dBBC Project Book

Version: June 2, 2010

Start date: The project started April 2010.
Documentation timeframe: Week 1 is the week of 31.5.2010

Project group:
Gino Tuccari
Amit Bansod
Alan Roy
Jan Wagner

Project goal

Have a standalone FiLa10G system capable of encapsulating data from a VSI bus and sending it over a single 10GbE SR fibre link as UDP/IP packets, with the payload being Mark5B, Mark5C and potentially VDIF 1.0 data frames.

Acronyms

1PPS	One Pulse Per Second
VSI	VLBI Standard Interface – VSI-H - essentially an unidirectional LVDS data bus

Reference material

See project SVN, 'doc' directory. Some of the relevant files are:

Mark5C Data Frame Spec sensimemo13.pdf	MIT Haystack Memo #58, Mark5C and Mark5B
haystackmemo016.pdf	MIT Haystack Memo #16, VSI4 Geo/Astro Modes
VDIF specification Release 1.0 ratified.pdf	www.vlbi.org
fila10g_sch.pdf	Gino Tuccari, Version 1.0 FiLa10G Schematic

Contents

Output of work

The project has the following deliverables:

Done	Title	Assigned	Week
	Firmware and source code for FiLa10G that fulfill the system requirements	Amit	8
	Documentation of the user configuration interface	Amit / Jan	4..8
	Documentation of signals/bits used by PowerPC to configure User Logic	Amit / Jan	4..8
	Final documentation of system design	Amit / Jan	8

System Requirements

Essential requirements that the FiLa10G firmware has to fulfill:

1. Interface for system configuration – serial port
2. Synchronization of internal clock to external VSI 1PPS
3. Input of 32-bit (VSI4 geo mode) and 64-bit (VSI4 astro mode) data from VSI bus
4. Framing of the VSI data into Mark5B data format with 64-bit word size
5. Configurable own 10G IP address and gateway
6. Configurable destination 10G IP address and port
7. Transmit data frames over one 10G link to configured destination IP address
8. Timing requirements: input 16..128MHz, system 150MHz

Interface Requirements for PowerPC-to-UserLogic

The minimum signals that need to be passed between the PowerPC processor and the User Logic are:

1. Real-time clock base second : PowerPC => UserIP 32bit
2. Real-time clock current value : UserIP => PowerPC 32bit
3. Years Since 2000: PowerPC => UserIP min. 8bit
4. System configuration bits: PowerPC => UserIP 32bit (?)
5. System status bits: UserIP => PowerPC 32bit (?)
6. UDP/IP Destination IP address: PowerPC => UserIP 32bit
7. UDP/IP Destination Port: PowerPC => UserIP 16bit
8. Station ID (2-char ASCII): PowerPC => UserIP 16bit

Additional signals may be necessary.

System configuration bits: at least following bits need to be passed from PowerPC to the User Logic: [1bit: Arm for next 1PPS], [1bit: Select Input Format between 32b/64b], [4bit: Select Output Format Mark5B/5C/VDIF], [1bit: Global Reset/Halt], [1bit: Enable Test Vector Generator Mode], …

System status bits: at least the following bits need to be passed from User Logic to the PowerPC: [1PPS Sync Gained], [10G Link#1 Up], …

10GbE User IP: Note that the "own IP", gateway IP and ARP table entries of the 10G UserIP can already be configured from the PowerPC software via the IP's PLB interface and configuration memory area, this doesn't need to be done from the User Logic. Only the Destination IP Address and Port have to be passed from PowerPC to the User Logic…

User IP/VHDL Subcomponent Waypoints

Table of waypoints is below. For implementation requirement details see entries after the table.

Done	Waypoint		Due Week
(X)	VHDL/System System clocking setup, at least two clocks (VSI, onboard osc)	Amit	1
	Module Real time clock: 32-bit Unix seconds counter with arm / trigger / count features	Amit	2
	Module or toplevel Two data input methods: 1) actual physical, 2) internal test	Amit	2
	Module Data framing blocks (implement at least: Mark5B)	Amit	3
	User IP Block OPB/PLB-bus register to pass values between User/PowerPC	Amit	2..3
	VHDL/System Implement all registers needed in the "Interface Requirements" section and pass them to the top-level VHDL	Amit	3
	VHDL/System Use implemented registers to actually configure User Logic	Amit	4..5

1. System clocking setup, at least two clocks:
    - external VSI clock input : ClockDomain#1(VSI) : 16..128 MHz constraint
    - onboard 50MHz DCM'ed to 150MHz : ClockDomain#2(UserLogic, PowerPC)
2. Real time clock: 32-bit Unix seconds counter with arm/trigger/count
    - base second configurable from software / separate register
    - reference clock rate configurable from software / separate register
    - arm and (re)trigger from first 1PPS coming from VSI#1 connector
    - after trigger: count seconds based on reference clock rate
3. Two data inputs methods: physical, test
    - {1PPS,Valid,32bit/64bit} true VSI#1/#2 data into VSI Receive FIFO
    - {1PPS,Valid,32bit/64bit} test vector counter data into VSI Receive FIFO
    - Direct input (geo 32bit and astro 64bit), no reformatting required
4. Data framing block

    1) Read from VSI Receive FIFO using 200MHz clock

2) Add frame headers, use RTC IP/VHDL block for frame timestamps

3) Add sub-second frame sequence counter to headers (reset to 0 at each seconds-crossing)

3) Generate 10G UDP payload in formats

- Mark5B required *Format described in MIT Haystack Memo #58*
- Mark5C currently optional
- VDIF(?) currently optional

5. User IP : OPB/PLB-bus register to pass 32-bit or wider value between User Logic<->PowerPC
   - Can be two unidirectional registers
   - Or one bidirectional register?
6. Implement all registers needed in the "Interface Requirements" section and pass them to the top-level VHDL
7. Use implemented registers to actually configure User Logic

Embedded Software Waypoints

Done	Waypoint		Due Week
	Basic TinySH OS running in hardware over Serial Port	Amit	1
	Add default values for system configuration to TinySH	Amit	3
	Add required new configuration commands to TinySH	Amit	
	Time synchronization		4
	Data format selection		3
	Test vector mode enable/disable		3
	… other?		…8
	Add required new status commands to TinySH	Amit	3

Details:

1. Basic TinySH OS running in hardware over Serial Port
2. Add default values for system configuration to TinySH (10G IP addresses, geo/astro mode, …)
3. Add following new configuration commands to TinySH
   - Initiate time synchronization of User Logic real-time clock to next 1PPS, the Unix timestamp of next 1PPS is entered by user together with the command
   - allow data format selection between 5B/5C/VDIF (even if 5C&VDIF not yet required to be implemented)
   - allow enable/disable of test vector generation mode (default: enabled)
4. Add following new status commands to TinySH
   - command to print the real-time clock value
   - command to show system status bits in user-readable format

Hardware Testing Waypoints

Done	Waypoint		Due Week
	Board running, programmable over JTAG	Gino	1
	Access to basic TinySH over serial console	Amit	1

	Time synchronization to external 1PPS *Checking via: print the live seconds counter on serial console for ~10 seconds*	Amit/Jan/Gino	3
	10G link running in loopback *Checking via: ChipScope capture of received data, verify e.g. timestamps*	Amit/Jan/Gino	1
	10G link runs with direct connection to computer with 10G NIC, data capture verification ok *Checking via: tcpdump, …, verify e.g. timestamps*	Amit/Jan/Gino	3..4
	10G link runs with direct or switch connection to Mark5C, data capture verification ok *Checking via: Mark5C software, record all channels to disk*	Amit/Jan/Gino	3..4

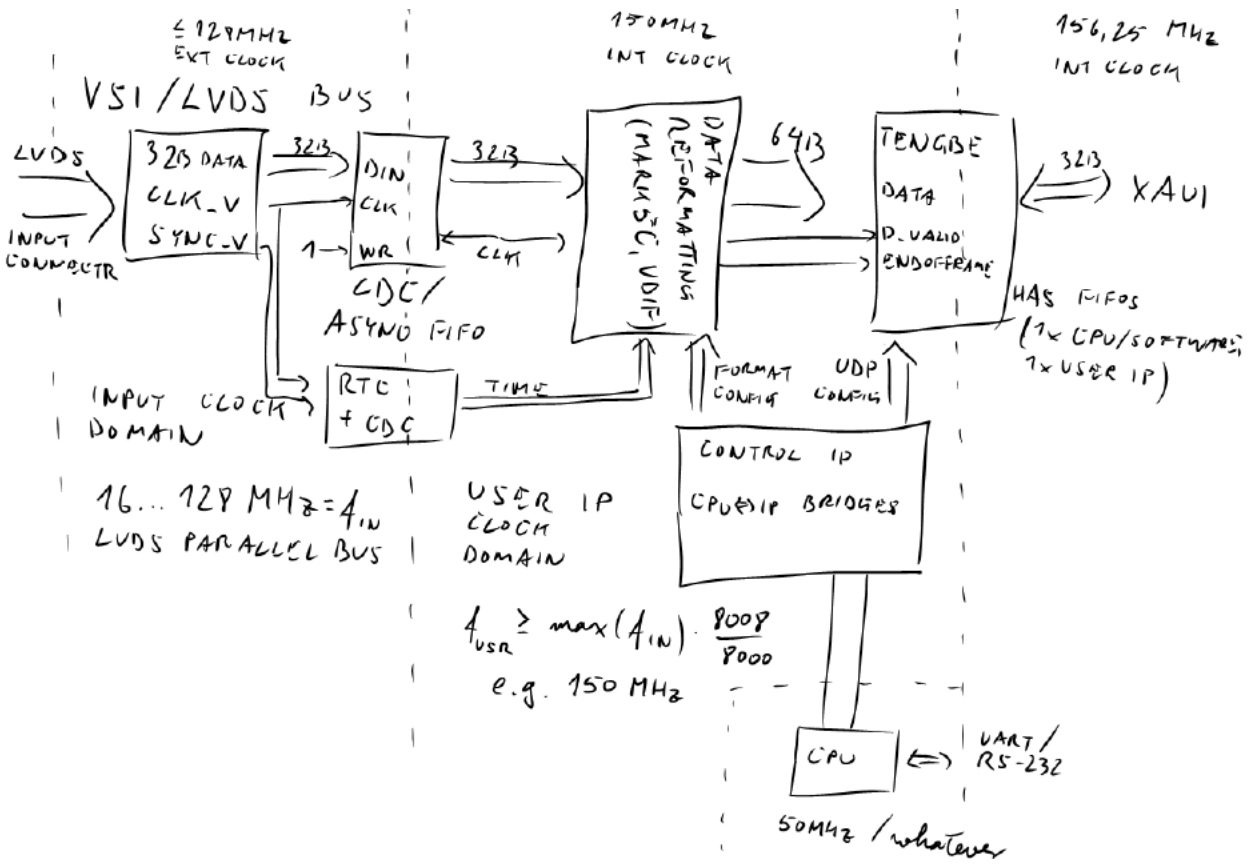
Open Issues

Conversion from Unix seconds into the epoch and MJD representations required by some of the data framing formats?

Conversion from any timestamp to VLBA-style binary-coded-decimal format for Mark5B headers?

System Diagram and Progress

TODO: pretty it up


≤ 128MHz
EXT CLOCK
VSI/LVDS BUS
LVDS
INPUT CONNECTR
32b DATA
CLK_V
SYNC_V
32b
DIN
CLK
WR
1
CDC/
ASYNC FIFO
32b
CLK
INPUT CLOCK
DOMAIN
RTC
+ CDC
TIME
150MHz
INT CLOCK
DATA REFORMATTING
(MARK5C, VDIF)
64b
FORMAT CONFIG
UDP CONFIG
TENGBE
DATA
D_VALID
ENDOFFRAME
32b
XAUI
156,25 MHz
INT CLOCK
HAS FIFOS
(1x CPU/SOFTWARE,
1x USER IP)
CONTROL IP
CPU⇔IP BRIDGES
16... 128 MHz = f_IN
LVDS PARALLEL BUS
USER IP
CLOCK
DOMAIN
f_USR ≥ max(f_IN) · 8008/8000
e.g. 150 MHz
CPU
UART/
RS-232
50MHz / whatever