Application Note

ÚTIA Akademie věd České republiky
Ústav teorie informace a automatizace AV ČR, v.v.i.

Data Movers in DTRiMC tool for TE0726-03M-07S board

Jiři Kadlec, Zdeněk Pohl, Lukáš Kohout, Raissa Likhonina
kadlec@utia.cas.cz xpohl@utia.cas.cz kohoutl@utia.cas.cz likhonina@utia.cas.cz

Revision history

Rev.	Date	Author	Description
0	6.04.2020	J. Kadlec	Initial draft
1	11.08.2021	J. Kadlec	App note addressing: Data Movers in DTRiMC tool for TE0726-03M-07S board

Table of Contents

Table of Figures

Acknowledgement

This work has been partially supported from project FitOptiVis, project number ECSEL 783162 [9], [10] and the corresponding Czech NFA (MSMT) institutional support project 8A18013.

1 DTRiMC tool data movers for TE0726-03M-07S board

This application note describes package for evaluation of data movers with Design Time Resource integration of Model Composer DTRiMC tool. See Figure 1. The DTRiMC tool serves for integration AXI-S IPs for Zynq device on TE0726-03M-07S board [1], [2].

Due to the limited size of the PL logic, the AXI-S IPs is simple AXI-Steam 1024x32bit FIFO with AXI-S I/O connected to data movers. The data movers are generated in the DTRiMC tool by the Xilinx SDSoC 2018.2 compiler [3]. This application note serves for description of definition of these data movers and for comparison of basic properties (area used, performance) of these data movers. Figure 2 describes the top level, SW/HW view of the generated Zynq system.

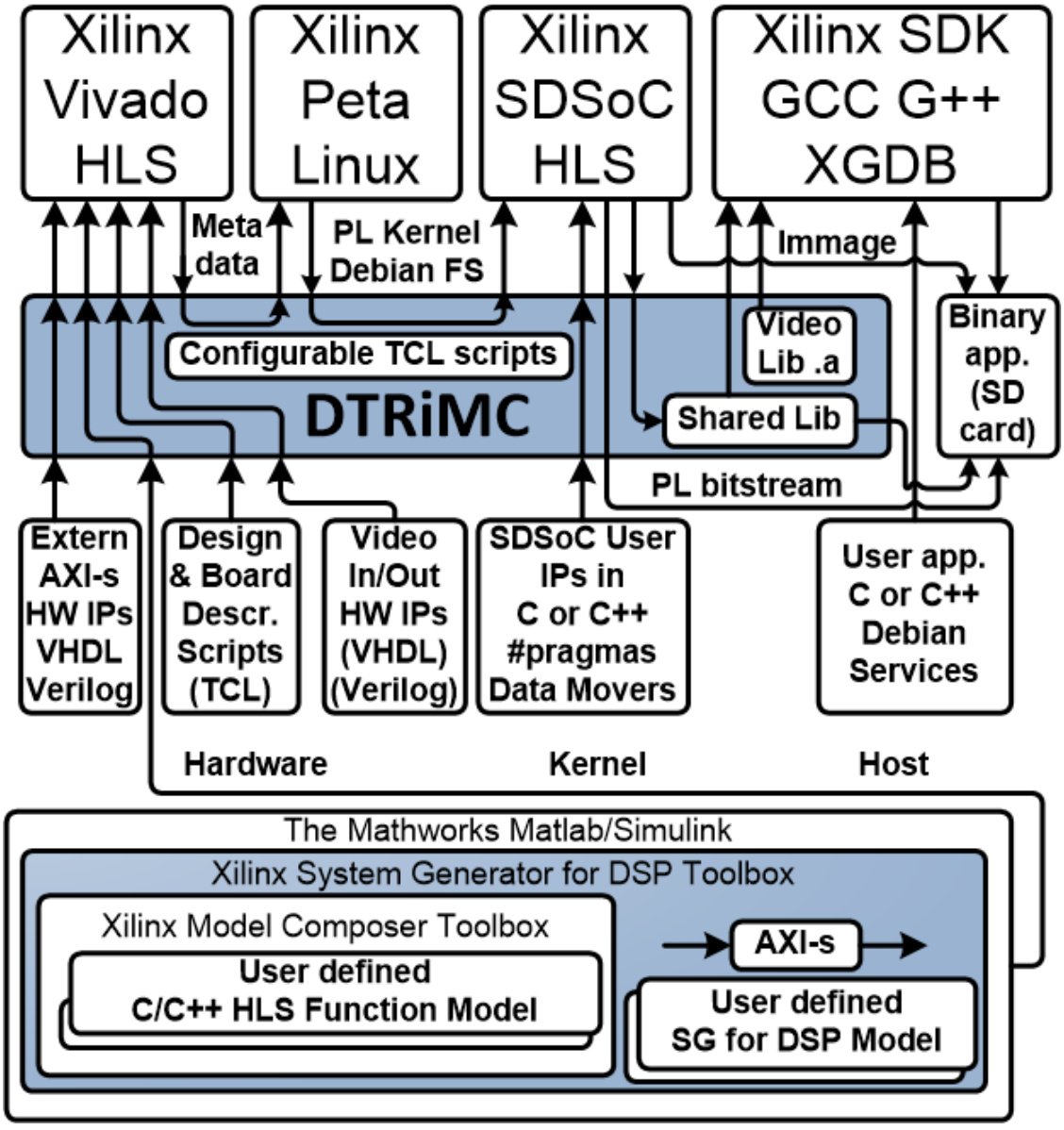


Figure 1: Design Time Resource integration of Model Composer DTRiMC tool.

ZynqBerry TE0726-03M-07S [1] works with Xilinx XC07007S-1C device with a single core ARM A9 32 bit processor, 512 MB of DDR3 memory and limited size of programmable logic on the single 28 nm chip. The ZynqBerry board has RaspBerryPi 2 form factor. It can be extended with RaspBerryPi 2 compatible shields. The ZynqBerry board TE0726-03M-07S is designed and manufactured by company Trenz Electronic [1], [2], [6].

User SW can be cross-compiled by the g++ compiler in Xilinx SDK [4] on Win 10 PC. The Debian utility „make“ can be also used for compilation of user C++ SW on A9 processor.

The HW data communication is represented for the SW developer as a shared C++ library with simple SW API, identical for several HW data-mover alternatives generated by the Xilinx SDSoC 2018.2 compiler [3].

2 8xSIMD FP03x8 floating point accelerator for TE0726-03M-07S board

The FP03x8 HW accelerator serves for run-time reprogrammable 8xSIMD single precision floating point computations. The internal structure of data path is described in Figure 2.

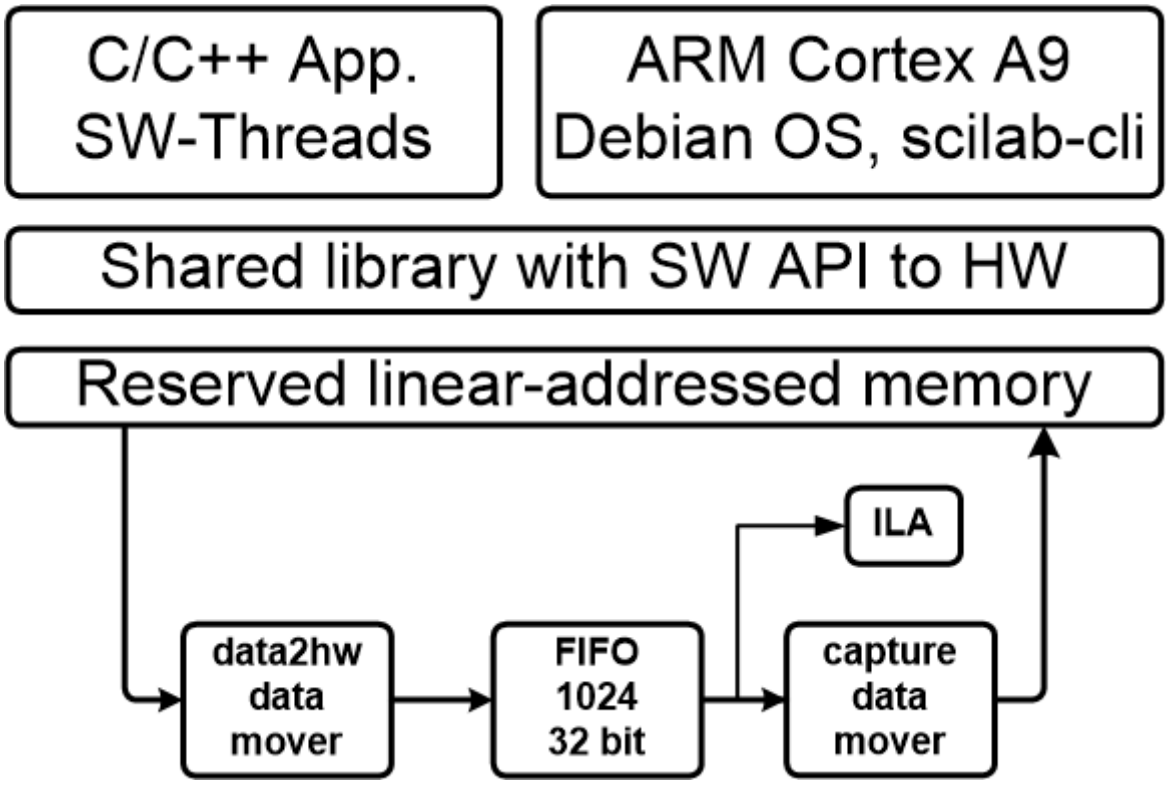


Figure 2: Data path implemented in the programmable logic of the TE0726-03M-07S board

Input:

- Data copied via AXI stream interface controlled by HW data mover `data2hw` from the reserved part of the ARM processor DDR3 memory.

Output:

- Data copied via AXI stream interface controlled by HW data mover `capture` to the reserved part of the ARM processor DDR3 memory.

Connectivity:

- AXI stream data input from ARM to the HW data path. The side channel indicates the last transferred word sent.
- AXI stream data output from the HW data path to ARM. The output AXI stream side channel indicates the last transferred word.

Interfaces:	Device:	Clock:
• Data stream AXI-S 32 bit	xc7z007sclg225-1	115 MHz
• ARM A9 system clock	xc7z007sclg225-1	666 MHz

3 ARM SW API for Streaming of Data

Serial streaming SW API for HW accelerated data movement from/to the non-cacheable linear address space memory is defined by sequence of two calls to these asynchronous non-blocking functions:

```
void data2hw_wrapper(unsigned *src, unsigned len);
void capture_wrapper(unsigned *storage, unsigned len);
```

Example:

```
data2hw_wrapper((unsigned*)A1_A2, len);    //1
capture_wrapper((unsigned*)B1_B2, len);    //2
…
sds_wait(1);
sds_wait(2);
```

`unsigned *src` is pointer to memory start of vector of 32bit wide words of data source
`unsigned *storage` is pointer to memory start of vector of 32bit wide words of data destination

`sds_wait()` synchronization functions are implemented as:

- Functions performing SW pooling in case of DMA and Zero Copy (ZC) data movers.
- Interrupt service routines initiated by an interrupt from the HW data mover in case of the Scatter Gather (SG) HW data movers.

Arm A9 processor can execute some additional SW instructions in parallel with HW accelerated copy of data. This SW code should be located in place marked by the … dots.

Calls to blocking functions `sds_wait(1)` and `sds_wait(2)` is obligatory. ARM A9 processor SW waits there for the complete end of the HW supported data transfer.

All HW data movers supporting the data communication are represented for the SW developer in shared C++ Debian OS libraries.

4 C++ projects for evaluation of HW accelerated copy of data

The evaluation package accompanying this application note contains the Xilinx Vivado 2018.2 base support package HW project, four Xilinx SDSoC 2018.2 HW projects and four Xilinx SDK C++ SW application projects serving for evaluation of four HW data movers.

The base support package HW project contains in PL part of the device one 32 bit wide AXI-Stream FIFO HW IP. The SDSoC projects generate four versions of HW data-movers from specification in format of CPP functions with pragmas. The access functions for these SDSoC generated HW data movers are exported by the SDSoC projects into four shared libraries. The shared libraries are linked with the Debian OS SW user applications in the Xilinx SDK SW projects. Test applications run on the 32 bit single core, ARM A9 processor of the xc7z007sclg225-1 device on the TE0726-03M-07S board.

- **Zero-Copy (ZC)** data movers. HW accelerated copy of data from/to reserved, non-cacheable, linear address space memory area.

  Directory (C++): **copy_zc_1x1_sw**
  SW C++ project: **copy_1x1_sw**
  Shared C++ library: **./Debug/sd_card/libcopy_zc_1x1_hw.so**
  Shared C++ library: **./Release/sd_card/libcopy_zc_1x1_hw.so**

- **Dma (DMA)** data movers. HW accelerated copy of data from/to reserved, non-cacheable, linear address space memory area.

  Directory (C++): **copy_dma_1x1_sw**
  SW C++ project: **copy_1x1_sw**
  Shared C++ library: **./Debug/sd_card/libcopy_dma_1x1_hw.so**
  Shared C++ library: **./Release/sd_card/libcopy_dma_1x1_hw.so**

- **Scatter-Gather (SG)** data movers. HW accelerated copy of data from/to reserved, non-cacheable, linear address space memory area.

  Directory (C++): **copy_sg_1x1_sw**
  SW C++ project: **copy_1x1_sw**
  Shared C++ library: **./Debug/sd_card/libcopy_sg_1x1_hw.so**
  Shared C++ library: **./Release/sd_card/libcopy_sg_1x1_hw.so**

- **Scatter-Gather Malloc (SG-malloc)** data movers. HW accelerated copy of data from/to standard Debian OS memory area.

  Directory (C++): **copy_sg_malloc_1x1_sw**
  SW C++ project: **copy_1x1_sw**
  Shared C++ library: **./Debug/sd_card/libcopy_sg_malloc_1x1_hw.so**
  Shared C++ library: **./Release/sd_card/libcopy_sg_malloc_1x1_hw.so**

The shared Debian Stretch 9.8 OS libraries for the SDK 2018.2 C++ SW flow (g++ compiler) provide interfaces to the HW data movers. In all four cases, the **copy_1x1_sw** project demonstrates performance of HW supported data copy for a single precision floating point matrix [64x64].

HW data mover performance is compared with the optimized (-O3) ARM host SW implementation of SW data copy of a single precision floating point matrix [64x64] from user

space memory to linear addressable non-cacheable memory area and back to user space memory.

TE0726M-07S	***MByte/s***
ZC HW data movers	170.4
ZC SW copy	19.9
DMA HW data movers	151.1
DMA SW copy	19.9
SG HWdata movers	79.1
SG SW copy	19.9
SG-malloc HW data movers	9.8
SG-malloc SW copy	229.6

Figure 3: Performance of data copy for different data movers and for SW.

Matrix [64x64] has 4096 32 bit FP32 (single precision floating point) words. HW supported copy is performed in four blocks of 1024 32 bit words. HW data movers can copy one 32 bit word each 125 MHz clock in all four cases. This corresponds to the peak performance 500 Mbyte/s. However, this performance is not reached due to the ARM SW overhead related to data mover initialization.

The ARM SW data mover initialization overhead is relatively short in case of `ZC` and `DMA` data movers. Data are present in the linear addressable, non-cacheable memory.

The initialization overhead is longer in case of `SG` data mover even if data are present in the linear addressable, non-cacheable memory. This is due to the overhead related to creation, and release of SW threads implementing the interrupt service functions.

`SG-malloc` performs copy of data allocated by standard C++ `malloc()` function in standard, cached Debian user space memory. `SG-malloc` data movers have relatively large overhead and relatively low performance if transferred data vectors are short (like in the implemented example).

The advantage of `SG-malloc` data movers is the possibility to work directly with the Debian user space data allocated by the standard C/C++ `malloc()` function. Also SW copy performance is high in the `SG-malloc` case. SW copy performed by ARM works with potentially cached data in the Debian user space.

`SG` and `SG-malloc` data movers use interrupts and interrupt service routines to indicate the end of data mover operation. Interrupt service routines are implemented as non-active Debian OS process threads activated only by the coming interrupt. This solution removes the SW pooling and results in reduced ARM processor load. The associated cost is an additional SW overhead related to creating, execution and termination of interrupt service process threads. `ZC` and `DMA` data movers use SW pooling in the `sds_wait()`synchronisation functions. It requires 100% load of one ARM A9 processor core. The TE0726M-07S device works with single core ARM A9 processor.
Four versions of data movers provided in the evaluation package accompanying this application note is presented in Figure 4, Figure 5, Figure 6 and Figure 7.

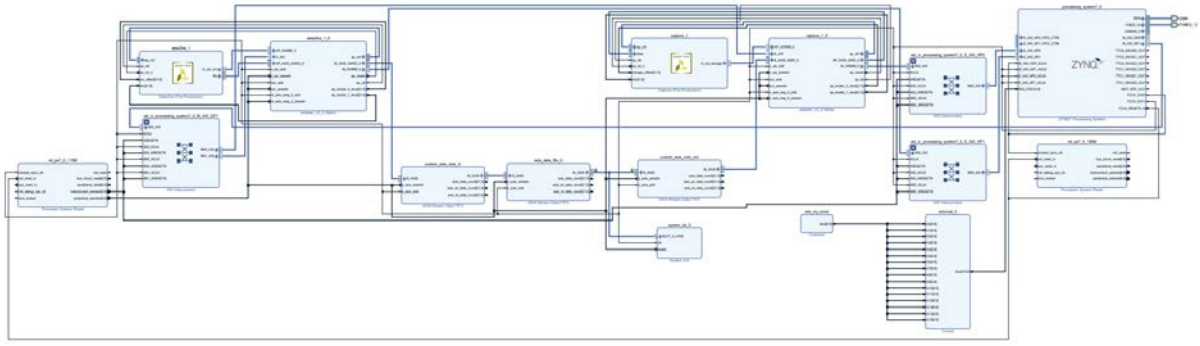


Figure 4: Design with ZC data movers.

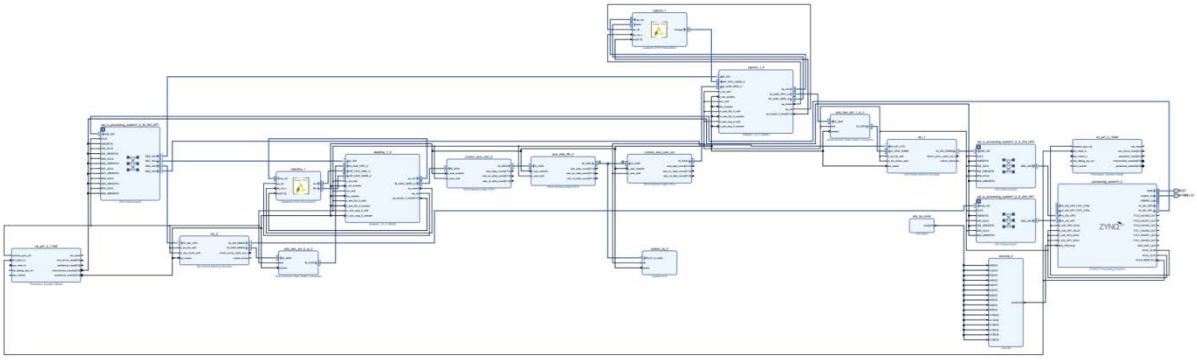


Figure 5: Design with DMA data movers.

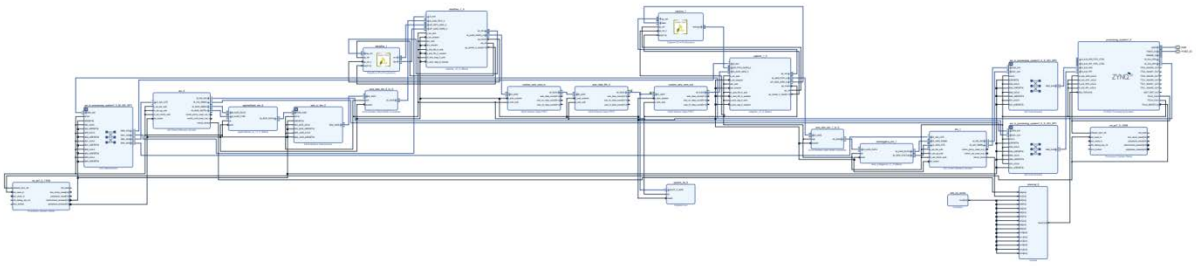


Figure 6: Design with SG data movers.

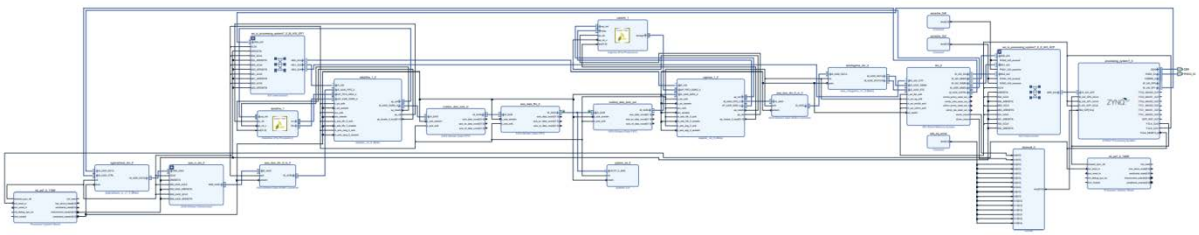


Figure 7: Design with SG-malloc data movers.

Designs with different data movers have these programmable logic resource requirements. See Figure 8.

TE0726M-07S	*Site Type*	*Used*	*Available*	*Util%*
ZC	Slice LUTs	7581	14400	52.65
ZC	Slice Registers	10864	28800	37.72
ZC	Block RAM Tile	9	50	18.00
DMA	Slice LUTs	8421	14400	58.48
DMA	Slice Registers	12492	28800	43.38
DMA	Block RAM Tile	12.5	50	25.00
SG	Slice LUTs	12819	14400	89.02
SG	Slice Registers	20555	28800	71.37
SG	Block RAM Tile	22	50	44.00
SG-malloc	Slice LUTs	10856	14400	75.39
SG-malloc	Slice Registers	16715	28800	58.04
SG-malloc	Block RAM Tile	17.5	50	35.00

Figure 8: Programmable logic resources used in designs with different data movers.

Design with ZC data movers use minimal PL resources. See Figure 4 and Figure 8.

Design with DMA data movers is using two AXI Direct Memory Access HW IPs. It is presented in Figure 5.

Design with SG data movers is using two AXI Direct Memory Access HW IPs configured for SG DMA data transfers. Data movers are connected to S_AXI_HP0 and S_AXI_HP1 high performance ports of the ZYNQ processor. Design is using two interrupts. Design is presented in Figure 6.

Design with SG-malloc data movers is using one AXI Direct Memory Access HW IPs configured for SG DMA data transfers. It is connected to the S_AXI_ACP advanced cache coherent port of the ZYNQ processor. Design is using two interrupts. Design is presented in Figure 7.

Design with SG and SG-malloc data movers require nearly all PL resources of the small TE0726M-07S device. See Figure 6, Figure 7 and Figure 8.

Details of all four designs can be analysed in the block diagrams (in pdf vector format). These diagrams are included in the evaluation package accompanying this application note.

Designs can be recompiled in SDSoC 2018.2 and Vivado 2018.2 can be used to see all details and configuration of all HW IPs.

Debian console listing from SW application `copy_1x1_sw.elf` on system with ZC data movers is presented in Figure 9.

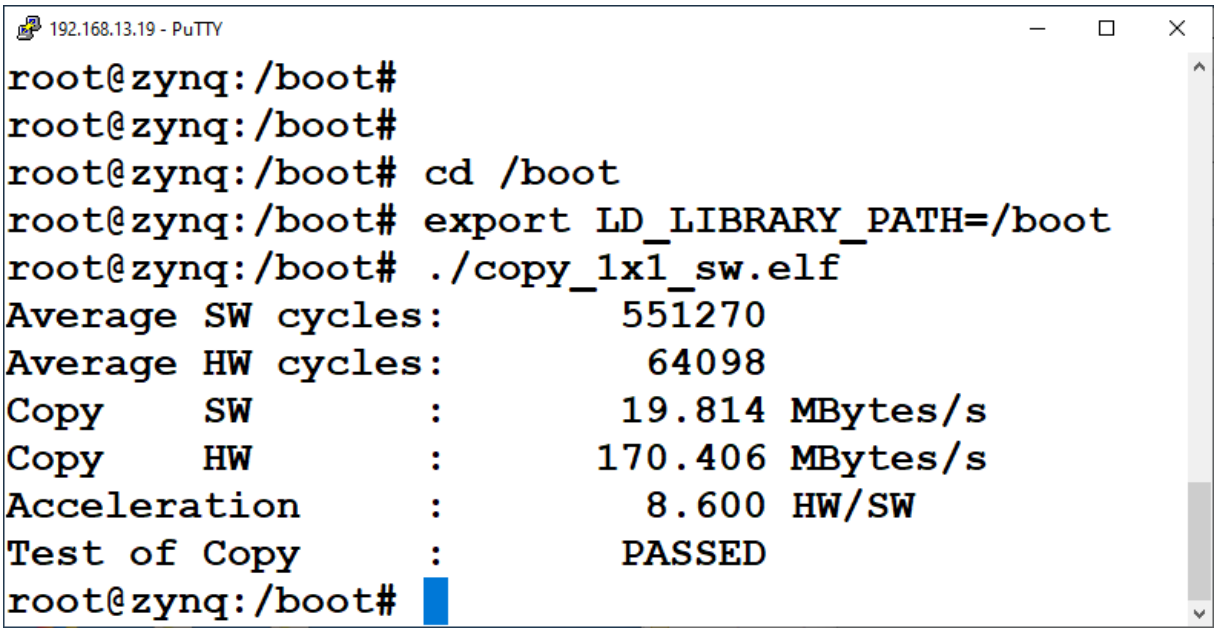


Figure 9: Debian terminal, TE0726M-07S device with ZC data movers in the PL.

5 Power consumption

Power consumption is measured on input power line 5V. All power supply is derived from this single power source.

TE0726M-07S	***Power consumption [W]***
ZC HW data movers, running copy_1x1_sw	2.75
ZC Debian OS, idle	2.45
DMA HW data movers, running copy_1x1_sw	2.75
DMA Debian OS. idle	2.45
SG HW data movers, running copy_1x1_sw	2.75
SG Debian OS, idle	2.50
SG-malloc HW data movers, running copy_1x1_sw	2.70
SG-malloc Debian OS, idle	2.50

Figure 10: Power consumption for data copy for different data movers and Debian OS.

TE0726M-07S device with ZC data movers in the PL.with Debian OS (right part) is presented on Figure 11. The left part of the screen is a remote X11 desktop (running on a Win 10 PC with VMWare player Ubuntu OS).

TE0726M-07S is connected via wired Ethernet and Ubuntu PuTTY terminal with enabled X11 forwarding to the screen and executes Debian `mousepad` text editor (running on Arm A9) with Debian file system. Mouse and keyboard for the `mousepad` application is also forwarded to the Arm by the remote X11 connection.

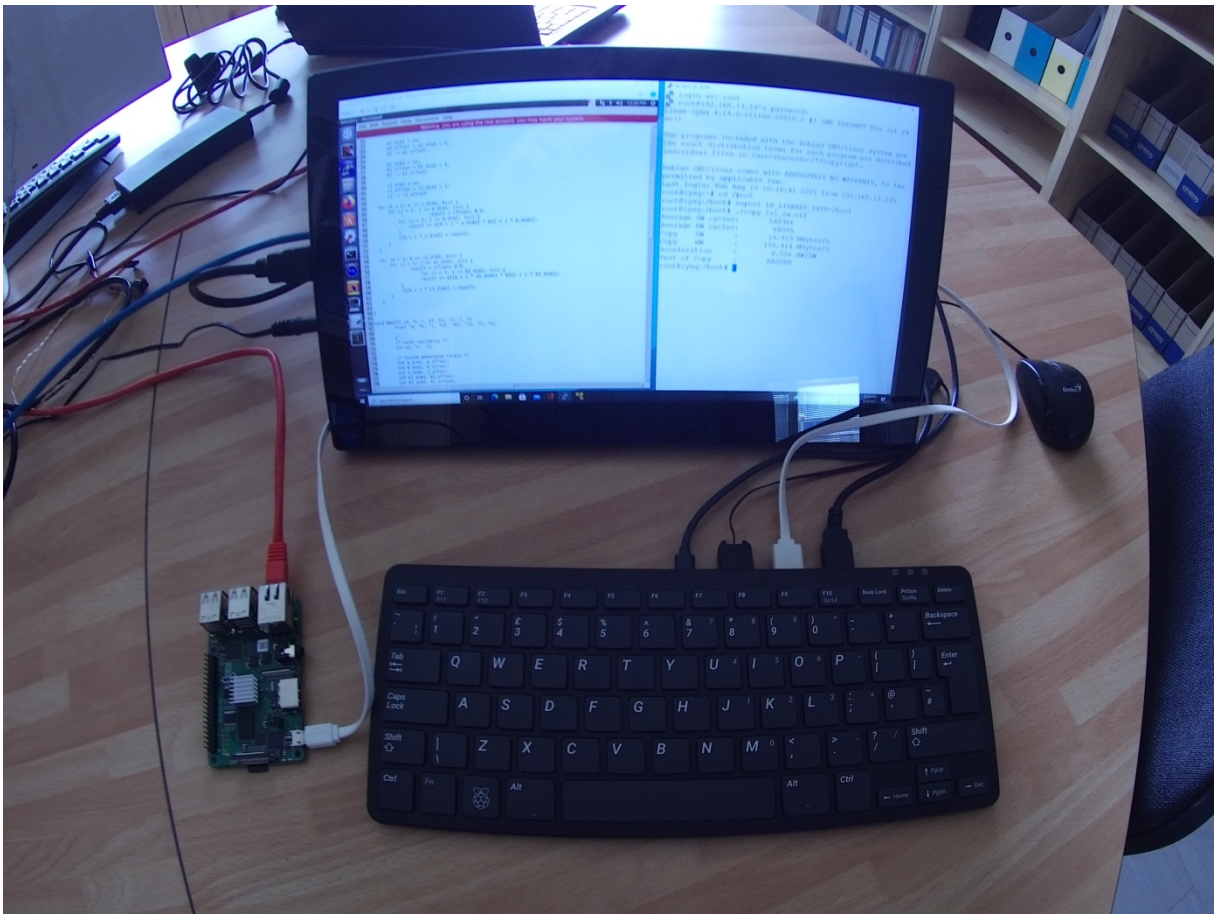


Figure 11: TE0726M-07S device, mousepad editor on remote X11 desktop, terminal.

6 ILA – In-circuit Logic Analyzer

System created by the Design Time Resource integration of Model Composer DTRiMC tool default scripts includes HW IP of the Vivado-Lab tool 2018.2 ILA – In-circuit Logic Analyzer. It is connected to the output of the 1024x32bit FIFO HW IP. See Figure 2. The start of ILA capturing can be triggered by specific logic combination of input values defined by user.

ILA monitor displays values of AXI-S stream bus with the 115 MHz clock resolution. It is configured to store 1024 data samples. See Figure 12.

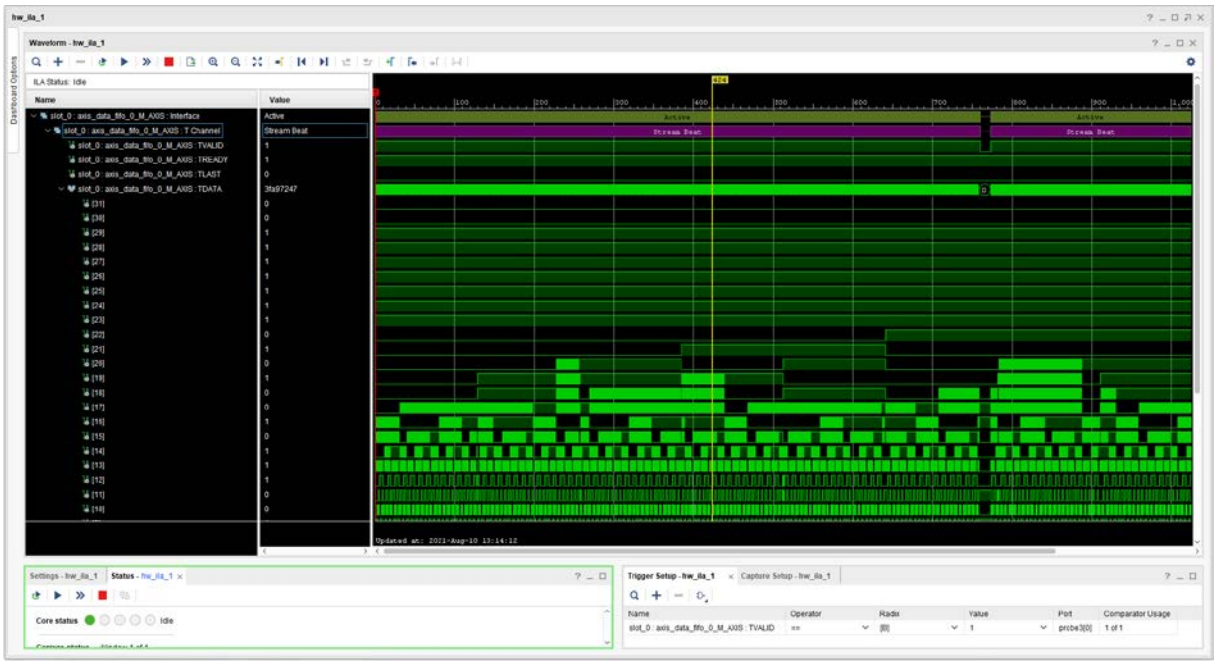


Figure 12: AXI-S bus captured by ILA for design with ZC data movers.

Figure 12 presents example of data transfer captured by ILA. It corresponds to ARM host SW `copy_1x1_sw.elf` and HW design with ZC data movers. It performs HW accelerated copy of [64x64] floating point matrix. AXI-S control signals and part of 32 bit wide floating point data are displayed at bit level.

7 References

[1] "ZynqBerry" Module with Xilinx Z-7007S single-core in Raspberry Pi Form Factor
https://shop.trenz-electronic.de/en/TE0726-03-07S-1C-ZynqBerry-Module-with-Xilinx-Z-7007S-single-core-in-Raspberry-Pi-Form-Factor?c=350

[2] Trenz Electronic Wiki – TE0726 Resources
https://wiki.trenz-electronic.de/display/PD/TE0726+Resources

[3] SDSoC - 2018.2 Full Product Installation
https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vitis/archive-sdsoc.html

[4] Software Development Kit Standalone Web Install Client - 2018.2
https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vitis/archive-sdk.html

[5] Vivado Lab Solutions - 2018.2
https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/archive.html

[6] "ZynqBerry" Module with Xilinx Zynq-7010 in Raspberry Pi Form Factor
https://shop.trenz-electronic.de/en/TE0726-03M-ZynqBerry-Module-with-Xilinx-Zynq-7010-in-Raspberry-Pi-Form-Factor?c=350

[7] FP01x8 Accelerator on TE0726-03M
http://sp.utia.cz/index.php?ids=results&id=te0726_fp01x8

[8] DTRiMC tool for TE0726-03M board
http://sp.utia.cz/index.php?ids=results&id=2018_2_te0726_fp01x8_ila_DTRiMC

[9] FitOptiVis From the cloud to the edge – smart IntegraTion and OPtimisation Technologies for highly efficient Image and VIdeo processing Systems
https://fitoptivis.eu/

[10] FitOptiVis From the cloud to the edge - smart IntegraTion and OPtimization Technologies for highly efficient Image and VIdeo processing Systems
http://sp.utia.cz/index.php?ids=projects/fitoptivis

8 APPENDIX - Confidence test

This is basic confidence test of the evaluation package.

Unzip evaluation package to Win 10 directory of your choice. We will use:
c:\home\work\TS82fp01x8_TE0726_DTRiMC_xc7z07s

Precompiled HW and SW projects are located in directory:
c:\home\work\TS82fp01x8_TE0726_DTRiMC_xc7z07s\xc7z07s_deb_eval_fifo_ila_release\copy_zc_1x1_sw

Compressed SD card image with ARM Debian OS is located in directory:
c:\home\work\TS82fp01x8_TE0726_DTRiMC_xc7z07s\xc7z07s_deb_eval_fifo_ila_release_sdcard

INSTALLATION OF TOOLS

- Install Xilinx SDK 2018.2 on Win 10 PC 64 bit [3].
- Install Xilinx Lab Tools 2018.2 on Win 10 PC 64 bit [5].
- Install Win32DiskImager for writing of image to 16 GB SD card, (Class 10).
- Install Putty (for USB based serial console and Ethernet based serial console).
- Unzip ARM Debian OS disk image on Win 10 PC and use the Win32DiskImager to write the disk image (16 GB) from the PC to the 16 GB SD card (Class 10).

Before test on the ZynqBerry board, you have to write to the on-board FLASH the correct BOOT.BIN file with the bit-stream. It is done by performing these steps:

- Remove SD card from the TE0726-03M-07S board.
- Connect the TE0726-03M-07S board to PC by the USB serial terminal cable.
- Copy the BOOT.BIN file from

  c:\home\work\TS82fp01x8_TE0726\xc7z07s_deb_eval_fifo_ila_release\copy_zc_1x1_sw\Release\sd_card\BOOT.BIN
  to
  c:\home\work\TS82fp01x8_TE0726\xc7z07s_deb_eval_fifo_ila\zsys\prebuilt\boot_images\m\NA\BOOT.BIN

- Change directory to
  c:\home\work\TS82fp01x8_TE0726\xc7z07s_deb_eval_fifo_ila\zsys

  Execute this script in Win 10 terminal:

  program_flash_binfile.cmd

  This script will write content of the **BOOT.BIN** file to the ZynqBerry board flash.
- Power-off the TE0726-03M-07S board by removing the USB cable from the PC.

HW SETUP

- Insert the SD card with Debian OS disk image to the TE0726-03M-07S board.
- Connect PC and TE0726-03M-07S board to Ethernet.

- Connect USB serial terminal cable to TE0726-03M-07S to PC. This will power-on the board.

TEST

- TE0726-03M-07S board will start to boot Debian OS. The boot process starts by reading data from the BOOT.BIN present in the internal flash. Only the second stage of the boot process is performed from the SD card.
- In Win 10 PC, open Putty terminal. Set it to:
  (115200 bps, 8 data bits, stop bit 1, parity none, flow control off)
- Use Putty terminal to login as user: **`root`** password: **`root`**
- Change directory to **`/boot`**
- Export path to the shared library. Type in the Putty Debian OS terminal:

  ```
  export LD_LIBRARY_PATH=/boot
  ```

- Start application code by typing in the Putty Debian OS terminal:

  ```
  ./copy_zc_1x1_sw.elf
  ```

RESULT

- The application will copy single precision floating point by:
  - SW on host ARM A9 processor
  - HW acclerated on host ARM A9 processor by zero copy (ZC) data movers.
- Results of ARM SW and HW accelerated copy are compared to be identical and Mbyte/s performance is measured, computed and displayed. See Figure 9.

Compilation and debug of projects from source code

The evaluation package includes SW projects for Xilinx SDK 2018.2 tool running on Win 10.

These projects can be recompiled for ARM and executed on Zynq with or without debugging support. Open SDK 2018.2 tool, in this working directory:

`c:\home\work\TS82fp01x8_TE0726_DTRiMC_xc7z07s\xc7z07s_deb_eval_fifo_ila_release\copy_zc_1x1_sw\`

Projects in this directory link to this shared library:
`c:\home\work\TS82fp01x8_TE0726_DTRiMC_xc7z07s\xc7z07s_deb_eval_fifo_ila_release\copy_zc_1x1_sw\Release\sd_card\libcopy_zc_1x1_hw.so`
or
`c:\home\work\TS82fp01x8_TE0726_DTRiMC_xc7z07s\xc7z07s_deb_eval_fifo_ila_release\copy_zc_1x1_sw\Debug\sd_card\libcopy_zc_1x1_hw.so`

Projects have two configurations:

- **Debug** for debugging with –O0 flag with debug information symbols included.
- **Release** for maximal performance with -O3 flag and without debug symbols.

You can modify and re-compile the SW code in the Xilinx SDK 2018.2 tool on Win 10 PC.

DEBUG of SW application from Xilinx SDK 2018.2

The application can be executed or debugged from the SDK 2018.2 tool. SDK debugger needs environment information about the location of the actual shared library on the board.

Before start of Debug, copy complete content of this Debug directory:

```
c:\home\work\TS82fp01x8_TE0726_DTRiMC_xc7z07s\xc7z07s_deb_eval_fifo_ila_release\copy_zc_1x1_sw\Debug\sd_card\*.*
```

to the top directory of the SD card visible in the Win 10 PC:

```
*.*
```

Insert the SD card to the board and power on the board by connecting USB cable to the PC.

Alternatively, you can also use the Ethernet connection to perform binary copy to the SD card. If you use Ethernet, you have to type in the Debian OS console

```
reboot
```

to reboot the Debian OS.

To debug from the PC in the Xilinx SDK debugger GUI, the Zynq TCF server has to be accessible from the PC via Ethernet. This can be tested in the SDK. See Figure 13.

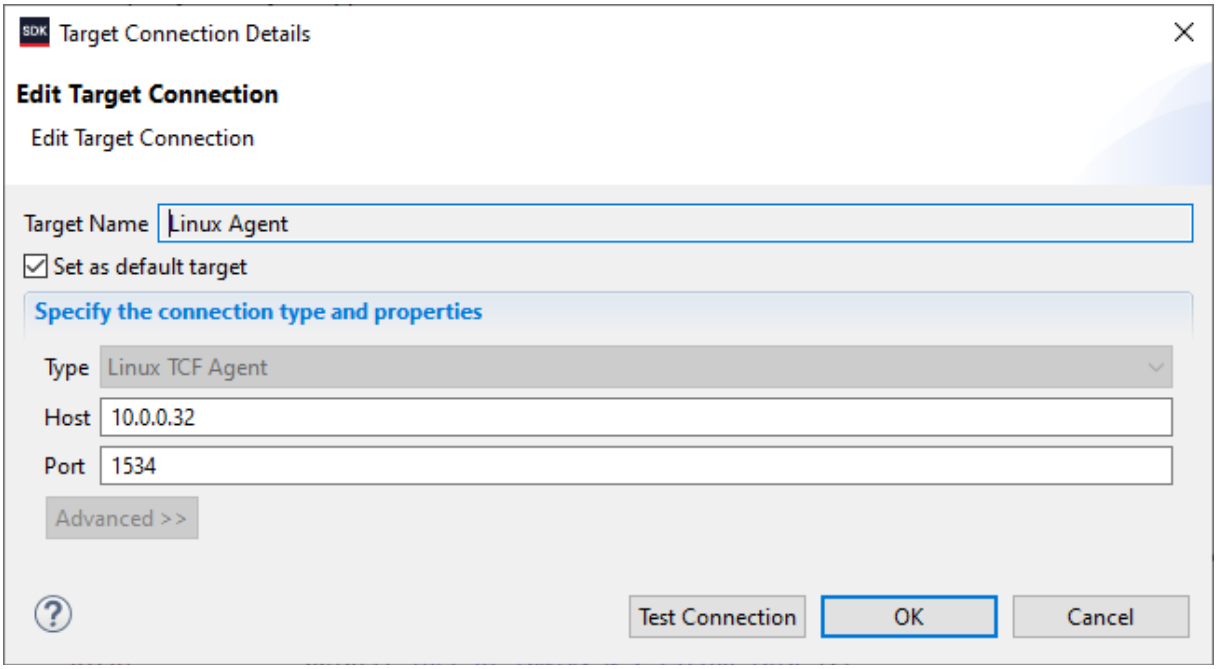


Figure 13: Test connection to Linux TCF Agent.

Recompile the ARM host SW application directly on the TE0726-03M-07S board

Xilinx SDK 2018.2 tool creates files for the **make** utility, which can be used for compilation of SW application directly on the board with use of the **g++** (C++) compiler of the ARM Debian OS.

You can copy complete SDK 2018.2 project to the Debian file system and compile on board by copy complete content of the C++ SDK project directory:

c:\home\work\TS82fp01x8_TE0726_DTRiMC_xc7z07s\xc7z07s_deb_eval_fifo_ila_release\copy_zc_1x1_sw

to the ARM host Debian OS directory:

/home/copy_zc_1x1_sw/

Change the directory in ARM Debian OS to:

```
cd /home/copy_zc_1x1_sw/Debug/
```

Export the relative path to the Debug version of the shared library:

```
export LD_DATA_PATH=../../Debug/sd_card
```

In the Debian OS terminal, clean and then recompile the project by typing:

```
make clean
make
```

Finally, execute the re-compiled C++ Debug version of the SW application compiled by the ARM host Debian OS g++ compiler. Type in the Debian console:

```
./copy_zc_1x1_sw.elf
```

You are done. The compiled application is running on the TE0726 board. See Figure 9.
To close correctly the Debian OS, type in the Debian OS terminal:

```
halt
```

This will close all open files on the SD file system and halt the ARM Debian OS.

Now you can safely remove the SD card. The USB serial terminal can remain connected.
You can modify the SD card in the Win 10 PC.
You can insert modified SD card.
You have to press the reset on the board to initiate a new Debian OS boot process (without the power-off power-on step).

Guide for compilation and use of C MEX functions in `scilab-cli`

The Debian OS image includes **`scilab-cli`** SW interpreter. To use it take these steps.
In Debian OS terminal, change directory to

cd /**`home/xc7z07s_deb_eval_fifo_ila_release_scilab/cc/mmultf`**

In ARM Debian OS terminal, Start **`scilab-cli`** interpret by typing

`scilab-cli`

In **`scilab-cli`**, execute script **`mmultf_cc.sce`** by command

`exec("mmultf_cc.sce")`

This script will compile C MEX function **`mmultf.c`** to shared library **`libmex_mmultf.so`** in the same directory
Quit **`scilab-cli`** by typing

`quit`

Copy created shared library **`libmex_mmultf.so`**

`/home/xc7z07s_deb_eval_fifo_ila_release_scilab/cc/mmultf/libmex_mmultf.so`

to

`/home/xc7z07s_deb_eval_fifo_ila_release_scilab/test/test_mmultf_4xB/libmex_mmultf.so`

In Debian terminal, change directory to

`/home/xc7z07s_deb_eval_fifo_ila_release_scilab /test/test_mmultf_4xB`

Start **`scilab-cli`** by typing

`scilab-cli`

In scilab-cli execute script **`mmultf_4xB_test.sce`** by command

`exec("mmultf_4xB_test.sce")`

scilab-cli will execute **`mmultf()`** C MEX function present in shared library **`libmex_mmultf.so`** and generate reference header files in the current directory . Files contain single precision floating point reference data used for testing of 8xSIMD HW accelerators. Quit **`scilab-cli`** by typing.

`quit`

Use same process to compile and use all other reference MEX C functions.

9 APPENDIX – DTRiMC tool guidelines

Guide for compilation of HW in the DTRiMC tool

1. Unpack the DTRiMC evaluation package to Win 10 directory.
   In this guide, we unpack DTRiMC package to this directory:
   `c:\home\work\TS82fp01x8_TE0726_DTRiMC_xc7z07s\xc7z07s_deb_eval_fifo_ila\`
   Change directory to:
   ```
   c:\home\work\TS82fp01x8_TE0726_DTRiMC_xc7z07s\xc7z07s_deb_eval_fifo_ila\zsys\
   ```
2. On Win 10, open dos terminal window, change directory to the folder
   `c:\home\work\TS82fp01x8_TE0726_DTRiMC_xc7z07s\xc7z07s_deb_eval_fifo_ila\zsys\`
3. To overcome limitations of Win 10 related to the need of short directory paths, use the script *_use_virtual_drive.cmd* to create a virtual short path to your directory drive *X:\zusys* Type command:
   ```
   _use_virtual_drive.cmd
   ```
   Select `X` as name of the virtual drive and select `0` to create the virtual drive.
   Go to the created virtual short-path directory by typing in the win 10 terminal:
   ```
   X:
   cd zsys
   ```
4. Use text editor of your choice and open and modify script `design_basic_settings.sh`
   Select correct path to SDSoC 2018.2 tool installed on your Win7 or Win 10. Line 38:
   ```
   @set XILDIR=C:/Xilinx
   ```
   Select proper Xilinx device:
   ```
   @set PARTNUMBER=4
   ```
   The selected number corresponds to the number defined in file
   `X:\zusys\board_files/TE0808_board_files.csv`
   Verify, if line 78 of script *design_basic_settings.sh* sets the SDSoC flow support by: *ENABLE_SDSOC=1*
   ```
   @set ENABLE_SDSOC=1
   ```
5. Start the Xilinx Vivado 2018.2 and create the design by executing of script:
   ```
   X:\zsys\vivado_create_project_guimode.cmd
   ```
6. Optional:
   You can use *Vivado* automation and to the created HW design Xilinx *In Circuit Logic Analyzer* (ILA) monitor to enable capturing of selected accelerator outputs of your choice.
7. In *Vivado* console, execute command:
   ```
   TE::hw_build_design -export_prebuilt
   ```
   After the Vivado compilation, new hardware description file `zsys.hdf` is generated in folder:
   ```
   X:\zsys\prebuilt\hardware\m\zsys.hdf
   ```

Guide for configuration and compilation of PetaLinux in the DTRiMC tool

The configuration and compilation of the *Petalinux 2018.2* kernel and *Debian 9.8 Stretch* image as the FitOptiVis run time resource for the Zynq TE0726-03M-07S board is described now. The configuration has to be performed in the Ubuntu 16.04 LTS OS.

The DTRiMC tool is configured for use of Ubuntu 16.04 LTS in the *VMware Workstation Player* in Win 10. The Petalinux 2018.2 distribution can be downloaded to the Ubuntu 16.04 LTS from

https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/embedded-design-tools/2018-2.html

and installed to the default Ubuntu directory:

```
/opt/petalinux/petalinux-v2018.2-final
```

The standard PetaLinux 2018.2 distribution requires few modifications.

1. Copy content of these Win 10 directories:

   ```
   X:\zsys\prebuilt
   X:\zsys\os
   ```

   to Ubuntu directories:

   ```
   /home/devel/work/TS82fp01x8_TE0726_DTRiMC_xc7z07s/xc7z07s_deb_eval_fifo_ila/zsys/prebuilt/
   /home/devel/work/TS82fp01x8_TE0726_DTRiMC_xc7z07s/xc7z07s_deb_eval_fifo_ila/zsys/os/
   ```

2. In Ubuntu, open terminal window and set path to the PetaLinux 2018.2:

   ```
   source /opt/petalinux/petalinux-v2018.2-final/settings.sh
   ```

3. Change directory to the directory copied from the evaluation package with pre-defined configuration:

   ```
   cd /home/devel/work/TS82fp01x8_TE0726_DTRiMC_xc7z07s/xc7z07s_deb_eval_fifo_ila/zsys/os/petalinux/
   ```

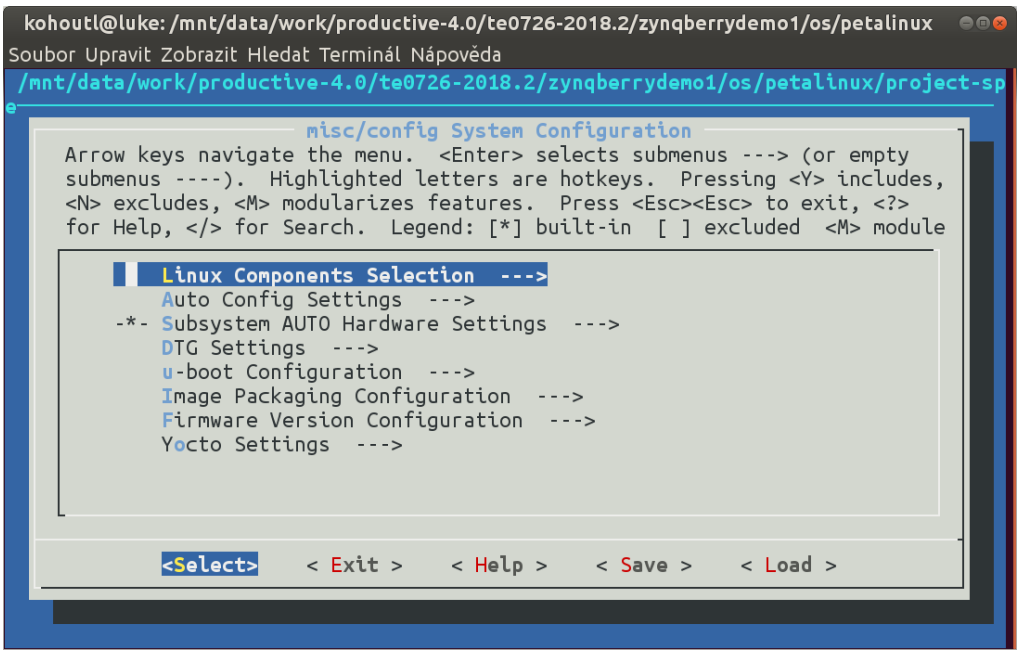
   It contains a predefined configuration according to Zynq TE0726-03M-07S board requirements.

4. The zsys.hdf file created in Win 10 in Vivado 2018.2 tool is present in the Ubuntu folder:

   ```
   /home/devel/work/TS82fp01x8_TE0726_DTRiMC_xc7z07s/xc7z07s_deb_eval_fifo_ila/zsys/prebuilt/prebuilt/hardware/m/
   ```

5. Use the `zusys.hdf` file as input for the PetaLinux configuration by (on single line)

   ```
   petalinux-config --get-hw-description=/home/devel/work/TS82fp01x8_TE0726_DTRiMC_xc7z07s/xc7z07s_deb_eval_fifo_ila/zsys/prebuilt/prebuilt/hardware/m/
   ```



6. Verify if the PetaLinux filesystem location is changed from the ramdisk to the extra partition on the SD card, select:

```
Image Packaging Configuration  --->
      Root filesystem type (SD card)  --->
```

7. Verify if option to generate boot args. automatically is disabled and if user defined arguments are set to:

```
earlycon clk_ignore_unused root=/dev/mmcblk0p2 rootfstype=ext4 rw
rootwait quiet
```

Leave the configuration, 3x *Exit* and *Yes*.

8. Build PetaLinux, from the bash terminal execute

```
petalinux-build
```

9. Files *image.ub*, *u-boot.elf* and *bl31.elf* are created in:

```
/home/devel/work/TS82fp01x8_TE0726_DTRiMC_xc7z07s/xc7z07s_deb_eval_fifo_i
la/zsys/os/petalinux/images/linux/image.ub
/home/devel/work/TS82fp01x8_TE0726_DTRiMC_xc7z07s/xc7z07s_deb_eval_fifo_ila/z
sys/os/petalinux/images/linux/u-boot.elf
/home/devel/work/TS82fp01x8_TE0726_DTRiMC_xc7z07s/xc7z07s_deb_eval_fifo_ila/z
sys/os/petalinux/images/linux/bl31.elf
```

Guide for configuration and compilation of Debian OS in the DTRiMC tool

The file system is based on the latest stable version of Debian 9.8 Stretch distribution (03. 25. 2019). Follow the steps below.

10. In Debian, cd to the folder with PetaLinux:

```
cd
/home/devel/work/TS82fp01x8_TE0726_DTRiMC_xc7z07s/xc7z07s_deb_eval_fifo_ila/zsys/os/petalinux/
```

11. The 32bit Debian image will be created by execution of the *mkdebian.sh* script. The script checks all the tools that are needed to create the image, most of them are a standard part of the Ubuntu 16.04 LTS distribution.

    When some of them are missing, install them by:

```
sudo apt install Package
```

    Table 1: tools with a corresponding package name.

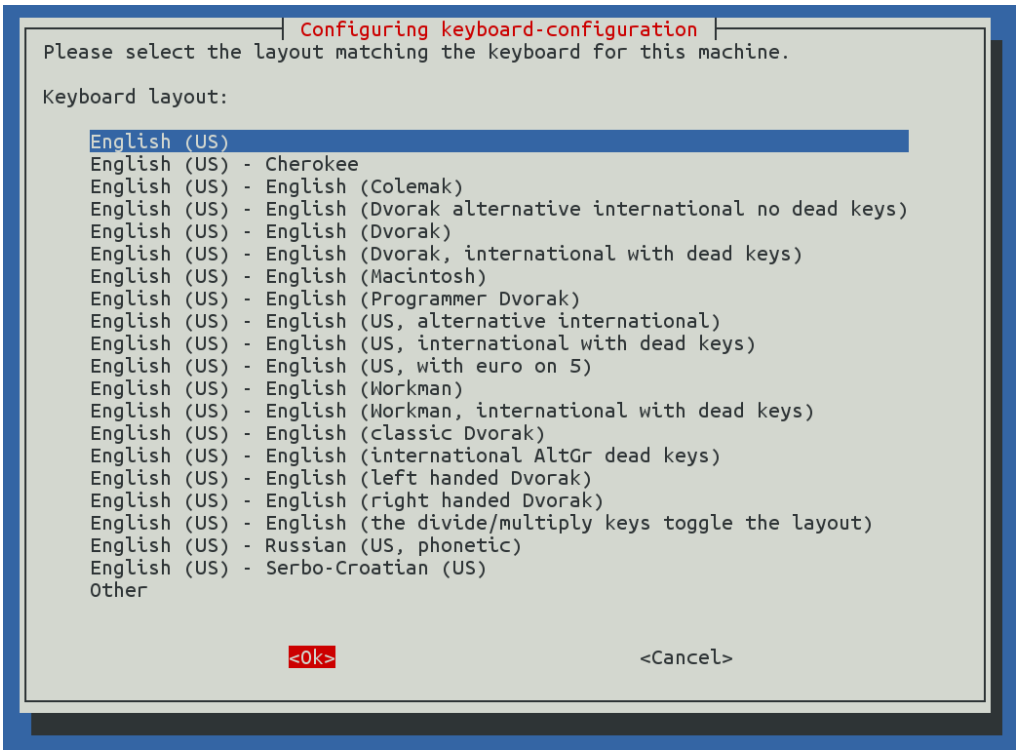
Tool	*Package*
dd	coreutils
losetup	mount
parted	parted
lsblk	util-linux
mkfs.vfat	dosfstools
mkfs.ext4	e2fsprogs
debootstrap	debootstrap
gzip	gzip
cpio	cpio
chroot	coreutils
apt-get	apt
dpkg-reconfigure	debconf
sed	sed
locale-gen	locales
update-locale	locales
qemu-ARM-static	qemu-user-static

12. Create the Debian image. It will consist of two partitions.

    The file system of the first one will be FAT32. This partition is dedicated for image of the PetaLinux kernel. The second partition will contain the Debian using EXT4 file system. Create the Debian image from the external Ethernet repositories by this command:

```
chmod ugo+x mkdebian.sh
sudo ./mkdebian.sh
```

    During the creation procedure, you will be asked to set language. Choose *English (US)*. The resultant image file will be called te0726-debian.img its size will be 7 GB.



13. Compress the created image to file te0726-debian.zip:

```
zip te0726-debian te0726-debian.img
```

14. Copy compressed image file from Ubuntu

```
/home/devel/work/TS82fp01x8_TE0726_DTRiMC_xc7z07s/xc7z07s_deb_eval_fifo_ila/zsys/os/petalinux/te0726-debian.zip
```

to Win 10 file:

```
X:\zsys\prebuilt\os\petalinux\default\te0726-debian.zip
```

15. Copy these files from Ubuntu

```
/home/devel/work/TS82fp01x8_TE0726_DTRiMC_xc7z07s/xc7z07s_deb_eval_fifo_ila/zsys/os/petalinux/images/linux/image.ub
/home/devel/work/TS82fp01x8_TE0726_DTRiMC_xc7z07s/xc7z07s_deb_eval_fifo_ila/zsys/os/petalinux/images/linux/u-boot.elf
/home/devel/work/TS82fp01x8_TE0726_DTRiMC_xc7z07s/xc7z07s_deb_eval_fifo_ila/zsys/os/petalinux/images/linux/bl31.elf
```

to Win 10 files:

```
X:\zsys\prebuilt\os\petalinux\default\image.ub
X:\zsys\prebuilt\os\petalinux\default\u-boot.elf
X:\zsys\prebuilt\os\petalinux\default\bl31.elf
```

16. In Ubuntu, clean Petalinux project files

```
petalinux-build -x mrproper
```

17. In Ubuntu, delete files

```
/home/devel/work/TS82fp01x8_TE0726_DTRiMC_xc7z07s/xc7z07s_deb_eval_fifo_ila/zsys/os/petalinux/TE0726-debian.zip
/home/devel/work/TS82fp01x8_TE0726_DTRiMC_xc7z07s/xc7z07s_deb_eval_fifo_ila/zsys/os/petalinux/TE0726-debian.img
```

18. In Ubuntu, close all applications and shut down Linux.

19. In Win 10, close the VMware Workstation Player.

You can continue with preparation of the Zynq board with created files:

- Petalinux kernel image `image.ub`
- Compressed Debian image `TE0726-debian.zip`
- U-boot program `u-boot.elf`

This ends the DTRiMC tool configuration and compilation steps for the Petalinux and Debian.

Guide for creation of SDSoC platform for TE0726-03M-07S in the DTRiMC tool

20. In the open Vivado 2018.2 console, create and compile the initial *BOOT.bin* file and the initial SW modules by execution of the command:

    ```
    TE::sw_run_hsi
    ```

    The resulting *BOOT.bin* file will be located in the folder

    ```
    X:\zsys\prebuilt\boot_images\m\u-boot\BOOT.bin
    ```

21. These files are created:

    ```
    X:\zsys\prebuilt\software\m\hello_te0726.elf
    X:\zsys\prebuilt\software\m\zynq_fsbl.elf
    X:\zsys\prebuilt\software\m\zynq_fsbl_flash.elf
    ```

    File `zynq_fsbl.elf` is correct first stage board loader (FSBL) file, while the `zynq_fsbl_flash.elf` is special FSBL file used only for programming of the on board flash.

22. Move `zynq_fsbl_flash.elf` file to some different temporary location before next step.
23. In Vivado 2018.2 console, create the SDSoC platform by execution of the command:

    ```
    TE::ADV::beta_util_sdsoc_project
    ```

    The SDSoC 2018.2 platform is generated in the directory

    ```
    X:\SDSoC_PFM\te0726\03m\zsys
    ```

    and it is also packed into the ZIP file in directory

    X:\SDSoC_PFM\te0726\

24. Return `zynq_fsbl_flash.elf` file back from the temporary location to

    X:\zsys\prebuilt\software\m\zynq_fsbl_flash.elf

    It will be used later on by the TE0726-03M-07S board flash programming script

    ```
    program_flash_binfile.cmd
    ```

Guide for creation of shared library and HW kernel in the DTRiMC tool

25. On Win 10, i*n the open dos terminal window, cancel the current virtual drive X: by executing from* the command line

    ```
    _use_virtual_drive.cmd
    ```

    and response (1)

26. Change directory to

    ```
    c:\home\work\TS82fp01x8_TE0726_DTRiMC_xc7z07s\xc7z07s_deb_eval_fifo_ila\SDSoC_PFM_src\te0726\03m\
    ```

27. In Win 10, open dos terminal window and use the copy of the script *_use_virtual_drive.cmd* to create a new virtual short path to get short SDSoC directory `X:\03m`

```
_use_virtual_drive.cmd
```
Select X as name of the virtual drive and select (0) to create the virtual drive.
Go to the created virtual short-path directory by:
```
X:
cd 03m
```

28. Open SDSoC project in directory
```
X:\03m
```
29. In SDSoC import HW kernel design project
`fp01x8_v26x1_hw`
from the directory
`c:\home\work\TS82fp01x8_TE0726_DTRiMC_xc7z07s\xc7z07s_deb_eval_fifo_ila\SDSoC_PFM_src\te0726\03m\fp01x8_v26x1_hw\`
Define the custom SDSoC platform
`X:\03m\zsys`
30. Change imported project from Debug to the Release compilation target
31. Compile project by the SDSoC 2018.2 compiler
32. Result of compilation are the SD cards with the `BOOT.BIN` file and the shared object library file `libcopy_zc_1x1_hw.so` in the directory:
`X:\03m\fp01x8_v26x1_hw\Release\sd_card\`
33. Copy content of the directory
`X:\03m\fp01x8_v26x1_hw\Release\sd_card\`
to
`c:\home\work\TS82fp01x8_TE0726_DTRiMC_xc7z07s\xc7z07s_deb_eval_fifo_ila_release\copy_zc_1x1_sw\Release\sd_card\`
and also to
`c:\home\work\TS82fp01x8_TE0726_DTRiMC_xc7z07s\xc7z07s_deb_eval_fifo_ila_release\copy_zc_1x1_sw\Debug\sd_card\`
34. Optional:
Copy ILA nets definition files `debug_nets.ltx` and `zsys_wrapper.ltx` from the directory
`X:\03m\fp01x8_v26x1_hw\Release\_sds\p0\vivado\prj\prj.runs\impl_1\`
to
`c:\home\work\TS82fp01x8_TE0726_DTRiMC_xc7z07s\xc7z07s_deb_eval_fifo_ila_release\copy_zc_1x1_sw\Release\sd_card\`
and also to
`c:\home\work\TS82fp01x8_TE0726_DTRiMC_xc7z07s\xc7z07s_deb_eval_fifo_ila_release\copy_zc_1x1_sw\Debug\sd_card\`
35. Clean SDSoC project to save disk space.
36. Close SDSoC 2018.2 tool.
37. The created `BOOT.BIN` file will be used for programming of TE0726-03M-07S board flash. The shared object library file `libcopy_zc_1x1_hw.so` is to be linked to applications compiled for ARM in SDK and also used in the runtime on ARM.

This is described in the first section of the chapter 10 APPENDIX - Confidence test.

Guide for retargeting to another board

The DTRiMC tool is configured for Trenz Electronic module with ID=4, board TE0726-03M-07S part xc7z007sclg225-1, memory 0.5GB with short module name 7s [1].

However, the DTRiMC tool scripts can be modified to target different Trenz Electronic board [6].

Use text editor of your choice and open and modify script *design_basic_settings.sh* Modify ID of the device from @set PARTNUMBER=4 to @set PARTNUMBER=3.

Trenz Electronic modules supported by this release of the DTRiMC tool.

Trenz Electronic modules supported by this release of the DTRiMC tool.

ID	Board	Partname	Memory	ShortName
3	te0726-03m	xc7z010clg225-1	0.5GB	m
4	te0726-03-07s-1c	xc7z007sclg225-1	0.5GB	7s

After the change of the target to ID=3, implement all design steps with the DTRiMC tool for the retargeted board [6]. It has dual core ARM A9 processor and larger PL resources.

You also download and use the related application notes and evaluation packages [7] and [8] for the te0726-03m board [6].

Disclaimer

This disclaimer is not a license and does not grant any rights to the materials distributed herewith. Except as otherwise provided in a valid license issued to you by UTIA AV CR, v.v.i., and to the maximum extent permitted by applicable law:

(1) THIS APPLICATION NOTE AND RELATED MATERIALS LISTED IN THIS PACKAGE CONTENT ARE MADE AVAILABLE "AS IS" AND WITH ALL FAULTS, AND UTIA AV CR V.V.I. HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and

(2) UTIA AV CR, v.v.i. shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under or in connection with these materials, including for any direct, or any indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or UTIA AV CR, v.v.i. had been advised of the possibility of the same.

Critical Applications:
UTIA AV CR, v.v.i. products are not designed or intended to be fail-safe, or for use in any application requiring fail-safe performance, such as life-support or safety devices or systems, Class III medical devices, nuclear facilities, applications related to the deployment of airbags, or any other applications that could lead to death, personal injury, or severe property or environmental damage (individually and collectively, "Critical Applications"). Customer assumes the sole risk and liability of any use of UTIA AV CR, v.v.i. products in Critical Applications, subject only to applicable laws and regulations governing limitations on product liability.