FE-I4 code for emulator, release V2.0

Laura Gonella
Bonn, 12/10/2010

Note

Keep in mind that every release of the FEI4 code for the emulator implements some functionality of FE-I4, and thus behaves as FE-I4. For a better and deeper understanding of what it does, I recommend to get familiar with FE-I4 before/instead of looking at the emulator code.
Some parts of the "FE-I4 Integrated Circuit Guide V7.7" (06/08/2010) have been copied here to help explaining the firmware.

Functionalities

This release of the FE-I4 code for the emulator board allows writing global registers, and reading the configuration back, as well as sending triggers and receiving data back. The data output speed is 40Mbps.

Target applications

- USBPix firmware/software development

Implementation

The main components of the code are

- A **DCM** for clock buffering and multiplication
- The command decoder block (**CMD**)
- The configuration memory (**CONFGMEM**)
- The End Of Chip Logic block (**EOCHL**)
- The Data Output block (**DOB**)
- A block to connect token, L1 trigger and read signal (**TOKEN**)
- A memory 4 bits deep and 20 bits wide storing PDR (Pixel Digital Region) data
    - This memory can of course be extended, or data could even be generated randomly. The choice of such a small set of known data was to ease debugging of the test systems. This can be changed in future versions of the firmware;
    - PDR data are in the form [19] NeiRB, [18] NeiRT, [17] NeiLB, [16] NeiLT, [15:12] RB, [11:8] RT, [7:4] LB, [3:0] LT (Nei = Neighbor, R = Right, L = Left, B = Bottom, T = Top).

The CMD, EOCHL and DOB code is the one written for FE-I4 with some minor modifications to allow correct synthesis and implementation in an FPGA.
The CONFGMEM and the TOKEN blocks have been written ad-hoc.

The connections between the blocks are minimal, and they are basically only those needed for the above mentioned tasks. In particular following the register table in the "FE-I4 Integrated Circuit Guide V7.7" (06/08/2010):

- Reg 29, bit [11:4]: EmptyRecordConfig
- Reg 29, bit 12: clk2OutConfig
- Reg 29, bit 13: no8b10b (1 = raw data, 0 = 8b10b encoded data)
- Reg 2, bit 11: Conf_AddrEnable (1 = address read back, 0 = no address read back)

- Reg2, bit [15:12]: Trigger_count (internal trigger multiplication of the incoming trigger)

Instruction for use

- First of all remember that the FE-I4 has two mode of operation: ConfMode and RunMode (see explanation below). Remember thus to set the chip in the correct mode according to the type of command you send!!!

RunMode The chip has two modes which the command decoder can be placed in. There is a run mode, where the command decoder only acts on trigger and fast commands. There is also a configuration mode, where the command decoder only acts on slow commands. More concisely, the command decoder always processes the commands you send it, but depending on its mode it either does or does not produce output. This command toggles the FE between RunMode and ConfMode. In order to set the FE in RunMode one has to write 111000 in Field 5. A value of 000111 puts the chip in ConfMode. Any other combination of Field 5 does not change the RunMode status of the chip.

- The firmware accepts the following commands (see tables below)
  - LV1 command
  - Fast commands
    - BCR
    - ECR
  - Slow commands
    - RdRegister
    - WrRegister
    - GlobalReset
    - RunMode

Name	Field 1	Field 2	Description
size (bits):	5	4	
LV1	11101[1]	-	Level 1 Trigger
BCR	10110	0001	Bunch Counter Reset
ECR	10110	0010	Event Counter Reset
CAL	10110	0100	Calibration Pulse
Slow	10110	1000	Slow command header

1: A single bit flip in the LV1 command will still result in a LV1 being decoded

Name	Field 3	Field 4	Field 5	Field 6	Description
size (bits):	4	4	6		
RdRegister	0001	ChipId	Address	-	Read addressed global memory register
WrRegister	0010	ChipId	Address	Data[1]	Write into addressed global memory register
WrFrontEnd	0100	ChipId	xxxxxx	Data[2]	Write conf data to selected shift register(s)
GlobalReset	1000	ChipId	-	-	Reset command; Puts the chip in its idle state
GlobalPulse	1001	ChipId	Width[3]	-	Has variable pulse width and functionality
RunMode	1010	ChipId	sssccc	-	Sets RunMode or ConfMode

1: 16 bits 2: 672 bits 3: 1-64 bits

Table 11. Slow Commands

- The type of data that can be received back are (see list below)
  - Pixel data
  - Configuration with address read back
  - Configuration with no address read back
  - Idle

1. Pixel Data:
   raw: (1)×DH, (0/n)×DR, (0/1)×SR, (1)×ER
   framed: SOF, (1)×DH, (0/n)×DR, (0/1)×SR, EOF

2. Configuration with address read back:
   raw: (1/n)×(AR, VR)
   framed: SOF, (1/n)×(AR, VR), EOF

3. Configuration with no address read back:
   raw: (1/n)×VR
   framed: SOF, (1/n)×VR, EOF

4. Service message:
   raw: (1/n)×SR
   framed: SOF, (1/n)×SR, EOF

5. Idle:
   raw: (1/n)×ER
   framed: (1/n)×Idle

24-bit Record Word	Field 1	Field 2	Field 3	Field 4	Field 5	Comments
Data Header DH	11101	001	xxxx	LV1ID [3:0]	bcID [0:7]	xxxx reserved for later use
Data Record DR	Column [6:0]	Row [8:0]	ToTtop [3:0]	ToTbot [3:0]		Column numbering: 0000001 to 1010000
Address Record AR	11101	010	Type	Address [14:0]		Type 0: Global Register Type 1: Shift Register Position
Value Record VR	11101	100	Value [15:0]			Value Record without previous address Record allowed
Service Record SR	11101	111	Message [15:0]			Service Message (e.g. error codes)
Empty Record ER	00000000	00000000	00000000			Idle = K.28.1 commas (8b/10b coding case)

Table 16. The Six 24-bit Record Words

Note: Service Message: ServCode[15:10], Counter[9:0]

- Comments on the pixel data
  - The column and row number are generated by counters. This eases the task of debugging, as explained in the example below.
  - The number of Data Records (DR) that is sent with every pixel data is set via a pseudo random generator.
  - The possible combinations of ToTtop and ToTbot are
    - Tot=[ToTtop,ToTbot] =[14, 6]
    - Tot=[5, 15]
    - Tot=[4, 3]
    - Tot=[14, 15]
    - Tot=[8, 6]
    - Tot=[14, 15]
    - Tot=[14, 7]
    - Tot=[4,14]
    - Tot=[5, 8]
    - Tot=[14, 15]
    - Tot=[14, 4]
    - Tot=[8, 15]
    - Tot=[14, 6]
    - Tot=[5, 15]
    - Tot=[8, 4]
    - Tot=[14,15]

    in this order!

  - Example
    Let's assume:

- Trigger_count = 2

This means that every trigger sent to the chip will be internally multiplied by 2. Let's send two trigger signals to the chip. The output will look like this

DH: 11101 001 – 1000 - LV1ID=[0] bcID=[22]
DR - col=[3] row=[2] Tot=[14, 6]
DR - col=[3] row=[4] Tot=[5,15]
DR - col=[4] row=[3] Tot=[4, 3]
DR - col=[4] row=[5] Tot=[14,15]
DR - col=[5] row=[5] Tot=[8, 6]
DR - col=[5] row=[7] Tot=[14,15]
DR - col=[6] row=[4] Tot=[14, 7]
DR - col=[6] row=[6] Tot=[4,14]
DR - col=[7] row=[7] Tot=[5, 8]
DR - col=[7] row=[9] Tot=[14,15]
DR - col=[8] row=[6] Tot=[14, 4]
DR - col=[8] row=[8] Tot=[8,15]
DR - col=[9] row=[8] Tot=[14, 6]
DR - col=[9] row=[10] Tot=[5,15]
DR - col=[10] row=[9] Tot=[8, 4]
DR - col=[10] row=[11] Tot=[14,15]
SR - ServCode=[9] Counter=[19]

11101 001 - 0000 - LV1ID =[0] bcID =[23]
DR - col=[11] row=[10] Tot=[14, 6]
DR - col=[11] row=[12] Tot=[5,15]
DR - col=[12] row=[11] Tot=[4, 3]
DR - col=[12] row=[13] Tot=[14,15]
…
DR - col=[37] row=[37] Tot=[8, 6]
DR - col=[37] row=[39] Tot=[14,15]
DR - col=[38] row=[36] Tot=[14, 7]
DR - col=[38] row=[38] Tot=[4,14]

11101 001 - 1000 - LV1ID =[1] bcID =[3]
DR - col=[39] row=[39] Tot=[5, 8]
DR - col=[39] row=[41] Tot=[14,15]
DR - col=[40] row=[38] Tot=[14, 4]
DR - col=[40] row=[40] Tot=[8,15]
…
DR - col=[61] row=[61] Tot=[8, 6]
DR - col=[61] row=[63] Tot=[14,15]
DR - col=[62] row=[60] Tot=[14, 7]
DR - col=[62] row=[62] Tot=[4,14]
SR - ServCode=[9] Counter=[106]

11101 001 - 0000 - LV1ID =[1] bcID =[4]
DR - col=[63] row=[63] Tot=[5, 8]
DR - col=[63] row=[65] Tot=[14,15]
DR - col=[64] row=[62] Tot=[14, 4]
DR - col=[64] row=[64] Tot=[8,15]
…
DR - col=[79] row=[79] Tot=[5, 8]
DR - col=[79] row=[81] Tot=[14,15]
DR - col=[80] row=[78] Tot=[14, 4]
DR - col=[80] row=[80] Tot=[8,15]
DR - col=[1] row=[80] Tot=[14, 6]
DR - col=[1] row=[82] Tot=[5,15]
DR - col=[2] row=[81] Tot=[8, 4]
DR - col=[2] row=[83] Tot=[14,15]
DR - col=[3] row=[82] Tot=[14, 6]
DR - col=[3] row=[84] Tot=[5,15]
DR - col=[4] row=[83] Tot=[4, 3]
DR - col=[4] row=[85] Tot=[14,15]

How to check if that is the correct data:

1) First of all one can check if the number of pixel data (i.e. events) read back is correct. Two trigger were sent and multiplied by 2, this mean we expect 4 events -> correct.
2) Afterwards one can check the L1 trigger ID. 2 triggers were sent, so one expects to have two consecutive numbers for the L1 trigger ID. Here we see LV1ID[0] and LV1ID [1] -> correct.
3) Now let's check that the bcID is correct. Again, we multiplied the trigger by two, so for every level one trigger we should have two events with consecutive bcID number. Here we have 2 events associated to LV1ID[0] with bcID [22] and bcID [23], and 2 events associated to LV1ID[1] with bcID [3] and bcID [4] -> correct
4) Next thing to check is if the data are correct.
    a. Look at column numbers, they should be consecutive numbers. That's the case here. When 80 is reached the column number starts from 1 again (see last event);
    b. Check the row numbers in one PDR* (Pixel Digital Region): they have to be adjacent numbers, but they will not be in consecutive order. Example:
    DR - col=[3] row=[2] Tot=[14, 6]
    DR - col=[3] row=[4] Tot=[5,15]
    DR - col=[4] row=[3] Tot=[4, 3]
    DR - col=[4] row=[5] Tot=[14,15]
    c. Finally check that the ToT combinations are the ones listed above, in that order.

*NB: explaining a PDR and how data in a PDR are read out is out of the scope of this document. Please refer to the FE-I4 literature.

- Comments on field 3 of Data Header (DH)

- A 0000 in field 3 of DH means there are no service messages attached to the DR
- A 1000 in field 3 of DH means there is a service message attached to the DR
- See example and known issues

Known issues

Every other event will send back also a service message indicating that the FIFO is full (see example and the following table, copied from the document "EOCHL V5" (5/2/2010)). This might be due to the low speed used to read out the FIFO. The firmware is in fact designed to send out data at 40Mbps, which implies reading the FIFO at 4MHz if 8b10b encoding in on, or 5MHz if it is off. However, the FIFO was designed to be read out at 16MHZ or 20MHz, using the default FE-I4 output speed of 160Mbps. This has to be checked in the next version of the firmware with output at 160MHz.

As one can see in the example above, this error does not prevent correct reading of the data.

Error Code	Error Source	description	
0	Hamming Error Word 0	Hamming Error Word 0 Fifo Output	jd
1	Hamming Error Word 1	Hamming Error Word 1 Fifo Output	jd
2	Hamming Error Word 2	Hamming Error Word 2 Fifo Output	jd
3	L1 Trigger Id Counter Error	Triple Counter not equal	jd
4	Bunch Counter Error	Triple Counter not equal	jd
5	L1 In Counter Error	Triple Counter not equal	jd
6	L1 Register Error	Triple Register not equal	jd
7	L1 Reqest Counter Error	Triple Counter not equal	jd
8	State Machine Error	Triple State Machine Not equal	jd
9	Fifo Full	Read Out of Fifo is to slow	jd
10	Prompt Radiatoin Detector	At least 1 prompt radiation detector fired	maurice
11	Hit Or clock high	Counts HitOr pulses when PLL MUX2 is high	Abderrezak
12	Hit Or clock low	Counts HitOr pulses when PLL MUX2 is low	Abderrezak
13	CNFGMEM SEU error	Error out global OR from TRE registers 0-31	Mohsine
14	Efuse shadow SEU error	Error out global OR from Efuse shadow registers	Julien
15	CMD SEU	SEU in 1 of 3 state machine	Roberto
16	CMD bit-flip	Bit flip in a CMD TRE register	Roberto
17	Trig bit-flip	Bit flip in received trigger command	Roberto
18	Fast bit-flip	Bit flip in received fast command	Roberto
19	Slow bit-flip	Bit flip in received slow command	Roberto
20	DC Hamming Error Word0	DC Hamming Error Word0	Tomasz
21	DC Hamming Error Word1	DC Hamming Error Word1	Tomasz
22	DC Hamming Error Word2	DC Hamming Error Word2	Tomasz
23	DC token TRE error	Error in triple redundant DC token	Tomasz
24	DC too many read request	A serious error Tomasz to define	Tomasz
25	PLL feedback too fast	PLL feedback too fast = loss of lock	Andre
26	PLL feedback too slow	PLL feedback too slow = loss of lock	Andre
27	Invalid address	CMD puts a non-existing 6 bit register address	Dario
28			
29			
30	VDDA power glitch	Instantaneous VDDA dips below 80% of average	Abderrezak
31	VDDD power glitch	Instantaneous VDDD dips below 80% of average	Abderrezak