DBBC FILA10G Project Book

Version: July 7, 2010

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

Project group:
Amit Bansod
Gino Tuccari
Jan Wagner
Alan Roy

Editor:
Jan Wagner

Project goal

Have a standalone FiLa10G system capable of encapsulating data from one (32-bit) or two (64-bit) VSI-H busses 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
vlbasensimemo012.pdf	VLBA Sensitivity Upgr Memo #12, Mark5C Specs
VDIF specification Release 1.0 ratified.pdf	www.vlbi.org
fila10g_sch.pdf	Gino Tuccari, Version 1.0 FiLa10G Schematic

Contents

Project goal 2

Acronyms 2

Reference material 2

Output of work 3

System Requirements 3

Interface Requirements for PowerPC-to-UserLogic 4

Network Data Format Requirements 5

Test Vector Generator Requirements 6

Corner-turning Module Requirements 7

User IP/VHDL Subcomponent Waypoints 8

Embedded Software Waypoints 9

Hardware Testing Waypoints 10

Open Issues 10

System Diagram and Progress 12

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, RX/TX/GND and no flow control
2. Synchronization of internal clock absolute time to external VSI 1PPS
3. Three software-selectable input modes to accept data from one or both VSI bus connectors:
   input of 32-bit data from a single VSI bus (VSI4 geo mode)
   input of 64-bit data from both VSI busses in parallel (VSI4 astro mode)
   input of 32-bit/64-bit from one or both VSI busses in parallel (DBBC PFB mode)
4. Generating 64bit UDP payload data from actual 32bit or 64bit VSI data
5. Generating 64bit UDP payload data with Test Vector Generator data patterns
6. Support for most common Test Vector Generator types: VSI-H, Mark5
7. Framing of VSI data into Mark5B format with 64-bit word size and 10,016-byte frame size
8. Re-framing of Mark5B/other data into smaller equal-sized UDP packets with 64bit sequence#
9. Software-configurable own 10G IP address and gateway
10. Software-configurable destination 10G IP address and port
11. Transmit data frames over one 10G link to the configured destination IP address
12. Ethernet frame payload needs to fulfill Mark 5C recording feature capability limitations
13. Timing requirements: input rate 16…128 MHz (VSI-H), system rate 150 MHz
    Base input rate: 32 MHz from DBBC no matter the baseband signal bandwidth
    Wide input rate: 64 MHz from DBBC for baseband signal bandwidths of 32 MHz
    High input rate: 128 MHz from DBBC, optional as per VSI-H standard.

There are some additional considerations:

No input data reordering: The input data is output in entirety onto the 10G link without any pre-filtering. No dedicated bit selection, channel selection or reformatting of the VSI input data stream(s) is required

internally in the FiLa10G system. VSI data in different source modes will come from the VSI-H bus, using one or two VSI ports, as defined and routed by the backend.

High input rate: While a 128 MHz VSI-H input rate will be useful to achieve 8.192 Gbps, no support for this rate is presently planned in the Haystack Mark5C recorder. Other systems and network distributed storage can however cope with almost any rate. In addition, future DBBC may facilitate data transfer between two FILA10G at 8.192 Gbps.

VSI port data phase offsets: Each VSI input port receives data and clock from its own input cable. While the clocks originate from same source clock in the DBBC, cable length differences may cause phase offset between the signals on one cable and the signals on the other cable.

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	at least 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	at least 16bit
8.	Station ID (2-char ASCII):	PowerPC => UserIP	at least 16bit

Additional signals may be necessary. All signals may be made 32bit for convenience.

System configuration bits: at least following bits need to be passed from PowerPC to the User Logic:

[1bit: Arm for next 1PPS]	'0' = stop and clear RTC clock, '1' = enable and arm RTC
[2bit: Select Input Format and Source]	'00' = VSI#1+#2 (64bit), '01' = VSI#1 (32bit), '10' = VSI#2 (32bit)
[2bit: Select Output Format]	'00' = Mark5B, '01' = VDIF, others are future extension
[1bit: Global Reset/Halt]	'1' = reset TenGBE, FIFOs, RealTime Clock, own IP
[2bit: Enable Test Vector Mode]	'00' = disabled/real data, '01' = VSI-H TVG, '10' = Mark5 TVG

System status bits: at least the following bits need to be passed from User Logic to the PowerPC:

[1bit: 1PPS Sync Gained]	'0' = when RTC not running, or armed but no 1PPS received yet
[1bit: VSI FIFO Empty]	'1' = when selected VSI input FIFO is empty (no-data –indicator)
[1bit: VSI FIFO Full]	'1' = when selected VSI input FIFO is full (overflow –indicator)

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…

Network Data Format Requirements

According to Haystack, their Mark 5C recording design uses Mark 5B as VLBI data payload in UDP data frames with a 64-bit packet sequence number prepended to the VLBI Data Payload. The optional 64-bit PSN consists of 32-bit high-bit padding and a 32-bit counter. The PSN is used by the Mark5C to reorder incoming packets, independently of the actual VLBI Payload format used.

The VLBI Data Payload size must be multiples of 64-bit word (8-byte).

The Mark5B-emulation format has a frame of 10,016 bytes. It must be sent as two 5,008-byte UDP/IP packets, the first contains the Mark5B header and some data, the second contains remaining data. When a 64-bit (8byte) PSN is added to each UDP packet, the final packet size is 5,016-byte.

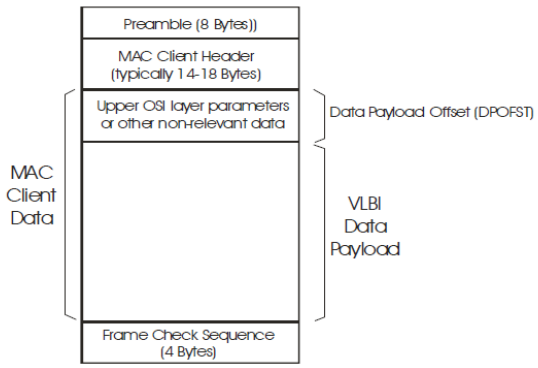

Figure 1 - Mark 5C network data format -- normal/jumbo Ethernet frame including payload. Source: VLBA Sensitivity Upgrade Memo #12

Ethernet frame capture on the Mark 5C can be configured such that it extracts data from customizable offsets within the Ethernet frame (Figure 1). The upper OSI layer headers (IP, UDP/IP) in front of the final VLBI Payload can be removed in the Mark5C recorder. Relevant Mark5C software settings include Data Playload Offset (DPOFST), Data Frame Offset (DFOFST), VLBI Payload Size.

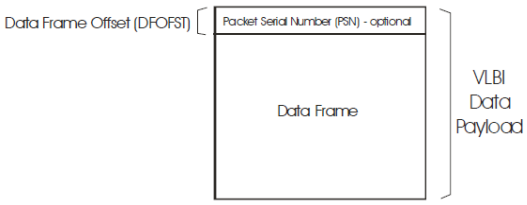

Figure 2 - Mark 5C network data format -- custom payload of Ethernet frame. Source: VLBA Sensitivity Upgrade Memo #12

The preferred structure to use (as used by Haystack) is: [MAC Ethernet Header | IP Header | UDP Header | 8 bytes for Packet serial number, of which the upper 4 bytes are padding | VLBI frame in Mark5B disk format]

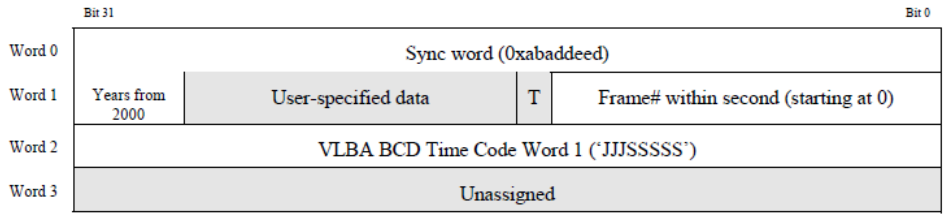

Figure 3 - Mark5B-emulation Disk Frame Header format
Source: VLBA Sensitivity Upgrade Memo #13

The VLBI frame format to primarily implement in this project is the Mark5B disk format. See Figure 3. The 'Unassigned' field should be set to 0. The 'User-specified data' section should be set to 0.

Test Vector Generator Requirements

See the VSI-H standard. Three TVG modes have to be supported: all '0', all '1', and the pseudo-random number generator on page 29 of the VSI-H PDF (or Figure 4 below). Note that the TV14 output is logical XOR of the *output* TV0 and the topmost flip-flop Q, whereas other outputs are the logical XOR of neighboring flip-flop Q's.

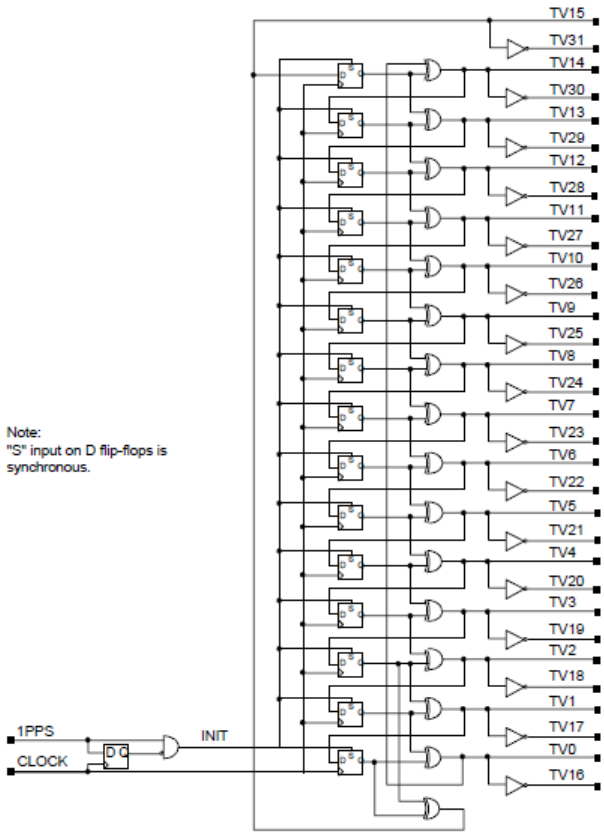

Figure 4. VSI-H Test Vector Generator

Corner-turning Module Requirements

This is to document the requirements for future work. At the moment, no implementation is required.

Corner turning is later required to support for VDIF 1.x data streams. There is only a single channel in each data stream and streams can be sent to different IP network destinations. The parallel (packed) layout of sample channels should be transposed into a serial layout.

Software correlators may want to be able to operate in two modes. Firstly in standard multi-channel mode, where nodes process a time-divisioned chunk of data that contains all channels. Secondly, in a mode that assigning computing nodes for correlating individual channels only.

Single-channel data streams are provisioned for in the VDIF 1.0 Standard. The FILA10G should be able to directly stream individual channels to assigned computing nodes (i.e. several Destination IPs configurable in the FILA10G, one for each channel). This way, no cluster resources are needed for splitting a multichannel data stream. In addition, cluster network bandwidth can be kept down by not transmitting data not required by some subset of nodes.

Table 1 : Mark5B-emulation output: one single multi-datachannel UDP/IP output stream

Time	UDP Stream 1
0	Mark5B payload, 64-bit words containing {ch0,ch1,ch2,…chN | time 0}
1	Mark5B payload, 64-bit words containing {ch0,ch1,ch2,…chN | time 1}
2	Mark5B payload, 64-bit words containing {ch0,ch1,ch2,…chN | time 2}
…	…

Table 2: VDIF output: multiple single-channel UDP/IP output streams, optionally to different IP destinations

Time	UDP Stream 1	UDP Stream 2	…	UDP Stream N
0	VDIF payload, 64-bit words {ch0,ch0,…|time 0}		…	
1				
2				
…				

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
X	Module Real time clock: 32-bit Unix seconds counter with arm / trigger / count features	Amit	2
X	Module or toplevel Two data input methods: 1) actual physical, 2) internal test	Amit	2
90%	Module Data framing blocks (implement at least: Mark5B)	Amit	3
X	User IP Block OPB/PLB-bus register to pass values between User/PowerPC	Amit/Jan	2..3
X	VHDL/System Implement all registers needed in the “Interface Requirements” section and pass them to the top-level VHDL	Amit	3
50%	VHDL/System Use implemented registers to actually configure User Logic	Amit	4..5
X	VHDL/System Test vector generator implementing VSI (or Mark5B) modes	Jan	4..5
0%	Module Corner turn for VDIF, transpose of channels from parallel multi-channel into serial single-channel format	Amit	N/A

1. System clocking setup, at least two clocks:
    - external VSI clock input : ClockDomain#1(VSI) : 16..128 MHz constraint
    - onboard 50 MHz xtal osc: ClockDomain#2(UserLogic): 100 or 150 MHz via DCM
      note: new board revisions may use a 150MHz xtal osc, 100 MHz preferred for power constraints and 4.096 Gbps VSI-H input (max at 100MHz is 6.4 Gbps)
    - onboard 50 MHz xtal osc: ClockDomain#3(PowerPC): 200 MHz via DCM
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-H Input FIFO
    - {1PPS,Valid,32bit/64bit} test vector counter data into VSI-H Input FIFO
    - Direct input (geo 32bit and astro 64bit), no reformatting required
4. Data framing block
   1) Read from VSI Receive FIFO using ClockDomain#2 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)
   4) Generate 10G UDP payload in formats
    - Mark5B required *Format described in MIT Haystack Memo #58*
    - Mark5C currently optional
    - VDIF(?) currently optional
5. Data frame numbering block
    - read data from input-side FIFO, re-output it with an extra frame sequence number in the serial format [1 x 64bit-word FSN | 626 x 64bit-words of data]
    - in front of every 5008-byte (626-word x 64-bit) block, output a 64-bit counter value
    - increment 64-bit counter for every new block, wrap to zero on overflow
6. User IP : OPB/PLB-bus register to pass 32-bit or wider value between User Logic<->PowerPC
    - Can be two unidirectional registers (preferred)
    - Or one bidirectional register
7. Implement all registers needed in the “Interface Requirements” section and pass them to the top-level VHDL
8. Use implemented registers to actually configure User Logic
9. Corner-turning module for later use with VDIF

Embedded Software Waypoints

Done	Waypoint		Due Week
X	Basic TinySH OS running in hardware over Serial Port	Amit	1
X	Add default values for system configuration to TinySH	Amit	3
	Add required new configuration commands to TinySH	Amit	
50%	Time synchronization		4

X	Data format selection		3
X	Test vector mode enable/disable		3
	Corner turn enable/disable for VDIF		N/A
	… other?		…8
X	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
100%	Board running, programmable over JTAG	Gino	1
100%	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
100%	10G link running in loopback *Checking via: ChipScope capture of received data, verify e.g. timestamps*	Amit/Jan/Gino	1
50%	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
0%	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
0%	Zero baseline testing with two DBBC, Mark5C	Amit/Jan/Gino	final
Bonus	Streaming over the 10G link from Effelsberg to MPIfR Correlate or analyze with DiFX	Amit/Jan/Gino	final

Open Issues

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

Conversion from any timestamp (or rather, just from Unix seconds plus some starting year) into a VLBA-style binary-coded-decimal (BCD) format for the Mark5B headers? The VLBA format is a Modified Julian Date expressed in BCD, with 3 digits for the day, 5 digits for the second (60*60*24 = 38400).

System Diagram

Raw data either from two 32-bit VSI or a selectable 32-bit VSI time multiplexed 32-to-64-bit.
VSI FIFOs element counts are used for FIFO read-enable when both have more than 1 element.
Real-time clock (RTC) increments from a PPC-provided base second by +1 on every VSI 1PPS.
Mark5B data framing module uses RTC and adds headers in front of data read from FIFO(s).
Packet framing module adds a 64-bit counter in front of every 5008-byte block.
PowerPC software configures 10GbE, RTC, data modes.