NSWCDD: The Leader in Warfare Systems Development and Integration

CAPT Godfrey Weekes
Commanding Officer

Mr. John Fiore, SES
Technical Director

Publication Release ID: 5523
Distribution Statement A; Approved for Public Release; Distribution is unlimited.

Functional test with FPGA AND MICROCONTROLLER

Teradyne 2018
Technical Interchange Meeting

Sid Fluhrer, Alliance Support Partners, Inc.
Lin Yang, Alliance Support Partners, Inc.
Robert J. Trahan, NAVSEA
Keith J. White, NAVSEA

NAVSEA WARFARE CENTERS DAHLGREN

NAVAL SURFACE WARFARE CENTER
DAHLGREN DIVISION
DAHLGREN | DAM NECK

Publication Release ID: 5523

Test Dilemma

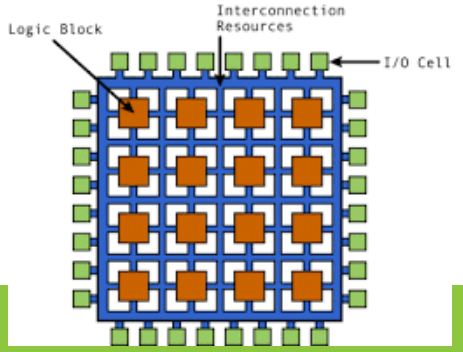
- 4 circuit cards
  - 2 with NIOS processors
  - 1 with an STM32F4
  - All with FPGAs.
- Complex circuit cards - simulation is not available to create test and diagnostics
- Need to create tests quickly that have good coverage and meaningful diagnostics.

Solution: Leverage Microprocessors and FPGAs to enhance tests

USE THE BUILD-IN SMART

FPGA and MCU

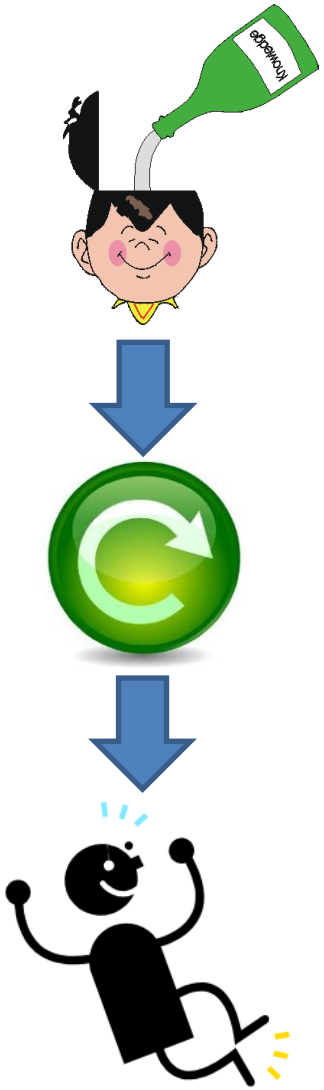
- FPGAs are ideal for wide variety of applications
  - Each series of FPGA include different features, such as embedded memory, DSP blocks, high speed transceivers, high speed IO pins, etc.
- FPGA loads configuration into its configurable logic cells when first powered up.
  - Can be re-programmed with no changes to the hardware
  - Can implement logic structures in parallel
- MCU executes program loaded in its Flash memory
  - Can load different programs at run time.
  - Can execute instructions sequentially



Publication Release ID: 5523

In-system programming (ISP)

- When power is first applied, the FPGAs must be loaded with their configuration before the circuit card can function.
- Non-volatile memories are used to store FPGA configuration when power is off.
- A temporary configuration can be loaded into the FPGA that will be lost when power is removed.
- Both non-volatile memory and temporary configurations can be programmed through the JTAG interface using a USB POD or Di.



ISP – using DI or PODs

Di	USB POD
Programming routine in SVF format	FPGA manufacturers programming software ✓
May need to reload pin memory if large amount of data needed	Should be close to device theoretical speed ✓
Standard S9 tester hardware ✓	Third party hardware built in ITA
No diagnostic message running SVF	Generate useful message during execution ✓

Programming Pods

Platform Cable USB II

- Programmer for Xilinx® PROM, FPGA and CPLD devices
- JTAG interface
- USB control

USB Blaster II

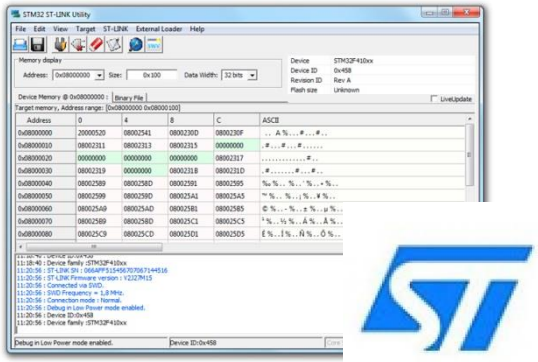
- Programmer for Altera® devices
- JTAG interface
- USB control

ST-LINK®

- Programmer for STM8 and STM32 family
- JTAG interface
- USB control

JTAG TO DEVICE, USB TO PC

Programming SOFTWARE



Xilinx iMPACT®

- Programmer for Xilinx PROM, FPGA and CPLD devices
- Part of Xilinx ISE Design Suite

Quartus®

- Programmer for Altera devices

ST-LINK® Utility

- Supports S19, HEX and binary format
- Erase, program, view and verify

OFFERS COMMAND LINE INTERFACE

Create programming command

- XILINX iMPACT® and ALTERA Quartus® can create command file from GUI

```
setMode -bs
setCable -port auto
Identify
attachflash -position 1 -spi "N25Q32"
assignfiletoattachedflash -position 1 -file
"C:/Production/1553B/1553Btest/Build/Xilinx_Runtime/CM_1553_FPGA_ITR3_1.mcs"
Program -p 1 -dataWidth 4 -spionly -e -v -loadfpga
quit
```

- ST-LINK® requires user to create BATCH file

```
C:\"Program Files”\ST-LINK Utility"\ST-LINK_CLI -Q -c UR JTAG
FREQ=9000 -ME -P
C:\Production\1553B\1553Btest\Build\ST_Runtime\Functional_Test.hex
0x08000000
-V "after_programming"
> C:\Production\1553B\1553Btest\Build\ST_Runtime\Test.log
```

Launching programmer

- Using Visual Studio's CreateProcess() to launch programmer in background. Use 'CREATE_NO_WINDOW' flag to suppress black command window. Wait for process to terminate before continuing. Collect messages to display on Test Studio console window.

```
bSuccess = CreateProcess(NULL,
        szCmdline,// command line
        NULL,// process security attributes
        NULL,// primary thread security attributes
        TRUE,// handles are inherited
        CREATE_NO_WINDOW,// creation flags
        NULL,// use parent's environment
        NULL,// use parent's current directory
        &siStartInfo,// STARTUPINFO pointer
        &piProcInfo);// receives PROCESS_INFORMATION
```

- Get result of ISP programming

```
GetExitCodeProcess(piProcInfo.hProcess, &ExitCode)
```

Example Commands

- Configure FPGA with IMPACT

```
impact -batch
C:\\Production\\1553B\\1553Btest\\Build\\Xilinx_Runtime\\1553B_U8_Tactical.cmd
```

- Run Quartus programmer Command File

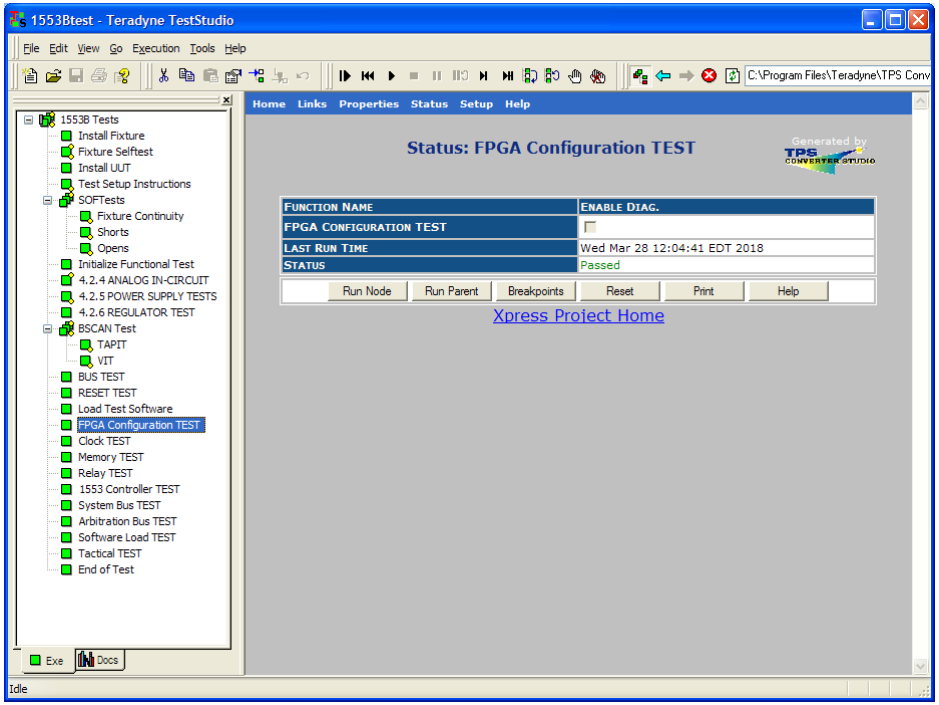
```
quartus_pgm
C:\\Production\\SM1553\\SM1553test\\Build\\Quartus_Runtime\\SM1553_FPGA_Erase.cdf
```

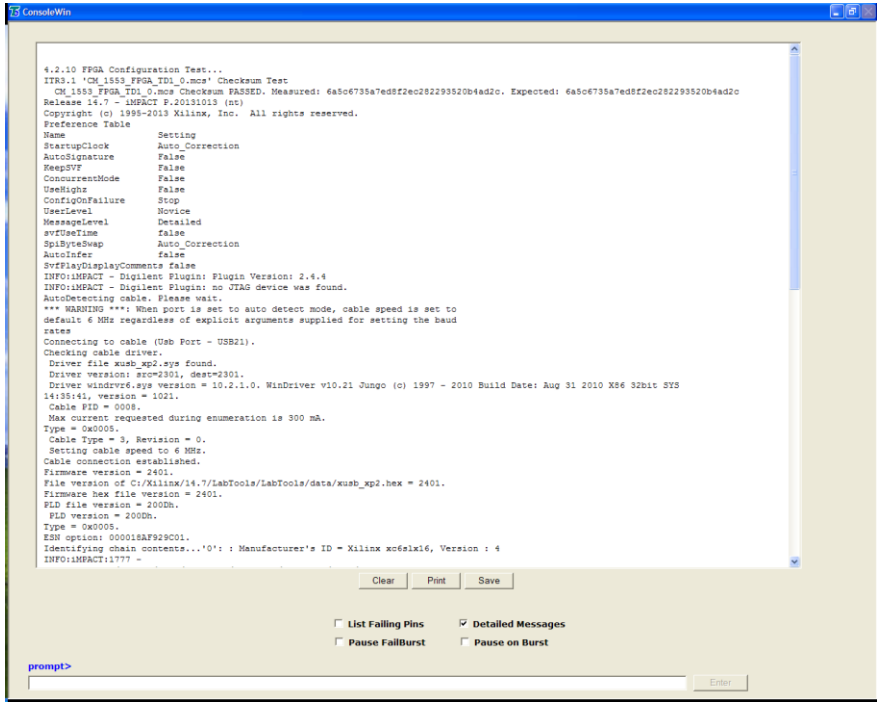
- Run ST-Link Batch File

```
CMD /C C:\\Production\\1553B\\1553Btest\\Build\\ST_Runtime\\Tactical.bat
```

Example Test Studio console window

- Power cycle and live board with configuration/program

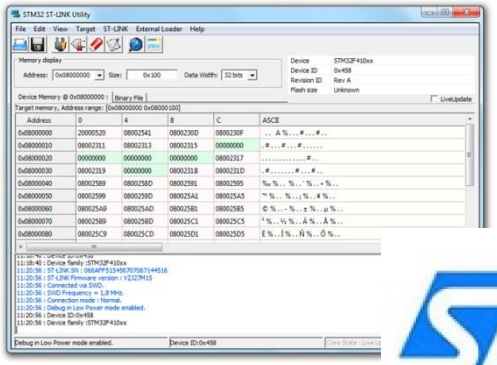




I am alive!

XILINX
ALL PROGRAMMABLE™

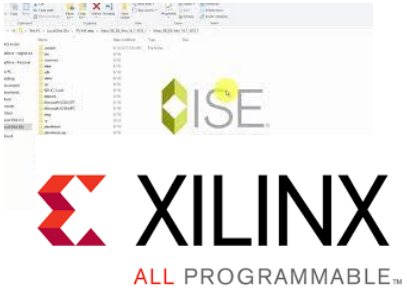
ALTERA



HOW DO I EXECUTE TEST AND GET RESULT?

®Xilinx is a registered trademark of Xilinx Corp., San Jose CA
®Altera is a registered trademark of Altera Corp., San Jose CA
®ST is a registered trademark of STMicroelectronics International N.V.

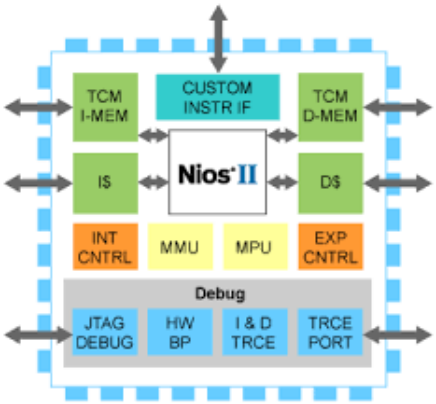
Programming SOFTWARE



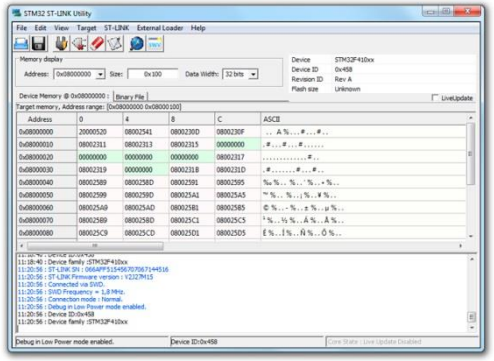
FPGA now have test configuration

Use Di to provide IO control, and acquire

Same as any digital test



Communicate thru NIOS® terminal



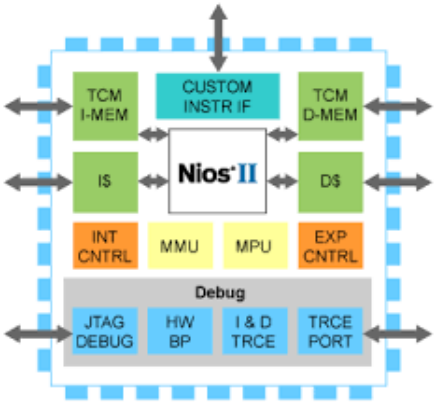
Communicate thru Open OCD

MORE COMPLICATED

®Xilinx is a registered trademark of Xilinx Corp., San Jose CA
®NIOS is a registered trademark of Altera Corp., San Jose CA

Communication with NIOS Processor

- Using the NIOS® terminal to display messages from MCU
- FPGA based NIOS® processor had a very small memory to store test program. Created many small test programs that could be loaded and then executed out of internal FPGA memory.



®NIOS is a registered trademark of Altera Corp., San Jose CA

NIOS Terminal capture routine

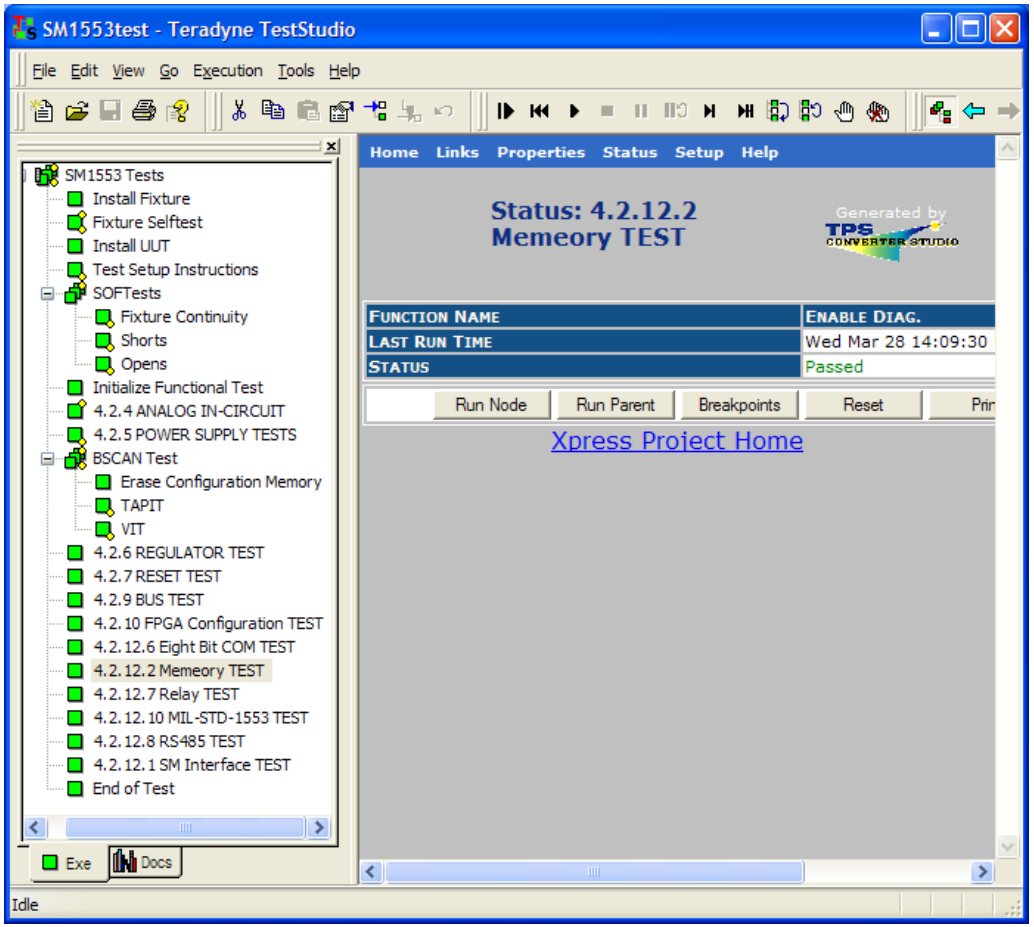
- Using CreateProcess() to launch NIOS® Terminal in background.
  - Use ‘CREATE_NO_WINDOW’ flag to suppress black command window.
  - Collect messages to display on Test Studio console window
  - Wait for MCU to send pass fail status
- Test Studio will terminate process when status is received
- Example: running memory test

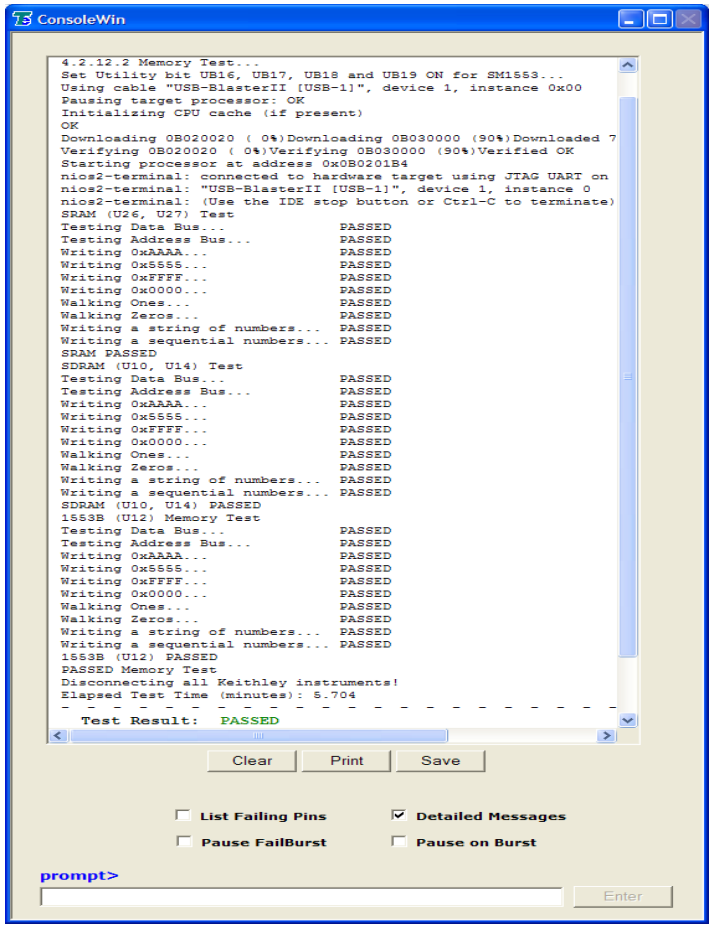
```
CMD /C
C:\\altera\\13.0sp1\\nios2eds\\\"Nios II Command Shell.bat\"
nios2-download -g /cygdrive/c/Production/SM1553/SM1553test/Build/Quartus_Runtime/MemTest.elf
```

®NIOS is a registered trademark of Altera Corp., San Jose CA

NIOS TERMINAL OUTPUT

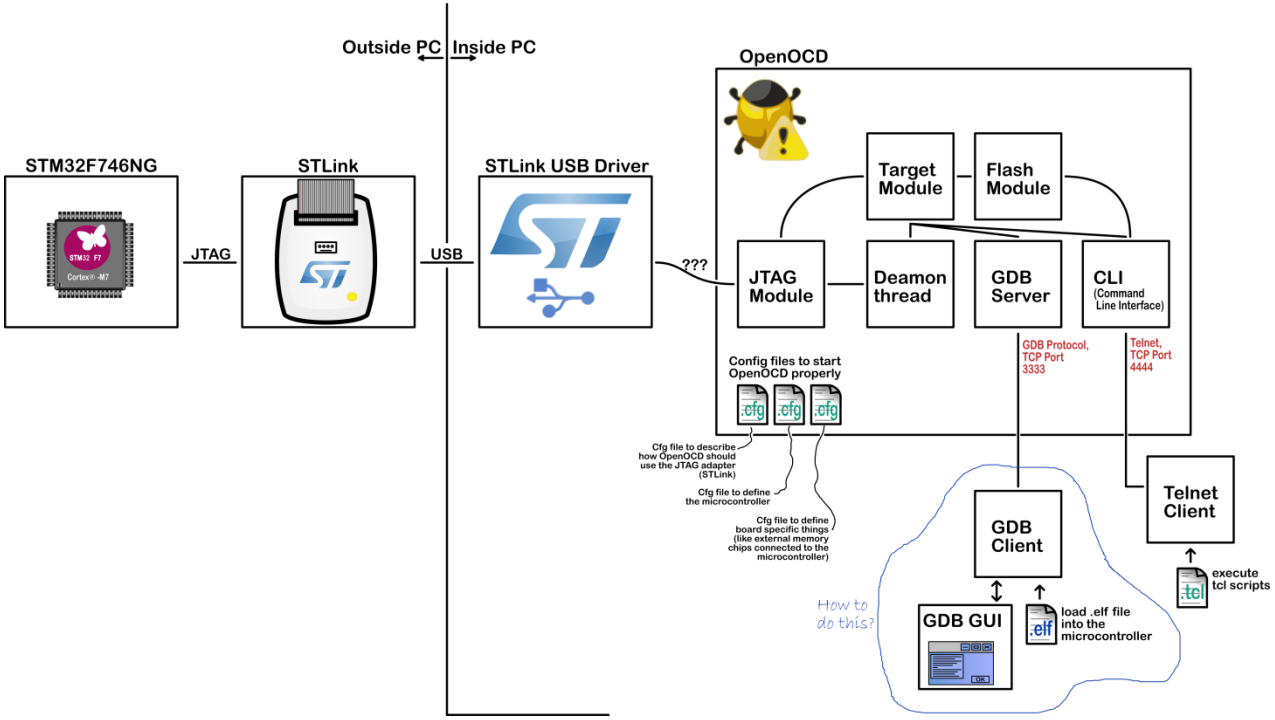
Example Test Studio Console Window





Communication with MCU – Open OCD

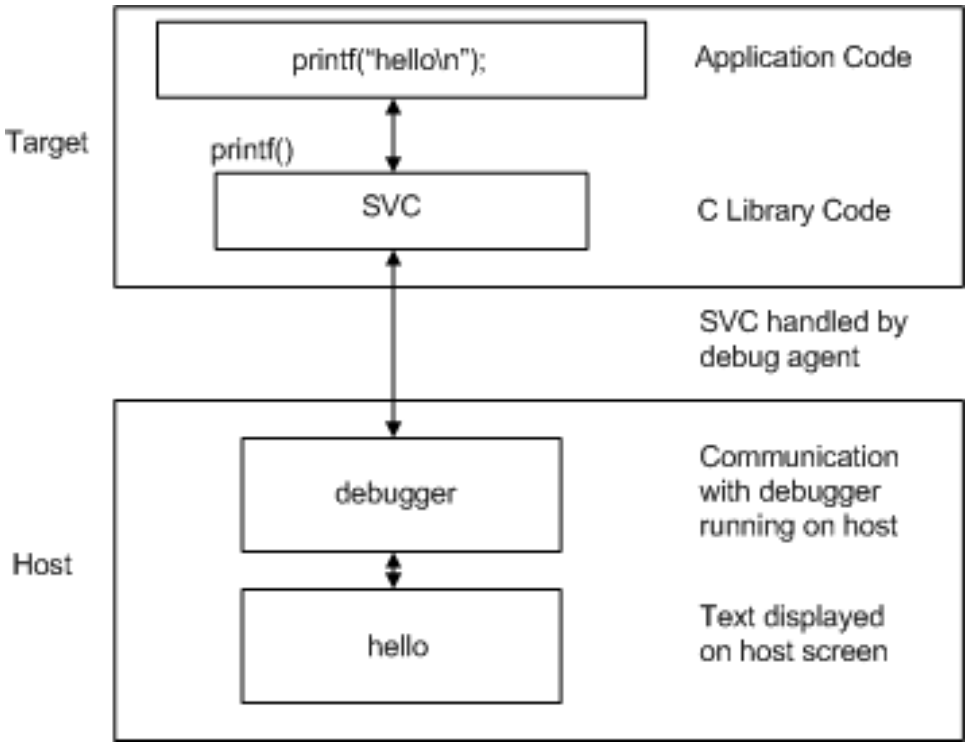
- The Open On-Chip Debugger(OCD) is going to run as a daemon process on a host PC, making use of a JTAG compliant hardware interface that connects to the target system



Semi-hosting for processor messages

Semihosting enables code running on an ARM target to communicate and use USB on tester PC running OpenOCD

- Code need to compiled with semihosting enabled
- OpenOCD need to have semihosting enabled



HOW TO EXECUTE TEST THRU OPEN OCD

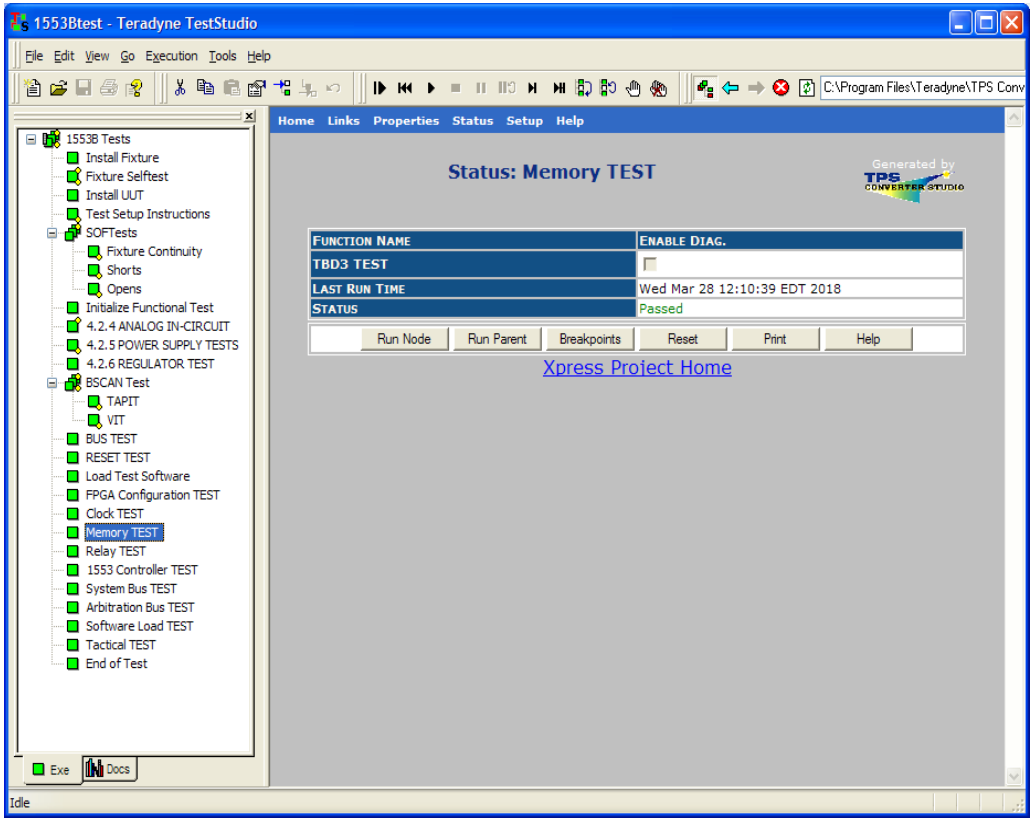
- SOIC processor, STM32F4, had large flash memory for test program storage. Loaded one large program that could handle all tests.
- Used OPENOCD to modify a register value to select which test to run.

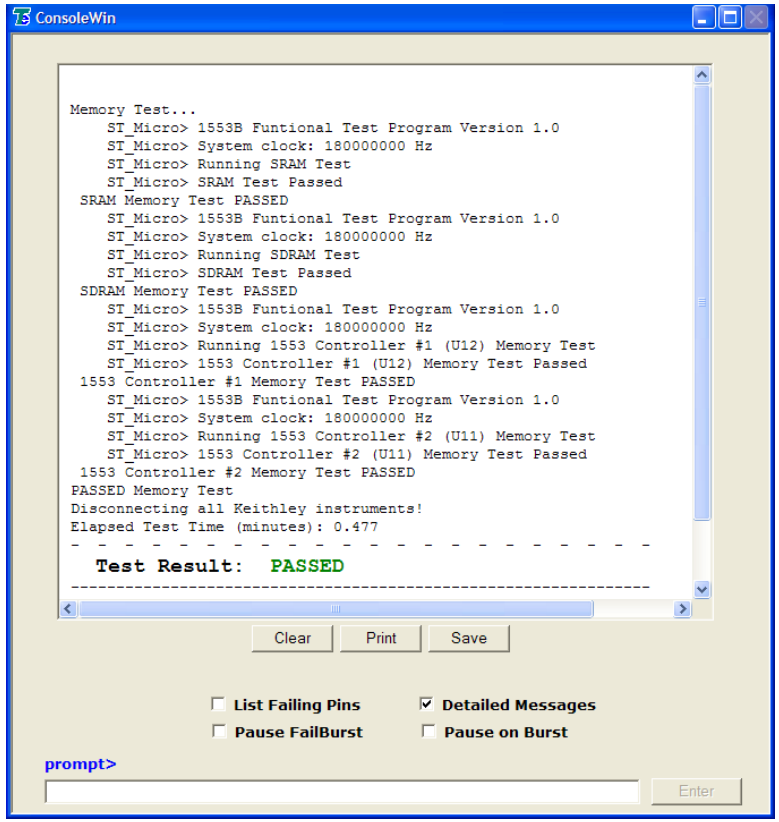
```
sprintf (ConsoleCmd,"%s %s %s %s %s %d %s %s %s %s",
                    GDB_EXECUTABLE,
                    "-ex \"target remote localhost:3333\"",        // Connect GDB Server to ST-Link
                    "-ex \"monitor arm semihosting enable\"",      // Enable Semihosting
                    "-ex \"monitor reset halt\"",                  // Reset and then Halt the processor
                    "-ex \"monitor reg r12", test_num, "\"",       // Register r12 = 'test_num'
                    "-ex \"monitor resume\"",                      // Start Program execution
                    "-ex \"disconnect\"",                          // Disconnect GDB Server from ST-Link
                    "-ex \"quit\"");                               // Quit GDB Server

if (TsTerminal_Run_ST_Link(ConsoleCmd) != TS_TERMINAL_SUCCESS)
{
                    set_cfail_(TRUE_);
                    return(BAD_4B_);
}
```

EXAMPLE TPS Output

- Example Test Studio Console Window





Example Open OCD log File

st_micro.log - Notepad

File Edit Format View Help

```
Open On-Chip Debugger 0.8.0 (2014-05-02-12:11)
Licensed under GNU GPL v2
For bug reports, read
http://openocd.sourceforge.net/doc/doxygen/bugs.html
none separate
Info : This adapter doesn't support configurable speed
Info : STLINK v2 JTAG v25 API v2 SWIM v4 VID 0x0483 PID 0x3748
Info : using stlink api v2
Info : Target voltage: 3.288772
Info : stm32f4x.cpu: hardware has 6 breakpoints, 4 watchpoints
Info : accepting 'gdb' connection from 3333
Info : device id = 0x20016419
Info : flash size = 2048kbytes
semihosting is enabled
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x080002dc msp: 0x20030000, semihosting
r12 (/32): 0x00000001
Info : dropped 'gdb' connection
   ST_Micro> 1553B Funtional Test Program Version 1.0
   ST_Micro> System clock: 180000000 Hz
   ST_Micro> Running SRAM Test
   ST_Micro> SRAM Test Passed
Info : accepting 'gdb' connection from 3333
semihosting is enabled
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x080002dc msp: 0x20030000, semihosting
r12 (/32): 0x00000002
   ST_Micro> 1553B Funtional Test Program Version 1.0
Info : dropped 'gdb' connection
   ST_Micro> System clock: 180000000 Hz
   ST_Micro> Running SDRAM Test
   ST_Micro> SDRAM Test Passed
Info : accepting 'gdb' connection from 3333
semihosting is enabled
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x080002dc msp: 0x20030000, semihosting
r12 (/32): 0x00000003
Info : dropped 'gdb' connection
   ST_Micro> 1553B Funtional Test Program Version 1.0
   ST_Micro> System clock: 180000000 Hz
   ST_Micro> Running 1553 Controller #1 (U12) Memory Test
   ST_Micro> 1553 Controller #1 (U12) Memory Test Passed
Info : accepting 'gdb' connection from 3333
semihosting is enabled
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x080002dc msp: 0x20030000, semihosting
r12 (/32): 0x00000004
Info : dropped 'gdb' connection
   ST_Micro> 1553B Funtional Test Program Version 1.0
   ST_Micro> System clock: 180000000 Hz
   ST_Micro> Running 1553 Controller #2 (U11) Memory Test
   ST_Micro> 1553 Controller #2 (U11) Memory Test Passed
Info : accepting 'gdb' connection from 3333
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x080002dc msp: 0x20030000, semihosting
shutdown command invoked
```

What is in a configuration

- Tactical Configuration
  - Firmware loaded after the UUT is verified and ready for deployment
  - Provided by customer
- Test Configuration
  - Firmware specifically designed to test all UUT functions
  - Provided by customer
- Custom Test Configuration (1 out of 4 in this test case)
  - When no test configuration was provided by customer
  - Tactical configuration is not well documented to be used
  - If the circuit card is primarily an FPGA surrounded by simple support circuitry, creating a test configuration is best

Test case – custom Configuration Using FPGA as HELPER

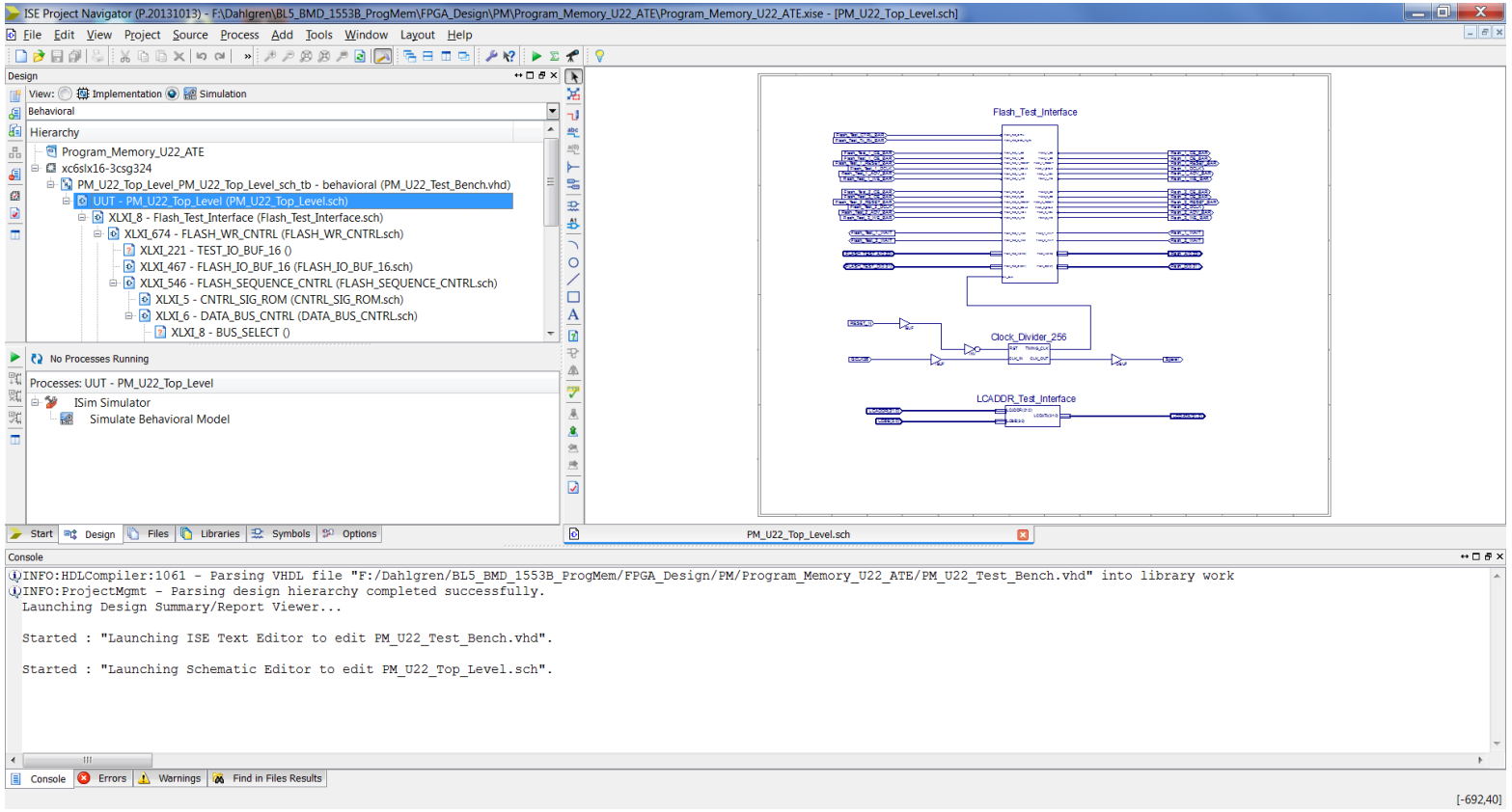
- Access to FLASH memory was through FPGA only.
- Initially FPGA created a direct connection to FLASH memory. Di had to generate the programming sequence for every memory location written. The Di pin memory load time overhead made the programming time too long!
- FPGA was modified to perform programming sequence. All the Di had to do was apply the address and data to the FPGA and wait for it to finish writing to the FLASH memory. This change reduced the FLASH memory programming time from 20 to 5.1 minutes.

Test case – custom configuration control

- FPGA was surrounded by simple support logic. This logic was divided into 13 logic blocks that required no more than 6 control signals.
- The FPGA only had 10 direct connections to a Di channel. Custom FPGA configuration was created that would route 6 of these channels to the logic block control signals. The remaining 4 channels were used to select which logic block would be tested.
- Response of the selected logic block was measured on a connector pin with Di access, or routed back through the FPGA and then back out onto the Data bus.
- The 10 direct FPGA connections were previously tested with boundary scan.

How to create custom configuration

- Use the FPGA vendor's development tool to create FPGA design

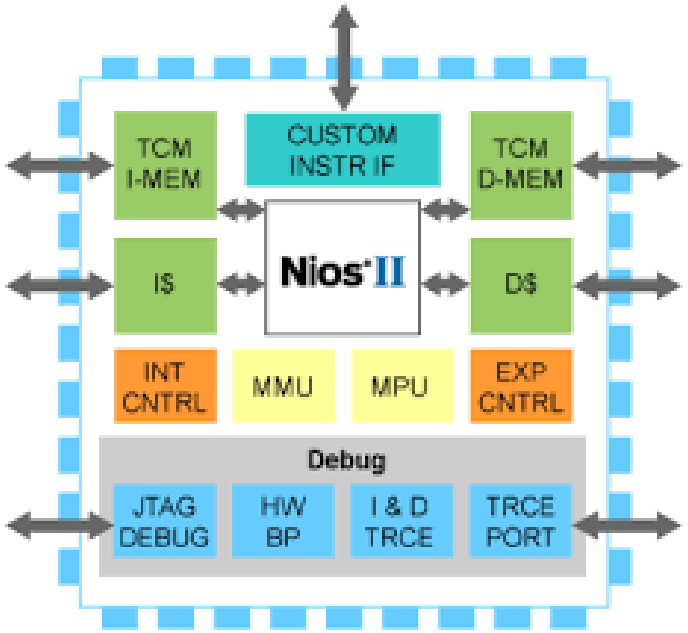


Leveraging EXISTING Code

- 3 circuit cards were Microprocessor based.
  - Tactical source code was provided in C++ / Assembly / ADA
  - Design verification source code was provided in C++
- Existing source code was used to determine base address of control registers, peripheral configuration.
  - No design documentation was provided
- Design verification code was reused in functional test.

TEST CASE – Altera functional test

- Altera FPGA uses external SRAM to store tactical software
  - How do you test the SRAM that is storing the program?
  - If SRAM is bad, other part of the circuit cannot be tested

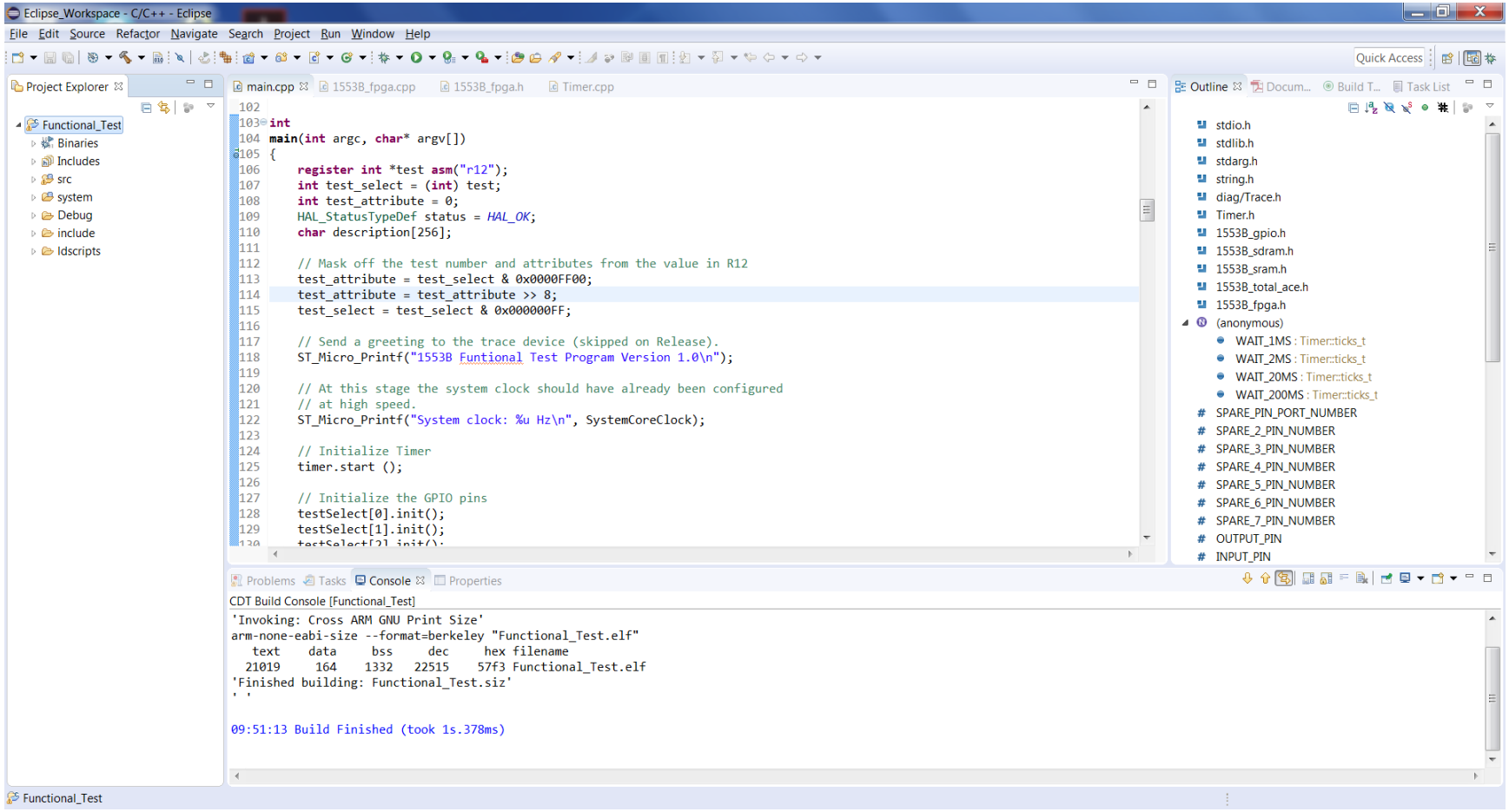


TEST CASE – Altera functional test

- The NIOS processor Altera FPGA had small internal memory.
- Write SRAM memory test that can fit in the internal SRAM
- Can also break up functional test sequence into smaller chunks
- Load individual test and then execute.

Eclipse IDE

Eclipse Workspace




Q
&
A