TI Designs

Using TI's TMS320C6678 Processor to Implement a Power-Efficient Scalable H.265 / HEVC Solution

Design Overview

HEVC is an efficient but processor-intensive video standard that can double the data compression ratio compared to H.264 / MPEG-4 at the same level of video quality. This design shows how a power efficient, software H.265 / HEVC solution that scales across resolutions, frame rates, and profiles can be implemented in real time using one or more TMS320C6678 devices. A specific use case of a single-channel HEVC 720p30 real time encoder and single channel HEVC 1080p60 real time decoder is also included. TI's HEVC C66x HEVC encoder shows a bitrate saving, for the same visual quality, of ~40% compared with an existing H.264 encoder.

Design Resources

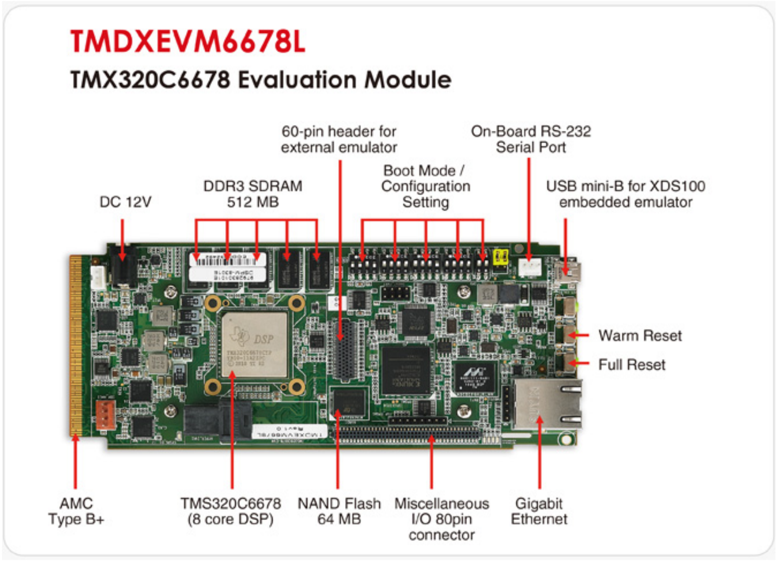
TMDSEVM6678	TMDSEVM6678 (L/LE) information
TMDXEVMPCI	AMC to PCIe adapter card information
C66x Codecs	Video and Speech Codecs information
HEVC Decoder	Download latest HEVC Decoder
HEVC Encoder	Download latest HEVC Encoder
MCSDK Video	MCSDK Download and details
Demo Guide	MCSDK Video 2.0 Demo Guide
TIDEP0037	TI Designs TIDEP0037

Design Features

- This reference design is tested, and includes a hardware reference (EVM), software, and a user's guide
- TMDSEVM6678 EVM for a high performance, cost-efficient, standalone development platform, using theTMS320C6678 high-performance DSP based on TI's C66x Keystone multicore architecture.
- This design includes schematics, design files, and a bill of materials.
- HEVC/ H.265 encoder and decoder, MCSDK framework, and other software packages
- Design guide discusses performance and scalability across DSP cores and devices to achieve the desired HEVC configuration

TI E2E™ Community

ASK Our E2E Experts
Ask the Keystone Experts


TMDXEVM6678L
TMX320C6678 Evaluation Module
60-pin header for external emulator
On-Board RS-232 Serial Port
Boot Mode / Configuration Setting
DDR3 SDRAM 512 MB
USB mini-B for XDS100 embedded emulator
DC 12V
Warm Reset
Full Reset
AMC Type B+
TMS320C6678 (8 core DSP)
NAND Flash 64 MB
Miscellaneous I/O 80pin connector
Gigabit Ethernet

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

1 Introduction

This design shows HEVC codecs running in real-time on the TMDSEVM6678 board (on both TMDSEVM6678L and TMDSEVM6678LE versions).

High-Efficiency Video Coding (HEVC) is the latest video compression standard, prepared in a joint effort between ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG). The main goal of the HEVC standardization effort is to improve compression performance compared to its predecessor (H.264) in the range of 50% bit rate reduction for equal perceptual video quality.

HEVC, also called H.265, employs the same hybrid approach (inter-/intra prediction and 2D transform coding) used in all video compression standards, to create a bitstream conforming to the standard.

In HEVC, a picture is partitioned into coding tree blocks (CTBs). The size of the CTBs (64×64, 32×32, and 16×16) can be chosen by the encoder, depending on input characteristics. Within CTBs, there can be multiple intra/inter prediction CUs (32×32, 16×16, and 8×8), as compared to H264, where the picture was divided into fixed size MBs.

Table 1 shows the new and enhanced features in HEVC compared to H.264

Table 1. New HEVC Features

Tool	H.264/AVC	HEVC
Coding Unit	16×16, intra or inter-coded MB	64×64, 32×32, or 16×16, hybrid intra/inter in a LCU, quad-tree signaling of CU partitioning
Intra prediction	Luma8×8: angular (8 modes), DC Luma16×16/Chroma4×4: Ver. Hor. DC, plane	Luma/Chroma (4×4 to 32×32): Planar, DC, Angular (33 modes), MDIS
Inter prediction	6-tap for luma, bilinear for chroma , WP	8/7-tap for luma, 4-tap for chroma, WP
Inter prediction partitions	16×16, 16×8, 8×16, 8×8, 8×4, 4×8, 4×4	2N×2N, 2N×N, N×2N, N×N (N = 4, 8, 16, 32) 2N×nU, 2N×nD, nL×2N, nR×2N (N = 8, 16, 32)
Skip/direct/merge mode	Skip/direct, single MVP candidate	Skip/merge, up to 5 MVP candidates, merge_idx signaled
MVD coding	Single MVP candidate, median filter of MVPs	AMVP list up to 2 candidates, mvp_idx signaled
Transform	Luma: 4×4/8×8, 2×2 DC transform Chroma: 4×4	4×4 (DST for Intra, DCT for Inter), 8×8, 16×16, 32×32 (32×32 for luma only, size signaled w/ RQT)
Entropy coding	CABAC/CAVLC	CABAC (high throughput), sign data hiding
De-blocking	4×4 edges based	8×8 edges based
In-loop filters	No additional in-loop filters	SAO (Sample Adaptive Offset)
Parallel processing	Regular slices	Regular slices, Tiles, Wavefront Parallel Processing
Lossless coding	I_PCM	I_PCM, T&Q bypass

TI’s HEVC C66x encoder shows a bitrate saving, for the same visual quality, of approximately 40% compared with TI’s H.264 HP DM816x encoder (also known as truView) for HD resolutions in Random Access (IBBBBBBBBP) configuration.

TI’s HEVC solution is feature-rich, and can support a variety of resolutions, profiles, and frame rates.

For DSP performance details on a variety of HEVC configurations, see Section 11.

2 System Overview

This TI design uses MCSDK video software framework as a software package, which provides video specific software modules, video codecs, and analytics algorithms, along with several out-of-box demos. MCSDK Video, in conjunction with BIOS-MCSDK and Desktop Linux® SDK, provides a complete development environment [host + DSP] to offload real-time video applications to TI C66x multi-core DSPs.

MCSDK Video executes highly compute-intensive video processing on TI C66x multi-core DSPs; it also provides a host application built on Linux Desktop SDK to allow a user-friendly interface to run video demos on a Linux PC. As shown in Figure 1, DSP and the host processor share common header files for message interpretation and communication through a host-DSP mailbox.

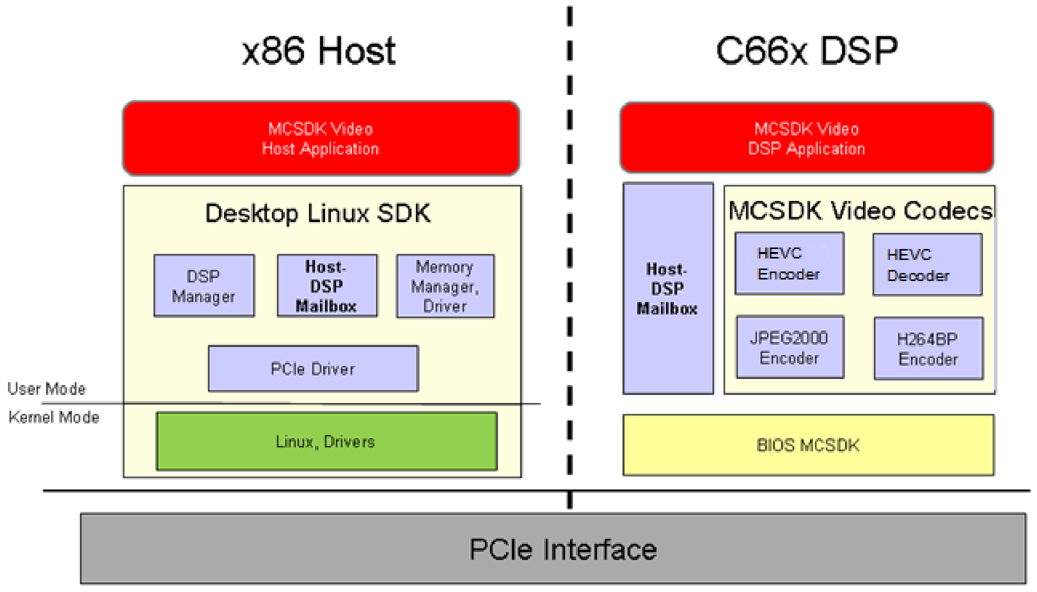


Figure 1. DSP and Host Processor

MCSDK video provides a multicore and multichip framework that can be used for different codecs and video algorithms. For this TI design, we will focus on running HEVC codecs on a single C6678 chip.

3 Required Hardware

- Evaluation board TMDSEVM6678L / TMDSEVM6678LE
- PCIe adapter TMDXEVMPCI
- Linux desktop PC with Ubuntu 12.04 LTS
- Motherboard with a PCIe slot for plugging TI's DSP card

4 Required Software

The software packages required and overall steps are:

1. Install the Desktop Linux SDK
2. Install the required Linux packages
3. Install the BIOS MCSDK
4. Install the MCSDK Video 2.x
5. Install Active Perl

Each MCSDK video release is paired with a Desktop Linux SDK and a BIOS-MCSDK version. Refer to the MCSDK video download page for version details. For this TI design, MCSDK video 2.2.0.45 release was used: http://software-dl.ti.com/sdoemb/sdoemb_public_sw/mcsdk_video/02_02_00_45/index_FDS.html.

1. Installing the Desktop Linux SDK

   After downloading Desktop Linux SDK, change the attribute of the installer to executable and run the installer as shown below.

   ```
   chmod +x desktop-linux-sdk_<version>_setuplinux.bin
   ./desktop-linux-sdk_<version>_setuplinux.bin
   ```

   Video demos require large and pre-reserved contiguous memory allocation. The following is an example for 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
   ```

   Note that this procedure only works if the memory on the desktop is more than 4 GB. If the memory is <= 4 GB, this is not recommended. Skip this step if using < 4 GB.

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

2. Installing the required Linux packages

   ```
   sudo apt-get install libpciaccess-dev
   sudo apt-get install binutils-dev
   sudo apt-get install freeglut3-dev
   sudo apt-get install libglew1.6-dev
   ```

3. Installing BIOS MCSDK

   ```
   chmod +x bios_mcsdk<version>_setuplinux.bin
   ./bios_mcsdk<version>_setuplinux.bin
   ```

4. Installing MCSDK Video

   Choose the installation directory and components to install.

   ```
   chmod +x mcsdk_video_<version>_setuplinux.bin
   sudo ./mcsdk_video_<version>_setuplinux.bin
   ```

   Select HEVC ENC 01.00.00.44 and HEVC DEC 01.00.00.29. Along with the MCSDK VIDEO 02.02.00.45 component.

   For additional details, refer to http://processors.wiki.ti.com/index.php/MCSDK_VIDEO_2.x_Getting_Started_PCIe_Demo_Guide#Software_Installation.

5 MCDSK Video DSP Application

1. DSP memory map for running single Shannon EVM

   To ensure that the MCSDK video DSP application (sv04.out) can run on a single Shannon EVM, check that “HOST_MANAGED_MEM” inside <*MCSDK Video Install Dir>>/dsp/ggcfg/build/hdg/sv04/ggvf0.becmd* has a length of 0x02000000. Below is an example of an external DDR memory map:

```
 /*-----------------------------------------------------------------------*/
  /* ---------------------- External DDR memory (0x80000000-0xBFFFFFFF) --- */
  /*-----------------------------------------------------------------------*/
    MAILBOX_DSP_TO_HOST      : origin = 0x80000000,  length = 0x00180000
    MAILBOX_HOST_TO_DSP      : origin = 0x80180000,  length = 0x00180000
    DDR_UNCACHED             : origin = 0x80300000,  length = 0x01B00000
    MAILBOX_CHIP2CHIP_RX     : origin = 0x81E00000,  length = 0x00200000
    CHIP2CHIP_XFER_PCIE      : origin = 0x82000000,  length = 0x00E00000
    CHIP2CHIP_XFER_HLNK      : origin = 0x82E00000,  length = 0x00E00000
    HOST_MANAGED_MEM         : origin = 0x83C00000,  length = 0x02000000 //value Increased
from 0x01200000
    DDR_CACHED               : origin = 0x85C00000,  length = 0x1A1FFF00 //value decreased
from 0x1AFFFF00
    DIAG_APP                 : origin = 0x9FDFFF00,  length = 0x00000100
```

   NOTE: The above example was extracted from MCSDK video 2.2.0.45. There, HOST_MANAGED_MEM was increased and DDR_CACHED was decreased. Due to these changes, some other codecs such as JPEG, MPEG2, and so forth won’t fit in DDR. Refer to Section 12 for an example on removing unnecessary codecs from sv04.out.

2. Rebuilding the DSP side of MCSDK video demos

   To rebuild sv04.out, go to <*MCSDK Video Install Dir>\dsp\mkrel*

   - Open setupenvMsys.sh and check TI_TOOL_DIR, which points to your installation path directory TI_TOOL_DIR=/home/alice/ti
   - In a terminal window, go to <*MCSDK Video Install Dir>\dsp\mkrel* and run the lines below in **bold** for configuring the environment and rebuilding sv04.out.

```
ubuntu:~/ti/mcsdk_video_x_x_x_x/dsp/mkrel$ source setupenvMsys.sh

Please wait... configuring environment
*****************************************************************************
Note: Each dependant RTSC package will also do verify tools.  To stop this,
re-run this script with "bypass" argument
*****************************************************************************
===================== ENVIRONMENT SUCESSFULLY CONFIGURED ======================
ubuntu:~/ti/mcsdk_video_x_x_x_x/dsp/mkrel$ make clean
ubuntu:~/ti/mcsdk_video_x_x_x_x/dsp/mkrel$ make sv04
```

   For more details, go to http://processors.wiki.ti.com/index.php/MCSDK_VIDEO_2.x_Getting_Started_TFTP_Demo_Guide#Rebuild_MCSDK_Video_Demo_Build.

6 MCDSK Video HOST Application

1. HOST source files modifications

   - Inside <*MCSDK Video Install Dir>/host/src/dspMemMap.h*, enable #define SINGLE_DSP_EVM_BUILD
   - Inside <*MCSDK Video Install Dir>/host/src/demoBufferPoolCfg.h* , check that the single Shannon EVM DSP DDR memory regions are correctly defined. DEMO_C667X_DEV_OUTPUT_POOL_BASEADDR should correspond to the previously defined DSP side HOST_MANAGED_MEM origin, and DEMO_C667X_DEV_OUTPUT_POOL_SIZE + DEMO_C667X_DEV_INPUT_POOL_SIZE should be equal to HOST_MANAGED_MEM length. See the example below.

```
/*Configuration for input buffer pool in the DSP DDR memory.*/
#define DEMO_C667X_DEV_OUTPUT_POOL_BASEADDR          0x83C00000
#define DEMO_C667X_DEV_OUTPUT_POOL_SIZE              0x01000000

/*Configuration for output buffer pool in the DSP DDR memory.*/
#define DEMO_C667X_DEV_INPUT_POOL_BASEADDR           0x84C00000
#define DEMO_C667X_DEV_INPUT_POOL_SIZE               0x01000000
```

2. Environment variables

   Modify *<MCSDK Video Install Dir>/host/setEnv.sh* to set the installation directory of Desktop Linux SDK and MCSDK Video. For example, if Desktop Linux SDK is installed at */home/alice/ti/desktop-linux-sdk_01_00_03_00* and MCSDK Video is installed at */home/alice/ti/mcsdk_video_2_2_0_45*, then modify setEnv.sh as shown below:

   ```
   #!/bin/sh
   export DESKTOP_LINUX_SDK_DIR=/home/alice/ti/desktop-linux-sdk_01_00_03_00
   export MCSDK_VIDEO_DIR=/home/alice/ti/mcsdk_video_2_2_0_45
   ```

3. Building the host side of MCSDK video demos

   In a terminal window, go to *<MCSDK Video Install Dir>/host/* and run the following commands to both set up the environment, and build Desktop Linux SDK and the host side of MCSDK video demos (demo_c667x):

   ```
   cd host
   source setEnv.sh
   ./makeCleanSdk.sh
   ./makeSdk.sh
   ./ makeCleanVideoApp.sh
   ./ makeVideoApp.sh
   ```

7 EVM Preparation

1. IBL Flash

   (a) Unzip I2crom_0x51_c6678_le.zip to obtain and use an updated i2crom_0x51_c6678_le.bin, in which the BAR region sizes have been tuned for the use case of video demos.

   (b) Copy i2crom_0x51_c6678_le.bin (from the previous step) to mcsdk_2_00_xx_xx\tools\writer\eeprom\evmc6678l\bin. Rename this copied file to app.bin.

   (c) Open eepromwriter_input.txt in mcsdk_2_00_xx_xx\tools\writer\eeprom\evmc667#l\bin. Set file_name equal to app.bin and bus_addr equal to 0x51. Ensure start_addr and swap_data are set to 0. Save and close eepromwriter_input.txt.

   (d) Turn on and connect your EVM. Open CCSv5, load the appropriate Target Configuration, connect to Core 0, and load the corresponding GEL file.

   (e) Load the EEPROM writer program by going to Run -> Load Program, and browse for the eeprom writer DSP executable. For example, eepromwriter_evm6678l.out in the same folder as app.bin for C6678 EVM.

   (f) View the memory browser (go to View -> Memory Browser). Browse to address 0x0C000000.

   (g) Right-click on the memory window, and select Load Memory. Select app.bin (by default, the browse menu only displays .dat files. You will have to change the option TI Data Format (*.dat) to Raw Data Format (*.bin) to find your binary file.)

   (h) Change the Start Address to 0x0C000000 if it is not already. Leave the swap checkbox unchecked. Click Finish. Select 32-bits for Type-Size option in CCS.

   (i) Run the program. This programs the EEPROM.

   (j) A sample successful eeprom writer output would look 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 ...
   EEPROM programming completed successfully
   ```

2. Hardware Setup
   (a) Set the EVM card to PCIE boot through the following switch settings (For SW3, pin 1: OFF: Little Endian;)

SW3	SW4	SW5	SW6	SW9
(pin1, 2, 3, 4)	(pin1, 2, 3, 4)	(pin1, 2, 3, 4)	(pin1, 2, 3, 4)	(pin1)
(off, on, on, off)	(on, on, on, on)	(on, on, on, off)	(off, on, on, on)	(off)

   (b) Assemble the EVM card into the adaptor card.
   (c) Completely shut off the PC power supply (by disconnecting the power cord).
   (d) Insert the AMC adapter card (with TMS320C6678L EVM card mounted) into an open PCIE slot in the PC motherboard.
   (e) Supply power to PC, wait for a few seconds, and power on the PC.
   (f) Ensure that the PCIE device is correctly enumerated by the PC by typing "lspci –n" under the Linux command shell after Linux OS is loaded; a TI device (VENDOR_ID: 0x104c) should be in the list:

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

8 Install and Run HEVC Encoder and Decoder Demo

1. Installing HEVC demos

   Before running the demos, update setDemoEnv with the correct Desktop Linux SDK and MCSDK video directory path. See example below:

   <MCSDK Video Install Dir>/demos/codecs/pcie_h265enc_demo/scripts/setDemoEnv.sh

   <MCSDK Video Install Dir>/demos/codecs/pcie_h265dec_demo/scripts/setDemoEnv.sh

```
#!/bin/sh
export DESKTOP_LINUX_SDK_DIR=/home/alice/ti/desktop-linux-sdk_01_00_03_00
export MCSDK_VIDEO_DIR=/home/alice/ti/mcsdk_video_2_2_0_45
```

   Then, run the following commands to set up the environment variable if this has not been done before. Finally, install the scripts and configuration parameters used by the codecs running installDemo.sh:

```
cd demos/codecs/pcie_h265xxx_demo/scripts
source setDemoEnv.sh
./installDemo.sh
```

2. One time Setup and Initialization

   Every time following a reboot of the PC, run source setDemoEnv.sh:

```
cd demos/codecs/pcie_h265xxx_demo/scripts
source setDemoEnv.sh
```

   Then, initialize the DSPs by running the following commands, with the CPU frequency specified for DSP initialization.

```
./init_dsp.sh 1250
```

   Note that the above init_dsp.sh must execute just once after powering up the demo Linux PC.

3. Running HEVC encoder demo

```
cd demos/codecs/pcie_h265enc_demo/scripts
./reset_dsp.sh 1250
./dnld_dsp.sh 1
```

   Check that the demoSave2File.sh script has the correct information for running in a single C6678 chip (8 cores), and also check that the clip resolutions are correct, as in the following example:

```
#!/bin/sh

CLIP_CONFIG=multiClip.cfg
CODEC_PARAMS=codecParams.cfg

$MCSDK_VIDEO_HOSTBIN_DIR/demo_c667x --dsp-image=$MCSDK_VIDEO_DSPIMAGE_DIR/sv04.out -v --
channel="H265 Encode" -a HEVCENC -f $CLIP_CONFIG -c $CODEC_PARAMS -t "0 1 2 3 4 5 6 7" --
width=1280 --height=720
```

Check that codecParams.sh has the correct information for running your use case resolution, bitrate, and frames per second. For more information about configuration parameters, refer to: *C:\TI\Codecs\C66x_h265venc_XX_XX_XX_XX_ELF\packages\ti\sdo\codecs\h265venc\Docs\HEVC_Encoder_C6678_UserGuide.pdf*.

If you must modify the clip, input/output clips locations, or number of frames to encode, update multiClip.cfg. The example below is for five frames:

```
../testVecs/input/airshow_p1280x720.yuv
../testVecs/output/airshow_p1280x720.265
5
```

Finally, run the demo script:

```
./demoSave2File.sh
```

4. Running HEVC decoder demo

```
cd demos/codecs/pcie_h265dec_demo/scripts
./reset_dsp.sh 1250
./dnld_dsp.sh 1
```

Check that the demoSave2File.sh script has the correct information for running in a single C6678 chip (8 cores), and also check that the clip resolutions are correct, as in the following example:

```
#!/bin/sh

CLIP_CONFIG=multiClip.cfg
CODEC_PARAMS=codecParams.cfg

$MCSDK_VIDEO_HOSTBIN_DIR/demo_c667x --dsp-
image=$MCSDK_VIDEO_DSPIMAGE_DIR/sv04.out -v --channel="H265 Decode" -a HEVCDEC
-f $CLIP_CONFIG -c $CODEC_PARAMS -t "0 1 2 3 4 5 6 7" --width=1920 --height=1080
```

If you must modify the clip, input/output clips locations, or number of frames to encode, update multiClip.cfg. The example below is for five frames:

```
../testVecs/input/airshow_p1920x1080.yuv
../testVecs/output/airshow_p1920x1080.265
5
```

If you must modify the HEVC decoder configuration parameters, check codecParams.cfg. Finally, run the demo script.

```
./demoSave2File.sh
```

9 Additional Documentation

- Development Guide for MCSDK Video:
  http://processors.wiki.ti.com/index.php/MCSDK_VIDEO_2.x_Development_Guide
- Desktop Linux SDK Development Guide:
  http://processors.wiki.ti.com/index.php/Desktop-linux-sdk_01.00.00_Development_Guide
- White Paper: The Next Frontier in Video Encoding
  http://www.ti.com/lit/pdf/spry246
- White Paper: Multicore Video Processing on TMS320C6678
  http://www.ti.com/lit/pdf/sprabi6

10 Latest HEVC Codec Libraries

http://software-dl.ti.com/dsps/dsps_public_sw/codecs/C6678/HEVC_D/latest/index_FDS.html

http://software-dl.ti.com/dsps/dsps_public_sw/codecs/C6678/HEVC_E/latest/index_FDS.html

11 HEVC Encoder and Decoder Target Performance

Table 2. HEVC Encoder Target Performance

Configuration	Resolution[1]	# of C6678 devices at 1.25 GHz
Low Delay	720p30	6 cores of 8
Low Delay	1080p30	1+ DSP (10 cores)
Low Delay	1080p60	2.5 DSP (20 cores)
Low Delay	4kp30	5 DSP (40 cores)
Low Delay	4kp60	10 DSP (80 cores)
Standard	720p30	1 DSP (8 cores)
Standard	1080p30	2 DSP (16 cores)
Standard	1080p60	4 DSP (32 cores)
Standard	4kp30	8 DSP (64 cores)
Standard	4kp60	16 DSP (128 cores)
Broadcast	720p30	2 DSP (16 cores)
Broadcast	720p60	4 DSP (32 cores)
Broadcast	1080p30	4 DSP (32 cores)
Boradcast	1080p60	6 DSP (48 cores)
Broadcast	4kp30	12 DSP (96 cores)
Broadcast	4kp60	24 DSP (192 cores)

[1] 4kp60 performance numbers are theoretically derived from 4kp30.

Table 3. HEVC Decoder Target Performance

Configuration	Resolution	# of C6678 devices at 1.25 GHz
Low Delay	720p30	2 cores of 8
Low Delay	1080p30	3 cores of 8
Low Delay	1080p60	5 cores of 8
Standard/Broadcast	720p30	2 cores of 8
Standard/Broadcast	720p60	4 cores of 8
Standard/Broadcast	1080p30	4 cores of 8
Standard/Broadcast	1080p60	1 DSP (8 cores)

For additional performance details, refer to the data sheet at http://software-dl.ti.com/dsps/dsps_public_sw/codecs/C6678/HEVC_E/latest/index_FDS.html.

Table 4. HEVC Decoder Target Performance Multichip

Configuration	Resolution[1]	# of C6678 devices (1.25 GHz)[2]
Low Delay	4kp30	2 DSP (16 cores)
Low Delay	4kp60	3 DSP (24 cores)
Standard/Broadcast	4kp30	2 DSP (16 cores)
Standard/Broadcast	4kp60	4 DSP (32 cores)

[1] 4kp60 performance numbers are theoretically derived from 4kp30.
[2] Multichip decoder requires using equally divided tiles. Tiles can be horizontal or vertical sub-pictures. The number of required tiles is based on the number of used chips.

For additional performance details, refer to the data sheet at http://software-dl.ti.com/dsps/dsps_public_sw/codecs/C6678/HEVC_E/latest/index_FDS.html.

12 Appendix A

Below is an example snippet code which shows how to take out codecs from the MCSDK video demo DSP application, to free DDR memory.

1. Inside *<MCSDK Video Install Dir>/dsp/ggcfg/build/hdg/sv04/ggvf0.becmd*.
   - Commenting EABI/COFF aliases for other codecs:

```
/*_text_h264hpvenc_start  = __text_h264hpvenc_start;*/
/*_text_h264hpvenc_end    = __text_h264hpvenc_end;*/
_text_h265venc_start    = __text_h265venc_start;
_text_h265venc_end      = __text_h265venc_end;
/*_text_avciuvenc_start   = __text_avciuvenc_start;*/
/*_text_avciuvenc_end     = __text_avciuvenc_end;*/
/*_text_mpeg2venc_start   = __text_mpeg2venc_start;*/
/*_text_mpeg2venc_end     = __text_mpeg2venc_end;*/
/*_text_mp4venc_start     = __text_mp4venc_start;*/
/*_text_mp4venc_end       = __text_mp4venc_end;*/
/*_text_jpgealg_start     = __text_jpgealg_start;*/
/*_text_jpgealg_end       = __text_jpgealg_end;*/
/*_text_h264hpvdec_start  = __text_h264hpvdec_start;*/
/*_text_h264hpvdec_end    = __text_h264hpvdec_end;*/
/*_text_h264dec_start     = __text_h264dec_start;*/
/*_text_h264dec_end       = __text_h264dec_end;*/
_text_h265vdec_start    = __text_h265vdec_start;
_text_h265vdec_end      = __text_h265vdec_end;
/*_text_mpeg2vdec_start   = __text_mpeg2vdec_start;*/
/*_text_mpeg2vdec_end     = __text_mpeg2vdec_end;*/
/*_text_m4h3dec_start     = __text_m4h3dec_start;*/
/*_text_m4h3dec_end       = __text_m4h3dec_end;*/
/*_text_jpegdec_start     = __text_jpegdec_start;*/
/*_text_jpegdec_end       = __text_jpegdec_end;*/
/*_text_j2e_j2d_start     = __text_j2e_j2d_start;*/
/*_text_j2e_j2d_end       = __text_j2e_j2d_end;*/
```

   - Commenting other codecs libraries:

```
/*-l ti/sdo/codecs/h264hpvenc/lib/h264hpvenc_ti.le66*/
/*-l ti/sdo/codecs/mp4venc/lib/mp4venc_ti.le66*/
/*-l ti/sdo/codecs/m4h3dec/lib/m4h3dec_ti.le66*/
/*-l ti/sdo/codecs/mpeg2vdec/lib/mpeg2vdec_ti.le66*/
/*-l ti/sdo/codecs/h264vdec/lib/h264dec_ti.le66*/
/*-l ti/sdo/codecs/h264hpvdec/lib/h264hpvdec_ti.le66*/
/*-l ti/sdo/codecs/jpegdec/lib/jpegdec_ti.le66*/
/*-l ti/sdo/codecs/jpegenc/lib/jpgealg_ti.le66*/
/*-l ti/sdo/codecs/j2d/lib/j2d_ti_c66.ae66*/
/*-l ti/sdo/codecs/j2e/lib/j2e_ti_c66.ae66*/
/*-l ti/sdo/codecs/avciuvenc/Lib/avciuvenc_ti_c66x.lib*/
/*-l ../../siu/vct/codec/encoder/hevc/alg/h265venc_ti.le66*/
/*-l ../../siu/vct/codec/decoder/hevc/alg/h265vdec_ti.le66*/
-l ti/sdo/codecs/h265vdec/lib/h265vdec_ti.le66
-l ti/sdo/codecs/h265venc/lib/h265venc_ti.le66
/*-l ti/sdo/codecs/mpeg2venc/lib/mpeg2venc_ti.le66*/
/*-l ti/sdo/codecs/h264venc/lib/h264venc_ti.le66*/
```

   -

Commenting other codecs in nonsharedfar:

```
.nonsharedfar_ll2 fill=0x00 > LL2_SRAM
{
       __far1_core_begin = .;
       ../../mkrel/c64x/bioscfg/package/cfg/bios6_pe66.oe66
(.far:taskStackSection)
       h265venc_ti.le66 (.fardata)
       h265vdec_ti.le66 (.fardata)
       h265vdec_ti.le66 (.far)
       /*avciuvenc_ti_c66x.lib (.fardata)*/
       /*j2e_ti_c66.ae66 (.fardata)*/
       /*j2d_ti_c66.ae66 (.fardata)*/
       __far1_core_end = .;
}
```

- Commenting other codecs in load and run sections:

```
UNION: run = MSMC_SRAM START(__text_msmc_start) END(__text_msmc_end)
{
    /*.text_h264hpvenc: load = DDR_CACHED, { h264hpvenc_ti.le66 (.text) }
START(__text_h264hpvenc_start) END(__text_h264hpvenc_end)*/
    .text_h265venc:   load = DDR_CACHED, { h265venc_ti.le66 (.text) }
START(__text_h265venc_start) END(__text_h265venc_end)
    /*.text_avciuvenc:  load = DDR_CACHED, { avciuvenc_ti_c66x.lib (.text) }
START(__text_avciuvenc_start) END(__text_avciuvenc_end)*/
    /*.text_mpeg2venc:  load = DDR_CACHED, { mpeg2venc_ti.le66 (.text) }
START(__text_mpeg2venc_start) END(__text_mpeg2venc_end)*/
    /*.text_mp4venc:    load = DDR_CACHED, { mp4venc_ti.le66 (.text) }
START(__text_mp4venc_start) END(__text_mp4venc_end)*/
    /*.text_jpgealg:    load = DDR_CACHED, { jpgealg_ti.le66 (.text) }
START(__text_jpgealg_start) END(__text_jpgealg_end)*/
    /*.text_h264hpvdec: load = DDR_CACHED, { h264hpvdec_ti.le66 (.text) }
START(__text_h264hpvdec_start) END(__text_h264hpvdec_end)*/
    /*.text_h264dec:    load = DDR_CACHED, { h264dec_ti.le66 (.text) }
START(__text_h264dec_start) END(__text_h264dec_end)*/
    .text_h265vdec:   load = DDR_CACHED, { h265vdec_ti.le66 (.text) }
START(__text_h265vdec_start) END(__text_h265vdec_end)
    /*.text_mpeg2vdec:  load = DDR_CACHED, { mpeg2vdec_ti.le66 (.text) }
START(__text_mpeg2vdec_start) END(__text_mpeg2vdec_end)*/
    /*.text_m4h3dec:    load = DDR_CACHED, { m4h3dec_ti.le66 (.text) }
START(__text_m4h3dec_start) END(__text_m4h3dec_end)*/
    /*.text_jpegdec:    load = DDR_CACHED, { jpegdec_ti.le66 (.text) }
START(__text_jpegdec_start) END(__text_jpegdec_end)*/
    /*GROUP: load = DDR_CACHED START (__text_j2e_j2d_start)
END(__text_j2e_j2d_end)
    {         .text_j2e:        { j2e_ti_c66.ae66 (.text) }
              .text_j2d:        { j2d_ti_c66.ae66 (.text) }
    }*/
  }
```

2. Inside *<MCSDK Video Install Dir>/dsp/siu/vct/codec/siuVctSupportedCodecs.c*, comment other codecs:
   - Commenting other codecs headers:

```
/*#include "vctMpeg4EncClient.h"*/
/*#include "vctH264EncClient.h"*/
/*#include "vctMpeg4DecClient.h"*/
/*#include "vctH264HpDecClient.h"*/
/*#include "vctMpeg2DecClient.h"*/
/*#include "vctH264BpMpDecClient.h"*/
/*#include "vctMctnfClient.h"*/
/*#include "vctJ2KDecClient.h"*/
/*#include "vctJ2kEncClient.h"*/
/*#include "vctAvciuEncClient.h"*/
#include "vctHevcEncClient.h"
#include "vctHevcDecClient.h"
/*#include "vctH264HpEncClient.h"*/
/*#include "vctMpeg2EncClient.h"*/
/*#include "vctJpegDecClient.h"*/
/*#include "vctJpegEncClient.h"*/
/*#include "vctAACEncClient.h"*/
/*#include "vctMp3DecClient.h"*/
/*#include "vctSorsparkDecClient.h"*/
/*#include "vctDv100DecClient.h"*/
/*#include "vctDv50DecClient.h"*/
/*#include "vctDv25DecClient.h"*/
/*#include "vctVc1DecClient.h"*/
```

   - Commenting other codecs from a supported list:

```
vidEncoderAPI_t supportedVidEncoders[] = {
   /*mpeg4EncAPI,*/
   /*h264EncAPI,*/
   /*j2kEncAPI,*/
   /*avciuEncAPI,*/
   /*h264hpEncAPI,*/
   /*mpeg2EncAPI,*/
     hevcEncAPI,
   /* INSTALL MORE CODECS HERE */
     vidEncNullAPI
};

vidDecoderAPI_t supportedVidDecoders[] = {
   /*mpeg4DecAPI,*/
   /*h264hpDecAPI,*/
   /*h264bpmpDecAPI,*/
   /*mpeg2DecAPI,*/
   /*j2kDecAPI,*/
     hevcDecAPI,
   /*sorsparkDecAPI,*/
   /*dv100DecAPI,*/
   /*dv50DecAPI,*/
   /*dv25DecAPI,*/
   /*vc1DecAPI,*/
   /* INSTALL MORE CODECS HERE */
   vidDecNullAPI
};
```

Revision History

Changes from Original (June 2015) to A Revision **Page**

- Removed HEVC Functionality table. 2
- Updated HEVC Encoder Target Performance table. 9
- Updated HEVC Decoder Target Performance table. 9
- Added HEVC Decoder Target Performance Multichip table. 9

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.