b2tt: Belle II Trigger Timing Core for Frontend Electronics

Mikihiko Nakao, KEK

November 18, 2013

- version 0.04 (2013.10.15) — for b2tt beta SVN release 0.04 (2013.10.13)
- version 0.05 (2013.10.19) — minor update
- version 0.07 (2013.11.18) — b2tt_ddr split into v5 and v6, fifodata changed to 96-bit

1 Introduction

The Belle II Trigger Timing **(b2tt)** receiver core is a set of firmware components that are to be included in the FPGA firmware on the frontend electronics boards of Belle II subdetector readout systems. The frontend board has to be designed according to the Belle II DAQ requirement, to receive the clock **(CLK)** and encoded-trigger **(TRG)** signal and to return the encoded status acknowledgment **(ACK)** signal via a RJ-45 connector. The **b2tt** core takes care of decoding the TRG signal and encoding the ACK signal, and does more useful things that are needed to read out the detector and to ensure the integrity of recorded data.

This document describes the **beta** version of **b2tt** firmware. There was an alpha release of **b2tt** firmware which has been tested on many systems. The beta version is hopelessly incompatible with the alpha version.

2 Target Devices

The target FPGA of the **b2tt** core are Xilinx Virtex-5, Virtex-6 and Spartan-6, and tested on following devices.

CDC readout board — with Virtex 5 (XC5VLX155T-FF1738), both version 3 and version 4 boards, together with Belle2link core

UT3 — universal trigger board 3, with Virtex 6 (XC6VHX380T-FF1923), together with Belle2link core

HSLB — Belle2link receiver with Virtex 5 (XC5VLX50T-FF1136)

SP605 — Xilinx Spartan 6 evaluation board, together with a handmade FMC card to attach a general purpose RJ-45 connector

Since alpha version of **b2tt** core works on ECL collector (Spartan 6), SVD FTB (Spartan 6), PXD DHH (Virtex 6), the beta version core is expected to work on these systems, too.

There is no **b2tt** code tested on 7-series of Xilinx FPGAs, but it is not so difficult to port the firmware. Porting the **b2tt** firmware to an ALTERA device is more challenging, but there is an on-going effort to port it to a Stratix IV FPGA for the SVD FADC board.

3 Target Setup

The target setup of the beta version of **b2tt** core is from a small system with a single frontend timing switch **(FTSW)** module, but it is basically designed to work with the final Belle II system. The beta version should evolve into the final version without drastic change in the protocol and the module structure.

The target FTSW is a **FTSW version 2.1** with **ft2u** firmware. There is a separate document for the usage of **ft2u**. The **ft2u** firmware handles up to 8 frontend readout system plus up to 4 TT-RX cards (i.e. up to 4 COPPER boards). As for now HSLB cards also need to receive the same clock, and they can be taken from any of 8+4 frontend+TT-RX connection ports or additional FTSW module. The **ft2u** firmware can be used to cascade the clock signal only, but different type of firmware is needed to cascade the trigger distribution tree.

The example firmware for HSLB (and similarly for other devices) should work by connecting one of the FTSW output port (odd number of OUT port from 5 to 19) to the RJ-45 connector of the HSLB board with a CAT7 cable. There are 4 LEDs directly connected from **b2tt** output:

LED(0) `b2clkup` to detect the clock from FTSW
LED(1) `b2ttup` to detect the **b2tt** link is established
LED(3:2) `trgtag[1:0]` to show the lowest two bits of the trigger tag.

There are few more LEDs and signals for some other signals.

4 Files

Following files are provided:

- Common files
  b2tt.vhd — top module of **b2tt** firmware
  b2tt_decode.vhd — **b2tt** protocol decoder
  b2tt_encode.vhd — **b2tt** protocol encoder
  b2tt_8b10b.vhd — 8b10b decoder and encoder
  b2tt_symbols.vhd — 8b10b and **b2tt** protocol symbol definitions
- FPGA dependent files
  b2tt_clk_v5.vhd — clock management block for Virtex-5
  b2tt_clk_v6.vhd — clock management block for Virtex-6
  b2tt_clk_s6.vhd — clock management block for Spartan-6
  b2tt_ddr_v5.vhd — double data rate (DDR) I/O handler for Virtex-5
  b2tt_ddr_v6.vhd — double data rate (DDR) I/O handler for Virtex-6
  b2tt_ddr_s6.vhd — double data rate (DDR) I/O handler for Spartan-6
  b2tt_fifo.vhd — FIFO storage of event header data for Virtex-5/-6
  b2tt_fifo_s6.vhd — FIFO storage of event header data for Spartan-6
- HSLB example:
  hslb_b2tt.{vhd,ucf} — top module and UCF file
  hslb_b2tt.{xise,gise} — ISE12.4 project
  hslb_b2tt.{xst,prj,ut},Makefile — command-line based ISE12.4 project
- SP605 example:
  sp605_b2tt.{vhd,ucf} — top module and UCF file
  sp605_b2tt.{xise,gise} — ISE12.4 project
  sp605_b2tt.{xst,prj,ut},Makefile — command-line based ISE12.4 project

5 Code Structure

The **b2tt** source file is organized into following structure.

```
b2tt                      (top module)
 +-- b2tt_clk               (clock management)
 +-- b2tt_fifo              (FIFO management)
 +-- b2tt_decode            (decoder top)
      +-- b2tt_iddr             (DDR receiver: serial => 2-bit)
      +-- b2tt_decomma          (comma/boundary finder)
      +-- b2tt_debit2           (2-bit decoder => 10-bit)
      +-- b2tt_debit10          (10-bit decoder => octet)
           +-- b2tt_de8b10b          (8b10b decoder)
      +-- b2tt_detrig           (trigger code finder)
      +-- b2tt_deoctet          (octet decoder => packet)
      +-- b2tt_depacket         (packet decoder => command)
      +-- b2tt_detag            (trigger tag checker)
 +-- b2tt_encode            (encoder top)
      +-- b2tt_encounter        (counters for encoding)
      +-- b2tt_enpacket         (packet encoder <= various info)
      +-- b2tt_enoctet          (octet encoder <= packet payload)
      +-- b2tt_enbit2           (2-bit encoder <= octet)
           +-- b2tt_en8b10b          (8b10b encoder)
      +-- b2tt_oddr             (DDR driver: serial <= 2-bit)
```

6 b2tt Interface

The **b2tt** firmware has quite a few in and out signals. Here is the list of parameters:

VERSION (integer) version of the**b2tt** firmware, do not change.

DEFADDR (20-bit) default address (for debug), do not change.

FLIPCLK (1-bit) flag to flip the clock polarity. Not needed, do not use.

FLIPTRG (1-bit) flag to flip the TRG signal polarity. `FLIPTRG='1'` when `trgp/trgn` polarity at the RJ-45 connector is flipped due to the board design (default = `'0'`).

FLIPACK (1-bit) flag to flip the ACK signal polarity for the same reason as `FLIPTRG` (default = `'0'`).

USEFIFO (1-bit) flag to whether read event header data FIFO. If `USEFIFO='1'` and `fifonext` is not asserted for every trigger, it will block the trigger distribution after 1020 triggers (default = '1').

CLKDIV1 (integer 1 to 72) division factor for reduced `divclk1`.

CLKDIV2 (integer 1 to 72) division factor for reduced `divclk2`.

USEPLL (1-bit) flag to use FPGA's PLL for `sysclk` (default = '0').

USEICTRL (1-bit) to instantiate `idelayctrl` inside b2tt, only for Virtex-5/-6 (default = '1').

And here is the list of signals:

clkp/clkn : in std_logic (LVDS pair)
LVDS pair for clock input, directly connected to the IO pin pairs for the RJ-45 connector's 7-8 pins.

trgp/trgn : in std_logic (LVDS pair)
LVDS pair for encoded **b2tt** protocol input, directly connected to the IO pin pairs for the RJ-45 connector's 3-6 pins.

rsvp/rsvn : out std_logic (LVCMOS25)
Unused, tied to ground inside **b2tt**, directly connected to the IO pin pairs for the RJ-45 connector's 4-4 pins.

ackp/ackn : out std_logic (LVDS pair)
LVDS pair for encoded **b2tt** protocol output, directly connected to the IO pin pairs for the RJ-45 connector's 1-2 pins.

id : out (16-bit)
Up to 16-bit board id number which should be unique to each board of the same detector subgroup. This is to guarantee the correct cable connection is made. Different subgroup may have the same number.

b2clkup : out std_logic
Flag to tell that every clock management (DCM, PLL, MCMM) is locked inside **b2tt**. With `b2clkup = '0'`, nothing is expected to work, and the `clkp/clkn` signal has to be checked. And even with `b2clkup = '1'`, it does not guarantee that the clock frequency is correctly at 127 MHz.

b2ttup : out std_logic
Flag to tell that the firmware is receiving correct **b2tt** protocol, and is ready to receive various reset signals, trigger signals and clock related signals.

sysclk : out std_logic
System clock of 127 MHz. All other **b2tt** output signals are synchronous to `sysclk`, so do not introduce another 127 MHz clock domain in the design unless it is necessary (e.g., `rx/txusrclk2` of RocketIO is such an avoidable example).

rawclk : out std_logic
Same as `sysclk`, but before feeding into `bufg`, for output through an I/O pad.

utime : out (32-bit)
32-bit time counted in second. One second corresponds to about 127216000 system clocks, where the exact number of clock may be changed through **b2tt** protocol.

ctime : out (27-bit)
Sub-second time counted in clocks. It rounds back to 0 when utime is incremented after 127216000 (or other defined) system clocks. The utime/ctime combination has the same set of values throughout the **b2tt** controlled system for the same `trgout` signal.

divclk1 : out (2-bit)
a reduced clock divided by `CLKDIV1`, generated as a flip-flop output driven by `sysclk`. This is meant to be used together with ODDR which is driven by `sysclk` to a small jitter reduced clock output from FPGA. Using

two bits as the ODDR input makes it possible to generate a reduced clock with a 50% duty cycle from the 127 MHz clock divided by an odd number. If the duty cycle does not matter, the second bit is not necessary. For the use inside the FPGA fabric, one can use `divclk1(0)` directly, or generate a one `sysclk` long pulse out of `divclk1(0)` and use it as an enable signal of a logic driven by `sysclk`. The reduced clock's rising edge is aligned between different frontend boards if `CLKDIV1` can divide 1280×9, as it is synchronized with the `revo` or `revo9` signal.

divclk2 : out (2-bit)
a reduced clock divided by `CLKDIV2`, and the rest is the same as `divclk1`.

exprun : out (32-bit)
10-bit experiment number and 22-bit run number, delivered from FTSW.

runreset : out std_logic
A reset signal to clear counters before starting a run. For every reset signal, there will be no trigger for the next 1279 clock cycles.

feereset : out std_logic
A reset signal for the front-end electronics.

gtpreset : out std_logic
A reset signal for the GTP RocketIO component.

trgout : out std_logic
A one-clock long trigger signal.

trgtyp : out (4-bit)
A 4-bit trigger type, which is available at the same time as `trgout`. The trigger type is defined somewhere else.

trgtag : out (32-bit)
A 32-bit trigger number. It is counted inside the **b2tt** firmware, and verified at every 1280 `sysclk` cycle with the trigger number counted at the FTSW. The first trigger will be a begin-run trigger with a trigger number 0, and the nominal trigger will start from a trigger number 1.

revo : out std_logic
A revolution signal, which is set to '1' at every 1280 `sysclk` cycle. Note that it will be arbitrary different from the bunch 0 position from the accelerator.

revo9 : out std_logic
A 9th revolution signal, which is set to '1' at every 1280×9 `sysclk` cycle. This signal is needed to synchronize a reduced clock with a division by 3, 9 or 72.

revogap : out std_logic
Place holder for a possible signal to tell where the beam bunch is not filled.

injveto : out (2-bit)
Place holder for the injection signal.

busy : in std_logic
An optional busy signal if the frontend electronics can decide to suspend the trigger well before the trigger decision. For example, if there is a FIFO for the event data, an "almost-full" signal may be used to generate the `busy` signal, but using a "full" signal is too late. Needed

time to transfer back the `busy` signal and the amount of maximum expected data size in this period have to be rigorously calculated, in order to avoid generating an unnecessary dead time.

err : in std_logic;
An error signal which will not recover until `runreset` is received. A fatal condition such as an unrecoverable FIFO full or inconsistency in some of the data / signals can be reported. An inconsistency is expected to happen by an SEU under a radiation controlled area.

b2plllk : in std_logic
A PLL lock signal from Belle2link, monitored by the **b2tt** link.

b2linkup : in std_logic
A link-up signal from Belle2link, monitored by the **b2tt** link.

b2linkwe : in std_logic
A copy of write-enable signal to Belle2link, monitored by the **b2tt** link.

b2lclk : in std_logic
A copy of the clock that is used to drive the Belle2link. If `sysclk` is used (recommended), `sysclk` should be provided here.

seuinit : in std_logic
seubusy : in std_logic
seuactiv : in std_logic
seuscan : in std_logic
seudet : in std_logic
seucrc : in std_logic
seumbe : in std_logic
Initialization, busy, active, scan, SEU detection, CRC error detection and multi-bit error detection signals from Virtex 5 SEU mitigation code (XAPP864). This and other SEU related signals are directly connected to the `virtex5_seu_controller` component, and it may need a redefinition for FPGAs other than Virtex-5.

fifordy : out std_logic
`fifodata` is ready to read. This signal is raised 4 to 8 `sysclk` cycles after `trgout`. It is not necessary to monitor this signal, unless the fifodata has to be immediately read out.

fifodata : out (96-bit)
96-bit information to be attached to the event header, available when `fifordy = '1'`. Data contains:

fifoerr (bit 95) should be '0' for a valid `fifodata`,

ctime (bit 94:68) sub-second event time counted in clock

trgtyp (bit 67:64) trigger type of the event

trgtag (bit 63:32) lowest 16-bit of trigger number of the event

utime (bit 31:0) lowest 16-bit of event time counted in second

fifonext : in std_logic
Signal to switch `fifodata` to the next event.

octet : out (8-bit)
isk : out std_logic
Raw octet data and K-symbol flag, which may be used in a front-end firmware if further decoding of the trigger signal has to be made somewhere else.

sigbit2 : out (2-bit)

bitddr : out std_logic
revoclk : out (11-bit)
cntbit2 : out (3-bit)
regdbg : in (8-bit)
dbg : out (32-bit)
dbg2 : out (32-bit)
Debug information, not for users.

7 Missing Items

There are still missing items from this beta version of **b2tt**, some of which are already described above. Here is the most likely incomplete list of items to be implemented before the final version.

Altera support — in particular, for SVD FADC board

Kintex-7, Artix-7 support —

Proper reset sequence definition — especially the way to properly reset the Belle2link firmware is not established yet.

Point-to-point communication — between an FTSW and a front-end board is not possible yet, as FTSW just distributes the packets. Some address based point-to-point communication method is implemented in the **b2tt** core, but as FTSW is not ready yet, it is not tested.

Run start sequence definition — to start a run within the fast **b2tt** system only, without including slow network devices.

Fix ctime rounding bug — because the length of 1 second is not transmitted yet to the frontend yet, there is the ctime/utime mismatch between the recorded data and FTSW at the beginning of a second. It currently happens at a probability of about 1/10,000.

Extending fifodata to 96 bit? — as 16-bit trigger number is not enough at various places, `fifodata` may need to be enlarged to add one more 32-bit word.

SEU detection code — for Virtex-5 is just copied from existing example, and not tested yet (it looks like not working yet). Also the Spartan-6 and Virtex-6 cases should be surveyed and implemented.

revogap, injveto signals — are yet to be implemented.

SFP module monitoring — based on I2C using three `modedef` lines to be added.