Application Note

Memory Allocation for TIDL Usage

Adam Hua, Sotirios 'Chris' Tsongas

ABSTRACT

TIDL is TI's AI inference framework that runs on TDA4x series and AM6xA series processors, utilizing the built-in AI accelerator C7xMMA for efficient AI model inference. TI processors adopt a shared memory architecture, where all cores, including A-cores running high-level operating systems (Linux, QNX) and the C7xMMA for AI inference, share the same physical memory. The memory space pre-allocated to C7xMMA is inaccessible to other cores, which may result in insufficient runtime memory for the main system when memory space is limited. This paper describes how to pre-allocate C7xMMA memory based on the actual model runtime conditions to minimize its memory footprint.

Table of Contents

Trademarks

1 Introduction

TI's SoC uses a shared memory approach, meaning the NPU shares physical memory with other cores. In the default system, a relatively large amount of memory is reserved for the NPU based on the SoC's resources. The reserved memory cannot be used by other cores, which in memory-constrained situations may cause other processes to fail due to the inability to obtain required memory, while the NPU may not fully use the pre-allocated memory space, resulting in a waste of precious resources. Therefore, we need to allocate memory space for C7xMMA according to the actual runtime conditions of the model to maximize resource utilization.

2 C7xMMA DDR Usage Analysis and Optimization

When running models on the board, setting the debug level can output the memory requirements during model execution. Use /opt/tidl_test/TI_DEVICE_armv8_test_dl_algo_host_rt.out and set debugTraceLevel to 2, or use edgeai-tidl-tools and set debug_level to 2, or when using TIOVX applications, set traceLogLevel to 2 in TIDL_CreateParams. Before inferring the model, run /opt/vision_apps/vision_apps_init.sh to enable C7xMMA logging. At this point, inferring the model will produce logs as shown in the figure below. These logs illustrate the storage space allocated for each part of TIDL.

```
[C7x_2 ]     52.423052 s: 0           , DDR Non-cacheable   , Persistent  ,  128, 19.42   , 0x10000000
[C7x_2 ]     52.423071 s:
[C7x_2 ]     52.423106 s: 1           , DDR Cacheable       , Persistent  ,  128, 0.66    , 0x1043de40
[C7x_2 ]     52.423123 s:
[C7x_2 ]     52.423158 s: 2           , L1D                 , Scratch     ,  128, 16.00   , 0x7f83c000
[C7x_2 ]     52.423176 s:
[C7x_2 ]     52.423210 s: 3           , L2                  , Scratch     ,  128, 224.00  , 0x7f800000
[C7x_2 ]     52.423228 s:
[C7x_2 ]     52.423263 s: 4           , L3/MSMC             , Scratch     ,  128, 2048.00 , 0x7e200000
[C7x_2 ]     52.423281 s:
[C7x_2 ]     52.423316 s: 5           , DDR Cacheable       , Persistent  ,  128, 2755.41 , 0x1043e180
[C7x_2 ]     52.423333 s:
[C7x_2 ]     52.423368 s: 6           , DDR Non-cacheable   , Scratch     ,  128, 4.00    , 0x10300000
[C7x_2 ]     52.423386 s:
 Freeing memory for user provided Net
[C7x_2 ]     52.423421 s: 7           , DDR Non-cacheable   , Persistent  ,  128, 148.25  , 0x10005000
[C7x_2 ]     52.423439 s:
[C7x_2 ]     52.423473 s: 8           , DDR Non-cacheable   , Scratch     ,  128, 0.13    , 0x10301000
[C7x_2 ]     52.423492 s:
[C7x_2 ]     52.423526 s: 9           , DDR Non-cacheable   , Scratch     ,  128, 3.13    , 0x10301400
[C7x_2 ]     52.423544 s:
[C7x_2 ]     52.423578 s: 10          , DDR Cacheable       , Persistent  ,  128, 530.19  , 0x106eefc0
[C7x_2 ]     52.423596 s:
[C7x_2 ]     52.423631 s: 11          , DDR Cacheable       , Scratch     ,  128, 4096.25 , 0x10e00000
[C7x_2 ]     52.423649 s:
[C7x_2 ]     52.423684 s: 12          , DDR Non-cacheable   , Persistent  ,  128, 2048.00 , 0x1002a400
[C7x_2 ]     52.423701 s:
[C7x_2 ]     52.423736 s: 13          , DDR Cacheable       , Persistent  ,  128, 1390.06 , 0x10773900
[C7x_2 ]     52.423754 s:
[C7x_2 ]     52.423788 s: 14          , DDR Non-cacheable   , Persistent  ,  128, 0.00    , 0x1022a400
[C7x_2 ]     52.423806 s:
[C7x_2 ]     52.423840 s: 15          , DDR Cacheable       , Persistent  ,  128, 4381.13 , 0x108cf1c0
[C7x_2 ]     52.423857 s:
```

Figure 2-1. Model Runtime Memory Space Allocation

The table shows the memory allocation records, i.e., the printing of memRec information. There are 16 memory records in MemRec, recording the storage space occupied by various processes during model operation. The following briefly describes the purpose of each description:

Record	Description	Cacheable	Attribute
0	TIDL Instance	No	Persistent
1	Space occupied by TIDL_CreateParams	Yes	Persistent
2	L1D SRAM	-	Scratch
3	L2 SRAM	-	Scratch
4	L3 SRAM	-	Scratch
5	Stores fixed inputs for each layer, such as weights	Yes	Persistent
6	Intermediate layer output results (for model conversion)	No	Scratch
7	Intermediate layer output results (model inference)	No	Persistent
8	Temporarily used space for each layer, for accumulating output values, and taking the maximum values.	No	Scratch
9	Stores statistics and some recording information	No	Scratch
10	Sum of each layer's sTIDL_AlgLayer_t, recording basic information of each layer	Yes	Persistent
11	Space used by each layer's algorithm application	Yes	Scratch

Record	Description	Cacheable	Attribute
12	Stores context when handling preemption	No	Persistent
13	Network structure	Yes	Persistent
14	Used for multi-core synchronization in multi-core mode	No	Persistent
15	Stores some constants, such as convolution parameters	Yes	Persistent

To improve CPU operating efficiency, DDR should be set as cacheable as much as possible. For memory that needs to be accessed by other cores, it must be set as non-cacheable to avoid cache coherency issues in multi-core systems. Persistent indicates that the data in the memory block remains unchanged for a long time, while Scratch indicates that the data changes frequently.

When configuring the memory map with reference to the documentation, it can be noted that the following four sizes need to be configured for each C7 heap usage:

- C7x_x_ddr_local_heap_size corresponds to cacheable Persistent memory blocks
- C7x_x_ddr_scratch_size corresponds to cacheable Scratch memory blocks
- C7x_x_ddr_local_heap_non_cacheable_size corresponds to non-cacheable Persistent memory blocks
- C7x_x_ddr_scratch_non_cacheable_size corresponds to non-cacheable Scratch memory blocks

We only need to classify and sum the memory blocks in Figure 1 according to whether they are cacheable and Persistent or Scratch to obtain the respective sizes that need to be configured for the above four blocks. In chips with multiple C7xMMAs, each C7x needs to configure its own four memory blocks according to this method.

Readers may also notice that in gen_linker_mem_map.py, in addition to the C7x heap, there are some memory blocks named c7x_x_ddr_size. These memory blocks are responsible for storing IPC communication, MCU startup firmware, basic runtime required stack, etc., and can be reduced as appropriate based on actual usage.

Note that when configuring the memory map, please strictly follow the requirements for memory alignment and minimum memory in the documentation.

3 Testing Based on J722S and MobileNet

This article was tested on J722S EVM using SDK version 11.01. While other SDK versions may work, we recommend using SDK 11.01 and running on a J722S EVM.

3.1 Firmware Compilation and Environment Configuration

Configure the environment and perform the first compilation according to the SDK User Guide. Configure the environment required by TIDL and perform the first compilation according to the TIDL Compilation Guide.

3.2 Model Export and On-Board Inference

After completing 3.1, the default test model MobileNet will be installed in the SDK. Run in ti-processor-sdk-rtos-j722s-evm-11_01_00_04/c7x-mma-tidl/ti_dl/utils/tidlModelImport:

./out/tidl_model_import.out ../../test/testvecs/config/import/public/caffe/tidl_import_mobilenet_v1.txt

After the import completes, the model artifacts must be copied to the target board. Copy the exported model to the board, configure the inference parameters in infer.txt. The infer.txt file should contain the following configuration:

```
inFileFormat = 1
numFrames = 1
postProcType = 1
postProcDataId = 0
inData = input.bin
outData = "output.bin"
netBinFile = tidl_net_mobilenet_v1.bin
ioConfigFile = tidl_io_mobilenet_v1_1.bin
writeTraceLevel = 0
debugTraceLevel = 2
coreNum = 2
```

Please note to set debugTraceLevel = 2. The paths in this infer.txt file assume the model artifacts (tidl_net_mobilenet_v1.bin and tidl_io_mobilenet_v1_1.bin), and input.bin are in the same directory as the TI_DEVICE_armv8_test_dl_algo_host_rt.out executable.

Then run:

/opt/tidl_test/TI_DEVICE_armv8_test_dl_algo_host_rt.out s:infer.txt

Run /opt/vision_apps/vision_apps_init.sh to get the result in Figure 1.

3.3 Memory Statistics

Perform memory statistics according to the values in Figure 1, and obtain the following table:

Memory Type	Size
Cacheable Persistent	9058.16 KB
Cacheable Scratch	4096.25 KB
Noncacheable Persistent	2215.67 KB
Noncacheable Scratch	7.26 KB

3.4 Modifying the Memory Map

Modify gen_linker_mem_map.py as follows:

```
diff --git a/platform/j722s/rtos/gen_linker_mem_map.py b/platform/j722s/rtos/gen_linker_mem_map.py
index 18a008d..2cf290b 100755
--- a/platform/j722s/rtos/gen_linker_mem_map.py
+++ b/platform/j722s/rtos/gen_linker_mem_map.py
@@ -236,18 +236,18 @@ c7x_1_ddr_scratch_size         = 64*MB;
 # C7x_2 Non-Cache sections
 c7x_2_ddr_local_heap_non_cacheable_addr       = c7x_1_ddr_scratch_addr + c7x_1_ddr_scratch_size;
 c7x_2_ddr_local_heap_non_cacheable_addr_phys  = c7x_1_ddr_scratch_addr_phys + c7x_1_ddr_scratch_size;
-c7x_2_ddr_local_heap_non_cacheable_size       = 64*MB;
+c7x_2_ddr_local_heap_non_cacheable_size       = 3*MB;
 c7x_2_ddr_scratch_non_cacheable_addr          = c7x_2_ddr_local_heap_non_cacheable_addr + c7x_2_ddr_local_heap_non_cacheable_size;
 c7x_2_ddr_scratch_non_cacheable_addr_phys     = c7x_2_ddr_local_heap_non_cacheable_addr_phys + c7x_2_ddr_local_heap_non_cacheable_size;
-c7x_2_ddr_scratch_non_cacheable_size          = 64*MB;
+c7x_2_ddr_scratch_non_cacheable_size          = 1*MB;

 # C7x_2 Cache sections
 c7x_2_ddr_local_heap_addr      = c7x_2_ddr_scratch_non_cacheable_addr + c7x_2_ddr_scratch_non_cacheable_size;
 c7x_2_ddr_local_heap_addr_phys = c7x_2_ddr_scratch_non_cacheable_addr_phys + c7x_2_ddr_scratch_non_cacheable_size;
-c7x_2_ddr_local_heap_size      = 64*MB;
+c7x_2_ddr_local_heap_size      = 10*MB;
 c7x_2_ddr_scratch_addr         = c7x_2_ddr_local_heap_addr + c7x_2_ddr_local_heap_size;
 c7x_2_ddr_scratch_addr_phys    = c7x_2_ddr_local_heap_addr_phys + c7x_2_ddr_local_heap_size;
-c7x_2_ddr_scratch_size         = 64*MB;
+c7x_2_ddr_scratch_size         = 5*MB;

 total_c7x_1_local_ddr   = c7x_1_ddr_local_heap_non_cacheable_size + c7x_1_ddr_local_heap_size;
 total_c7x_1_scratch_ddr = c7x_1_ddr_scratch_non_cacheable_size + c7x_1_ddr_scratch_size
```

Figure 3-1. Memory Map Modification

The modification is based on the statistical results in section 3.3, considering that the model only runs on the second C7xMMA. Therefore, only the heap segment memory corresponding to C7x_2 is modified. Among them, Cacheable Persistent occupies 9058.16 KB. In the configuration, the SDK uses 1000 instead of 1024 for KB to MB conversion. Additionally, considering the 1MB memory alignment requirement, c7x_2_ddr_local_heap_size is configured to 10MB. Cacheable Scratch occupies 4096.25 KB, and considering the 1MB alignment requirement, c7x_2_ddr_scratch_size is configured to 5MB. Noncacheable Persistent occupies 2215.67 KB, and c7x_2_ddr_local_heap_non_cacheable_size is configured to 3MB. Noncacheable Scratch occupies 7.26 KB, and c7x_2_ddr_scratch_non_cacheable_size is configured to 1MB.

Sysconfig modification is as follows:

```
diff --git a/platform/j722s/rtos/c7x_2/example.syscfg b/platform/j722s/rtos/c7x_2/example.syscfg
index de7cb27..23cdd1f 100644
--- a/platform/j722s/rtos/c7x_2/example.syscfg
+++ b/platform/j722s/rtos/c7x_2/example.syscfg
@@ -234,7 +234,7 @@ mmu_armv814.attribute          = "MAIR7";
 mmu_armv815.$name              = "DDR_C7X_2_HEAP_SCRATCH_NON_CACHEABLE";
 mmu_armv815.vAddr              = 0x110000000;
 mmu_armv815.pAddr              = 0x890000000;
-mmu_armv815.size               = 0x8000000;
+mmu_armv815.size               = 0x400000;
 mmu_armv815.attribute          = "MAIR4";

 /**
@@ -245,7 +245,7 @@ mmu_armv815.attribute          = "MAIR4";
  * Attribute is MAIR7.
  */
 mmu_armv816.$name              = "DDR_C7X_2_HEAP_SCRATCH_CACHEABLE";
-mmu_armv816.vAddr              = 0x118000000;
-mmu_armv816.pAddr              = 0x898000000;
-mmu_armv816.size               = 0x8000000;
+mmu_armv816.vAddr              = 0x110400000;
+mmu_armv816.pAddr              = 0x890400000;
+mmu_armv816.size               = 0xF00000;
 mmu_armv816.attribute          = "MAIR7";
```

Figure 3-2. Sysconfig Memory Configuration Modification

For instructions on implementing the memory map modification, please refer to the documentation.

3.5 Recompiling SDK and Updating to Board

Perform RTOS SDK compilation and update according to the SDK User Guide. Compile U-Boot and update tiboot3.bin according to the Linux SDK User Guide.

3.6 On-Board Testing

Copy the files tidl_net_mobilenet_v1.bin, tidl_io_mobilenet_v1_1.bin, infer.txt, and input.bin to /opt/tidl_test on the device.

Run on board:

```
cd /opt/tidl_test/
./TI_DEVICE_armv8_test_dl_algo_host_rt.out s:infer.txt
```

Obtain test results:

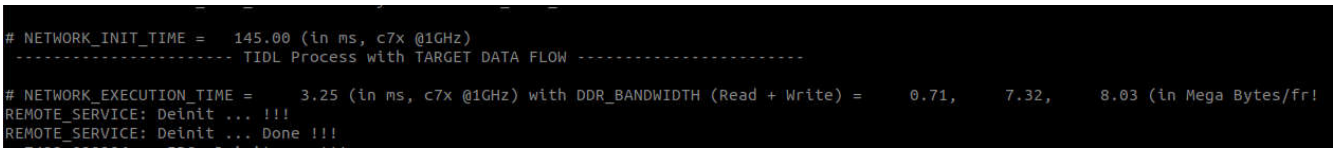

Figure 3-3. On-Board Inference Results

4 Summary

This article discusses the actual memory usage of AI models during inference, describes the purposes of each memory block, and provides guidance on how to allocate memory space reasonably based on actual usage to conserve DDR space resources. This article also provided a test case, using a model that comes with the SDK for testing, successfully reducing the memory occupied by a single C7x from 256MB to 19MB, which can effectively reduce memory usage and lower the cost of the entire board.

5 References

1. https://www.ti.com/product/AM62A7
2. https://www.ti.com/tool/PROCESSOR-SDK-AM62A
3. https://www.ti.com/product/AM67A
4. https://www.ti.com/product/TDA4VM
5. https://www.ti.com/product/TDA4VE-Q1
6. https://www.ti.com/product/TDA4VM-Q1
7. https://www.ti.com/product/TDA4AL-Q1
8. https://www.ti.com/product/TDA4VL-Q1
9. https://www.ti.com/product/TDA4VP-Q1
10. https://www.ti.com/product/TDA4VH-Q1
11. https://www.ti.com/product/TDA4VEN-Q1

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you fully indemnify TI and its representatives against any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale, TI’s General Quality Guidelines, or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.

Last updated 10/2025