Vivante Driver Release Notes

For GCCORE/VIPCORE

For

Software Release Version

6.3.2.5

Jan2019

Legal Notes

This document contains proprietary information of Vivante Corporation. Vivante Corporation reserves the right to make changes to any products herein at any time without notice. Vivante Corporation does not assume any responsibility or liability arising out of the application or use of any product described herein, except as expressly agreed to in writing by Vivante Corporation; nor does the purchase or use of a product from Vivante Corporation convey a license under any patent rights, copyrights, trademark rights, or any other of the intellectual property rights of Vivante Corporation or third parties.

DISCLOSURE/RE-DISTRIBUTION LIMITATIONS

The information contained in this Vivante Confidential document may be adapted for re-distribution by Vivante customer SoC vendors who have licensed the related core IP, with the understanding that re-distribution is limited to those customers of the SoC vendor who have active and valid NDA or licenses applicable for the SoC which contains the related Vivante core IP. Otherwise, the information contained herein is not to be used by or disclosed to the third parties without the express written permission of an officer of Vivante Corporation.
(Vivante Distribution LevelC CONFIDENTIAL – DISTRIBUTION RESTRICTED).

Upon request Vivante provides this document in Word format for adaptive inclusion in documentation intended for your customers. Vivante requests a pre-release copy of the adaptation for review and approval. In the adapted document please include a reference to the copyright owner, for example: “This chapter contains copyright confidential and proprietary material disclosed with permission of Vivante Corporation. Vivante has authorized re-distribution of this material restricted to those NDA partners and licensees of *[the SoC vendor]* who are engineering *[SoC vendor’s product]* which includes the discussed Vivante IP.”

TRADEMARK ACKNOWLEDGMENT

Vivante Corporation and the Vivante Corporation logo design are trademarks or registered trademarks of Vivante Corporation. All other brand and product names may be trademarks of their respective companies.

Contents

General Notes

Version: 6.3.2.5

Release Date:Jan, 2019

Release Format: Compressed tar file .tgz

Package Filenames:

Vivante 6.3.2.5 software release packages will have names similar to the following examples:

Beginning	Type	Version	extension
VIVANTE_GAL_Unified_Src_	**drv** or **tst**	**6.3.2.5**	**.tgz**
VIVANTE_GALVIP_Unified_Src_	**drv** or **tst**	**6.3.2.5**	**.tgz**

Compatibility:

With compatible Vivante GCCORE/VIPCORE IP, Vivante software release packagesinclude support for the following operating systemsand standard APIs.

Operating Systems:(some OSs require add on packages)

- Android 9.0 (Pie), 8.x (Oreo) / Android 7.x (Nougat) / Android 6.0 (Marshmallow) / 5.1 (Lollipop)
- Chrome OS version 47 and later
- Linux / Linux Embedded
- Yocto X11 Desktop
- Wayland Desktop
- QNX
- Windows Embedded Compact 7

APIs:(some APIs require compatible hardware and add on packages)

- Vulkan 1.0
- OpenGL ES3.2 / OpenGL ES3.1 / OpenGL ES3.0 / OpenGL ES2.0
- OpenGL ES1.1
- OpenVX 1.1
- OpenVX NN Extension API 0.4
- Android NNAPI 1.1
- OpenVG 1.1
- OpenCL 1.2Full Profile

- OpenGL 3.1
- DirectFB1.7.x
- GDI/DirectDraw
- Vivante2D API
- VivanteVG API
- VivanteVX API

Hardware Limitations:

Not all Vivante GPU IPs can support advanced graphics/compute features such as tessellation shaders and geometry shaders.The advanced graphics features in OpenGL ES 3.2 API and Android Extension Pack (AEP) require VivanteGCCORE XS option support. The OpenVX API requires Vivante GCCORE VX option support, and is also supported by OpenCL backend. Vivante VIP cores support OpenVX and Neural Networking, while CCCOREs are compatible with OpenCL but do not include support for OpenGL ES. Please check with Vivante for detailed information on GCCORE/VIPCORE IP capabilities.

Conventions:

Issues are numbered sequentially within each release and subsection. Additional numbers may precede the issue descriptions.

- Numbers with a CL prefix reference a changelist number in Vivante's internal Software Configuration Management system.
- Numbers without an alphabetic prefix reference an issue number in Vivante's internal Issue Tracking system.

Build Options:

- If the Vivante release package includes a build script for a specific OS system, the build option values specified in the build script should not be changed.
- The build options for a specific OS system are determined during system integration. Changing the build options to values that are not validated may cause undefined behavior.

Key Features of the Vivante 6.3.x Driver Series

Support for Vivante OpenVX Neural Network Extension version 0.4 in 6.3.1

Vivante's OpenVX Neural Network Extension version 0.4 is a superset of the Khronos Neural Network Extension version 1.2. It has some fused APIs to enable VIP's built-in Neural Network engine in a more efficient way. For detailed information about this extension, please check with Vivante.

Support for Android NN API 1.1

The Android Neural Networks API (NNAPI) is an Android C API designed for running computationally intensive operations for machine learning on mobile devices. NNAPI is designed to provide a base layer of functionality for higher-level machine learning frameworks (such as TensorFlow Lite, Caffe2, or others) that build and train neural networks. The API is available on all devices running Android 8.1 (API level 27) or higher.

Backward Compatible with earlier 6.x, 5.x and 4.x Drivers

The 6.2.x drivers are backward-compatible with earlier drivers. Existing OpenGL ES 3.x/2.0 applications can run on these drivers without any changes. Application performance on 6.3.x driver is better than or equal to that with previous drivers.

Key Features of the Vivante6.2.x/6.1.x Driver Series

Full support for KhronosVulkan 1.0 API and WSI extension APIs

Vulkan is a new generation graphics and compute API that provides high-efficiency, cross-platform access to modern GPUs used in a wide variety of devices.Vivante GC7000* GPU cores have been certified by Khronosas Vulkan 1.0 compliant products.

Full support for Khronos OpenGL ES 3.2 API

The new OpenGL ES 3.2 and OpenGL ES Shading Language 3.20 specifications bring AEP, plus additional functionality, into core OpenGL ES. Vivante GC7000XS* GPU core has been certified by Khronosas an OpenGL ES 3.2 compliant product.

Better support for Khronos OpenCL 1.2Full-Profile API

Vivante'sOpenCL 1.2 implementationis much improvedin terms of functionality, reliability, and performance in the 6.2.x release. It can support a wide spectrum of applications including scientific computing, vision processing, and neural network training and inferencing.

Full support for KhronosOpenVX 1.1 API

Starting with the Vivante 6.2.5 driver, it can fully supportthe KhronosOpenVX 1.1 API which is used in areas such as face, body, and gesture tracking, smart video surveillance, augmented reality, visual inspection, and more.

Backward Compatible with 6.2.x, 6.0.x, 5.1.x and 4.x Drivers

The 6.2.x and the 6.1.x driversare backward-compatible with 6.0.x, 5.x.x and 4.x drivers. Existing OpenGL ES 3.x/2.0 applications can run on these drivers without any changes. Application performance on 6.2.x driver is better than or equal to that with previous drivers.

Key Features of the Vivante 6.0.x Driver Series

Full support for Khronos OpenGL ES 3.1 + AEP features

The 6.0.x driver fully supports Khronos OpenGL ES 3.1 + AEP (Android Extension Pack) features. It also supports KhronosOpenVX 1.0.1 API for vision processing and Khronos OpenCL 1.2 API for GPU computing.

The new features of the 6.0.x driver include:

- **Android Extension Pack (AEP)**
  - Tessellation shaders
  - Geometry shaders
  - Advanced blend equations
  - Copying subregions between image objects
  - Supporting blending on a per-draw-buffer basis
  - Miscellaneous new shader functionality
  - ASTC texture compression (LDR profile only)
  - Primitive bounding boxes
  - Shader image atomic operations
  - Shader interface blocks
  - Shadermultisample interpolation control
  - Sample shading control
  - Sample variables
  - Texture buffer objects
  - Texture border color
  - Texture cube map arrays
  - STENCIL8 texture formats
  - Texture multisample 2D arrays
  - EXT_texture_sRGB_decode
  - Debug messages
- **Draw calls specifying a base vertex parameter**
- **Floating-point framebuffers**
- **Robust buffer access control**
- **Support for querying CONTEXT_FLAGS, as needed by debug and robustbuffer access functionality.**
- **KhronosOpenVX 1.0.1 API**
- **Khronos OpenCL 1.2 API**

Backward-Compatible with 5.1.x and 4.x Driver

The 6.0.x driver is backward-compatible with5.x.x and 4.x drivers. Existing OpenGL ES 3.0/2.0 applications can run on 6.0.x driver without any changes. Application performance on the 6.0.x driver is better than or equal to that with the 5.x.x and 4.x driver.

Key Features of the Vivante 5.1.x Driver Series

Full support for Khronos OpenGL ES 3.1 API

The 5.1.x driver fully supports the Khronos OpenGL ES 3.1 API and passes the OpenGL ES 3.1 Conformance Test. The key features of the OpenGL ES 3.1 API include:

- **Compute shaders** – applications can use the GPU to perform general computing tasks, tightly coupled with graphics rendering. Compute shaders are written in the GLSL ES shading language, and can share data with the graphics pipeline;
- **Separate shader objects** – applications can program the vertex and fragment shader stages of the GPU independently, and can mix and match vertex and fragment programs without an explicit linking step;
- **Indirect draw commands** - the GPU can be instructed to take draw command data from GPU mapped memory rather than passing that data through drivers. For example, this allows a compute shader running on the GPU to perform a physics simulation and then generate the draw command information needed to display the results, without CPU synchronization;
- **Enhanced texturing functionality** – including multi-sample textures, stencil textures, and texture gather;
- **Shading language improvements** – new arithmetic and bitfield operations, and features to enable modern styles of shader programming;
- **Optional extensions** – per-sample shading, advanced blending modes, and more;
- **Backward compatibility with OpenGL ES 2.0 and 3.0** – programmers can add ES 3.1 functionality incrementally to working ES 2.0 and 3.0 applications.

Backward-Compatible with 5.0.x and 4.x Drivers

The 5.1.x driver is backward-compatible with the 4.x driver. Existing OpenGL ES 2.0 applications can run on the 5.1.x driver without any changes. Application performance on the 5.1.x driver is better than or equal to that with the 4.x driver.

Key Features of the Vivante 5.0.x Driver Series

Full support for Khronos OpenGL ES 3.0 API

The 5.0.x driver fully supports Khronos OpenGL ES 3.0 APIand passes the OpenGL ES 3.0 Conformance Test. The major functionalities in OpenGL ES 3.0 API include:

- **Rendering pipeline enhancements**to enable acceleration of advanced visual effects including: occlusion queries, transform feedback, instanced rendering and support for four or more rendering targets.
- **High quality ETC2/EAC texture compression**whicheliminates the need for a different set of textures for each platform.

- **Shading language enhancements** which include full support for integer and 32-bit floating point operations.
- **Enhanced texturing functionality** including guaranteed support for floating point, 3D, depth, vertex, NPOT, R/RG, immutable and 2D array textures, as well as for swizzles, LOD and mip level clamps, seamless cube maps and sampler objects.
- **Extensive set of required, explicitly sized texture and render-buffer formats,** which reduces implementation variability and makes it much easier to write portable applications.

Compatible with 4.x Driver for OpenGL ES 2.0 API

The 5.0.x driver is backward compatible with the 4.x driver. Existing OpenGL ES 2.0 applications can run on the 5.0.x driver without any changes. Applications performance on the 5.0.x driver is better than or equal to that with 4.x driver.

Enhanced Capabilities

- **An improved GL object management framework** is implemented, so the following GL objects can be better shared between contexts: texture, buffer, program, shader, renderbuffer, sync, and sampler objects. [Note that per the GLES spec, the following objects cannot be shared between contexts: framebuffer, query, transform feedback, and vertex array objects.]
- **A streamlined context switch framework** provides improved multi-thread / multi-context support.
- **The GPU state dumping mechanism** can dump GPU internal states, DMA buffers, driver call stack, etc. when a non-deterministic failure occurs in the system. Dumped information can be used to pin-point quickly afailure'sroot cause.
- **Run-time GPU profiling** can be enabled on a production release driver by setting an environment variable. When enabled, the built-in GPU profiling mechanism can generate a file of GPU profile data.
- **API logging capabilities**allow two levels of EGL/GLES API logging function to be enabled on a production release driver by setting an environment variable.
  - Level One API logging provides dumps with the EGL/GLES API call signature (API name with call parameters) to the console or a file for application behavior analysis and debugging.
  - Level Two API logging functions can record EGL/GLES API calls and data into a trace file which can be played back with vPlayer on different platforms or on a GPU software emulator (Cmodel) for debugging.

Changes in version 6.3.2.5

January2019: gal_6_3.2.5

Improvements:

1. Update android NN HAL to 1.1.
2. Fix issue of FULL_CONNECTED and RESHAPE operation in android nn.
3. Fix NN HAL crashed issue at the 2nd RESHAPE operation in android nn.
4. Driver support for V8 hw, depthwise.
5. Graph binary 1.1 multiple input buffer support, memory usage optimization.
6. Graph transform for MXN convolution, concat, tensor Add.
7. Optimize shader implementation for shader layer.
8. Refine code of tensor create.
9. Improve LSTM layer implementation.
10. Ovx1.2 support.
11. Add error message for public API.
12. Set correct interrupt trigger type to high_level to match hardware.
13. Set default coherent_dma_mask to 40bit for galcore device.
14. Update linux fence code to match corresponding kernel version
15. Improve several minor bugs in RA/CPP/CPF/DEC/SCPP/LOOP of backend compiler.
16. Fix a memory overrun issue for macro expansion in frontend compiler.

Changes in version 6.3.2.4

November2018: gal_6_3.2.4

Improvements:

1. Refine code to support hybrid data type in NN operations.
2. Fixed bugs in Android NN API operations.
3. Fixed multi-vip memory free issue.
4. Improved LSTM layer implementation.
5. Refine resource clean up logic for Android NN Hal.
6. Fixed shared tensor memory free issue.
7. Improve graph binary for multi-vip
8. Graph transform for tensor add and various bug fixes.
9. Compiler fixed for MOD and ARCTRIG functions.
10. Graph binary v1.1 format support.
11. Driver support v8 and huffman encoding.
12. Memory profiling support.
13. OVX1.2 support
14. kernel MMU programming change to support heterogenous 3D type hardware.

Changes in version 6.3.2.3

November2018: gal_6_3.2.3

Improvements:

1. More graph transform implementation fix.
2. vSimulator bug fixes.

3. Compiler fixes for atan2 to conform to OCL spec.
4. Graph binary bug fixes.
5. NN API OP implementation improvements.
6. Refine Android NN HAL driver structure.
7. Add uint8 dynamic fix point support.

Changes in version 6.3.2.2

November2018: gal_6_3.2.2

Improvements:

1. Fixed compiler OOB check.
2. More NN API 1.1 improvement.
3. SWTiling fix for phase3 hardware.

Changes in version 6.3.2.1

November2018: gal_6_3.2.1

Improvements:

1. Improve Android NN API 1.0 implementation.
2. Implement Android NN API 1.1 new OPs.
3. Fixed bugs in SWTiling path.
4. Added feature code for new VIP hardwares.
5. Fixed multi-vip issues, multi-vip vData support enhancement.
6. Added more LSTM API code framework.
7. Added OCL patch framework.
8. Fixed memory leak in vSimulator path.
9. Added more multi-VIP affinity mode control.
10. Improve arch model.
11. Improve graph transform implementation.
12. Driver memory footprint improve.
13. Fixed various compiler bugs.

Changes in version 6.3.2

November2018: gal_6_3.2

Improvements:

15. [NN] Disable aSYNC copy 256byte merge if hardware has the bug unfixed.
16. [kernel] Use dma_mapping_error() to check if dma_map_page does well.
17. [NN] Release WeightsBiasParament and temp tensor on time for LSTM Unit.
18. [NN] Remove useless code in _createWeightsBiasesParameterFromTensors
19. [NN] Only check NN core count for convolution layer
20. [NN] Save binary, refine code to add some check point
21. [Compiler] reset the storage qualifier for a function return variable.
22. [NN] Refine multi-TP output too small issue for CL179123.

23. [Compiler]use scratch mem pool for temp storages so it can be reused after it freed
24. [Vsimulator] set it to NULL after we destroyed globalFenceID for bug#21662
25. [Compiler] use std::map to contain the ops, instead of std::unordered_map that is the c11 feature.
26. [NN] Correct real kernel ptr for XYDP6/XYDP9 in calc zero run length code
27. [NN] adjust the dims of embeddiing_lookup ops to whcn and change the sw implement
28. [Compiler] Fix buddy memory system 64bit failure, need to access member of buddy node sysmetical to avoid 64bit size problem, and also need to allocate size no less than the size of VSC_BUDDY_MEM_BLOCK_NODE
29. [NN] Save binary, fix a tensor used by multiple nodes as input/output issue & refine code
30. [NN] Sync the implement of Op embedding lookup in hal driver with nnapi
31. [NN] Added huffman compression code.
32. [Compiler] Add saving/loading of shaderMode for shader inputs and outputs
33. [NN] enable 1xN feature if saving graph binary
34. [NN] Add a resize nearest neighbor shader operation
35. [NN] update shader path for RPNLayer
36. [NN] Add two resize bilinear shader operation to support u8 to fp16 and fp16 to u8
37. [NN] Support fp16 for embedding lut shader
38. [NN] Resize bilinear add 5 shader implementation for 2x u8 to u8/fp16, 4x u8/fp16 to u8/fp16
39. [NN] deconv kernel 4x4 stride 2 Uint8 shader implement
40. [NN] Fix tensor scale sw implementation issue, we must use the api of "vxoTensor_GetTensorViewMemory" to get tensor logical address.
41. [NN] Enabled new path of LSTM layer.
42. [NN] Add tensor pad tp operation implementation
43. [NN] Add tensor pad shader path to support pad mode "VX_PAD_REPLICATE"
44. [NN[Add Android NN 1.1 APIs(mean, squeeze, stride slice) support to head file.
45. [NN] TensorAdd add shader implementation to support Z-BroadCast
46. [NN] TensorMul add shader implementation to support Z-BroadCast
47. [NN] Add depthwise deconvolution nn/tp support.
48. [NN] Added forget_bias for LSTMUnit.
49. [NN] Add a tensor copy tp operation for reshape layer
50. [Compiler] Refined address space qualifier checking in parameter declaration.
51. [NN] Add the ops SUB & DIV of nnapi 1.1 and implement
52. [NN] Add framework of Android NN 1.1 new APIs.
53. [NN] Add the implement of op TRANSPOSE for nnapi 1.1
54. [VX] Refinments for vxImageCreateFromHandle();
55. [NN] MultiVIP NN/TP job splitting support.
56. [kernel] Refine paged and non-paged memory cacheable.
57. [NN] Adjust a bit virtual memory allocation algorithm to minimize whole memory.
58. [NN] Refine wrapping user VXC node to use shaderExecutable.
59. [NN] Upgrade arch model: archPerf: v0.122, swtiling: v0.100-, CTS version: CL182682

Changes in version 6.3.1.9

September2018: gal_6_3.1.9

Improvements:

1. Split tensorEltwise shader op to tensorAdd/Mul/Div to speed up compiler compilation
2. Make sure all other objects was destroyed before we release kernel->context object(bug#21629).
3. Vsimulator: add support for multiple graphs under a context

4. Sync LSTM & CONCAT in hal driver to nn api
5. Revert the code that set the attribute of bias tensor to high precision
6. Add a default kernel size [1x1xN] to optmize lstmlayer code
7. Vsimulator: detroy vx engine before we destroy default hardawre to fix more memory leak issue
8. Fix vxc binary build issue
9. Graph binary: refine code for patching nn/tp instruction pyhsical addrees
10. Enhancement TP_UPSAMPLE and TP_UPSAMPLE_CLIP for multi-tp(deconvolution).
11. Fix a typo for tensorDiv
12. Sync the implementation of OPs in hal driver to nnapi
13. Merge sw-tiling phase3 changes.
14. Graph binary: refine save nn/tp operation input/output/ks hard code
15. Update the algorithm that convert float32 to float16
16. Fixed bug #21650, add workaround for parameter changed if the image object is created from host ptr when it can't goes into wrapped path.
17. Added the code that convert rank and data format to hal driver
18. Fixed the memory leak in nnapi
19. Remove redundant embedding lut layer
20. Graph binary: Fix regression cause by CL177691. modify lut data type from vx_array to vx_tensor
21. Fixed the memory leak in node adapter
22. Fixed the memory leak when to convert batch FC to NN conv
23. Fixed more memory leak issue
24. Move part of CNNPerf info in front of gcfVX_Flush to track hang issue if any.
25. Force to use 8-byte alignment for structure 'vx_arch_perf_s'
26. Fixed some redefinition issue found by Fullhan tool chain
27. Revered the some code about releasing shared tensor for memory leak
28. Fixed the bug that identify the node type when to merge node
29. Vsimulator: enable NN conv1x1 to conv1xN optimization for vdata
30. Added NN_XYDP0 feature bit.
31. Added maxpool shader implementation to support kernel=1, stride=2
32. Compiler: Properly set the address space qualifiers for pointer variables and
33. Compiler: setting the type find to VIR_TY_POINTER accordingly.
34. Compiler: Missing replacing with _SetVariableQualifiers()
35. Compiler: Get uniform kind for pointer kernel arguments via type qualifiers.
36. Updated arch model changes from 5x(archPerf: v0.106, swtiling: 0.91.1-)
37. Vsimualtor: correct vsimulator target names for VIPNANOD/O/Q/S_PLUS_V8_0
38. Added nnvxc_binary project.
39. Converted rank and dataformat for ops in hal driver
40. Filter out non-tensor reference in opeartion inputs/outputs to avoid warning and comment out a bit TP reorder code.
41. Turn off multi-TP if output size is too small to avoid hang.
42. Save all members in archPerfHandle structure one by one for vdata.
43. Changed the condition that set the numParamters of Node from node->kernel->isUserkernel to node->kernel->enabled
44. Re-implemneted the function that tranform convoltion node to convlutionrelupooling2 node
45. Removed useless assert: hw support hybrid mode for quant8 format
46. Trimmed trailing for the file of gc_vx_layer.c.
47. Added a shader implementation for projection bias as float32 for lstmunit layer.
48. AB buffer does not support vip sram input yet.
49. Added a paramter for lstmunit state output sw implementation
50. Vcompiler: removed unused varaibles in saving kernel binary. (for bug#21574 - vcCompiler output size issue)
51. Fixed one memory leak issues. lstmLayer->lstmHiddenUnitNode->lstm_nn_operation.weights_biases was not released.

52. When to create tensor, set the param according to the data type of tensor
53. Handled multi cluster feature affecting the base address calculation of pointer to local address space variables.
54. Added the feature that check the chip feature when to merge conv & fc
55. Renamed the NNAPI lib from libNeuralNetworks to libneuralnetworks, which is synced to tensorflow
56. Corrected the input/output size of convolution/FC for LSTM unit.
57. Upgraded arch perf to v0.112
58. Set bias tensor to high precision in LSTM
59. Sync hal driver to nnapi CL179288:when to create tensor, set the param according to the data type of tensor
60. Checked whether allocate buffer for tensor when to check the attribute of tensors, which caused BUG 21724
61. Added limitation for softmax shader op
62. Adjust the logic to check the chip feature, which fix the bug 21714
63. Get Convolution size, pad and upsample pad with independent policy for Deconvolution.
64. Fixed NN pool stride info for profile calcuation under run mode.
65. Vsimulator: cannot read shader register value if the current running kernel is an TP or NN(bug#21716)
66. Vsimulator: added new vsimulator targets: VIPPICO_V1_PID0X87/VIPPICO_V2_PID0X93/VIPPICO_V3_PID0X99
67. Added tpliteCoreCount/NNFP16XYDPX/NNFP16XYDPY/NNF16ZDP chipspec.
68. Added a status checking for batchnorm shader op.
69. Added a shader implementation for lstmunit state output to support projection and fp16 data type
70. Fixed build issue also update chip revision for pid0x93 & pid0x99
71. Synced 179466 to nnapi and hal driver: check uint8 and int32 when to create tensor
72. Supported float32 bias for rnn
73. Supported float16 for hash LUT
74. Added ZRL_bits in fixed features.
75. Enabled relu for RNN NN implement, when rnn have relu parament.
76. Refined CL179340 to pass arch model CTS check
77. For lstm unit shader op, fix tanh overflow issue
78. Updated version number of arch model algorithm
79. Converted the dims to 4d in FC op of nnapi.
80. Fixed the bug when to convert the 3 dims.
81. Graph binary: save shader's axi-sram as a patch item

Changes in version 6.3.1.8

September2018: gal_6_3.1.8

Improvements:

1. Fixed dma map to usespace.
2. Changed some operation intput/output parameters from vx_array to vx_tensor for father-son relationship analysis
3. Separated VXC code to *.vx files
4. Added extern "C" to keep interface api of nnapi as C- style
5. Fixed stride setting for reshaped tensor when sw-tiling enabled.
6. Added nodeID to struct node to record the number of created nodes, which is unique
7. Added embedding&hash LUT shaders
8. Fixed profile calculation for FC operation used NN.
9. Updated deconvolution NN/TP implement, add upsample and clip tp support.
10. Updated vxcBuild

11. Supported kernel4.14(kernel_write instead of vfs_write)
12. Fixed copy directory duplicates issue
13. Make sure the lifetime of weights and biases are static for decovolution API.
14. Fixed shader path for adapterlayer caused by cl177293
15. Enable I2F and F2I for deconvolution upsample and clip(such as 0x7d), and enable multi-tp.
16. Fixed issue about sub image size calculation in driver(profile issue: bug#21610
17. Vsimualtor supports TP_LRN & TP_ROI_POOLING feature(bug#21573)
18. Adjust assertion for BitExtract to match spec and cmodel.
19. Dump NN status through register TX1.
20. Vsimulator: added more new vsimulator targets
21. Correct upsample clip pad for deconvolution.
22. Fix bug 21613, update image data when unmapimagepatch for the image created from host ptr when it used internal memory.
23. Dump 256 32-bit for NN status.
24. Fix a bug about wrong output for hash-lut
25. Refine LSTMLayer code
26. Flush stdout in vxInfo() to get sequential profile log
27. Fix sw-tiling issue if last Y tiling is beyond input. Set correct input start and size in this case.
28. Add the struct tensorAttribute that record the tensor's attribute{rank/precison/liftTime/valued/dataType} to operand
29. Vsimulator supports TP_MAX_POOLING_STRIDE1 feature.
30. Release a local vx_parameter object for memory leak.
31. Fixed a memory access issue: cannot convert temp vx_parameter got by vxGetParameterByIndex() into vx_image/vx_tensor.
32. Fix bug#21574 - vcCompiler output size issue
33. Correct upsample pad for deconvolution.
34. Add the convert rank and data format of static operand tensor in CONV & FC &depthwiseConv
35. Fix the bug21640 that caused by checking wrongly the attribute of tensor

Changes in version 6.3.1.7

September2018: gal_6_3.1.7

Improvements:

1. Convert static operand in LSTM / RNN / LSH and so on
2. Spit tensorEltwise shader op to tensorAdd/Mul/Div to speed up compiler compilation
3. make sure all other objects was destroyed before we release kernel->context object(bug#21629).
4. Vsimulator: add support for multiple graphs under a context
5. Sync LSTM & CONCAT in hal driver to nn api
6. Add a default kernel size [1x1xN] to optmize lstmlayer code
7. Vsimulator: destroy vx engine before we destroy default hardawre to fix more memory leak issue
8. Save binary, refine code for patching nn/tp instruction pyhsical addrees
9. Enhancement TP_UPSAMPLE and TP_UPSAMPLE_CLIP for multi-tp(deconvolution).
10. Sync the implement of OPs in hal driver to nnapi
11. Added sw-tiling phase3.
12. Fix bug #21650, add workaround for parameter changed if the image object is created from host ptr when it can't goes into wrapped path.
13. Add the code that convert rank and data format to hal driver
14. Clear the memory leak in nnapi
15. Remove redundant embedding lut layer
16. Save binary, fix regression cause by CL177691. modify lut data type from vx_array to vx_tensor

17. Clear the memory leak in node adapter
18. Clear the memory leak when to convert batch FC to NN conv
19. Move part of CNNPerf info in front of gcfVX_Flush to track hang issue if any.

Changes in version 6.3.1.6

September2018: gal_6_3.1.6

Improvements:

1. Fixed TP adapter for CL177293.
2. Change the environment variable for merging layer with -Merge:0|1 to switch the feature
3. Move rank/precision/data_lifetime from tensor to tensor->tensorBuffer.
4. Fixed bug #21563, refine code for tensor which created from host ptr.
5. Fixed for bug 21565. merge from CL177438.
6. Fixed in peephole pass, check symbol's addres sspace of load/store base operand, if they're global space, skip converting to load_l/store_l
7. For reshuffle shader OP, using dynamic uniform to hack hw bug#1891
8. Fixed performance drop which caused by cl176530
9. Fixed maxpooling performance drop which caused by cl176530
10. Program the ZP and SCALE correctly for the input and output in case one is non-tf-quant format.
11. Fixed scale precision check in vxnneGetDepthwiseConvShaderExecutable().
12. Changed a bit virtual buffer algorithm to satisfy swtiling/AB buffer branch request.

Changes in version 6.3.1.5

September2018: gal_6_3.1.5

Improvements:

1. Add an api to query maxWorkGroupSize
2. Fix the shader Z-Axis direction memory overflow issue
3. Hold the original weight if it is not converted to 2x2
4. Remove data type VX_TYPE_SCALAR from binary as loader only cares about size and format
5. Fixed memory issues reported by klocwork scan.
6. Fix bug#21543, init local work size on depth2space layer
7. Add the option:rank/precision/lifttime/valued to vxQueryTensor()
8. Optimize Batchnorm layer shader implementation, merge input scale/outputscale/output zeropoint into weight and bias
9. Save binary, save shader's shared video memory to LCD as a shader patch item
10. Save more sub weightsbiases info for vdata when SWTIling is on.(bug#21538)
11. Set node->numParameters for user node(bug#21536)
12. Vsimulator: remove old VIPNANOS_PLUS_PID0X89 and added new vsimulator target: VIPNANOD_PID0X89
13. Save binary, save shader's shared video memory to LCD as a shader patch item
14. Check whether the operand is empty before create virtural tensor, which cause the bug 21548
15. Fix depthwise conv and conv2d NN/TP support issue for mobilenet_quant8.
16. Enable combined mode for VX.
17. Reshape the dims of tensor accroding to the dimcount info when to create intermidiate tensor
18. Add relu feature to convolutionrelupooling and the others to vxActivationLayer()
19. Set VX_QUANT_DYNAMIC_FIXED_POINT when to create fp16 tensor to fix the bug 21553
20. Tranform the convolution node to crl2 though nodeCount that need to be merged is 1

21. Rebuild the topology graph after tranforming graph at each time
22. Set the attribute VX_TENSOR_VALUE with valued option of orignal tensor when to replace the orignal tensor in vxoNode_adapter();
23. Add vx_weights_biases_parameter_optimizations_t when to create vx_weights_biases_parameter
24. Save binary, patch in/outimage circular buffer end address for NN/TP instruction as a patch item
25. Fix TP tensor copy.
26. Correct the depth of input and output for Deconvolution weights reshuffle.
27. Add pad_right and pad_bottom support for deconvolution.
28. Enhance operand dump for special HW registers
29. A quick fix for MMU table not on case. We should do direct mapping if MMU is not on. Otherwise the GPU address(physical in fact) is not consistent with CPU physical because of shift mapping. Fixed v55 cmodel crash issue.
30. Correct pading right and bottom for deconvolution sw implement.
31. sync the graph optimization code from 5.X.
32. Add reference dose not support non-tensor reference yet.
33. Add gcvFEATURE_NN_ZDP3_NO_COMPRESS_FIX check for zdp zero length hack
34. Work around shader transpose operation in adapter layer.
35. Add TP operation in adapter layer.
36. Fixed shader debug info for VXC node which has mutiple kernel shaders

Changes in version 6.3.1.4

August 2018: gal_6_3.1.4

Improvements:

1. Fixed NN CTS issues.
2. VX NNE batch code refinement.
3. Refine virtual buffer algorithm to support AB buffering branch and better memory optimization.
4. Ranmesoc_platform code to gckPLATFORM to avoid name conflict.
5. Fixed graph binary bugs.
6. Fixed vcCompiler bus.
7. Enable virtual buffer feature and subimage tiling algorithm.
8. Fixed bugs exposed by enabling vxc binary path.
9. Refine NN API driver code to do static conversion and etc.
10. Static code analysis bug fixes.
11. VX driver debug message control.
12. Vsimulator bug fixes.

Changes in version 6.3.1.3

August 2018: gal_6_3.1.3

Improvements:

1 Fixed NN CTS Avg_pool op failure.
2 Fixed graph binary generation issues.
3 Compiler bug fixes.

Changes in version 6.3.1.2

July 2018: gal_6_3.1.2

Improvements:

1. Multi-VIP independent mode/multi-device mode support.
2. Fixed NN CTS issues.
3. Fixed graph binary generation issues.
4. VX NNE batch code refinement.
5. Compiler bug fixes.
6. Kernel driver crash dump refinement.
7. Supported graph binary v1.0 (alpha)
8. VX driver bug fixes.

Changes in version 6.3.1.1

July 2018: gal_6_3.1.1

Improvements:

1. Fixed graph binary generation issues.
2. Fixed NN CTS issues.
3. Fixed some 64 bit galcore driver issues.
4. Refine graph transformation framework code.
5. Compiler bug fixes

Changes in version 6.3.1

July2018: gal_6_3.1

Improvements:

6. SupportedVivante Neural Network API 0.4.
7. Supported v7.0.x/v7.1.x VIP hardware releases.
8. Supported NN API 1.0via Android HAL module and NN Adapter (alpha).
9. Supported first version of SW tiling algorithm (AB buffering).
10. Supported graph binary v1.0 (alpha) generation
11. Enhanced vData generation for v7.0.x/v7.1.x hardwares.
12. Kernel driver enhanced to support better GPU memory management, support multiple SRAMs and shared external memory.
13. Clean kernel driver 64 bit issues and make it more robust.
14. Compiler bug fixes and library file support
15. VX driver bug fixes and enhance performance.

Changes in version 6.2.5.p2

January 2018: gal_6_2_5.p2

Improvements:

16. Fixed OpenCV failure on Goke board.
17. Added CPU implementation for maxPooling and avgPooling with different strideX and strideY, different PoolX and PoolY.
18. Fixed activation layer bug and updated shader batch normalization.
19. Refined max workgroup size calculation in OCL.

Companion VTK version is 1.7.4.

Changes in version 6.2.5.p1.2

December2017: gal_6_2_5.p1.2

Improvements:

1. Refine VX driver memory leak found in a customer stress case.
2. Fixed OCL compiler memory leak in pre-linked shader binary code path.
3. Fixed OpenCV hang on Goke board.
4. Fixed 2D GPU hang on Goke board.
5. Fixed average pool rounding mode.

Companion VTK version is 1.7.4.

Changes in version 6.2.5.p1.1

November 2017: gal_6_2_5.p1.1

Improvements:

1. Refine VX driver message print.
2. Fixed vSimulatorshader executor.
3. Refine shader binary dump.

Companion VTK version is 1.7.4.

Changes in version 6.2.5.p1

November 2017: gal_6_2_5.p1

Improvements:

1. Refined NNE API implementations.
2. Expanded NNE API shader implementation, softrelu logistic and batch normalization, etc.
3. Fix for a general compiler back-end issue.
4. vSimulatorissues resolved.
5. Fix for driver static library build.

Companion VTK version is 1.7.4.

Changes in version 6.2.5

October 2017: gal_6_2_5

Improvements:

1. Common enhancement for vxMapxxx/vxUnmapxxx performance in OpenVX.
2. Support VX kernel offline compiler(vcCompiler).
3. Support new VivanteOpenVX NN Extension API 0.2 version.
4. Support loading pre-compressed NN using vdata.
5. Enhance vSimulator/IDE for VIP products.
6. Fixe some memory leaks in OpenVX driver.

Companion VTK version is1.7.4.

Changes in version 6.2.4

October 2017: gal_6_2_4

Improvements:

1. FixedVulkan CTS 1.0.2.xtestfailures to pass KhronosVulkan CTS 1.0.2.
2. Fixed all OpenVX 1.1 CTS test failures to pass the CTS.
3. Added support for X11 DRI3/DRM framework and WL GBM interface.
4. Support Android 8.0 (Oreo) Gralloc and HWC based on DRM/KMS framework.
5. Added dmabuf export support in kernel driver for VIDMEM and unified VIDMEM/VIRTUAL.
6. Enhanced support for GPU local memory in kernel driver for PCIe based systems.
7. Added support for EGL_EXT_image_dma_buf_import_modifiers extension.
8. Refined stream/vertex input setup driver implementation to reduce CPU overhead.
9. Optimized compiler implementation to reduce compile/link time.
10. Fixed the incorrect WL/Weston compositor FPS issue.
11. Fixed a multi-threaded race condition that causes random failure in stress tests.

Companion VTK version is1.7.4.

Changes in version 6.2.3

July 2017: gal_6_2_3

Improvements:

1. Fixed multiple Vulkan CTS 1.0.2.* failures.
2. Fixed big-endian driver issues exposed on the big-endian SoC platforms.
3. Improved vProfiler support in driver to have more accurate HW counter values.
4. Fixed Linux kernel panic issue when try to put_page for reserved memory.
5. Enhanced CL kernel compiler support forOpenCV library and applications.
6. Opened Wayland server side buffer protocol in EGL driver to enable WL applications.
7. Refined flat mapping kernel memory allocation. Use asingleallocation for all STLBs.
8. Fixed a kernel driver issue to enable repeatedgalcoreinsmod/rmmodtowork with mmu.
9. Fixed QNX driver issues to pass ES CTS and CL CTS.

10. Fixed the 3DVG SDK app upside down renderingissue.

Companion VTK versionis1.7.2.

Changes in version 6.2.2.p1

May 2017: gal_6_2_2_p1

Improvements:

1. Fixed multi-threadissues that cause fsl_2d_3d stress test to fail.
2. Fixed random hang/render error after repeat insmod/rmmod of galcore.
3. Fixed data overflow issue with flatMappingStart/flatMappingEnd.
4. Fixed gear_x11 memory leak issues.
5. Added complete EGLImage 4444 format support.
6. Added Kernel 4.9 support. Use sync_file/fence for android native fence sync.
7. Fixed errors when glcolorconvert is run.
8. Corrected mininumfree memory limit to avoid system freeze when memory is exhausted.
9. Fixed Wayland window resize issue to enable weston-simple-eglfullscreen mode.

Companion VTK version is1.7.1.

Changes in version 6.2.2

March 2017: gal_6_2_2

Improvements:

1. Addedmutex protection for referencing gctSIGNAL in gckOS_MapSignal to fix a MT race issue.
2. Streamlined GPU address calculation base on MC20 (0/1) and MMU (0/1) combinations.
3. Fixed multiple Android HWC 2.0 rendering issues. Refined HWC 2.0 driver implementation.
4. Improved HWC2.0 composition performance by composing damaged regions only.
5. Added QNX makefiles to enable driver build on QNX OS.
6. Added VXC compilation function in compiler for vision hardware and software support.
7. Enabled offline/online compiler IR assembly dump function for users.
8. Passed the latest Khronos open-source GLES CTS 3.2.2 from Github.
9. Implemented the direct rendering support (no-resolve) for Wayland platform.
10. Updated Wayland-viv protocol to support tile status sync from client to server.
11. Improved OpenCL 1.2 built-in function support with native GPU instructions.
12. Enabled OpenCL 1.2 API trace dump function controlled by VIV_TRACE environment variable.
13. Fixed multiple MesaGL application rendering issues with the new OpenGL driver.
14. Cleaned up driver code issues reported by Klocwork and Coverity.

Companion VTK version is 1.7.0.

VTK 1.7.0 Feature Enhancements and Bug Fixes

1. AddedvkCompiler to support offline Vulkan/SPIRV shader compilation.
2. Enabled shader IR assembly output from vCompiler, vcCompiler and vkCompiler.
3. Updated vProfiler/vAnalyzer to include some new counters.

4. Fixed vPlayer playback issues on 64-bit Windows platform. Add support for playing back 32-bit/64-bit trace files on different platforms.

Changes in version 6.2.1

January 2017: gal_6_2_1

Improvements:

1. Fixed all failures in Android N CTS dEQP tests.
2. Added native driver support for Android N HWC 2.0 interface.
3. Fixed Minecraft rendering issues on Android N.
4. Fixed failures inWebGL conformance test 1.0.2 on Chrome OS.
5. Improved GL4 driver quality to resolve all rendering issues in MesaGL samples.
6. AddedsRGB format support for Android Vulkan WSI and pass Android CTS VK tests.
7. Improved compiler optimizations to generate more efficient GPU instructions.
8. AddedVXCcompilation function in compiler for vision hardware and software support.
9. Fixed multiple bugs in OpenCL 1.2 driver.

Companion VTK version is 1.6.9.

Changes in version 6.2.0.p2

November 2016: gal_6_2_0_p2

Improvements:

1. Resolved Khronos GLES3.x CTS failures in all EGLConfig runs.
2. Fixed a YUV420 buffer allocation issue that caused 4K video OOM on Android N.
3. Fixed several GLES driver memory leak problems that cause random failures in Android CTS.
4. Corrected EGL buffer age calculation to resolve a GooglePlayUI flickering issue.
5. Passed Android N CTS Vulkan WSI extension check by disabling duplicated WSI extensions.
6. Fixed a MMU exception by limiting the number of sampler prefetch in GPU.
7. Enabled LunarGVulkan Loader support in the Vulkan driver.
8. Fixed driver code issues reported by static code analysis tools.
9. Add support for GL_LUMINANCE8_ALPHA8_EXT texture in the directTexVIV extension.
10. Fixed Xserver crash issues on Yocto X11 desktop.
11. Clamp 3Dblit rectangle size to surface area to fix a GPU hang in Wayland tests.
12. Set HW type correctly in Wayland server side to fix a memory leak in 2D composition.

Companion VTK version is 1.6.8.

Changes in version 6.2.0

September 2016: gal_6_2_0

New features:

1. Unified driver that includes OpenGL ES, Vulkan, and OpenVX, etc.
2. Android 7.0 (Nougat) support. Pass Android N CTS.

3. Brand new OpenGL 2.1 driver implementation which is based on ES3 driver code.
4. Improved vProfiler support to include many new HW counters.
5. Vulkan ICD, SDK layers support.
6. Compiler optimizations that improve benchmark/application performance.

Companion VTK version is 1.6.8.

Changes in version 6.1.1

July 2016: gal_6_1_1

New features:

1. Wayland GBM backend support.
2. Multi-GPU combined mode and individual mode support.
3. Multi-GPU affinity control interface.
4. Linux DMA-BUF buffer sharing to support Chrome OS.