Daniel Magalotti
University of Modena and Reggio Emilia
INFN of Perugia

Software for tests: AMB and LAMB configuration - Available tools

FTK Workshop – Pisa 13/03/2013

Outline

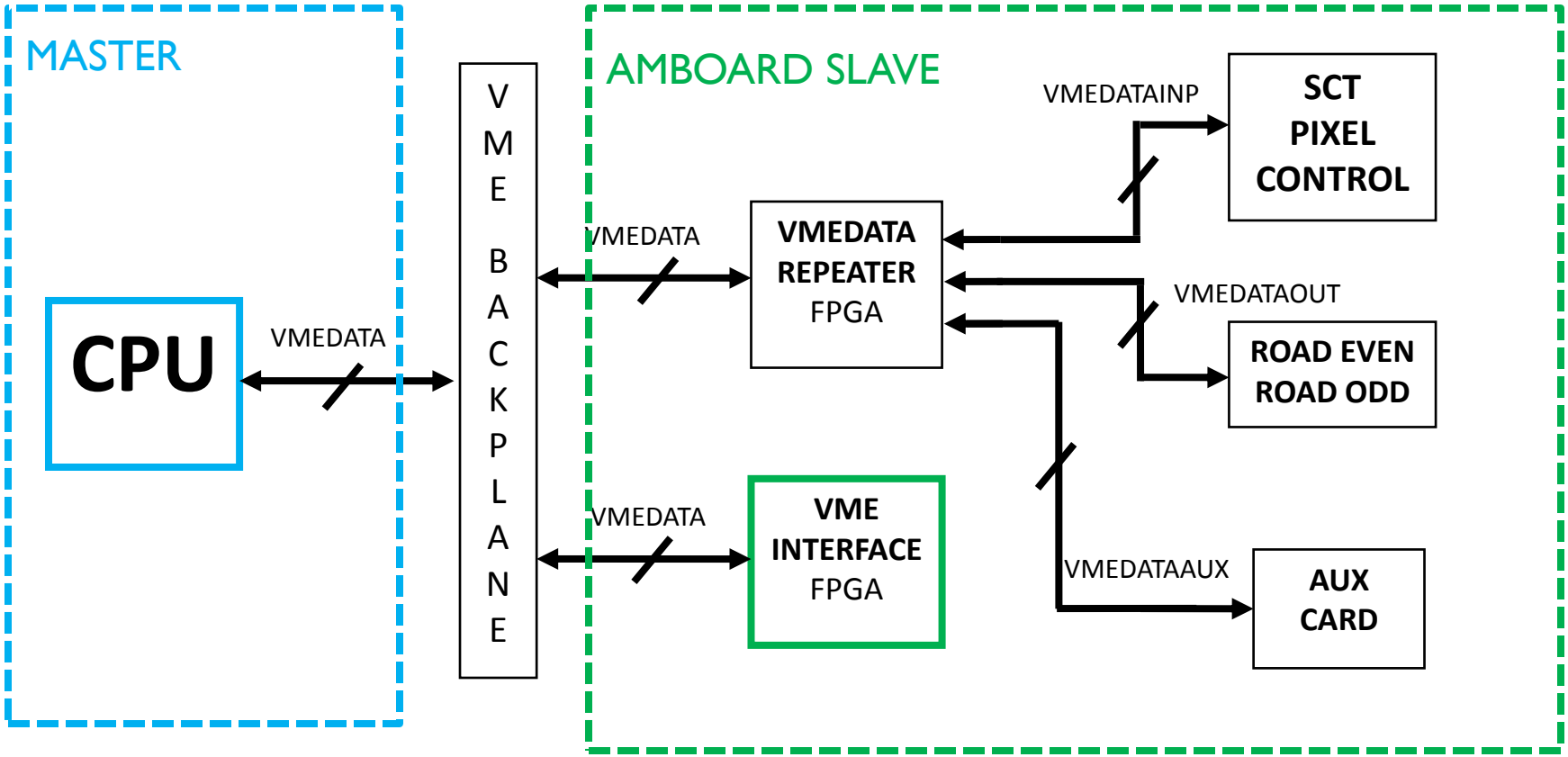
- VME standard protocol description
- AMBoard & LAMB configuration
- AMchip configuration and control
- Tools and software for testing

VME crate interface

- VME crate
  - Link to the document for the VME protocol
    http://www.interfacebus.com/Design_Connector_VME.html
  - 21 slot for the VME crate
  - In the **first slot** there is the master CPU
  - Until 20 slot for the AMBoard that is the slave
  - A **geographical address** identifies the position of the board

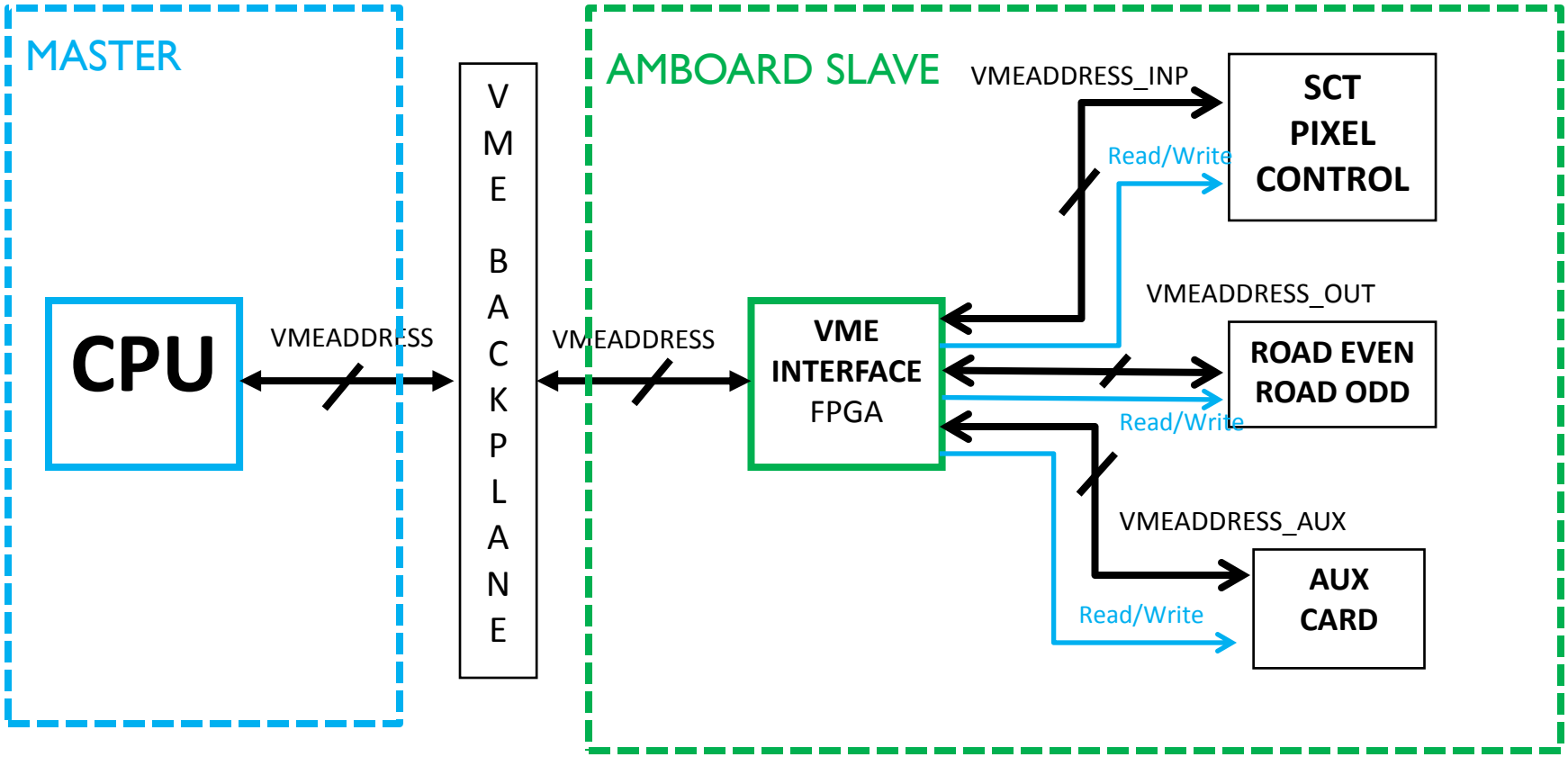
VME **data** distribution

- The VME data bus distribution inside the AMBoard



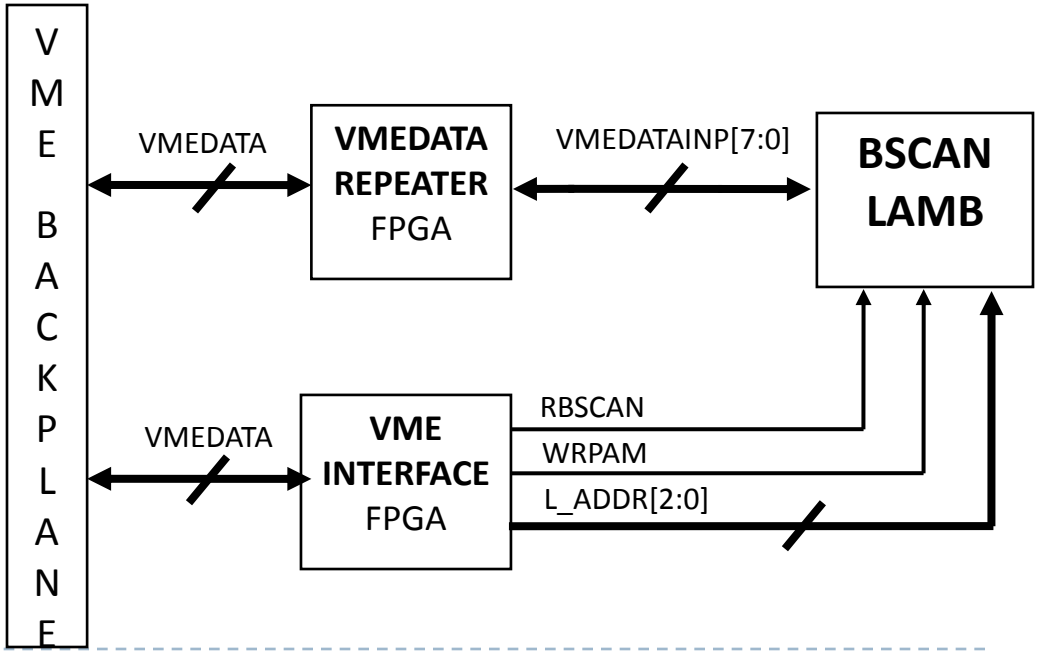
VME **address** distribution

- The VME address bus is distribute to all chip of the AMBoard with the Read/Write signal



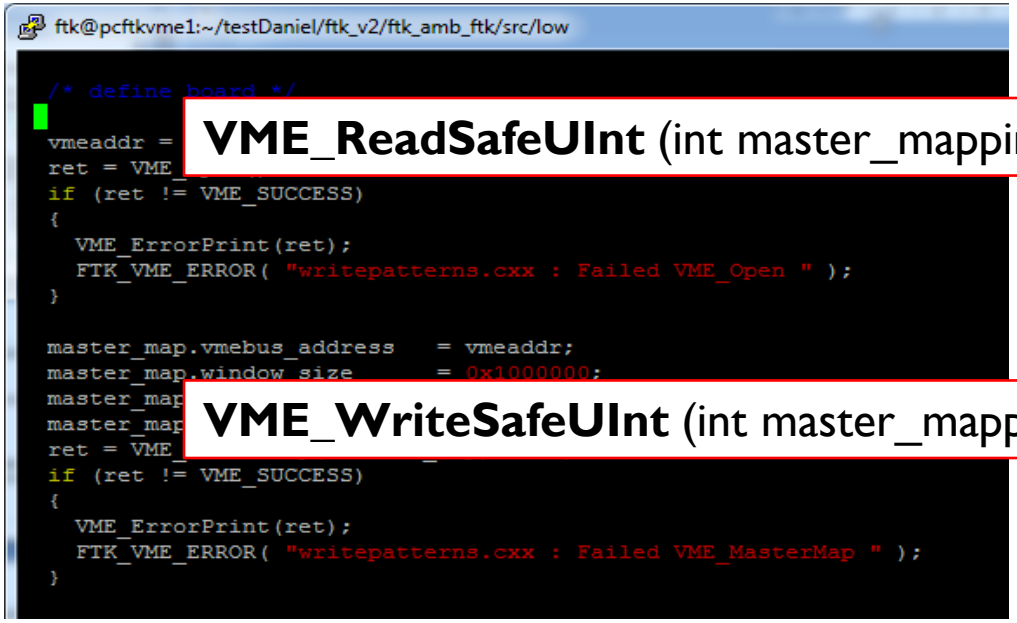
LAMB configuration

- The VME data bus is also distributed to the all 4 LAMBs
  - 8 vmedata to each LAMB so each one is controlled in parallel
- The data is received from BSCAN chip that perform the conversion from VME to JTAG interface
  - There is 8 AMChip JTAG chain so each chain is controlled in parallel
  - A 3 bit address is received to identify the JTAG operation (write TDI o TMS read TDO)



Software for test

- The CPU master controls the VME protocol: function in C++ code are implemented



VME_ReadSafeUInt (int master_mapping, u_int address_offset, u_int* value);

Read register of a specific location address

VME_WriteSafeUInt (int master_mapping, u_int address_offset, u_int value);

Write register of a specific location address

- All the functions are based on the basic read and write operation of a register

Software for test

- The principal operations with the software are
  - **Configure** and **control** the AMchips
  - **Testing the hardware** connection to/from AMchips with JTAG standard
  - **Readout** of the **SpyBuffer** and **writing** of **VmeFifo**
  - Automatic script to testing the entire system

AMChip pattern configuration

- Crate the map of the AMchip present into the board
  - A JTAG operation to define the mapping

Number of the LAMB present in the board

Define the **active columns** present in each LAMB

Total number of AMchips

AMChip pattern configuration

- Crate the map of the AMchip present into the board
  - A JTAG operation to define the mapping
- Write the "**CONFIGURATION REGISTER**" of the AMchip

DEF_THR <= JPATT_CTRL (3 downto 0) := Threshold
DEF_required_layers <= JPATT_CTRL (5 downto 4);
GEOADDR <= JPATT_CTRL(14 downto 8) := geografhical address
BOTTOMchip <= JPATT_CTRL(32) := input bus mirror
test_mode <= JPATT_CTRL(40);
enable_layermap <= JPATT_CTRL(44);
LASTchip <= JPATT_CTRL(48);
disable_PATT_FLOW <= JPATT_CTRL(52);
drive_strength <= JPATT_CTRL(60);
DCBits<=JPATT_CTRL(79 downto 64);

AMChip pattern configuration

- Crate the map of the AMchip present into the board
  - A JTAG operation to define the mapping
- Write the "**CONFIGURATION REGISTER**" of the AMchip
- Start the **writing pattern operation** into the AMchips
  - Write the value of the first address location into the **ADDRES_DATA** register
  - An iterative procedure on all the pattern
    - Write the pattern data into the **DATA REGISTER**
    - Send the **OPERATIONE_WRITE_INCREMENT** to write the pattern and increment the value of the **ADDRESS REGISTER**

AMChip pattern configuration

ftk@pcftkvme1:~/testDaniel/ftk_v2/ftk_amb_ftk/src/low

```
ConfigureScam(am1, slot, chipno, active_columns, BYPASS, 1);  /* goto Run-Test/Idle */

/* reset all patterns */

/* put all chips into LDADDR  and write in 0*/
ConfigureScam(am1, slot, chipno, active_columns, LDADDR, 1);  /* goto Run-Test/Idle */
GotoShiftDRfromIdle(am1, active_columns);
for (i=0; i<ADDR_SIZE*chainlength; i++)
  data[i] = 0;
AMshift(am1, slot, active_columns, ADDR_SIZE*chainlength, data, NULL, 1);
GotoIdlefromExit1(am1, active_columns);

/* put all chips into LDDATA */
ConfigureScam(am1, slot, chipno, active_columns, LDDATA, 1);  /* goto Run-Test/Idle */
GotoShiftDRfromIdle(am1, active_columns);
for (i=0; i<DATA_SIZE*chainlength; i++)
  data[i] = 0;

AMshift(am1, slot, active_columns, DATA_SIZE*chainlength, data, NULL, 1);
GotoIdlefromExit1(am1, active_columns);

//read the data register
CheckOperation(am1, slot, chipno, active_columns, RDDATA, DATA_S

// Reset all the patterns
for(i=0;i<MAXPATT_PERCHIP;i++)
  ConfigureScam(am1, slot, chipno, active_columns, OPWRINC, 1);  /* goto Run-Test/Idle */

ConfigureScam(am1, slot, chipno, active_columns, BYPASS, 1);  /* goto Run-Test/Idle */

/* start to write the patterns */
                                                              290,3         42%
```

Write the IR JTAG of AMchip

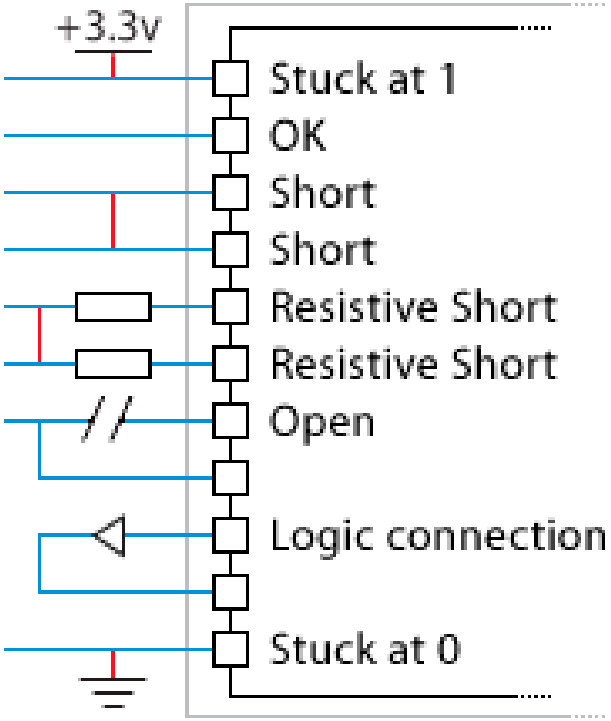
Write the data into the register addressing

AMChip configuration

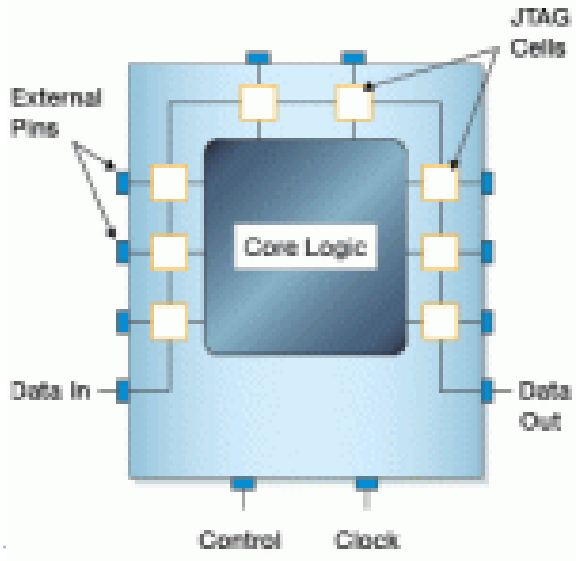
- **Check pattern** operation control the pattern bank written
  - This operation is done only with JTAG operation
  - Set the AMchip into TMODE operation so the input/output data is disable
    - Loading the pattern data into the **DATA REGISTER** of the AMchip
    - Send a **INIT_OPCODE** command to enable the match of the pattern
    - Control the **ADDRESS REGISTER** to compare the current value with the expecting value
    - Send an **SELECTION_BANK** command to clear the output match for the next pattern

LAMB JTAG Testing

- Testing with JTAG allows you to test printed circuit boards for manufacturing defects and functional failures



Chip manufacturers includes **special cell** on each of the ICs pins to facilitate test.



LAMB JTAG Testing

- Testing with JTAG allows you to test printed circuit boards for manufacturing defects and functional failures
  - By driving signals between connected devices, nets can be tested for opens, shorts and stuck-at failures.
- For testing all the pad of the AMchip we have to check
  - **INput DIstributor to input pad of AMchips [DONE]**
  - **AMchips output to GLUE [TO-DO]**
  - **AMchips output to AMchips input pad (pipeline connection) [TO-DO]**

LAMB JTAG Testing

ftk@pcftkvme1:~/testDaniel/ftk_v2/ftk_amb_ftk/src/low

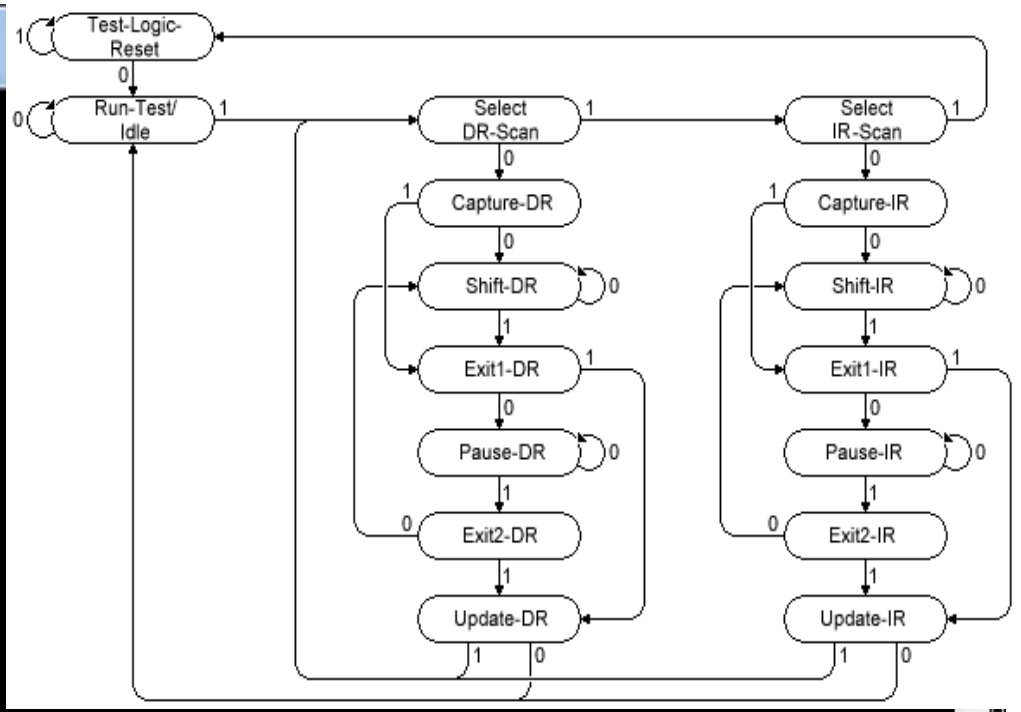
```
#ifndef AM_JTAG_FUNC_H
#define AM_JTAG_FUNC_H

#include "ftkvmelib.h"
#include "ambftk_vme_regs.h"

int GotoTestReset(int amb_handle);
int GotoIdlefromReset(int amb_handle, int column);
int GotoShiftIRfromReset(int amb_handle,int column);
int GotoShiftIRfromIdle(int amb_handle,int column);
int GotoIdlefromExit1(int amb_handle,int column);
int GotoSelectDRfromExit1(int amb_handle,int column);
int StayIdle(int amb_handle);
int GotoShiftDRfromIdle(int amb_handle,int column);
int AMshift(int slave, int slot,  u_int columns, u_int nwords, u_int* indata, u_int* outdata, int exit );
int ConfigureScam(int amb_handle, int slot, int chipno, int column, int scaminstruction, int gotoruntest);
int ConfigureChip(int amb_handle, int chipno, int column, int instruction);

#endif // AM_JTAG_FUNC_H
```

18,0-1 All



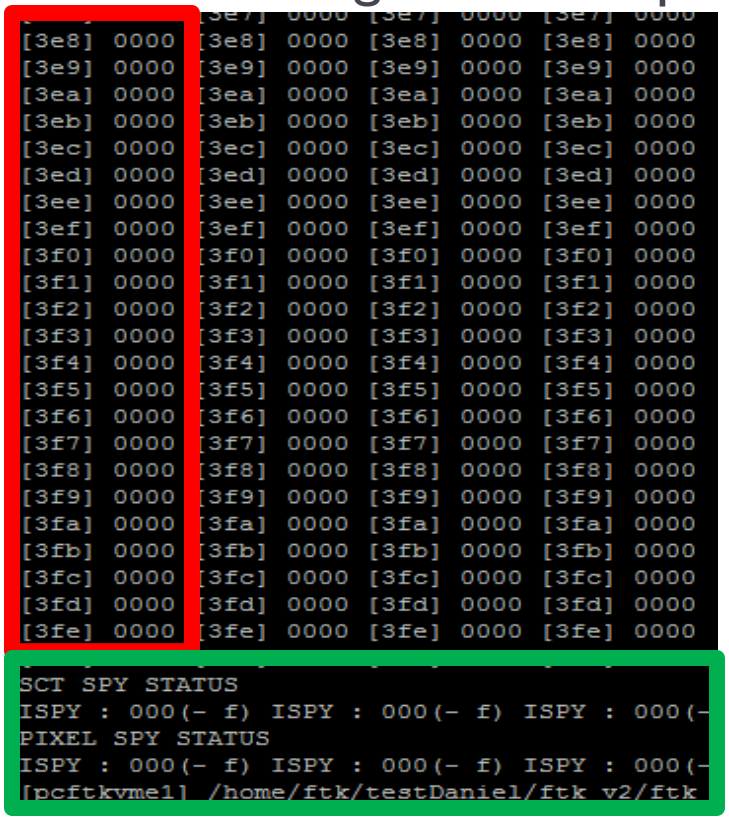
Monitoring and testing tools

- The Spybuffer is used to monitoring the data flow in each input and output link between AMBoard and AUX card
  - A function give this output file format

```
[3e8] 0000 [3e8] 0000 [3e8] 0000 [3e8] 0000 [3e8] 0000 [3e8] 0000 [3e8] 0000 [3e8] 0000 [3e8] 0000 [3e8] 0000 [3e8] 0000 [3e8] 0000
[3e9] 0000 [3e9] 0000 [3e9] 0000 [3e9] 0000 [3e9] 0000 [3e9] 0000 [3e9] 0000 [3e9] 0000 [3e9] 0000 [3e9] 0000 [3e9] 0000 [3e9] 0000
[3ea] 0000 [3ea] 0000 [3ea] 0000 [3ea] 0000 [3ea] 0000 [3ea] 0000 [3ea] 0000 [3ea] 0000 [3ea] 0000 [3ea] 0000 [3ea] 0000 [3ea] 0000
[3eb] 0000 [3eb] 0000 [3eb] 0000 [3eb] 0000 [3eb] 0000 [3eb] 0000 [3eb] 0000 [3eb] 0000 [3eb] 0000 [3eb] 0000 [3eb] 0000 [3eb] 0000
[3ec] 0000 [3ec] 0000 [3ec] 0000 [3ec] 0000 [3ec] 0000 [3ec] 0000 [3ec] 0000 [3ec] 0000 [3ec] 0000 [3ec] 0000 [3ec] 0000 [3ec] 0000
[3ed] 0000 [3ed] 0000 [3ed] 0000 [3ed] 0000 [3ed] 0000 [3ed] 0000 [3ed] 0000 [3ed] 0000 [3ed] 0000 [3ed] 0000 [3ed] 0000 [3ed] 0000
[3ee] 0000 [3ee] 0000 [3ee] 0000 [3ee] 0000 [3ee] 0000 [3ee] 0000 [3ee] 0000 [3ee] 0000 [3ee] 0000 [3ee] 0000 [3ee] 0000 [3ee] 0000
[3ef] 0000 [3ef] 0000 [3ef] 0000 [3ef] 0000 [3ef] 0000 [3ef] 0000 [3ef] 0000 [3ef] 0000 [3ef] 0000 [3ef] 0000 [3ef] 0000 [3ef] 0000
[3f0] 0000 [3f0] 0000 [3f0] 0000 [3f0] 0000 [3f0] 0000 [3f0] 0000 [3f0] 0000 [3f0] 0000 [3f0] 0000 [3f0] 0000 [3f0] 0000 [3f0] 0000
[3f1] 0000 [3f1] 0000 [3f1] 0000 [3f1] 0000 [3f1] 0000 [3f1] 0000 [3f1] 0000 [3f1] 0000 [3f1] 0000 [3f1] 0000 [3f1] 0000 [3f1] 0000
[3f2] 0000 [3f2] 0000 [3f2] 0000 [3f2] 0000 [3f2] 0000 [3f2] 0000 [3f2] 0000 [3f2] 0000 [3f2] 0000 [3f2] 0000 [3f2] 0000 [3f2] 0000
[3f3] 0000 [3f3] 0000 [3f3] 0000 [3f3] 0000 [3f3] 0000 [3f3] 0000 [3f3] 0000 [3f3] 0000 [3f3] 0000 [3f3] 0000 [3f3] 0000 [3f3] 0000
[3f4] 0000 [3f4] 0000 [3f4] 0000 [3f4] 0000 [3f4] 0000 [3f4] 0000 [3f4] 0000 [3f4] 0000 [3f4] 0000 [3f4] 0000 [3f4] 0000 [3f4] 0000
[3f5] 0000 [3f5] 0000 [3f5] 0000 [3f5] 0000 [3f5] 0000 [3f5] 0000 [3f5] 0000 [3f5] 0000 [3f5] 0000 [3f5] 0000 [3f5] 0000 [3f5] 0000
[3f6] 0000 [3f6] 0000 [3f6] 0000 [3f6] 0000 [3f6] 0000 [3f6] 0000 [3f6] 0000 [3f6] 0000 [3f6] 0000 [3f6] 0000 [3f6] 0000 [3f6] 0000
[3f7] 0000 [3f7] 0000 [3f7] 0000 [3f7] 0000 [3f7] 0000 [3f7] 0000 [3f7] 0000 [3f7] 0000 [3f7] 0000 [3f7] 0000 [3f7] 0000 [3f7] 0000
[3f8] 0000 [3f8] 0000 [3f8] 0000 [3f8] 0000 [3f8] 0000 [3f8] 0000 [3f8] 0000 [3f8] 0000 [3f8] 0000 [3f8] 0000 [3f8] 0000 [3f8] 0000
[3f9] 0000 [3f9] 0000 [3f9] 0000 [3f9] 0000 [3f9] 0000 [3f9] 0000 [3f9] 0000 [3f9] 0000 [3f9] 0000 [3f9] 0000 [3f9] 0000 [3f9] 0000
[3fa] 0000 [3fa] 0000 [3fa] 0000 [3fa] 0000 [3fa] 0000 [3fa] 0000 [3fa] 0000 [3fa] 0000 [3fa] 0000 [3fa] 0000 [3fa] 0000 [3fa] 0000
[3fb] 0000 [3fb] 0000 [3fb] 0000 [3fb] 0000 [3fb] 0000 [3fb] 0000 [3fb] 0000 [3fb] 0000 [3fb] 0000 [3fb] 0000 [3fb] 0000 [3fb] 0000
[3fc] 0000 [3fc] 0000 [3fc] 0000 [3fc] 0000 [3fc] 0000 [3fc] 0000 [3fc] 0000 [3fc] 0000 [3fc] 0000 [3fc] 0000 [3fc] 0000 [3fc] 0000
[3fd] 0000 [3fd] 0000 [3fd] 0000 [3fd] 0000 [3fd] 0000 [3fd] 0000 [3fd] 0000 [3fd] 0000 [3fd] 0000 [3fd] 0000 [3fd] 0000 [3fd] 0000
[3fe] 0000 [3fe] 0000 [3fe] 0000 [3fe] 0000 [3fe] 0000 [3fe] 0000 [3fe] 0000 [3fe] 0000 [3fe] 0000 [3fe] 0000 [3fe] 0000 [3fe] 0000
[3ff] 0000 [3ff] 0000 [3ff] 0000 [3ff] 0000 [3ff] 0000 [3ff] 0000 [3ff] 0000 [3ff] 0000 [3ff] 0000 [3ff] 0000 [3ff] 0000 [3ff] 0000
SCT SPY STATUS
ISPY : 000(- f) ISPY : 000(- f) ISPY : 000(- f) ISPY : 000(- f)
PIXEL SPY STATUS
ISPY : 000(- f) ISPY : 000(- f) ISPY : 000(- f) ISPY : 000(- f) ISPY : 000(- f) ISPY : 000(- f) ISPY : 000(- f) ISPY : 000(- f)
[pcftkvme1] /home/ftk/testDaniel/ftk_v2/ftk_amb_ftk/src/low >
```

Monitoring and testing tools

- The Spybuffer is used to monitoring the data flow in each input and output link between AMBoard and AUX card
  - A function give this output file format



In the red box is report the output of one Spybuffer:

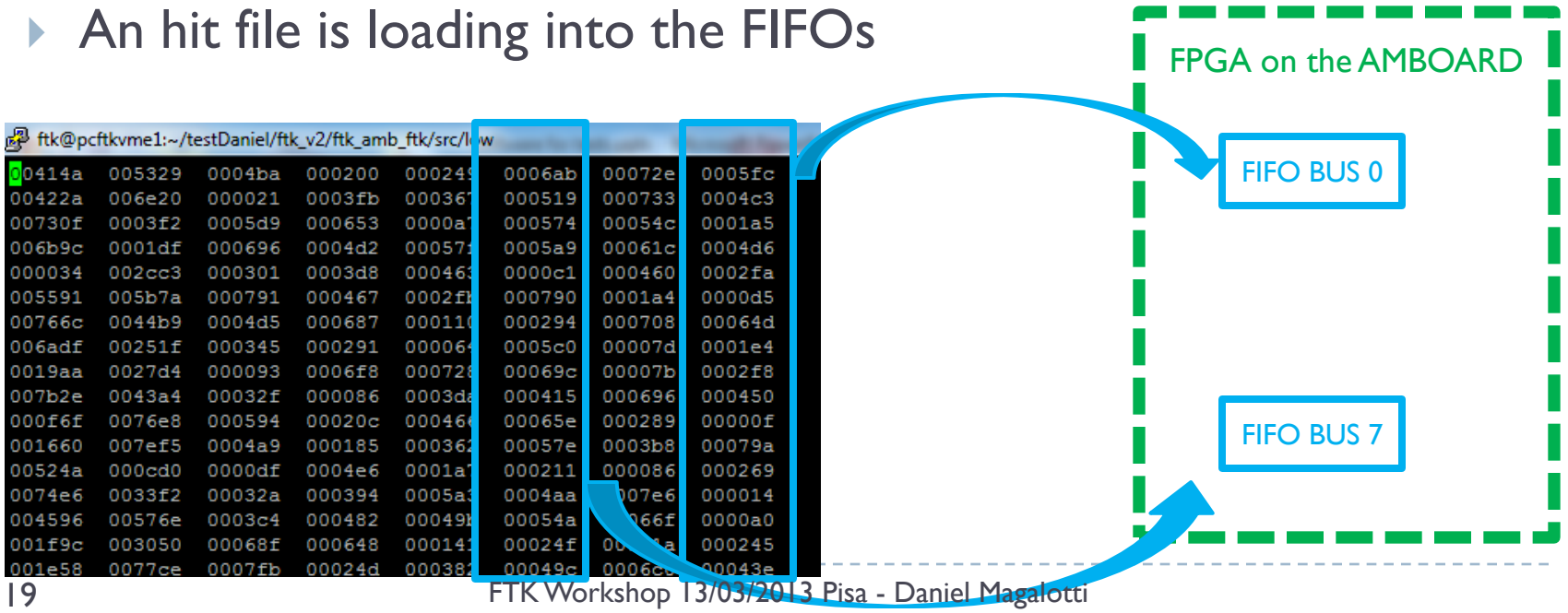
- address of memory location
- value of the memory

In the green box the are the information of the status of Spybuffer

- first free address location
- overflow status
- freeze status

Monitoring and testing tools

- The Spybuffer is used to monitoring the data flow in each input and output link between AMBoard and AUX card
  - A function give this output file format
- The VmeFifos are instantiate in each link for the input and output to simulate the eternal data flow
  - An hit file is loading into the FIFOs



Automatic test

- An automatic script is implemented to testing the entire system using tools describe above
- The list of operation
  - A simulation functions generate the **pattern bank** and both the **hit files** and the **expected road files**
  - The **hit file is loading** into the **VmeFifos**
  - A start command is sent to enable the hit distribution to the AMchip
  - The **output road** from the AMchip is stored into the output **Spybuffer**
  - The **content** of the **Spybuffer** is compared with the **expected road file**