MVT on FPGA Project Notes

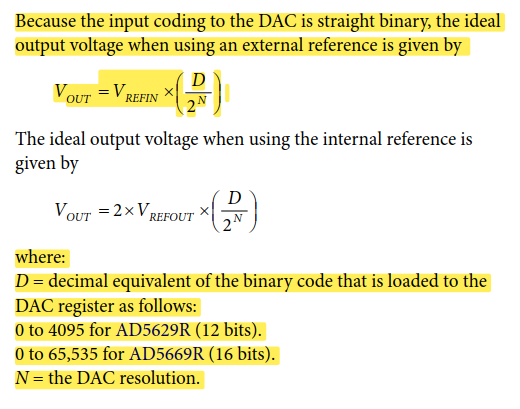
# LVDS Comparators Calibration

In order to calibrate the LVDS comparators. The following steps need to be followed:

1. Set up DAC channels to known theoretical values.
2. Measure and confirm the values are exactly as calculated.
3. Write some firmware that can measure the bias voltage of each of the comparators.
4. Apply a calibration triangle wave as well as a sync pulse to measure the bias voltages.
5. Adjust the values of the DAC to compensate for the bias.

## DAC Channel value calculations

The DACs receive 16-bit data values that determine the voltages. A external 1.8 V reference is used that comes from the MPSoC DevKit. From datasheet:

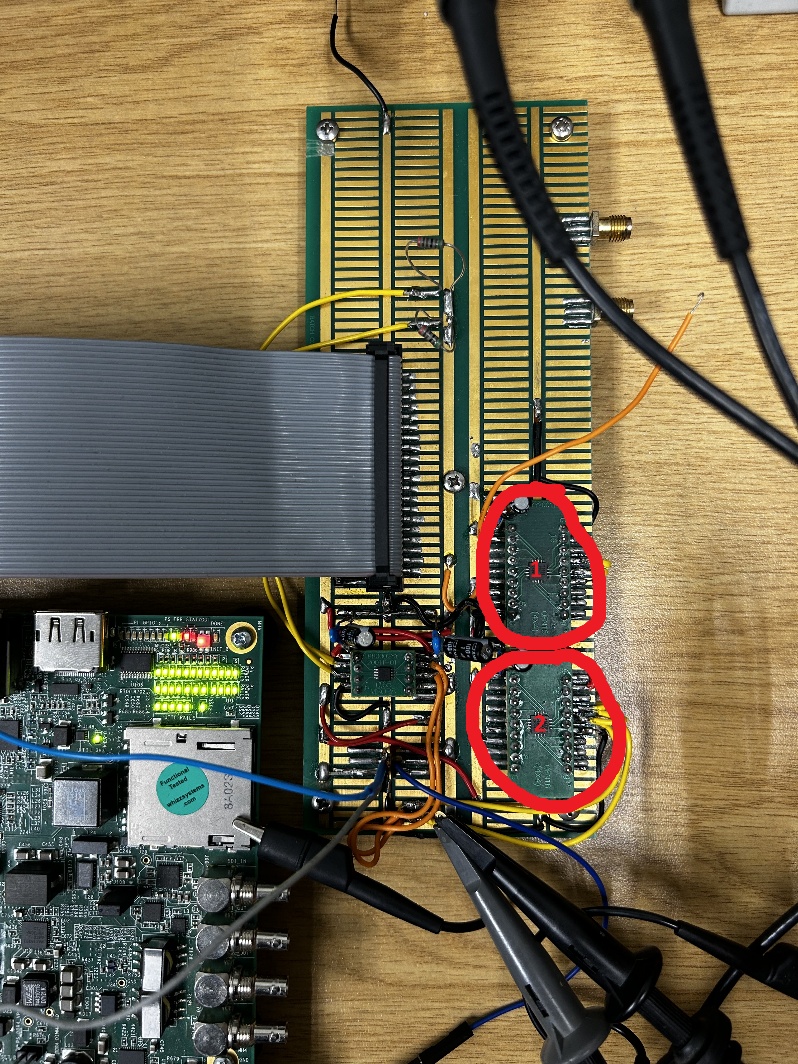


The following table show some theoretical values that the DAC will produce, given the corresponding hex value that is written to the DAC through the firmware:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Vref\_in** | **Output voltage** | **Decimal Value** | **Binary Value** | **Hex Value** | **Measured Value** |
| 1.80 | 0.00 | 0.00 | 0000000000000000 | 0000 | 0 |
| 1.80 | 0.10 | 3641.00 | 0000111000111001 | 0E39 | 0.1 |
| 1.80 | 0.20 | 7282.00 | 0001110001110010 | 1C72 | 0.2 |
| 1.80 | 0.30 | 10923.00 | 0010101010101011 | 2AAB | 0.3 |
| 1.80 | 0.40 | 14564.00 | 0011100011100100 | 38E4 | 0.399 |
| 1.80 | 0.50 | 18204.00 | 0100011100011100 | 471C | 0.499 |
| 1.80 | 0.60 | 21845.00 | 0101010101010101 | 5555 | 0.599 |
| 1.80 | 0.70 | 25486.00 | 0110001110001110 | 638E | 0.699 |
| 1.80 | 0.80 | 29127.00 | 0111000111000111 | 71C7 | 0.799 |
| 1.80 | 0.90 | 32768.00 | 1000000000000000 | 8000 | 0.898 |
| 1.80 | 1.00 | 36409.00 | 1000111000111001 | 8E39 | 0.998 |
| 1.80 | 1.10 | 40050.00 | 1001110001110010 | 9C72 | 1.098 |
| 1.80 | 1.20 | 43691.00 | 1010101010101011 | AAAB | 1.198 |
| 1.80 | 1.30 | 47332.00 | 1011100011100100 | B8E4 | 1.297 |
| 1.80 | 1.40 | 50972.00 | 1100011100011100 | C71C | 1.397 |
| 1.80 | 1.50 | 54613.00 | 1101010101010101 | D555 | 1.497 |
| 1.80 | 1.60 | 58254.00 | 1110001110001110 | E38E | 1.597 |
| 1.80 | 1.70 | 61895.00 | 1111000111000111 | F1C7 | 1.697 |
| 1.80 | 1.80 | 65535.00 | 1111111111111111 | FFFF | 1.797 |

## Project Setup

There are now two DACs on the prototyping board:



The top DAC (1) has its address bit (pin 16) not connected. According to datasheet this makes the DAC’s address 10101**10**0 or xAC.

The bottom DAC (2) pin 16 is tied to 3V3. According to datasheet, the address is therefore 10101**00**0 or xA8.

Both DACs have the external 1V8 rail connected as voltage reference, so there is no need to set-up the internal voltage reference (by default the device uses the external reference).

Also, I recently added (2025-09-18) a setting in the firmware that allows user to talk to different DACs, depending on a specific bit. The control word is built as follows:

|  |  |
| --- | --- |
| Bits | Description |
| 31 | Trigger bit. When a 1 is read here after a previous 0, this transaction is sent through to DAC. I.e., first write all zeros to AXI GPIO, then write the actual value. |
| 30 | Write/Read. Write : 0, Read : 1 |
| 29:25 | Reserved |
| 24 | DAC ID. Top DAC (1) : 0, Bottom DAC (2) : 1. The addresses are hardcoded in the firmware |
| 23:16 | DAC Command |
| 15:8 | Upper data byte |
| 7:0 | Lower data byte |

Currently the commands to set up any value are as follows (Example is of the top DAC 1):

1. First create all axi transactions:
   1. create\_hw\_axi\_txn wr\_zeros [get\_hw\_axis hw\_axi\_1] -address 80020000 -data 00000000 -type write
   2. create\_hw\_axi\_txn wr\_enable [get\_hw\_axis hw\_axi\_1] -address 80020000 -data 804f0000 -type write
   3. create\_hw\_axi\_txn wr\_dac\_chA [get\_hw\_axis hw\_axi\_1] -address 80020000 -data 80300E39 -type write
2. Now run the transactions one by one:
   1. run\_hw\_axi wr\_zeros
   2. run\_hw\_axi wr\_enable
   3. run\_hw\_axi wr\_zeros
   4. run\_hw\_axi wr\_dac\_chA

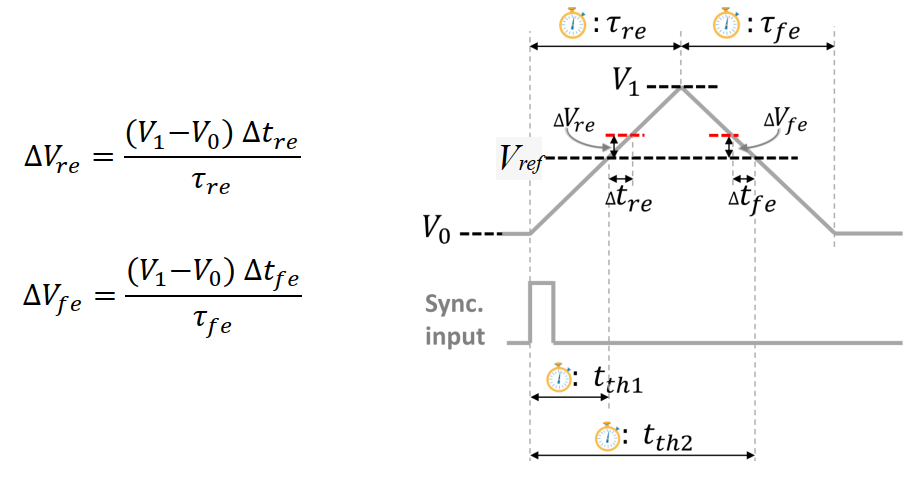
## Proposed calibration methods

There are two proposed methods of calculating the bias voltages of the comparators:

1. Sync pulse plus triangle wave
2. Calibration by metastability

### Method 1 – Triangle wave calibration

The sync pulse plus triangle method requires a signal generator with two outputs. One output provides a sync pulse, and the second a slow-moving triangle wave. A firmware block is used to start a timer when the sync pulse is received. The triangle wave input is placed on the non-inverting input of one of the LVDS comparators. A known reference voltage from the DAC is placed on the inverting input of the same comparator. The firmware block measures the timestamp when the LVDS input transitions to a rising edge, as well as on the falling edge. With the timestamp known, the bias voltage can be calculated. The below image illustrates this principle.



### Method 1 – Results

I have managed to find a pulse generator that can generate a sync pulse as well as a semi-triangular pulse (HP 8112A). The second pulse is actually just a normal pulse with a long slope. I have set the pulse generator to produce a pulse with leading edge of 50 ms. Measuring this with the scope (Tektronix DPO4054), it looks like it is closer to 60 ms, but both the scope and the generator are long-past calibration, so I am not sure.

To do:

1. Solder all the DAC outputs to the corresponding n terminals of the LVDS pairs.
2. Fork the incoming analog signal and solder to each p terminal of the LVDS pairs.
3. Consider adding the bias resistor to pull the analog input voltage up to Vbias (also create a quiet Vbias voltage)
4. Apply reference voltages to each LVDS n-terminal, using the DAC outputs.
5. Apply triangle wave and read the value of each p-terminal transition using the calibration firmware block (counter values). Perform this step multiple times so that a cloud of points can be gathered in order to find the most accurate approximation of the bias voltage.
6. Apply the bias voltage found to each LVDS n-terminal using new values for the DAC outputs.
7. Perform step 5 again multiple times to gather cloud of points and confirm that bias has been compensated for.

I am now trying to prepare a data structure in the firmware that will be capable of storing lots of measurements (timestamps of when the triangle wave triggers a rising edge). I am trying to get a block memory generator IP core to generate some BRAM for me, and writing incrementing values to it to confirm that it works. I am reading the values out through the tcl console.

The idea is to create a pulse train with the pulse generator that can trigger the firmware multiple times, and each triangle wave measurement will be stored, so that an accurate measurement of the bias voltage can be derived.

The BRAM is working well. I managed to stimulate it with a process that increments a simple 32 bit counter and loads each value into the BRAM one by one. I managed to read the data from the BRAM as follows from tcl console:

* create\_hw\_axi\_txn rd\_dac\_bram [get\_hw\_axis hw\_axi\_1] -address 84000000 -type read -len 128
* run\_hw\_axi rd\_dac\_bram

The data output is one long string of 128 32-bit entries (128X8 = 1024 hex string). I use a python script to split this data into 128 separate 8 character hex words. This data I copy into Excel and use that to calculate the rest:

Formula to calculate the error in voltage (mV):

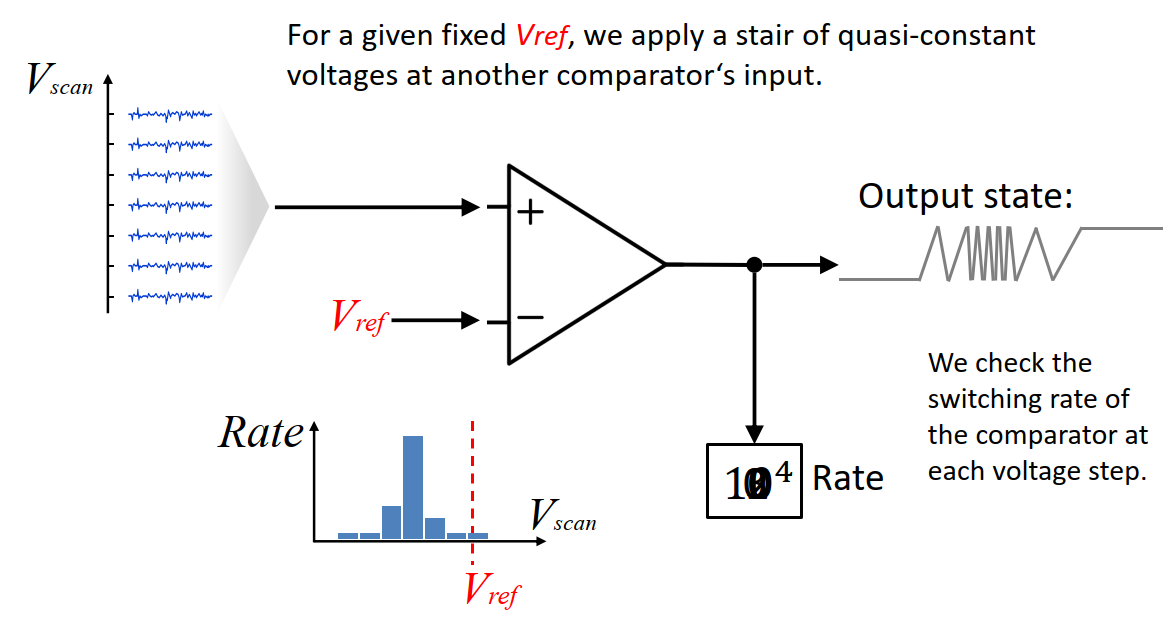
=($Z$4 - HEX2DEC(A3)) \*0.000000005\*1.8\*1000/0.0642

Where

* $Z$4 is the fixed theoretical decimal counter value
* A3 is the hex data read from the BRAM. It gets converted to decimal
* The difference is multiplied by 5 nanoseconds for the clock period used for the counter
* Multiplied by slope ratio used to stimulate the positive terminal (1.8V per 0.0642ms)
* Multiplied by 1000 to give answer in millivolts

### Method 2 – Calibration by Metastability

The second method, calibration by metastability, does not require a signal generator. The principle of metastability is used to measure the bias voltage of the comparators. Once again a reference voltage is applied at the inverting input of a comparator, but this time a quasi-constant voltage is applied at the non-inverting input of the comparator. For each step in the secondary voltage, the switching rate is measured on the FPGA. The voltage that produces the highest switching rate can be used to calculate the bias voltage. The image below illustrates this principle.



# MVT IP core

I am considering writing my own IP core to perform the MVT technique

Have to write report here about how I did the IP core, but it looks like it worked!

##################################

##################################

##################################

##################################

##################################

The following plot was generated by using the following voltages for the DACs:

0 – 0.1V ; 1 – 0.2V ; 2 – 0.3V ; …. ; 15 – 1.5V ; 16 – 1.6 V

This causes there to be 17 levels, and therefore the top comparator is meaningless, since we care only about 4 bits. A 17th level would need 5 bits to represent. So instead I will only use 15 comparators and split their values evenly across the full range.

I have set up an generated sine wave signal on the AFG port of the oscilloscope (now using Tektronix MDO3024). When measuring, the sine wave max is 1.73V and min is 0V. So I will use these values to split into 16 levels:

1.73V / 16 = 0.108125V per level. New levels are as follows:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Vref\_in** | **Output voltage** | **Decimal Value** | **Binary Value** |  | **Hex Value** | **Measured Value** |
| 1.80 | 0.000000 | 0.00 | 0000000000000000 |  | 0000 |  |
| 1.80 | 0.108125 | 3937.00 | 0000111101100001 |  | 0F61 |  |
| 1.80 | 0.216250 | 7873.00 | 0001111011000001 |  | 1EC1 |  |
| 1.80 | 0.324375 | 11810.00 | 0010111000100010 |  | 2E22 |  |
| 1.80 | 0.432500 | 15747.00 | 0011110110000011 |  | 3D83 |  |
| 1.80 | 0.540625 | 19684.00 | 0100110011100100 |  | 4CE4 |  |
| 1.80 | 0.648750 | 23620.00 | 0101110001000100 |  | 5C44 |  |
| 1.80 | 0.756875 | 27557.00 | 0110101110100101 |  | 6BA5 |  |
| 1.80 | 0.865000 | 31494.00 | 0111101100000110 |  | 7B06 |  |
| 1.80 | 0.973125 | 35430.00 | 1000101001100110 |  | 8A66 |  |
| 1.80 | 1.081250 | 39367.00 | 1001100111000111 |  | 99C7 |  |
| 1.80 | 1.189375 | 43304.00 | 1010100100101000 |  | A928 |  |
| 1.80 | 1.297500 | 47241.00 | 1011100010001001 |  | B889 |  |
| 1.80 | 1.405625 | 51177.00 | 1100011111101001 |  | C7E9 |  |
| 1.80 | 1.513750 | 55114.00 | 1101011101001010 |  | D74A |  |
| 1.80 | 1.621875 | 59051.00 | 1110011010101011 |  | E6AB |  |