

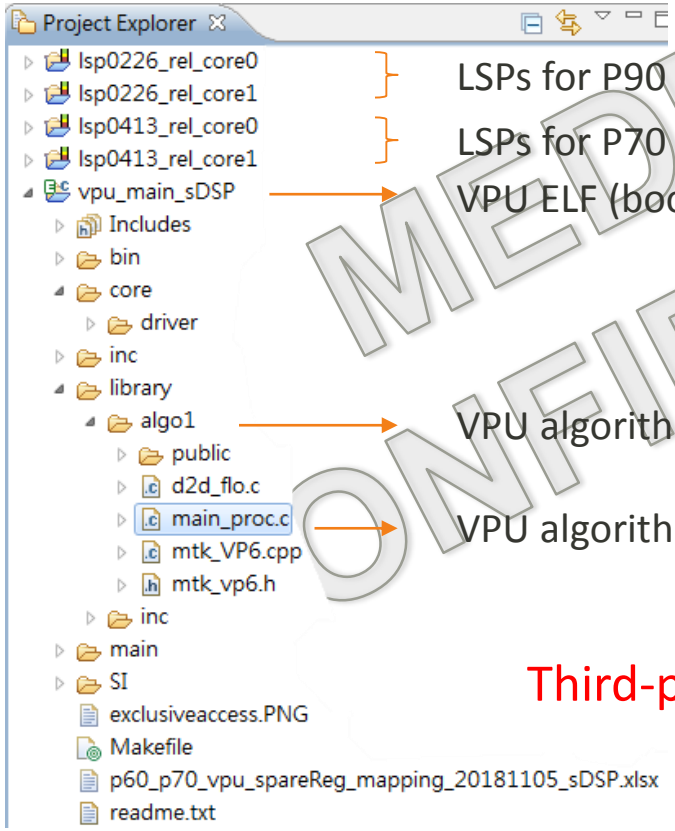

# 2019 MediaTek The New sDSP SDK

Secure DSP software platform for using VP6 HW shared with normal world

v1.1

# What's in legacy sDSP SDK

- A workspace of Xtensa Xplorer (most running at Win32 Env.)




LSPs for P90

LSPs for P70

VPU ELF (boot code + algorithm)

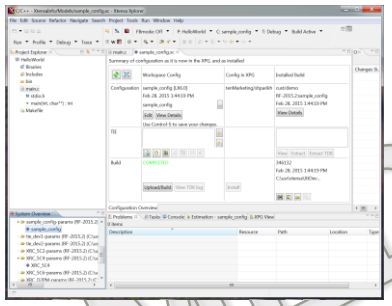
VPU algorithm directory

VPU algorithm implementation (sample code)

Third-party only engage with main_proc.c

# Build flow of legacy sDSP SDK

1. Open Xtensa Xplorer
2. Update VPU algorithm




+

3. Rebuild algorithm for Core#0

P: vpu_main_sDSP ▾ C: MVPU6_0226 ▾ T: Release0226Core0 ▾ Build Active ▾

+

4. Rebuild algorithm for Core#1

P: vpu_main_sDSP ▾ C: MVPU6_0226 ▾ T: Release0226Core1 ▾ Build Active ▾

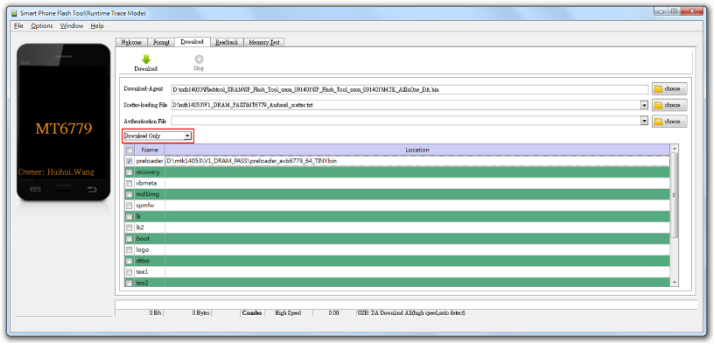
+

5. Rename the ELFs, and copy to Linux Env.

Linux Workstation

+

6. Rebuild gz.img, and download it to phone



# WHAT'S NEW IN THE NEW SDSP SDK

# sDSP SDK changes

- Linux build environment with CMake
  - No need to use Xplorer IDE
  - Only need Xtensa tools & VP6 HW Configuration (Linux version)
  - Multi-platform & parallel build support
  - ELF stripping & debugging message on/off support
- Source isolation between VPU boot code & algorithm
  - Boot code was maintained by MTK, and was released as a library (no source)
  - Sample source code for algorithm development
- Code was formatted using Linux kernel coding style
- Log message refinement
- LSP update for easy modifying the stack size
- Add drystone benchmark as an unit test (CMD: 0x16D)
- The new MTK VPU iDMA Driver

# Installing prerequisites

- Install commands on Linux:
  - Copy files into a convenient directory
  - tar xzf XtensaTools_<version>_linux.tgz
  - tar xzf <HW Configuration>_linux_redist.tgz
  - cd _<version>_linux/<HW Configuration>
  - ./install
  - Enter y
  - Enter tool path info (e.g. $HOME/RG-2018.10-linux/XtensaTools)
  - Enter y & enter & enter to finish installation
- Update toolchain path information to $sDSP_sdk/CMakeLists.txt
  - Update "MTK_PLATFORM" & "TOOL_PATH" & "LM_LICENSE_FILE"



| Platform | MT6771 | MT6779/MT6853 | MT6885/MT6873 |
|---|---|---|---|
| Tools | XtensaTools_RG_2017_7_linux.tgz<br>MVPU6_0413_Prod_linux_redist.tgz | XtensaTools_RG_2017_8_linux.tgz<br>MVPU6_0226_linux_redist.tgz | XtensaTools_RG_2018_10_linux.tgz<br>MVPU6F_1214_Prod_linux_redist.tgz |

# Toolchain license Issue?

- Any license-related issue, please contact Cadence support
  - Use “nc” command to check your license server is alive
    - nc -vz {host} {port}
    - nc -vz 192.168.2.254 2701
      - Connection to 192.168.2.254 2701 port [tcp/rtps-discovery] succeeded!

```
set(ENV{LM_LICENSE_FILE} "2701@192.168.2.254")
```

  - Open “~/.flexlmrc” file to check if previous license setting is correct

```
[  8%] Building C object CMakeFiles/vpu_main_sDSP_0.dir/library/algo1/d2d_flo.c.o
License checkout failed: Invalid license file syntax.
Feature:       XTENSA_XCC_TIE
License path:  /home/mtk16314/RG-2018.10-linux/XtensaTools/Tools/lic/license.dat:
FLEXnet Licensing error:-2,413
For further information, refer to the FLEXnet Licensing documentation,
available at "www.macrovision.com".
make[2]: *** [CMakeFiles/vpu_main_sDSP_0.dir/library/algo1/d2d_flo.c.o] Error 2
make[1]: *** [CMakeFiles/vpu_main_sDSP_0.dir/all] Error 2
make: *** [all] Error 2
```

# Build option & tool path settings in CMakeLists.txt

```
#=============================================================================================
# Please update "MTK_PLATFORM" & "TOOL_PATH" & "LM_LICENSE_FILE" according to your environment
#=============================================================================================

if("${MTK_PLATFORM}" STREQUAL "")
    set(MTK_PLATFORM mt6885)
    message("MTK_PLATFORM=${MTK_PLATFORM}")
endif()

set(DEBUG_PRINT 1)  -> 1 : Enable debug print
message("DEBUG_PRINT=${DEBUG_PRINT}")

set(ELF_STRIPPED 0) -> 0 : No stripping ELFs
message("ELF_STRIPPED=${ELF_STRIPPED}")

if(MTK_PLATFORM STREQUAL mt6885 OR MTK_PLATFORM STREQUAL mt6883)
    set(XTENSA_TOOL_VER RG-2018.10-linux)
    set(HW_CONFIG_FILE MVPU6F_1214_Prod)
    set(TOOL_PATH /mtkeda/xtensa/Xplorer-8.0.8/XtDevTools/install/tools/${XTENSA_TOOL_VER}/XtensaTools)
    set(VPU_HW_CONFIG 3)
elseif(MTK_PLATFORM STREQUAL mt6779 OR MTK_PLATFORM STREQUAL mt6785)
    set(XTENSA_TOOL_VER RG-2017.8-linux)
    set(HW_CONFIG_FILE MVPU6_0226)
    set(TOOL_PATH /mtkeda/xtensa/Xplorer-7.0.8/XtDevTools/install/tools/${XTENSA_TOOL_VER}/XtensaTools)
    set(VPU_HW_CONFIG 2)
elseif(MTK_PLATFORM STREQUAL mt6771)
    set(XTENSA_TOOL_VER RG-2017.7-linux)
    set(HW_CONFIG_FILE MVPU6_0413_Prod)
    set(TOOL_PATH /mtkeda/xtensa/Xplorer-7.0.7/XtDevTools/install/tools/${XTENSA_TOOL_VER}/XtensaTools)
    set(VPU_HW_CONFIG 1)
else()
    message(FATAL_ERROR "Unknown target: = ${MTK_PLATFORM}")
endif()
```

-> $HOME/RG-2018.10-linux/XtensaTools

-> $HOME/RG-2017.8-linux/XtensaTools

-> $HOME/RG-2017.7-linux/XtensaTools

# Source isolation of VPU boot code

```
|-- CMakeLists.txt
|-- README
|-- core -> Header files only
|   `-- driver
|       |-- dma
|       |-- hw
|       `-- utils
|-- docs -> sDSP SDK documentation
|-- elfs -> Prebuilt ELFs
|   |-- mt6771
|   |   |-- vpu0_main_sDSP
|   |   `-- vpu1_main_sDSP
|   |-- mt6779
|   |   |-- vpu0_main_sDSP
|   |   `-- vpu1_main_sDSP
|   `-- mt6885
|       |-- vpu0_main_sDSP
|       `-- vpu1_main_sDSP
|-- inc -> Header files only
|   |-- build_defs.h
|   |-- vpu_dbg.h
|   |-- vpu_drv.h
|   |-- vpu_event.h
|   `-- vpu_types.h
```

```
|-- library -> Sample code
|   |-- algo1
|   |   |-- d2d_flo.c
|   |   |-- dhrystone
|   |   |-- main_proc.c
|   |   |-- mtk_vp6.cpp
|   |   `-- mtk_vp6.h
|   `-- inc
|       `-- ikernel.h
|-- libs -> VPU boot code library
|   |-- libmt6771_vpu0.a
|   |-- libmt6771_vpu1.a
|   |-- libmt6779_vpu0.a
|   |-- libmt6779_vpu1.a
|   |-- libmt6885_vpu0.a
|   `-- libmt6885_vpu1.a
`-- lsp -> Memory map files
    |-- core0
    |   |-- memmap.xmm
    |   `-- specs
    `-- core1
        |-- memmap.xmm
        `-- specs
```

# Log message refinement

- Before:

VPU_SW_BUILD_DATE(0x19030616)

VPU_SW_VERSION(0x16122608)

VERSION_VPU_PORT_ST(0x0103b7f3)

vpu_lib_d2d_ksample

- After:

BootVer(0x19100818)

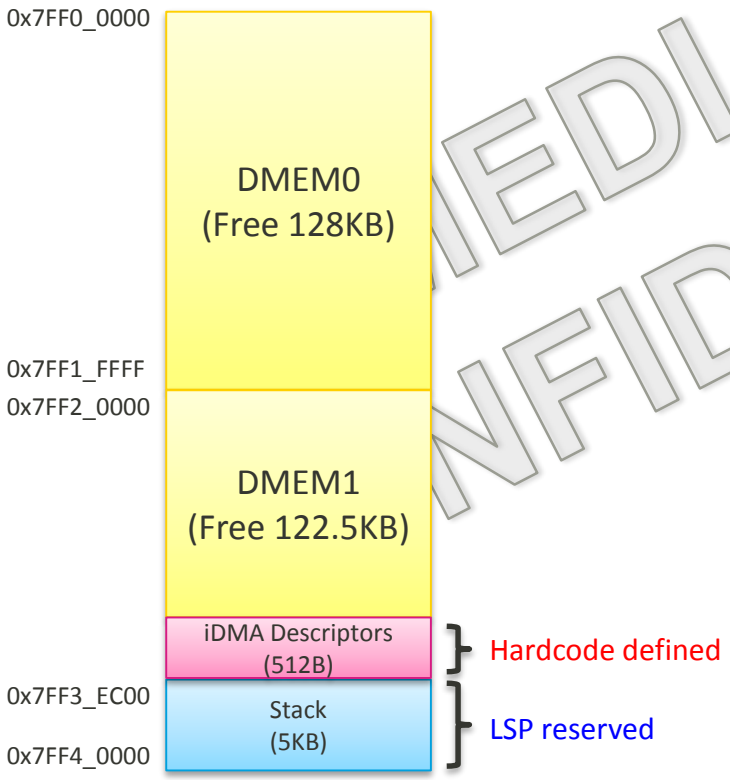
AlgoVer(0x19100818)



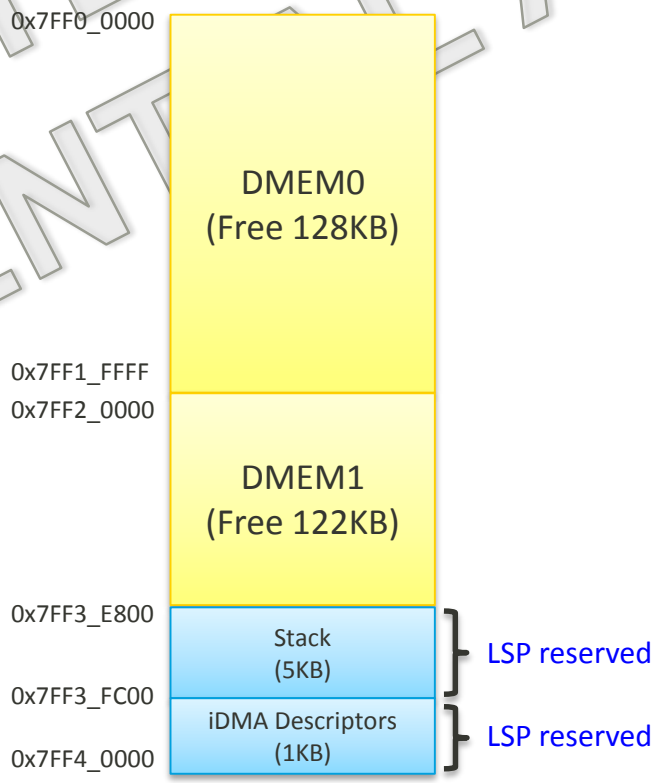
- BootVer: Build date of VPU boot code
  - Updated in each sDSP SDK release (by MTK)
- AlgoVer: Build date of VPU algorithm
  - Updated in each algorithm release (by third-party)

# LSP update for stack size modification

- Before:




- After:



# 2020 MTK iDMA driver

| MTK iDMA driver interface | Brief |
|---|---|
| init(void) | Reset control data & iDMA HW, and always return MTRUE |
| uninit(void) | Always return MTRUE |
| config(void *pSrc, void *pDst, tm_dma_direction direction) | Always return MTRUE |
| load(void *psrc, void *pdst, uint32_t srcPitchBytes, uint32_t dstPitchBytes, uint32_t numRows, uint32_t numBytesPerRow, uint32_t interruptOnCompletion)<br>The first argument of "xv_pdmaObject pdmaObj" was removed | Add a 2D descriptor to transfer data from DRAM to DMEM, return MTRUE if success, and return MFALSE if descriptor region was full |
| store(void *psrc, void *pdst, uint32_t srcPitchBytes,uint32_t dstPitchBytes, uint32_t numRows, uint32_t numBytesPerRow, uint32_t interruptOnCompletion)<br>The first argument of "xv_pdmaObject pdmaObj" was removed | Add a 2D descriptor to transfer data from DMEM to DRAM, return MTRUE if success, and return MFALSE if descriptor region was full |
| copy(void *psrc, void *pdst, uint32_t srcPitchBytes, uint32_t dstPitchBytes, uint32_t numRows, uint32_t numBytesPerRow, uint32_t interruptOnCompletion) | Copy data from psrc to pdst by VPU, and always return MTRUE |
| start(void)<br>The arguments of "MUINT32 sDesc" and "MUINT32 eDesc" were removed | Enable iDMA, it supports consecutive two "start()"s, and always return MTRUE |
| stop(void) | Always return MTRUE |
| stall(void) | Always return MTRUE |
| waitDone(void) | Return MTRUE if iDMA DONE, return MFALSE if iDMA ERROR, and reset iDMA HW |
| isDone(void) | Always return MTRUE |
| align_check(xv_pdmaObject pdmaObj, MUINT8 *psrc,MUINT8 *pdst, MUINT32 srcPitchBytes, MUINT32 dstPitchBytes, MUINT32 numRows, MUINT32 numBytesPerRow, tm_dma_direction direction) | Legacy unused API, removed |
| get_alignment(_IN_ xv_pdmaObject pdmaObj, _IN_ MUINT32 numBytesPerRow, _OUT_ MUINT32 *rowByteAlign, _OUT_ MUINT32 *sysMemAlign, _OUT_ MUINT32 *SMemAlign) | Legacy unused API, removed |

# Drystone benchmark

- Easy to add new sources without modifying the CMakefiles
  1. Copy dhrystone source to “$sDSP_sdk/library/algo1/” directory
  2. Call the entry point of new module in main_proc.c
  3. Rebuild the ELFs



```
|-- library
|   |-- algo1
|   |   |-- d2d_flo.c
|   |   |-- dhrystone
|   |   |-- main_proc.c
|   |   |-- mtk_vp6.cpp
|   |   `-- mtk_vp6.h
|   `-- inc
|       `-- ikernel.h
```

main_proc.c

```
case 0x16d:
    ret = dhrystone();
    if (ret == 0) {
        // Fill the result value of struct vpu_prop_t
        if (VPUReadReg(VPU_XTENSA_INFO17_REG) != 0)
            VPUWriteReg(VPU_XTENSA_INFO18_REG, 100);
        else
            pVPU_prop_t->result = 100;
    }
    break;
```

# SOPs to build sDSP device code

Make:

1. Change to $sDSP_sdk root directory
2. Update “MTK_PLATFORM” & Xtensa tools info in CMakeLists.txt
3. mkdir build
4. cd build
5. cmake ../
6. make –j8 → One “make” for all ELFs

Clean:

6. cd ..
7. rm -rf build

```
Scanning dependencies of target vpu0_main_sDSP
Scanning dependencies of target vpu1_main_sDSP
[ 10%] [ 20%] [ 40%] [ 40%] [ 50%] [ 80%] [ 80%] [ 80%] Building C object CMakeFiles/vpu0_main_sDSP.dir/library/algo1/dhrystone/dhry21b.c.o
Building C object CMakeFiles/vpu0_main_sDSP.dir/library/algo1/dhrystone/dhry21a.c.o
Building C object CMakeFiles/vpu0_main_sDSP.dir/library/algo1/d2d_flo.c.o
Building C object CMakeFiles/vpu0_main_sDSP.dir/library/algo1/main_proc.c.o
Building C object CMakeFiles/vpu1_main_sDSP.dir/library/algo1/dhrystone/dhry21b.c.o
Building C object CMakeFiles/vpu1_main_sDSP.dir/library/algo1/d2d_flo.c.o
Building CXX object CMakeFiles/vpu0_main_sDSP.dir/library/algo1/mtk_vp6.cpp.o
Building C object CMakeFiles/vpu1_main_sDSP.dir/library/algo1/dhrystone/dhry21a.c.o
[ 90%] Building C object CMakeFiles/vpu1_main_sDSP.dir/library/algo1/main_proc.c.o
[100%] Building CXX object CMakeFiles/vpu1_main_sDSP.dir/library/algo1/mtk_vp6.cpp.o
Linking CXX executable ../elfs/mt6885/vpu0_main_sDSP
[100%] Built target vpu0_main_sDSP
Linking CXX executable ../elfs/mt6885/vpu1_main_sDSP
[100%] Built target vpu1_main_sDSP
```

# Assertion support

- MTK assert function:
  - assert_mtk_print(int expression)
    - Raise exception & backtrace and print file name & line number
  - assert_mtk(int expression)
    - Raise exception and backtrace
- Example:
  - If expression evaluates to FALSE, MTK assertion raises exception & backtrace and aborts VPU execution

```
#include "assert_mtk.h"
...
int test(int val) {
  assert_mtk_print(val > 0);
  return 0;
}
```


*everyday genius*