Outline

- Requirement of documentation

- Work overview
  - Status
  - Block by block
  - For user and vhdl expert

- Difficulty and to be done
Requirement

- uTCA firmware documentation shows not only the **functionality** but also the **detailed information** about vhdl implementation
  - ports, signal assignment, RTL design
  - slices (LUT, shift register, mux)
  - fpga cores (DSP, pcie, gtx)
- Xilinx automatic generated RTL schematic
  - non-human-readable
- vhdl code in **illustration** with **document**
- for **vhdl expert** and **non-vhdl** user, BUT
  - manual work
  - time consuming
  - Schematic not automatic changeable 😞
Top-level of SiS8300 firmware documentation

81.25 MHz
from DWC
cav1 ADC Z⁻¹ delay sliding window input rot.
cav2 ADC Z⁻¹ drift calibr.
cav8 ADC Z⁻¹ 8pi/9 filter incl. in PVS

9.028 MHz
FIFO to uTC
cavity limiters

Board Management
attenuation
Memory Controller

attenuation
switch
PCle module

FPGA Memory
MUX
devbug mode
to server
to DWC
Chapter 1

Trigger detection

This chapter is devoted to the firmware implementation of trigger detection.

1.1 Trigger detection

Trigger detection consists of 2 blocks respectively in board part (Fig. 1.1) and app part (Fig. 1.2). In board part the 8 bit trigger line is from the AMC backplane and delivered to app part, the value of WORD_TRG_CNT is incremented at each incoming trigger. In app part the trigger signal is either detected as the trigger delivered from the board part or generated internally, depending on the setting of registers: WORD_TIMING_INT_ENA and WORD_TIMING_FREQ. The value of WORD_TIMING_CNT is incremented at each trigger. One of the 8 triggers is selected as the main trigger according to WORD_TIMING_TRG_SEL.

BLK_TRG_EVENTS

For 8 bits input trigger LTRG, the block BLK_TRG_EVENTS generates the same component 8 times, each with different index $i$ for the signals. 8 bit LTRG is delivered to O_TRG as a output to app part. The rising edge of each trigger is captured by 2 D-latch and 1 and-gate, used to enable the counter and increments TRG_CNT[$i$], which corresponds to one 32 bit word in the 8x32 bit array WORD_TRG_CNT, see Fig. 1.1.

BLK_TIMING

There are 4 registers for this block, 3 used as setting and 1 as indicator for the counters. The data format are shown in Fig. 1.2. Each bit of the register WORD_TIMING_INT_ENA decides respectively the mode of trigger detection for corresponding TRG[$i$], see Table 1.1. In external mode, the input trigger LTRG[$i$] from board part is used for trigger detection. For internal mode, the trigger TRG[$i$] is generated internally with the frequency set.
Interface for non-vhdl user

Table 1.1: WORD_TIMING_INT_ENA

<table>
<thead>
<tr>
<th>INT_ENA[i]</th>
<th>TRG[i]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>external mode</td>
</tr>
<tr>
<td>1</td>
<td>internal mode</td>
</tr>
</tbody>
</table>
Description of vhdl code with logic element

- **RTL (Register-transfer level) design**

  ```vhdl
  D <= not Q;
  process(clk)
  begin
    if rising_edge(clk) then
      Q <= D;
    end if;
  end process;
  ```

- **State machine**

  ![State machine diagram]

  - Xilinx automatic generated schematic
  - Manual schematic
VHDL Example: IIR filter order 2

```vhdl
-- vhdl code
```

Bin Yang | Review of Firmware Documentation | 13.05.2014

---

---

```vhdl
---
```

---

---

```vhdl
```
Xilinx automatic generated schematic: IIR filter order 2
Manual Schematic: IIR filter order 2

I_CLK
I_RESET
I_DATA
I_DRY
I_A1
I_A2
I_B0
I_B1
I_B2

Manual Schematic:

I_DATA: 18 bits
O_DATA: 18 bits
I_A1: 18 bits
I_A2: 18 bits
I_B0: 18 bits
I_B1: 18 bits
I_B2: 18 bits

Latency:
O_DONE: 6 x I_CLK

DSP48E
Difficult and to be done

> Massive manual work, time consuming
  - Parallel development, documentation has to be postponed
  - Schematic not automatically changeable to the firmware update

> Work done: application modules
  - Only partially done
  - Relative “stable” version
  - But still a few changes for DAQ, AP and timing module → doc needs update
  - Massive update for DCM have to be done

> Future work:
  - For rest of application
  - Board supporting part
  - More meaningful to do the work once for a stable version tag

> Time to continue the rest work?
Thanks for your attention!
Back up
IQ detection and input rotation

Diagram with signals and components labeled:
- I_RESET
- I_CLK
- I_DATA
- I_SIN
- I_COS
- TAB_IND
- COS_TMP
- SIN_TMP
- GEN_SINCOS_TAB_SIZE
- GEN_SAMPLES_PER_VALUE
- GEN_OUTPUT_SHIFT
- GEN_OUTPUT_WIDTH

Annotations:
- I_DATA: GEN_INPUT_WIDTH bits
- I_SIN: GEN_SINCOS_TAB_SIZE x GEN_OUTPUT_WIDTH bits
- I_COS: GEN_SINCOS_TAB_SIZE x GEN_OUTPUT_WIDTH bits
- O_I: GEN_OUTPUT_WIDTH bits
- O_Q: GEN_OUTPUT_WIDTH bits
- SIN_TMP: 2 x GEN_OUTPUT_WIDTH bits
- COS_TMP: 2 x GEN_OUTPUT_WIDTH bits
Drift calibration (1)
Drift calibration (2)

- Sending setting to the DCM (time delays TD1, TD2 and attenuation values)
- Setting DCM attenuators and switches
- Amplitude measurements (averaging)
- IQ detection
- Calculation of the calibration coefficients
- Sending value of reference amplitude to SIS
- Setting switches position
- "Pulse" trigger

Switches position:
- RF
- REF
- RF

Envelope of the input DWC signal:
- Reference
- RF-pulse
Drift calibration (3)
Computation of the partial vector sum

Diagram of the circuit for computing the partial vector sum.

---

Bin Yang | Review of Firmware Documentation | 13.05.2014
Vector sum: Xilinx automatic generated schematic
Data acquisition module

Table 1.1: CON_DAQx_CHANNELS and CON_DAQx_CHANNELS_TAB

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CON_DAQ0_CHANNELS</td>
<td>16</td>
</tr>
<tr>
<td>CON_DAQ0_CHANNELS_TAB</td>
<td>31, 30, 29, 28, 27, 26, 25, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10</td>
</tr>
<tr>
<td>CON_DAQ1_CHANNELS</td>
<td>8</td>
</tr>
<tr>
<td>CON_DAQ1_CHANNELS_TAB</td>
<td>7, 6, 5, 4, 3, 2, 1, 0</td>
</tr>
</tbody>
</table>

\[32 \times CON_{DAQx\_CHANNEL} = 256 \times M\]
\[\Rightarrow CON_{DAQx\_CHANNEL} = 8 \times M\] (1.1)
Amplitude and phase calculation
IIR filter

- $Y[i] = B0 \cdot \text{INSUB}[i] + \text{BSUM}[i - 1]$
- $\text{INSUB}[i] = X[i] - \text{ASUM}[i - 1]$
- $Y[i] = B0 \cdot (X[i] - \text{ASUM}[i - 1]) + \text{BSUM}[i - 1]$

- $\text{ASUM}[i] = A1 \cdot \text{INSUB}[i] + A2 \cdot \text{INSUB}[i - 1]$
- $\text{BSUM}[i] = B1 \cdot \text{INSUB}[i] + B2 \cdot \text{INSUB}[i - 1]$

- $Y[i] = B0 \cdot X[i] + B1 \cdot \text{INSUB}[i - 1] + B2 \cdot \text{INSUB}[i - 2]$
  - $-A1 \cdot (B0 \cdot \text{INSUB}[i - 1]) - A2 \cdot (B0 \cdot \text{INSUB}[i - 2])$