TI Designs

Reference Design Using TMS320C6678 to Implement a Real-Time Synthetic Aperture Radar (SAR) Algorithm

TI Designs

TI Designs provide the foundation that you need including methodology, testing and design files to quickly evaluate and customize the system. TI Designs help *you* accelerate your time to market.

Design Resources

TIDEP0045	Tool Folder Containing Design Files
TMDSEVM6678	Tool Folder
TMDXEVMPCI	Tool Folder
Desktop Linux SDK	Product Folder
Demo Guide	Desktop Linux SDK Getting Started
Development Guide	Desktop Linux SDK Development Guide

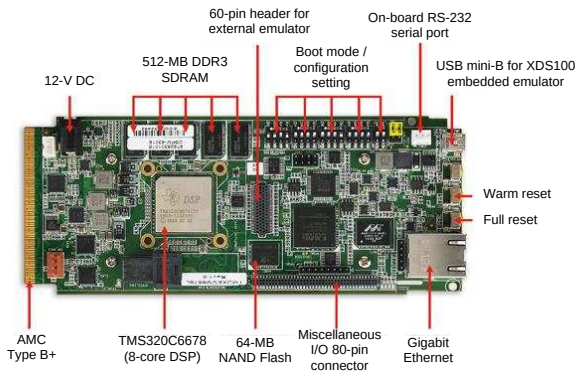
TI E2E™ Community — ASK Our E2E Experts

Design Features

- Tested and Includes a Hardware Reference (EVM), Software, and User's Guide.
- Hardware Platform Includes TMDSEVM6678 EVM for a High-Performance, Cost-Efficient, Standalone Development Platform.
- Includes the TMS320C6678 High-Performance DSP Based on TI's C66x Keystone Multicore Architecture.
- Includes Schematics, Design Files, and a Bill of Materials.
- Includes an SAR Algorithm, Input Binary Files, Display Scripts, and Links to Download the BIOS-MCSDK and SDK Software Framework.
- Features Algorithm Implementation, Required Hardware and Software Step by Step Guide to Building and Running the SAR Application.

Featured Applications

- Real-Time Airborne Radar
- High Resolution Imaging in Automotive



An IMPORTANT NOTICE at the end of this TI reference design addresses authorized use, intellectual property matters and other important disclaimers and information.

1 Design Overview

This TI design shows a real time Synthetic Aperture Radar (SAR) implementation running on a TI multicore DSP C6678 platform. One of the main challenges of SAR is to generate high resolution images in real-time, because forming the image involves computationally demanding signal processing procedures.

TI has implemented a SAR algorithm on the TMS320C6678 eight core fixed- and floating-point DSP to show the full application performance and how it scales across one, two, four, and eight DSP cores. Here, the range-Doppler SAR processing algorithm is modularized from a functional perspective and the computational task is mapped to multiple cores running in parallel. The task mapping procedure is accomplished using OpenMP.

2 Design Motivation

Commercial off-the-shelf (COTS) components have been gaining popularity in Synthetic Aperture Radar (SAR) and other defense and airborne applications. Their popularity is due to their widespread availability and low cost compared to custom-built solutions. For the user, the question of performance is vital, and the compute capabilities of these COTS devices has advanced to a level where real-time processing of complex algorithms, for SAR and other applications, has become feasible. In this design, the capability of the TMS320C6678 processor for SAR signal processing is shown in real time. This design also shows how the SAR algorithm may be modularized and mapped to C6678 architecture in order to achieve parallel computation. The design also demonstrates how the TMS320C6678 processor scales across one, two, four, and eight DSP cores.

3 Introduction to SAR

Synthetic Aperture Radar (SAR) achieves high resolution remote sensing imagery by moving the radar platform to create the effect of a large antenna. Typically, the radar is carried by a spaceborne or an airborne platform moving at known speed. A single physical antenna is used to gather signals reflected from the targets at different positions, at different times. The relative motion between the radar and the targets encodes the targets' information, which is processed to form a focused image of the surface area.

At each radar position, the antenna system transmits a short chirped waveform, and then the reflected echoes from the targets are collected, digitized, and typically stored for later processing. Advanced signal processing techniques are used to analyze the phase shift information and obtain fine resolution in the direction perpendicular to the beam direction. The acquisition geometry makes the SAR image processing a two-dimension operation. The original data collected from the radar is unfocused, and the raw data must be processed to achieve digital focusing and generate the final SAR image. The general processing method involves weighting, phase shifting and summing the phase histories of the responses. There are multiple algorithms for achieving this; the most widely used is the Range-Doppler (RD) algorithm, with the w–k 2D algorithm a close second. The RD algorithm features block processing efficiency and separability of processing in the two dimensions, range and azimuth, making it well-suited for parallel computing platforms. The RD algorithm has been implemented in this design for both its efficiency and its parallel nature which can be exploited by multicore architectures, such as the TMS320C6678 eight-core DSP, to produce large improvements in processing time.

4 Range-Doppler Algorithm Implementation

To efficiently implement the RD algorithm on the TMS320C6678 DSP, it is divided into five sequential tasks, as shown in Figure 1.

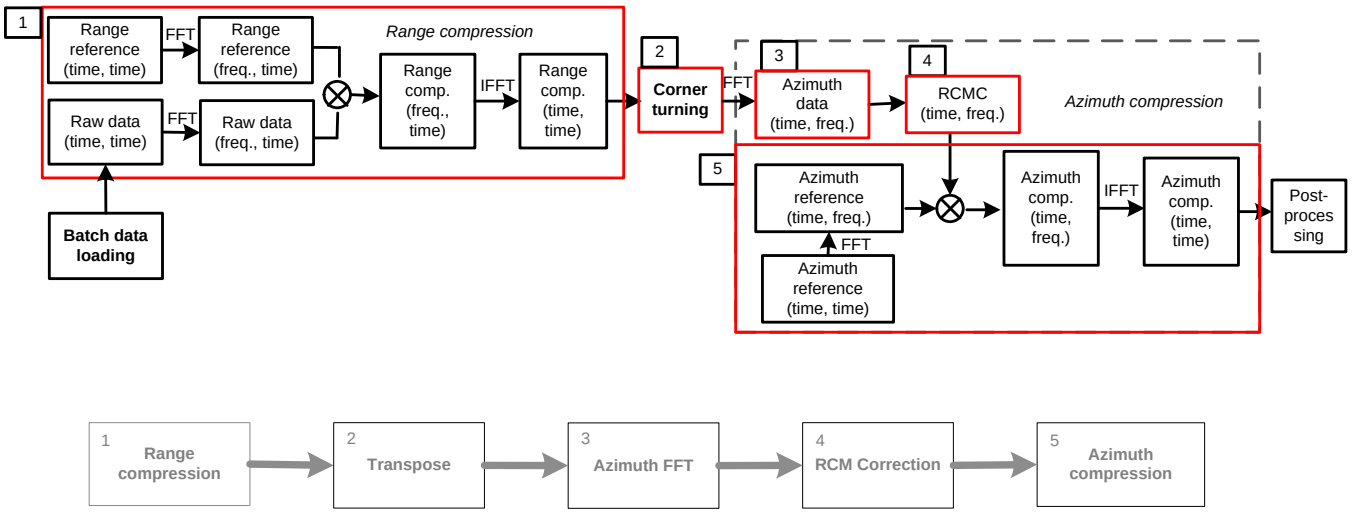


Figure 1. Range-Doppler Algorithm and Modulation

Specifically, these tasks are:

- Range compression
- Matrix transpose
- Azimuth FFT
- Range Cell Migration Correction (RCMC)
- Azimuth Compression

Each sequential task is considered as a module for implementation on the TMS320C6678 DSP. The whole program is written in standard *C* language with TI supplied *C* intrinsics, which allows efficient use of the DSP core instructions. TI supplied standard math and DSP functional libraries are also used in this design. In order to achieve the most efficient implementation across the multi-core architecture of this device, the computations inside each task is parallelized symmetrically across all the DSP cores. OpenMP software has been used to construct and perform this parallelization. Another feature of implementation is the use of the Enhanced Direct Memory Access (EDMA) engine available in the processor. This EDMA allows computation in parallel to data movement between memory hierarchies. The EDMA also enables 2-D scattering and gathering operation which is used to implement corner turning or transpose operations efficiently.

Although this design is specific to the RD algorithm of SAR image formation, the following two aspects of this design are common to other algorithms in SAR as well as many defense and airborne applications:

- Batch FFT and IFFT operations
- Transpose of corner turning

The optimization techniques used in this design are useful in other applications as well.

5 System Overview

In this design, the TMDSEVM6678 is connected to a host PC using a PCI adapter card. The host PC provides an 4096×4096 raw image in a Global Shared Memory (GSM), and signals DSP using mailbox. The DSP retrieves the input data and calls the SAR algorithm, which using OpenMP, distributes tasks between a pre-defined number of cores. After the DSP finishes processing an image, the DSP signals the host mailbox and writes back output to a GSM as a 3424×3072 pixel image. Finally, the demo front-end (python script on the host) will display, in grey scale, and inputs and respective outputs as outputs become available. Figure 2 shows the SAR demo setup.

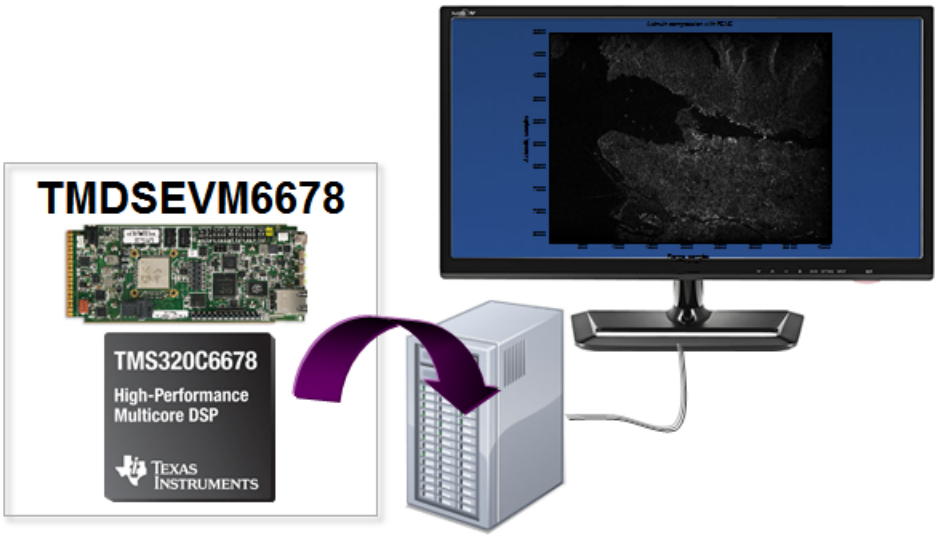


Figure 2. SAR Demo Setup

NOTE: Figure 2 shows an SAR image courtesy of MacDonald, Dettwiler and Associates Ltd. (MDA). Copyright: RADARSAT Data © Canadian Space Agency/Agence Spatiale Canadienne (2002). All Rights Reserved.

6 Linux Desktop SDK Framework Overview

This TI design uses a Linux Desktop SDK software framework as a software package. This package provides communication modules between a host PC and a DSP board, for demonstrating offload SAR processing in a multicore DSP. As shown in Figure 3, the desktop Linux SDK software has two components. One component runs on the Linux Host, and the other component runs on a DSP. The DSP and host processor share common header files for message interpretation, and the DSP reserved addresses for host memory management.

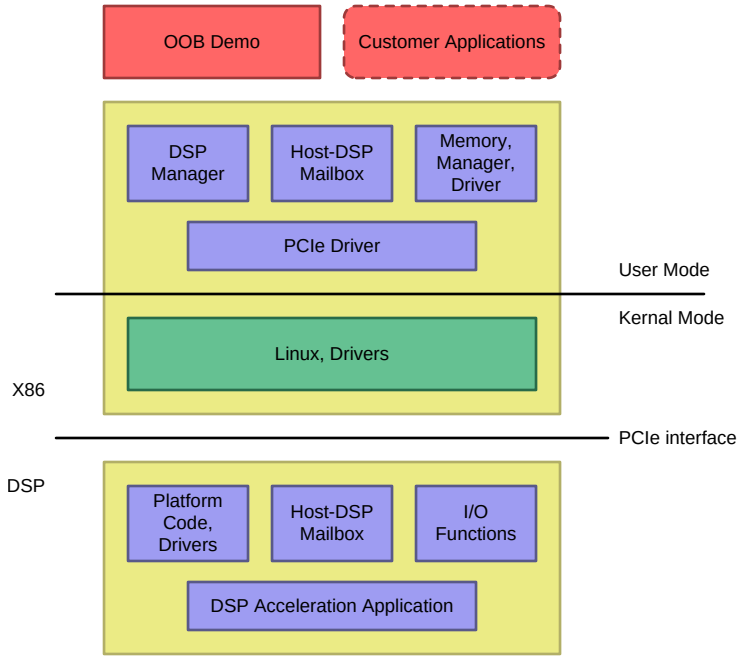


Figure 3. Desktop Linux SDK Software Framework Architecture

For more details on Linux Desktop SDK, refer to: Desktop-linux-sdk 01.00.00 Development Guide.

7 Required Hardware

- Evaluation board: TMDSEVM6678L / TMDSEVM6678LE
- PCIe adapter: TMDXEVMPCI
- Linux Desktop PC with Ubuntu 12.04 LTS (32-bit and 64-bit OS are supported)
- Motherboard with a PCIe slot for plugging TI's DSP card
- Memory RAM size bigger than 4 GB

8 Required Software

The software packages required and overall steps are:

- Install the Desktop Linux SDK.
- Install the required Linux packages.
- Install the BIOS MCSDK.
- Install the SAR package.
- Install Python.
- Install CCS (only required if the user wants to rebuild a DSP application).
- Install the CGT compiler (if not installed already).

Install the Desktop Linux SDK:

1. Download DESKTOP-LINUX-SDK 01_00_03_00
2. Change the attribute of the installer to executable
3. Run the installer as:

```
chmod +x desktop-linux-sdk_01_00_03_00_<OSversion>_setuplinux.bin
./desktop-linux-sdk_01_00_03_00_<OSversion>_setuplinux.bin
```

For additional installation details, refer to: http://processors.wiki.ti.com/index.php/Desktop-linux-sdk_01.00.00_Getting_Started_Guide.

Install the Required Linux Packages:

```
sudo apt-get install libpciaccess-dev
sudo apt-get install binutils-dev
```

Install BIOS MCSDK:

```
chmod + x bios_mcsdk_02_01_02_06_setuplinux.bin
./bios_mcsdk_02_01_02_06_setuplinux.bin
```

NOTE: Users using a 64-bit operating system find that they are not able to install BIOS MCSKD. When trying to run the installing, nothing happens and no error message appears. Because the MCSDK installer and integrated tools are 32-bit binaries, 32-bit versions of various packages are required to successfully install BIOS MCSDK. Install *sudo apt-get install ia 32-libslibgnomevfs2-0:i386 liborbit2:i386libjpeg62:i368* to fix the issue.

From the BIOS-MCSDK 2.1.2.6 Installation Package:

Users may install all of the components bundle on the BIOS MCSDK package. Listed below are the minimum tools required that must be installed for running and building the SAR demo.

1. Select PDK C6678 1.1.2.6.
2. Select XDC Tools 3.23.4.60.
3. Select SYSBIOS 6.33.6.50.
4. Select OMP 1.1.3.2.
5. Selet EDMA-LLD 2.11.5.2.
6. Select NDK 2.21.1.38.
7. Select IPC 1.24.3.32.
8. Select DSPLIB 3.1.0.0.

For additional details, refer to http://processors.wiki.ti.com/index.php/BIOS_MCSDK_2.0_Getting_Started_Guide.

Install the SAR Package:

1. Download SAR_python.tar.gz and SARdemo.tar.gz from http://www.ti.com/tool/TIDEP0045.
2. Untar and copy the SARdemo folder inside <DesktopLinuxSDK Install Dir>/demos/.

   The SAR Package contains the following folders:

C66x: SAR algorithm CCS project files.

Host: Host application files.

PlatformRepo: evmc6678 platform

Scripts: Input files and scripts for running the demo.

3. Untar and copy SAR_python folder to the home directory.

Install Python:

1. Install Python 2.7.3.
2. Install the *wxPython* component.
3. Install the *NumPy* component.
4. Install the *SciPy* component.
5. Install the *MatPlotLib* component.
6. You may use apt-get for installing these modules:

```
sudo apt-get install python-wxgtk2.8
sudo apt-get install python-numpy python-scipy python-matplotlib
```

Install CCSv5:

Install the Code Composer Studio for Linux. Download links and more at http://processors.wiki.ti.com/index.php/Download_CCS#Code_Composer_Studio_Version_5_Downloads and http://processors.wiki.ti.com/index.php/Linux_Host_Support.. The last link gives important tips for Linux users.

Install CGT Compiler:

Check to make sure that CGT 7.4.0 was installed inside *<CCs_install_path>/tools/compiler*. If the CGT 7.4.0 was not installed, download it from http://software-dl.ti.com/codegen/non-esd/downloads/download.htm.

9 EVM Preparation

IBL Flash:

1. Copy i2crom_0x51_c6678_le.bin from *mcsdk_2_01_02_06\tools\boot_loader\ibl\src\make\bin to mcsdk_2_01_02_06\tools\writer\eeprom\evmc6678l\bin*.
2. Rename this copied file to *app.bin*.
3. Open *eepromwriter_input.txt* in *mcsdk_2_01_02_06\tools\writer\eeprom\evmc6678l\bin*.
4. Set *file_name* equal to *app.bin* and *bus_addr* equal to 0×51.
5. Ensure *start_addr* and *swap_data* are set to 0.
6. Save and close *eepromwriter_input.txt*.
7. Turn on and connect the EVM.
8. Open CCSv5.
9. Load the appropriate target configuration.
10. Connect to Core 0.
11. Load the corresponding GEL file for the evmc6678l.
12. Load the EEPROM writer program by going to *Run→Load Program*.
13. Browse for the eeprom writer DSP executable. For example, *eepromwriter_evm6678l.out* in the same folder as *app.bin* for the C6678 EVM.
14. View the memory browser (go to *View→Memory Browser*).
15. Browse to address 0×0C000000.
16. Right-click on the memory window.
17. Select *Load Memory*.
18. Select *app.bin* (by default, the browse menu only displays .dat files).
19. Change the option TI Data Format (*.dat) to Raw Data Format (*.bin) to find your binary file.
20. Change the Start Address to 0x0C000000 if it has not been changed.
21. Leave the swap checkbox unchecked.
22. Click Finish.
23. Select 32-bits for Type-Size option in CCS.
24. A sample successful eeprom writer output appears like this:
    - [C66xx_0] EEPROM Writer Utility Version 01.00.00.05 Writing 52264 bytes from DSP memory address 0x0c000000 to EEPROM bus address 0x0051 starting from device address 0x0000 ... Reading 52264 bytes from EEPROM bus address 0x0051 to DSP memory address 0x0c010000 starting from device address 0x0000 ... Verifying data read ...
25. EEPROM programming successfully completes.

Hardware Setup:

1. Set the EVM card to PCIE boot via following the switch setting (For SW3, pin 1: OFF: Little Endian;) Table 1 shows the hardware setup.

Table 1. Hardware Setup

SW3	SW4	SW5	SW6	SW9
(pin 1,2,3,4)	(pin 1,2,3,4)	(pin 1,2,3,4)	(pin 1, 2, 3, 4)	(pin 1)
(off, on, on, off)	(on, on, on, on)	(on, on, on, off)	(off, on, on, on)	(off)

2. Assemble the EVM card into the adaptor card.
3. Shut off the PC power supply completely (by disconnecting the power cord).
4. Insert the AMC adaptor card (with the TMS320C6678L card mounted) into an open PCIE slot in the PC's motherboard.
5. Supply the power to the PC.
6. Wait for a few seconds.
7. Power on the PC.
8. Make sure the PCIE device is correctly enumerated by the PC by typing *lspci–n* under the Linux command shell after the Linux OS is loaded. A TI device (VENDOR_ID: 0×104c) should be in the list:

```
local-ubuntu:$ lspci -n | grep "104c"
01:00.0 0480: 104c:b005  (rev 01)
```

10 Building the Linux Desktop SDK–(Optional: Build/Run Loopback Demo)

Before continuing, user needs to rebuild SDK components using PDK 1.1.2.6 and CGT 7.4.0 tools. Optionally, we suggest to build and run loopback file test demo included in Linux Desktop SDK package. This file test demo, reads contents of a input binary test file, sends the data to DSP memory and send a message to DSP through mailbox (Mailbox message includes pointer to input and output data location). For Loopback demo, the DSP build simply send back buffer to host through mailbox. The host then receives message through mailbox, reads data from output buffer and writes to the output file.

Mandatory Steps:

1. Update <*DesktopLinuxSDK Install Dir*>*/setup_dsp_build_env.sh* script to use CGT 7.4.0 and the PDK 1.1.2.6.

   Example:

   export C6X_GEN_INSTALL_PATH=/home/alice/ti/TI_CGT_C6000_7.4.0/

   export PDK_INSTALL_PATH=/home/alice/ti/pdk_C6678_1_1_2_6

   export DESKTOP_LINUX_SDK_DIR=$(pwd)

2. Build the DSP test application and the SDK DSP images.

   (a) Go to <*DesktopLinuxSDK Install Dir*> in a terminal window.

   (b) Run the following commands:

   ```
   source setup_dsp_build_env.sh
   make clean
   make all
   ```

The SAR demo requires using larger and pre-reserved contiguous memory allocation. An example of reserving 520 MB:

```
cd demos/scripts
./install_grub.sh 520 8
Please confirm to replace line (Y/n)
./install_cmem_autoload.sh
./install_udev.sh
```

The system must be powered down and restarted for the reserve memory driver to work.

Optional Steps:

1. For running the file test demo, a binary file is required which is used as the input file. Currently, the scripts assume the file name is test.file at the directory location testfiles on the same base directory the Desktop Linux is installed: <DesktopLinuxSDK Install Dir>/testfiles/test.file.
2. The user may copy any file and rename it *test.file*.
3. TI recommends the minium file size is >64 MB (2×8 cores×4 MB), so that all cores in the card are used in the test at least twice.

   Example:

   ```
   mkdir ../testfiles
   cp <anyfile.file> ../testfiles/test.file
   ```

4. Download the initial DSP image to initialize the DSPs DDR.

   ```
   cd demos/scripts
   ./init_evm6678_1250.sh
   ```

5. Download the initial DSP image to all of the cores.

   ```
   cd ../filetestdemo/scripts
   ./dnld_demo_loopback.sh 1
   ```

6. Run the Loopback demo desktop application.

   ```
   ./run_memcpytest.sh 1  0×400000
   ```

7. DSP reset. To re-download the DSP, build or download to a different DSP build, first, the DSP build must be reset.

```
cd demos/scripts
./dspreset.sh <num_of_dsps>
```

For more details, refer to http://processors.wiki.ti.com/index.php/Desktop-linux-sdk_01.00.00_Getting_Started_Guide .

11 Building SAR Host Application

1. Go to *<DesktopLinuxSDK Install Dir>/demos/SARdemo/host/* in a terminal window.
2. Run the following commands to build the ***demo_filetest*** host application:

```
make clean
make
```

Input and Output file names and number of images to process in loop are hard-coded inside *<DesktopLinuxSDK Install Dir>/demos/SARdemo/host/demo_filetest.c*.

If the user wants to change the default names, search for the keywords **raw4096** and **Out4096**.

Currently, the demo has a hard code loop of 15 images.

After changes in the host files, if any, keep in mind to rebuild the application using *make* before re-running the demo.

NOTE: Input image was provided courtesy of MacDonald, Dettwiler and Associates Ltd. (MDA). Copyright: RADARSAT Data © Canadian Space Agency/Agence Spatiale Canadienne (2002). All Rights Reserved.

12 Running the Pre-Build DSP Application and SAR Demo Using Python Scripts

The files that must be edited are:

1. **python/ti_sar_demo_runner.py:** inside of the following script, three file paths that must be updated and verified.
   - BIN_PATH.
   - RUN_EVM_SCRIPT.
   - DEFAULT_SETTINGS_FILE .
2. **python/runEVM.sh:** Edit *p* with the correct path to the desktop SDK demo scripts folder, *<DesktopLinuxSDK Install Dir>/demos/scripts/* and *<DesktopLinuxSDK Install Dir>/demos/SARdemo/scripts*.
3. Cd to the python directory in a terminal.
4. Run the python scripts

   ```
   python ti_sar_demo_runner.py
   ```
5. A pop-up window appears.
6. Click the *Run Demo* button.
7. The user may use CTRL+C in the terminal to close the program, or restart if necessary.

13 Building and Running a DSP Application

1. Open CCS.
2. Import the *demo_SAR* project from *<DesktopLinuxSDK Install Dir>/demos/SARdemo/c66x/*.
3. Check that all tool versions are correctly selected in the project properties before building the project.
   (a) Right-click on the top of *demo_SAR* tab.
   (b) Go to *Properties*.
   (c) Go to *CCS General*.
   (d) Go to the *RTSC* tab.
   Tool Versions:
   - CGT 7.4.0.
   - XDC Tools 3.23.4.60
   - SYSBIOS 6.33.6.50.
   - PDK C6678 1.1.2.6.
   - OMP 1.1.3.2.
   - EDMA–LLD 2.11.5.2.
   - NDK 2.21.1.38.
   - IPC 1.24.3.32
   - DSPLIB 3.1.0.0.
4. Update the DESKTOP_LINUX_SDK_DIR and NDK_INSTALL_DIR paths in the *Project Properties→CCS Build and Variables* tab.
5. In the *Project Properties→CCS General*, Linker command file, check that the *demo_SAR.cmd* is selected.
6. Clean project.
7. Build project.
8. The user may find the *demo_SAR.out* for the debug configuration at *<DesktopLinuxSDK Install Dir>/demos/SARdemo/c66x/build/Debug/*.
9. For the release configuration at *<DesktopLinuxSDK Install Dir>/demos/SARdemo/c66x/build/bin/*.
10. If the user wants to change the number of used cores. modify the *config→nthreads* variable inside of the RDA.c file and rebuild the project. The demo runs with the 1, 2, 4, or 8 cores configuration.
11. The platform used for this demo is included inside of *<DesktopLinuxSDK Install Dir>/demos/SARdemo/platformRepo*.

Optional–Running File to File SAR demo:

1. In a terminal window, go to <DesktopLinuxSDK Install Dir>/demos/scripts.
   ```
   ./init_evmc6678l_1250.sh
   ```
2. Run the DSP reset script:
   ```
   ./dspreset.sh 1
   ```
3. Go to SAR scripts:
   ```
   cd ../SARdemo/scripts
   ```
4. Download and run release binary built of SAR demo:
   ```
   ./dnld_SARdemo_rel.sh 1
   ./run_hugebuf_rel_iter.sh 1 0×8000000 1
   ```
5. For re-running the demo reset DSP (see step 2 of this list).
6. Download and debug the binary built of SAR demo.

```
./dnld_SARdemo_dbg.sh 1
./run_hugebuf_dbg_iter.sh 1 0×8000000 5
```

14 Design Files

14.1 Software Files

To download the software files, see the design files at TIDEP-0045.

15 Additional Documentation

1. *Desktop Linux SDK Development Guide*, http://processors.wiki.ti.com/index.php/Desktop-linux-sdk_01.00.00_Development_Guide .
2. *White Paper: Synthetic Aperture Radar (SAR) implementation ona on a TMS320C6678 Multicore DSP*, http://www.ti.com/lit/pdf/spry276 .

16 FAQ

1. If the demo hangs after Mailbox Creation, check and make sure that you are using XDC Tools 3.23.4.60 along with CGT 7.4.0. Other XDC and compiler version combinations might not work correctly with OpenMP.
2. If the user has issues discovering BIOS MCSDK and PDK components in CCS with this reported error, *cannot be installed in this environment because its filter is not applicable*. The user must search for the ws="win32" reference and delete it. Find the *win32* reference inside MCSDK and PDK .xml files located in /features and /plugins folders. After deleting these references, clean ccstudio cache using: *sudo ./ccstudio -clean*.

17 About the Author

DAN WANG is a System Engineer in the Perception Lab in Silicon Development under Embedded Processing. She received her master degree from the department of Electrical Engineering at the University of Minnesota at Twin Cities, and her PhD degree from the department of Electrical Engineering at the University of Texas at Austin in 2012. Since she joined TI in 2012, Dan has been working on system development for various applications based on low power embedded processors including algorithm development in Synthetic Aperture Radar, software development in ADAS (advanced driving assistant system in automotive applications), and various industry applications.

PAULA CARRILLO is a Software Engineer for the Embedded Processing group at TI. She obtained her MSEE from Florida Atlantic University and her bachelor degree at Javeriana University, Colombia. Since joining TI in 2009, Paula has been working on different SoC and multi-core DSP platforms; developing applications for high-performance video codecs, syntechtic aperture radar (SAR), and industrial communication protocols.

MURTAZA ALI leads the Perception and Analytics Lab in the Silicon Development Group under the Processor Business Unit. His current activities include environmental sensing and perception, using various sensing technologies such as camera, radar, lidar, and more. Murtaza holds a Ph.D. from the University of Minnesotta in Electrical Engineering. In the past, Murtaza led research and development tems for mobile WiMAX, ADSL, and voice-band technologies in TI. He has also represented TI in various national and international standards organization including TIA, ITU, HomePlug, HPNA, and IEEE. Murtaza also holds 18 U.S. patentes and has published over 40 papers in refereed and invited forums. He is also a senior member of the IEEE.

IMPORTANT NOTICE FOR TI REFERENCE DESIGNS

Texas Instruments Incorporated ("TI") reference designs are solely intended to assist designers (“Buyers”) who are developing systems that incorporate TI semiconductor products (also referred to herein as “components”). Buyer understands and agrees that Buyer remains responsible for using its independent analysis, evaluation and judgment in designing Buyer’s systems and products.

TI reference designs have been created using standard laboratory conditions and engineering practices. **TI has not conducted any testing other than that specifically described in the published documentation for a particular reference design.** TI may make corrections, enhancements, improvements and other changes to its reference designs.

Buyers are authorized to use TI reference designs with the TI component(s) identified in each particular reference design and to modify the reference design in the development of their end products. HOWEVER, NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY THIRD PARTY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT, IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI REFERENCE DESIGNS ARE PROVIDED "AS IS". TI MAKES NO WARRANTIES OR REPRESENTATIONS WITH REGARD TO THE REFERENCE DESIGNS OR USE OF THE REFERENCE DESIGNS, EXPRESS, IMPLIED OR STATUTORY, INCLUDING ACCURACY OR COMPLETENESS. TI DISCLAIMS ANY WARRANTY OF TITLE AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, QUIET ENJOYMENT, QUIET POSSESSION, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS WITH REGARD TO TI REFERENCE DESIGNS OR USE THEREOF. TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY BUYERS AGAINST ANY THIRD PARTY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON A COMBINATION OF COMPONENTS PROVIDED IN A TI REFERENCE DESIGN. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, SPECIAL, INCIDENTAL, CONSEQUENTIAL OR INDIRECT DAMAGES, HOWEVER CAUSED, ON ANY THEORY OF LIABILITY AND WHETHER OR NOT TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, ARISING IN ANY WAY OUT OF TI REFERENCE DESIGNS OR BUYER’S USE OF TI REFERENCE DESIGNS.

TI reserves the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques for TI components are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

Reproduction of significant portions of TI information in TI data books, data sheets or reference designs is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards that anticipate dangerous failures, monitor failures and their consequences, lessen the likelihood of dangerous failures and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in Buyer’s safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed an agreement specifically governing such use.

Only those TI components that TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components that have ***not*** been so designated is solely at Buyer's risk, and Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265