:16] <= 8'b00000001;
    DAC_Data2[16-1:11] <= 5'b00001;
    DAC_Data2[11-1:0] <= 10'b0000000000;
    DAC_Ready[2] <= 1;

    DAC_Data3[24-1:16] <= 8'b00000001;
    DAC_Data3[16-1:11] <= 5'b00001;
    DAC_Data3[11-1:0] <= 10'b0000000000;
    DAC_Ready[3] <= 1;
```

(Chip 4 ~7에 대해서도 마찬가지로)

그리고 monitor하고자 하는 pin이 있으면 해당하는 pin을 monitor하도록 data bit를 다음과 같이 수정하기만 하면 된다. 이 경우 앞선 code로 인하여 연쇄적으로 전달되는 과정은 자동적으로 이루어진다.

```
else if(RXString[3*8-1:0] == "10D") begin
    DAC_Data1[24-1:16] <= 8'b00000001;
    DAC_Data1[16-1:11] <= 5'b00010;
    DAC_Data1[11-1:0] <= 10'b000000000;
    DAC_Ready[1] <= 1;
    TXLen <= 12;
    TXString <= "Data Fetched";
    state <= SPI_WAIT_BUSY_ANY;
end

else if(RXString[3*8-1:0] == "11D") begin
    DAC_Data1[24-1:16] <= 8'b00000001;
    DAC_Data1[16-1:11] <= 5'b00100;
    DAC_Data1[11-1:0] <= 10'b000000000;
    DAC_Ready[1] <= 1;
    TXLen <= 12;
    TXString <= "Data Fetched";
    state <= SPI_WAIT_BUSY_ANY;
end
```

지금까지 Read, Write하는 과정을 살펴보았다. 대부분의 code는 chip/pin number에 대하여 반복적으로 작성되었기 때문에 위 과정을 이해했다면 어렵지 않다.

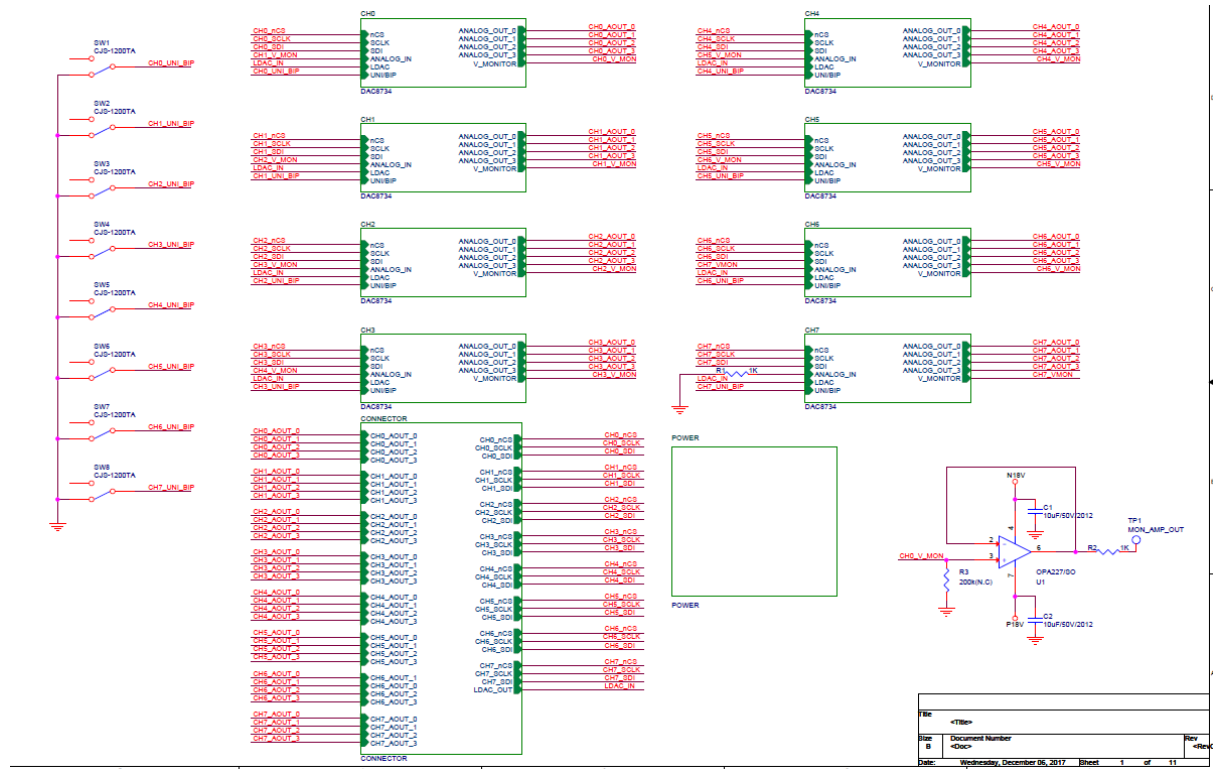
마지막으로 위와 같이 작성된 main code를 실제로 사용하기 위해서는 FPGA board에 programming하는 과정이 필요하다. 이 과정은 다음과 같다.

1. “Open Hardware manager” 클릭하여 FPGA board와 connection
2. “Generate Bitstream” 클릭하여 주어진 Verilog code로부터 main.bit file 생성
3. “Program Device” 클릭하여 생성된 FPGA board를 main.bit를 통해 program
(이 때 main.bit는 Serial_DAC_Control_v2.0.runs\impl_1 directory에 생성됨)

FPGA board가 정상적으로 작동하려면 위와 같이 program하는 과정이 필수적이다.

4. DAC board instruction

DAC board의 큰 구조는 다음과 같이 주어진다. 앞서 설명하였듯이 각 chip의 monitor pin은 하위의 chip에 대해서 연쇄적으로 연결 되어있다. 예를 들어 Chip7의 CH7_VMON은 Chip6의 ANALOG_IN(Ain)에 연결되어 있음을 알 수 있다.

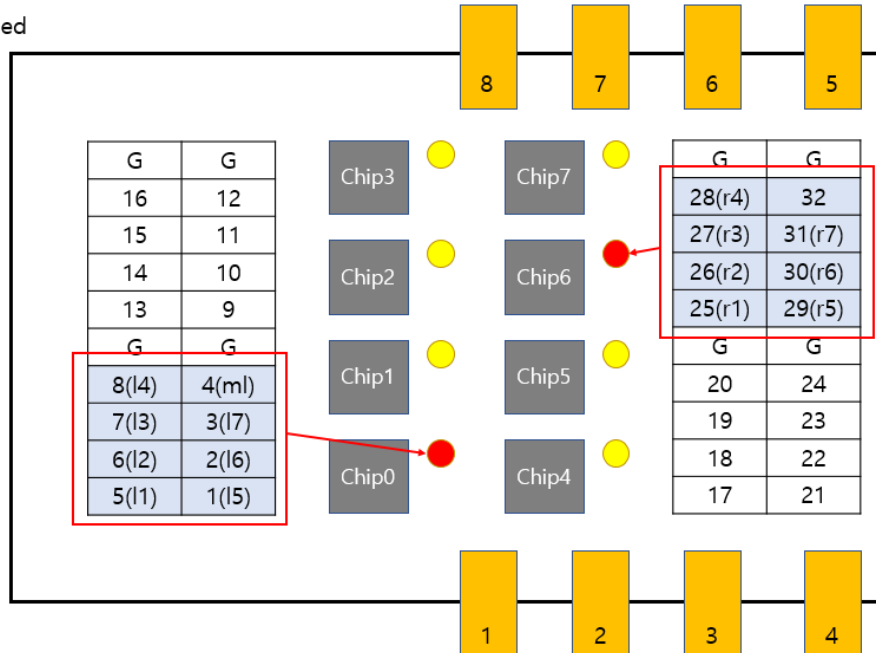


또한 우리가 사용하고 있는 DAC board는 reference voltage, polarization을 가지고 있다. Reference voltage는 7.5V, 2.5V가 가능하며 polarization은 bipolar, unipolar의 두 종류가 있다. 이때 사용되는 gain은 4로서 고정되어 있다. 예를 들어 reference voltage 7.5V, unipolar인 경우는 0~30V, bipolar인 경우는 -15~15V의 range를 가지게 된다. 각 chip마다 이 설정을 다르게 할 수 있다. DAC board의 좌측 하단의 switch에서 이를 조정할 수 있다. (Unipolar/bipolar, 7.5V/2.5V 순으로 되어있음) 실제로는 saturation으로 인하여 output이 30V까지는 나오지 않는다. (경험적으로 17V 정도에서 saturation이 일어났다)

5. DAC numbering and problems

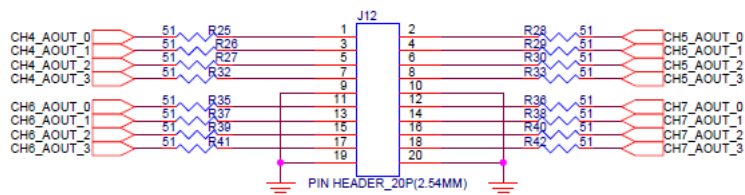
다음 그림은 DAC board의 8개의 chip과 해당하는 pin들을 나타낸 모식도이다.

25, 26 pin reversed



위 그림에서 Chip들 양 옆에 놓인 1~32의 Channel이 바로 각 Chip의 output에 해당하는 pin 들이다. (1~4는 Chip0, 5~8은 Chip1, ...) 또한 노랗게 표시된 부분은 cable에 연결할 수 있는 output으로서 Chip0, Chip1의 output을 나타낸다 (앞서 설명한 output pin들과 같은 값을 공유한다). G로 표시된 부분은 ground를 의미한다. 그러나 우리가 사용하고 있는 DAC board에는 몇 가지 문제들이 존재한다.

1. Pin 25 and Pin 26 are reversed



앞서 계속 언급하였듯이 pin 25, 26이 hardware적으로 서로 뒤바뀌어 있다. 즉 정상적이라면 CH6_AOUT_0가 J12_11, CH6_AOUT_1이 J12_13에 연결되어 있어야 하는데 이것이 서로 뒤바뀌어 있다(Multimeter를 통하여 connection을 체크하였다). 따라서 connection의 hardware적인 문제가 있다고 판단된다. **현재는 code 상에서 이 문제를 임시 방편으로 해결하고 있다. (문제가 해결된다면 code를 수정해야함!)**

2. Wrong and inconsistent results of some pins

위의 모식도에서 파란색으로 표시된 부분을 제외한 나머지 영역의 pin들은 reliable한 결과를 주지 못했다. 예를 들어 pin 9에 특정 전압을 주면 pin 11의 전압이 바뀌는 등 이상한 결과를 보였다. 더욱 문제인 부분은 어떤 경우엔 올바른 결과가 나오다가 다시 잘못된 결과가 나오는 등 결과의 일관성이 없었다는 점이다.

3. Failure of monitor pin chain

앞서 언급하였듯이 각 chip의 monitor pin들은 연쇄적으로 연결되어 있다. 각 Chip 오른 쪽에 있는 작은 원모양이 이러한 monitor pin을 나타낸다. 따라서 예를 들어 Chip7을 monitor하고자 하는 경우 Chip0 ~ Chip7의 monitor pin이 모두 동일한 결과를 보여야 한다. 그러나 Chip7을 monitor하고자 하는 경우 Chip6의 monitor pin까지만 값이 전달되는 문제가 발생했다. (확인 결과 Chip5부터는 제대로 값이 전달되지 못하였다.) 모식도에서 표시된 것처럼 Chip0~Chip1을 monitor하는 경우 Chip0의 monitor pin을, Chip6~Chip7을 monitor하는 경우 Chip6의 monitor pin을 사용하는 경우 문제가 일시적으로 해결되긴 한다. 하지만 근본적인 해결책은 되지 못한다. Chip에 상관없이 하나의 pin에서만 monitor를 하려면 이 문제가 해결되어야 할 것이다.

6. Further task

앞선 논의로부터 DAC board가 정상적으로 작동되기 위해서는 먼저 pin 25, 26의 hardware적인 결함이 발견되었으므로 이 문제가 해결되어야 한다. 또한 모식도에서 푸른색 영역을 제외한 나머지 부분의 동작이 의심되므로 이 부분에 대한 hardware적인 확인도 필요하다고 생각된다. 마지막으로 monitor pin chain이 제대로 동작하지 못하고 있는데, 이에 대한 확인작업 역시 필요할 것이다. DAC board에 hardware적인 문제가 없다면 Verilog code에 문제가 있다고 판단하고 FPGA 쪽을 수정하는 작업이 진행되어야 할 것이다.

(FPGA board, DAC board의 더 자세한 manual과 spec은 첨부된 file들을 참고하자)