Generic FPGA board version 3

From Quantum Optics Wiki

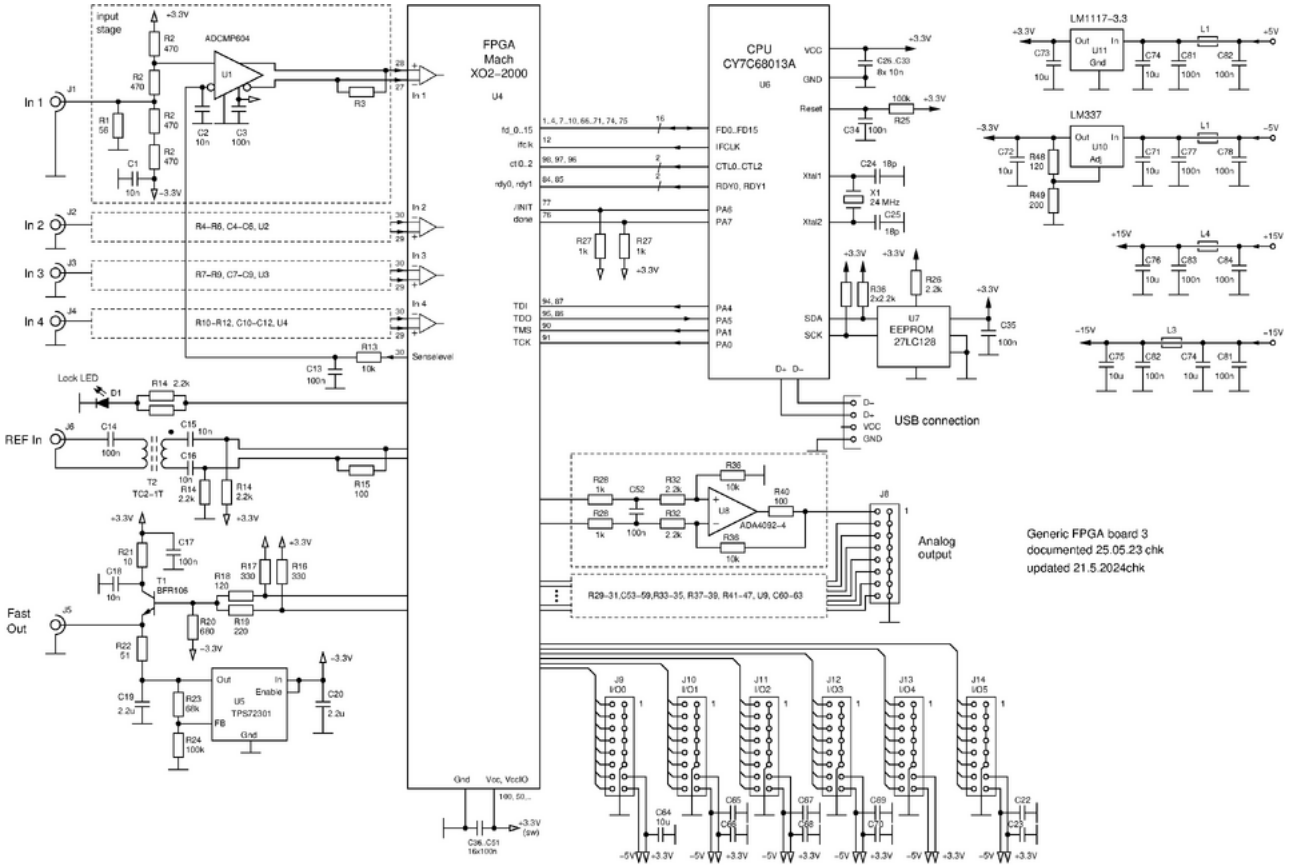
This board is an extension of the Generic FPGA board with NIM in/out, clock reference, digital I/O, in that it provides more digital I/O lines (6x8+1), additional 8 analog outputs, and more precise input discriminators. The footprint of the FPGA is larger, and thus can hold more complex code, but the front plate interface remains the same.

Contents

Hardware

Circuit diagram

The circuit diagram of this board is shown below(pdf version here). Note that the diagram is currently outdated as the wiki is still not yet correctly configured to allow editing images.



Core device: The core element is an Mach XO2 FPGA with 2000 or more gate blocks. The default desgin works with a XO2-2000HC-6 device. It features 48 digital output lines grouped in 6 blocks of 8 bits, which can be connected to an 9-fold NIM output board, or an 8-fold TTL output board. (to be documented).

Fast output port: Additionally, there is one on-board output line that can be changed from TTL level to NIM level compliance. This output has a relatively fast rise/fall time (<1ns), and can drive a 50 Ohm load. It can drive either 0V, -1V, or +1.5V (TBC), and allows for dual polarity outputs.

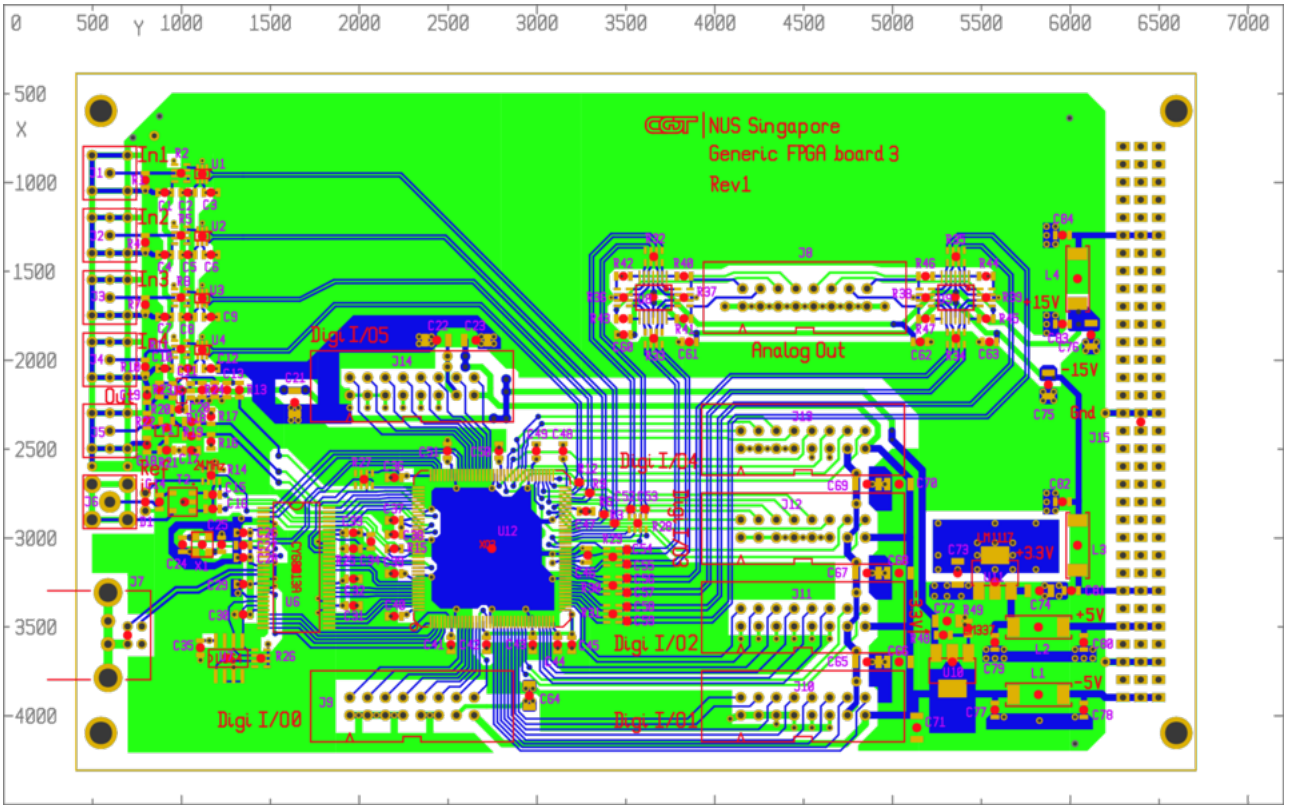
Input lines: There are also four input lines, which have a fast discriminator with adjustable voltage. All input lines are terminated with 50 Ohm. They can be used to read in signals from detectors or other devices. The whole bank of 4 inputs has a single common threshold voltage, which can be programmed from the FPGA. With the voltage shifter, the full range for threshold voltages should be roughly -3.3V...+3.3V (minus headroom of the output stages) for a Sigma-Delta DAC in the FPGA.

Analog outputs: The device features 8 analog output lines, which are intended to be driven by a Sigma-Delta DAC in the FPGA, with a subsequent low pass filter and gain amplifier that should bring up the output voltage to a range of ±10.15 Volt. The reconstruction filter of the gain chain has a nominal first order filter time constant of 140 usec (can be lowered to 14 usec with a 10nF capacitor). Note that these outputs are not designed to have a high precision, as the output voltage depends on the gain resistor tolerances, and its value is directly derived from the +3.3V supply of the digital circuitry. If you need any absolute stability, or have specific noise requirements, consider using a dedicated DAC with a proper analog voltage part.

Reference clock input: The device has a reference clock input to provide a nominal 10 MHz system clock to the device. Any standard 10 MHz signal can be used, as long as it has a sufficiently large slope. A rectangular signal would work best, but a low-noise sine signal with an amplitude of 1V would work as well. The clock input is transformer-coupled for clock noise suppression and has a termination impedance of effective 50 Ohm at 10 MHz, but no DC path.

Board layout

The outer two layers and the component locations are shown below:



The component-side inner layer is for ground only, the bottom-side inner layer carries mostly a +3.3V supply.

Useful daughter boards

This FPGA board comes with a number of common I/O boards to drive or receive signals from 50 Ohm lines, and are conneced with ribbon cables with a standardized pinout (description to come).

- 8-fold NIM output board - converts the CMOS levels from the FPGA into negative NIM levels for 50Ohm coaxial cables.
- 8-fold TTL output board - contains a line driver and 8 BNC connections such that TTL-like signals can be sent via cables with an impedance of 50 Ohm.
- Analog output board - simply a connection to the 8 analog outputs to BNC connectors.
- Lock-in front end board
- 8-fold TTLinput board for more branches

Implemented devices: Pattern generator V3

This is essentially an extension of the Pattern generator V2 code of a state engine based on a small RAM section in the FPGA, with external inputs, internal state variables and some counting mechanism. The extension features more digital output lines, and some analog output lines.

As compared to the previous version, the increased size of the FPGA allows for more I/O lines, but as a consequence, the internal delays of the device led to a slow-down of the shortedt duration of decision states from 10ns to 20ns. As of svn-6, this should have been fixed.

Function overview

Outputs The device offers a number of digital outputs via the extension connectors, and the same boards for TTL out or NIM out drivers as in the old pattern generator can be used, leading to 48 output lines. An additional output line can be addressed via the single NIM or TTL output that is fixed on the board, and does not require an extension board.

Table The code implements a lookup table with 8x16 bit wide words for each row; the first three words of each row contain the output line data, the fourth currently only the extra output bit, words 5 and 6 are used to fill the DACS, word 7 contains the wait states of this row in clock cycles, and word 8 either the address of the next row, or a number of conditional jumps that decide either to jump to the specified row, or continue with the row located directly after the current row.

Clock The default clock to step through the table runs at 100 MHz, generated by a PLL from a 30 MHz interface clock provided by the FX2 chip. If an external reference clock of 10+/-1 MHz is provided, the 100 MHz internal clock is locked to this reference clock. Alternatively, the reference clock itself can be used as the master clock for the table device.

Decision input lines The four digital inputs on the card can be used as decision flags for conditional jumps in the table, or to decrement a counter for each line. These counters can be set to a predefined value, and their result can be used as criteria for conditional jumps. This allows to branch to different segments in the table if a certain number of events has been detected.

Internal variables Additionally, there are four similar counters that can be decremented by special commands in the table entry, permitting simple repetitive loops, or long delays with very few table entries.

Analog output lines The eight analog output lines are driven by Sigma-Delta DACS with a 16 bit resolution, and the gain stage translates this into a voltage range of approximately ±10V. The DAC lines are supposed to be semi-static, and can be set by writing into parameter registers 10..17. This is the default behaviour when the config register bits 12:11 (`DACconfig` bits) contain both 0. However, the DACs can also receive values from the pattern table by colums with offset 4 and 5. Here, the column with offset 4 (ramdaclines[15:0]) contains the 16 bit value to be written into a DAC register, and the lowest 8 bits in column 5 (ramdaclines[23:16]) contain a bit field which of the 8 DAC registers will be written with the word in column 4. For this to work, the DAC writing feature has to be enabled by the `DACconfig` bits in the configuration register. This rather tedious enabling sequence is supposed to prevent accidental generation of voltages from rouge patterns. See configuration register description for details.

To communicate the table state to a controlling code, four lines can be queried via the status request without interrupting the pattern generator. In the other direction, two configuration bits can be used as decision flags for conditional jumps, so the pattern does not have to be stopped to switch patterns.

Currently, the device has 512 table rows.

Firmware

The firmware for the FX2 can be found at http://qolah.org/FX2_firmware/pattgen3 - it emulates a virtual serial device, and understands currently the following commands:

```
*IDN?     Returns device identifier
*RST      Resets device
STATUS?   returns the content of the status register in the FPGA
CONFIG  <value>:
          sets the configuration register for the device. The parameter
          is a valid 16 bit integer.
```

```
CONFIG?   returns the last value written to the config register.
WRITEW <v1> [<v2> [<v3> [...]]]:
          writes a sequence of up to data words either into the lookup
          table or the parameter registers. Set the config register to
          choose destination and to reset the write address counter.
HOLDADR   freezes the table address to the startaddress
PARAM <p0> [<p1> [ <p2> [...]]]
          write one or more parameters into registers. The sequence is
          p0: startad, p1..p4: extcntpreload, p5..p8: intcntpreload
RAMPROG   switches from parameter writing to pattern transfer
RUN       releases the holdaddress mode
HOOKS <value>
          sets the value of the two hook bits (range: 0...3)
HOOKS?    queries the last setting of hooks
TSTAT?    extracts the readback pattern in outline[31:28] from the status word
INSTAT?   extracts the status of the input lines from the status word.
TTL       sets the input sensitivity to TTL level (positive polarity)
NIM       sets the input sensitivity to NIM level (negative polarity)
CLOCKSEL <value>
          sets the clock source. 0: 100MHz autoref, 1: 100MHz ext ref
          2: 100MHz internal ref, 3: External clock directly
CLOCKSEL? returns the current clock select value
# <text>  comment; text is ignored until the next statement
HELP      Print this help text.
```

Values can be entered either as decimal values, or as hexadecimal values if they are preceeded with an `0x` identifier. For example, a bit pattern of `0xaaaa` is an alternating bit pattern with bit 15 set. Its decimal representation would be 43690.

FPGA code

Communication with the FPGA and the FX2 takes place via a configuration register (FX2->FPGA), a status register (FPGA->FX2), a parameter register bank (FX2->FPGA), and a pattern RAM (FX2->FPGA). Data read and write takes place in 16 bit wide words. To accommodate for more ouptut lines, the RAM structure had to be modified compared to the previous pattern generator.

Configuration register

This register determines the overall behavior of the device. It is largely an extension of the configuration register for the smaller pattern generator. Its bits have the following meaning:

<table>
<tr><th>Bits</th><th>what</th><th>interpretation</th></tr>
<tr><td>12:11</td><td>DACconfig</td><td>This determines how the DAC outputs are controlled. See description below how to load DAC data from the parameter RAM.
<table><tr><th>value</th><th>DAC data source</th></tr><tr><td>00</td><td>All DAC lines are taking values from the static DAC registers</td></tr><tr><td>01</td><td>DAC0 gets its value from the pattern RAM, the rest from the dacreg registers</td></tr><tr><td>10</td><td>DAC0..DAC3 get their values from the pattern ram, the rest from dacreg</td></tr><tr><td>11</td><td>DAC0..DAC7 get their values from the pattern ram.</td></tr></table></td></tr>
<tr><td>10</td><td>auxline polarity</td><td>This bit determines if the output delivers. 0: NIM levels, 1: TTL levels</td></tr>
<tr><td>9:8</td><td>tablehooks</td><td>these two bits can be used in conditional jumps in the table entry to redirect the sequence to different branches</td></tr>
<tr><td>7:6</td><td>clockselect</td><td>these two bits determine the master clock selection for the table sequencer according as shown below:
<table><tr><th>value</th><th>clock source</th></tr><tr><td>00</td><td>100 MHz, with automatic reference clock selection (internal, or external reference if its freqency is between 9 and 11 MHz)</td></tr><tr><td>01</td><td>100 MHz, always use external 10 MHz reference clock</td></tr><tr><td>10</td><td>100MHz, always use internal reference clock</td></tr><tr><td>11</td><td>use external clock directly as table master clock</td></tr></table></td></tr>
<tr><td>5:4</td><td>auxlineselect</td><td>Determines the NIM output line on the main board according to the following selection:
<table><tr><th>value</th><th>NIM output</th></tr><tr><td>00</td><td>value of bit 48 of a table entry</td></tr><tr><td>01</td><td>value of bit 48 of a table entry, delayed by half a table master clock cycle</td></tr><tr><td>10</td><td>main table clock</td></tr><tr><td>11</td><td>reference clock input</td></tr></table></td></tr>
<tr><td>3</td><td>parameterwrite</td><td>if set to 1, data is written to parameter register bank. When 0, data is written to the pattern RAM</td></tr>
<tr><td>2</td><td>addressreset</td><td>when set, the table address is loaded with the startaddress register content. Otherwise, the next address is generated from the table entry (direct or conditional jumps</td></tr>
<tr><td>1</td><td>level</td><td>input line polarity. 0: input are negative active, 1: inputs are positive</td></tr>
<tr><td>0</td><td>tablereset</td><td>RAM register reset; not clear if this is needed in conjunction with addressreset</td></tr>
</table>

Status register

This 16 bit wide register reflects the status of the pattern generator; it can be read out at any time, with the following interpretation of the bits:

bits	What	Interpretation
15:12	debuglines	for debugging the device
11	not used	returns 0
10	level	reflects the setting of the input polarity bit (NIM:0 or TTL:1)
9	PLL_lock	internal PLL lock status (i.e., when set, it gives good 100 MHz)
8	CLK_OK	external clock status. Is set if external frequency is between 9 and 11 MHz
7:4	inlines	reflects the status of the four input lines, corrected for the respective discipline (TTL or NIM); 1 means line is active
3:0	patternstatus	reflects the pattern status lines stored in each table entry

Parameter registers

There are a number of 16 bit wide parameter registers which can be set when the `parameterwrite` line in the configuration register is set. The parameters are written consecutively to the write address, which is reset to 0 at a write to the configuration register, and incremented by 1 after each 16 bit word write. The following parameters are currently implemented:

Write adr	Register	Comment
0	tablestart	Start address of the RAM. When the `addressreset` bit in the configuration register is set, the value of this register is loaded in the read address pointer of the lookup table.
1	inthreshold	determines the voltage threshold for the input comparators. This is a signed 16 bit number covering the nominal span from -3.3V to +3.3V.
2...5	extcntload0..3	Reload value for the four counters that get decremented through external events
6...9	intcntload0..3	Reload value for the four internal counters that get decremented through a special command. This can be used to set up loops.
10..17	dacreg0..7	Static values for the Sigma-Delta DAC lines. A signed 16 bit integer value gets mapped onto an output range of -10..+10V.

Inthreshold specificaton: The input threshold is set by a sigma-delta DAC that covers a nominal voltage range from -3.3V to +3.3V for a signed 16 bit number. For example, a threshold of -0.5V is represented by value of $-0.5/3.3*2^{15} = -4965 = 0xEC9B$, a threshold voltage of +1.6V corresponds to a inthreshold value of $1.6/3.3*2^{15} = 15888 = 0x3E10$. Note that the exact values may vary, as there are offsets in the threshold voltage generation chain.

DACreg specification: The analog output lines are implemented with sigma-delta dacs with a 16 bit resolution, covering a output span of approximately -10.3V...+10.3V. The DACs are loaded again with a 16 bit signed number covering that range. A DAC value of 0x7fff corresponds to the largest positive voltage, a DAC value of 0x8000 to the maximal negative voltage.

NOTE: All parameter registers need to be written starting from parameter register 0. So if register 10 is to be set, all registers 0...10 must be written in. Subsequent registers don't need to be written if the old value is ok.

Pattern RAM

This is an internal RAM bank, organized in 512x128 bits for the table replay, and in 4096x16 entries for writing data to it. The content is written consecutively to a write address, which is reset to 0 at a write to the configuration register, and incremented by 1 after each 16 bit word write. The table gets replayed by the internal sequencer. Write address 0...7 correspsond to table row 0, write address 8...15 to row address 1, write address 16..23 to row address 3 and so on. The bits have the following meaning:

Write addr	What	Interpretation
row*8+0	outlines[15:0]	determines the output pattern for 8-bit outputs Port 0 (in outlines[7:0]), and Port 1 (in outlines[15:8])
row*8+1	outlines[31:16]	determines the output pattern for 8-bit outputs Port 2 (in outlines[7:0]), and Port 3 (in outlines[15:8])
row*8+2	outlines[47:32]	determines the output pattern for 8-bit outputs Port 4 (in outlines[7:0]), and Port 5 (in outlines[15:8])
row*8+3	outlines[63:48]	Currently a bit underused. Outline[48] can be connected to the single auxiliary output line on the main board, depending on the setting `auxlinesel` in the configuration register. outlines[63:60] can be read in via the status register to allow determine the current state.
row*8+4	ramdaclines[15:0]	Value to be updated into the DAC lines, if enabled
row*8+5	ramdaclines[31:16]	Transfer mask for updating DACs. When bit 16 is set, DAC0 is updated with the value in ramdaclines[15:0], when bit 17 is set, DAC1 is updated etc. Several DACs can be set to the same value by setting bit values to 1.
row*8+6	timelines	Determines the wait cycles before the sequencer jumps to the next address. A value of 0 means that the next table row is played after 1 master clock cycle, or 10ns for the standard 100 MHz clock. For example, an entry of 9999 corresponds to a switch to the next table row after (9999+1)*10ns = 100µsec for a 100 MHz clock. Note that the largest value is 65535, corresponding to 655.36µs duration for a table row (for a 100MHz master clock).
row*8+7	addresslines	Determines the next row for the table sequencer, or executes a special command (in which case, the row address following the current one is chosen as next row).

Address line interpretation: The choice of the next table entry, and some special events, are determined by the 16 bit wide addressline register in each table row. The bit pattern is interpreted as shown below:

Action	bits 15:12	bits 11:8	bits 7:4	bits 3:0
unconditional jump	0	next row address		
special command	1	decrement internal counter 3:0	load internal counter 3:0	load external counter 3:0
conditional jump on hook 0	2	address of next row entry if tablehook[0] is set (otherwise, the next address is chosen)		
conditional jump on hook 1	3	address of next row entry if tablehook[1] is set (otherwise, the next address is chosen)		
conditional jump on input line 1	4	address of next row entry if inline 1 is set (otherwise, the next address is chosen)		
conditional jump on input line 2	5	address of next row entry if inline 2 is set (otherwise, the next address is chosen)		
conditional jump on input line 3	6	address of next row entry if inline 3 is set (otherwise, the next address is chosen)		
conditional jump on input line 4	7	address of next row entry if inline 4 is set (otherwise, the next address is chosen)		
conditional jump on external counter 1	8	address of next row entry if extcounter 1 is nonzero (otherwise, the next address is chosen)		
conditional jump on external counter 2	9	address of next row entry if extcounter 2 is nonzero (otherwise, the next address is chosen)		
conditional jump on external counter 3	10	address of next row entry if extcounter 3 is nonzero (otherwise, the next address is chosen)		
conditional jump on external counter 4	11	address of next row entry if extcounter 4 is nonzero (otherwise, the next address is chosen)		
conditional jump on internal counter 1	12	address of next row entry if internal counter 1 is nonzero (otherwise, the next address is chosen). **CAUTION: as seen in fpga svn-6, the jump condition is when the internal counter has overreached.** (fixed with fpga svn-7)		
conditional jump on internal counter 2	13	address of next row entry if internal counter 2 is nonzero (otherwise, the next address is chosen) **CAUTION: as seen in fpga svn-6, the jump condition is when the internal counter has overreached.**(fixed with fpga svn-7)		
conditional jump on internal counter 3	14	address of next row entry if internal counter 3 is nonzero (otherwise, the next address is chosen) **CAUTION: as seen in fpga svn-6, the jump condition is when the internal counter has overreached.**(fixed with fpga svn-7)		
conditional jump on internal counter 4	15	address of next row entry if internal counter 4 is nonzero (otherwise, the next address is chosen) **CAUTION: as seen in fpga svn-6, the jump condition is when the internal counter has overreached.**(fixed with fpga svn-7)		

For the special command, the address counter is incremented by one.

Note that for conditional jumps, at least one wait cycle needs to be inserted due to the internal FPGA transit times for FPGA code pre-svn-5. If a decision state as no wait cycles, the branching may not work properly.

Code examples

Some patterns can be found in https://qolah.org/FX2_firmware/pattgen3/patterns.

Pulse test pattern

An example code for a simple pulse sequence is shown below. Note the optional use for hexadecimal notations for patterns now.

```
# Demo to generate 1, 2, 3 and 4 pulse bursts on four channels, one sync
# channel

# Set device to programming mode: reset table, reset RAM, program params
config 13
writew 0, 59000; # basic address is 0
config 4; # switch to RAM write

# This is the RAM sequence
writew 0x11,0,0,1,0,0, 9,1; # channel 1 pulse sync pulse 100nsec
writew 0,0,0,0,0,0, 989,2; # off for 9.9usec

writew 0x22,0,0,0,0,0, 9,3;
writew 0,0,0,0,0,0, 89,4;
writew 0x22,0,0,0,0,0, 9,5; # 2 pulses, total len 1.1usec
writew 0,0,0,0,0,0, 889,6; # pause for 8.9 usec

writew 0x44,0,0,0,0,0, 9,7;
writew 0,0,0,0,0,0, 89,8;
writew 0x44,0,0,0,0,0, 9,9;
writew 0,0,0,0,0,0, 89,10;
writew 0x44,0,0,0,0,0, 9,11; # 3 pulses, 2.1us
writew 0,0,0,0,0,0, 789,12; # wait 77.9 usec go back to 0

writew 0x88,0,0,0,0,0, 9,13;
writew 0,0,0,0,0,0, 89,14;
writew 0x88,0,0,0,0,0, 9,15;
writew 0,0,0,0,0,0, 89,16;
writew 0x88,0,0,0,0,0, 9,17;
writew 0,0,0,0,0,0, 89,18;
writew 0x88,0,0,0,0,0, 9,19; # 4 pulses, 2.1us
writew 0,0,0,0,0,0, 6689,0; # wait 66.9 usec go back to 0

# start pattern
config 0;
```

DAC test pattern

A simple pattern to load the on-board DACs with some values is shown below. This only uses the static registers, not dynamically loaded ones through the table:

```
# Demo to prepare the DACs into a increasing pattern
# This only uses the static DAC patterns, not any table-controlled feature

# Set device to programming mode: reset table, reset RAM, program params
config 13


# write parameter sets
# Tablestart:
```

```
writew 0

# Input threshold:
writew 12000

# counter reload registers - just stay at zero address
writew 0,0,0,0, 0,0,0,0

# write DAC registers
writew 1000,2000,3000,4000,5000,6000,7000,8000
```

Known issues

General issues:

- FPGA pre-svn-5 has DAC selector bits wrong, and output bits swapped
- FPGA svn-5 may have timing issues in decision lines. To play safe, try to insert a wait state into a decision statement
- Some boards (external manufacturing run 1) have wrong resistors for output amplifiers.

- Writing pattern files into the device with `cat file.pattern > /dev/serial/by-id/...` seems to fail sometimes by hanging the device. Temporary fix: pipe text slower into the device, wait 100ms after each line of text... Need to fix this in FX2 firmware (29.3.2025chk, observed in serial 05).

Direct observations: See serial numbers below

Built devices

Currently still testing, but 10 boards have been externally manufactured. The devices come with an USB serial identifier `PG3-QOxx`, where xx stands for the serial index. For the overal serial number, the base code `QOT-0045` is used. So far, the following devices have been built:

Serial	USB identity	FX2 firmware	FPGA code	Comments	Where used
QOT-0045V1-01	PG3-QO01	5	5	first device, hardware/software bugfixes, reflashed 21.5.2024	back to atom group
QOT-0045V1-02	PG3-QO02	5	5	testing 7.9.2023chk, externally manufactured	with Syed
QOT-0045V1-03	PG3-QO03	5	5	tested 13.4.2024, externally manufactured	Wit atom group
QOT-0045V1-04	PG3-QO04	5	5	tested 21.5.2024, externally manufactured	for FHG, new
QOT-0045V1-05	PG3-QO05	5	6	ext manufactured, testing faster DAC 2.7.2024	with CK
QOT-0045V1-06	PG3-QO06	7	6	ext manufactured, A0-A3: faster DAC resistors 6.4.2025	with CK for PC5271
QOT-0045V1-07	PG3-QO07	7	7	ext manufactured, A0-A3 faster DAC, corrected loop FPGA, 12.4.2025	with CK for PC5271 & testing

Back to Electronics