BAM – Block-based Acceleration Manager

Algorithm Framework

Version 02.08.00.00

Release Notes

Aug 2017

Build ID: 02.08.00.00

New In This Release

- Interface changes: added parameters blockWidthMax and blockHeightMax to BAM_CreateGraphParams::blockDimParams to limit the dimensions of the processing block when the option optimizeBlockDim is set to true.

Validation information

Release Configuration	Description	Validation Platform
ALGFRAMEWORK	BAM algorithm framework on DSP and EVE	TDA2x EVM TDA3x EVM

This release was validated on the following:

- TDA2x EVM with:
    - Code Composer Studio version 5.1.0.09000
    - Code Generation Tool version 7.4.2
    - EVE/ARP32 Code Generation Tool version 1.0.7
    - XDAIS version 7.24.00.04
    - EDMA3 LLD 02.12.00.20
    - DMAUTILS version 00.08.00.01

Fixed in this release

Defect ID	Description	Applicable Release Configuration	Defect found in BAM release	Defect fixed in BAM release
ADASALGOS-86	In some case, BAM_createGraph() returns the error BAM_E_SMART_MEM_ALLOCATOR=-269 due to a bug in the memory allocator.	BAM	02.07.00.00	02.08.00.00

Build ID: 02.07.00.00

New In This Release

- The DSP version has interface changes to the structure intMemParams, passed to BAM_createGraph().
- New licensing terms 'TI Text File license' apply to this release, please refer to files' header.

Validation information

Release Configuration	Description	Validation Platform
ALGFRAMEWORK	BAM algorithm framework on DSP and EVE	TDA2x EVM TDA3x EVM

This release was validated on the following:

- TDA2x EVM with:
    - Code Composer Studio version 5.1.0.09000
    - Code Generation Tool version 7.4.2
    - EVE/ARP32 Code Generation Tool version 1.0.7
    - XDAIS version 7.24.00.04
    - EDMA3 LLD 02.12.00.20
    - DMAUTILS version 00.08.00.01

Fixed in this release

Defect ID	Description	Applicable Release Configuration	Defect found in BAM release	Defect fixed in BAM release
ADASALGOS-68	When compile for DSP, BAM ignores the aliasBufOffset parameter in structure intMemParams	BAM	02.06.00.00	02.06.01.00

Build ID: 02.06.00.00

New In This Release

- This version is delivered as a standalone package instead of being included inside the EVE software release.
- This version adds support for DSP in addition to EVE.
- Updated the user's guide with instructions on how to build the libraries for either EVE or DSP.

Fixed in this release

Defect ID	Description	Applicable Release Configuration	Defect found in BAM release	Defect fixed in BAM release
EVESW-319	After a bam applet is executed, regions in VCOP memory will be overwrittenby data. Meta-data belonging to VCOP heap could be overwritten, causing subsequent call to vcop_malloc() to return invalid addresses. There is no issue if only BAM applets are used since they do not use the VCOP heap. But non-BAM applets use VCOP heap and if a non-BAM applet is called after a BAM applet, then erratic behaviour can happen. The fix is to have BAM call vcop_minit() after applet execution.	BAM	02.05.03.00	02.06.00.00
EVESW-342	Kernels that runs on ARP32, do not see their setMemRecFunc() being executed for the second context.	BAM	02.05.03.00	02.06.00.00

Build ID: 02.05.03.00

New In This Release

- N/A

Fixed in this release

Defect ID	Description	Applicable Release Configuration	Defect found in EVE SW release	Defect fixed in EVE SW release
EVESW-309	If within a graph, a node needs a scratch data-block of maximum allowable size of 16 KB either in image buffer L or H, BAM allocates it successfully. Once the scratch block is not needed anymore, BAM tries to free it. However a bug in the function bam_mem_free() in file bam_memory.c prevents the free from happening correctly and the data-block remains as if it was still being allocated, preventing any subsequent attempt to allocate a memory block in IBUFLA from happening correctly.	EVESW	01.12.00.00	01.13.00.00
EVESW-307	Allocating a memory record of 16KB may fail in host mode if the alignment is set to 32 bytes. The root cause is that, in host mode, the image buffers are defined as arrays ibuflamem, ibufhamem in file bam_memory.c and if these arrays are not already 32 bytes aligned then the allocation of a memory block of 16KB and of alignment 32 bytes will fail. The solution is to force the alignment of these arrays by using the compiler directive __declspec(align(#bytes)), which is for Visual C++.	EVESW	01.12.00.00	01.13.00.00
EVESW-45	Deactivated allocation of a memory record in DMEM because requesting a memory record in DMEM doesn't quiet work as at creation time since the returned pointer resides in DDR. Since any graph's DMEM heap is part of the graph object and at creation time, a graph object resides in DDR, any kernel's setMemRec() function gets a pointer in DDR, not in DMEM. This causes issue if this pointer is stored in the kernel's context structure to be used later at runtime. Our simple fix here is to prevent allocation in DMEM altogether. Since none of our TI kernel's BAM helper requests any memory records needing memory in DMEM, it should not be an issue. If a customer's kernel does request memory in DMEM, then BAM_createGraph() will return the error code BAM_E_INVALID_MEM_SPACE. This error code is new and has been added to the existing list of error codes defined in file bam_types.h .	EVESW	01.12.00.00	01.13.00.00

Build ID: 02.05.02.00

New In This Release

- Improved memory allocator used by BAM_createGraph() when : no more hole of 32 bytes are inserted between memory blocks when useSmartMemAlloc=1 .

Fixed in this release

- Fixed in user's guide documentation for BAM_CreateGraphParams::useSmarMemAlloc and BAM_CreateGraphParams::memRec.

Build ID: 02.05.01.00

New In This Release

- Code is MISRA-C compliant.

Build ID: 02.05.00.00

New In This Release

- Improved BAM_createGraph() error handling: If the kernel specifies more output and internal memory records than allowed by BAM then error BAM_E_NUM_MEM_RECORDS is returned. Maximum number of output memory records is defined by symbol BAM_MAX_NODE_INPUT_OUTPUT (=10) in bam_common.h and maximum number of internal memory records is defined by symbol BAM_MAX_NODE_INTERNAL (=8) in bam_construct_int.h .

Fixed in this release

Defect ID	Description	Applicable Release Configuration	Defect found in release	Defect fixed in release
OMAPS00312100	Bug fix: smart memory allocator misbehaves with output memory records that are BAM_MEMATTRS_CONST or BAM_MEMATTRS_PERSISTENT: if graph has any kernel that have output memory records that have attribute BAM_MEMATTRS_CONST or BAM_MEMATTRS_PERSISTENT then the smart memory allocator will be put in a bad state after the first BAM_createGraph() call. Subsequent calls to BAM_createGraph() might return an incorrect memory partitioning leading to block incorrectly	EVESW	01.06.01.01	1.07.00.00

	overlapping with each other.			
OMAPS00312106	Bug fix in automatic block dimensions calculation. blockWidthStep=0 or blockHeightStep=0 was causing a hang.	EVESW	01.06.01.01	1.07.00.00
OMAPS00312913	Bug fix smart memory allocator: Smart memory allocated doesn't free up all the memory records upon exiting, in the case it was unable to allocate an internal memory record.	EVESW	01.06.01.01	1.07.00.00

Build ID: 02.04.00.00

New In This Release

- Improved BAM_initKernelDB() so that it checks whether the database lists all the kernels according to their kernel ID values, which must be in increasing order. If there the list is not ordered or has any gap, then error BAM_E_DATABASE_KERNELID is returned. This type of error is simple to correct but could consume a lot of debugging time when undetected. That's why BAM_initKernelDB() has been improved.

Fixed in this release

Defect ID	Description	Applicable Release Configuration	Defect found in release	Defect Fixed in release
OMAPS00309963	BAM smart mem allocator misbehaves for output mem rec that are BAM_MEMATTRS_CONST or BAM_MEMATTRS_PERSISTENT. Output rec which have BAM_MEMATTRS_CONST or BAM_MEMATTRS_PERSISTENT attributes are still getting freed to enable reuse, as if they were BAM_MEMATTRS_SCRATCH.	EVESW	01.05.00.00	1.06.00.00

Build ID: 02.03.00.00

New In This Release

- Implemented custom memory allocator which replaces vcop_malloc() and also works in host mode.
- Added BAM_contolNode() API. See user's guide for migration of old code.
- Updated every applet to use the new BAM_controlNode() API to update EDMA parameters.

Fixed in this release

Defect ID	Description	Applicable Release Configurati	Defect found in release	Defect Fixed in release

		on		
OMAPS00308008	BAM_createGraph() fails when setting graphCreateParams.useSmartMemAlloc= false and passing a predefined memRec array	EVESW	01.04.01.00	1.05.00.00

Build ID: 02.02.00.00

New In This Release

- Added function BAM_printMemRec(): Print to the console all details (node name, type, attribute, base pointer) about each memory record in the table pointed by memRec. This is useful for inspecting the memory allocation produced by the smart memory allocator and decide to do any manual fine-tuning by manually assigning some kernel's memory records to specific memory space, in the kernel's getMemRec function.Fixed bug in the automatic block dimension's calculation: no upper bound was set for the block width and height so they could grow larger than the frame's width and height.
- Changed interface of BAM_activateGraph() and BAM_deactivateGraph(). Simplified the interface and also they no longer restore/save the graph object itself. They only take care of the internal memory records that are persistent.
- Added support for "open" output ports: All output ports of a compute node must have a connection to a downstream node, otherwise processing is incorrect. But sometimes the application desires to ignore a particular output port, characterized as "open" output port. To support this scenario, the BAM_NULL_NODE id can be used to connect any open output port.

Fixed in this release

Defect ID	Description	Applicable Release Configuration	Defect found in release	Defect Fixed in release
OMAPS00305404	Bug fixes in smart memory allocator, fixed the following issues: - Smart memory allocator does not release internal memory records that are scratched from one node to another. The symptom is that there is limited reuse of memory from one node to the next one, thus the automatic block dimension estimator cannot run to its full potential. - Smart memory allocator can sometimes hang in vcop_free() In addition allowed the user to override the space of the internal memory record that is scratch to any location.	EVESW	01.03.02.00	NA
OMAPS00305062	Bug fix for ARP32 nodes: If an APR32 context structure has some private variables issues will happen. As these values will be incorrect during processing. This patch fixes the issue by having	EVESW	01.03.02.00	NA

	the BAM_execute() function maintain the coherence of the two copies of the context.			
OMAPS00305404	Bug fix in smart memory allocator: Observed that when BAM_createGraph is called with the exact same parameters, the location of the internal memory records allocated by the smart memory allocator, changes. But they should remain the same as nothing changes from one call to another. This memory leak eventually causes the graph to be incorrect and cause a crash when executed. Root cause is that the allocator was not freeing all the internal memory records before returning to the caller, resulting in "orphan" memory buffers. This, combined with the usage of vcop_minit() were causing vcop_malloc() to return 0x00000020 instead of NULL when no more memory was available, fooling the memory allocator that the pointer was valid, as being not NULL. Fix is to free everyrthing before returning from the function.	EVESW	01.03.02.00	NA
OMAPS00304922	Overriding of memspace for internal memory record to WBUF if attribute is BAM_MEMATTRS_SCRATCH, was ineffective.	EVESW	01.03.02.00	NA

Build ID: 02.01.02.00

New In This Release

- Fixed bug in the automatic block dimension's calculation: if the application creating the graph was passing an initial block height not multiple of the frame height, BAM was not able to correct the condition.
- Fixed bug in the automatic block dimension's calculation: no upper bound was set for the block width and height so they could grow larger than the frame's width and height.

Build ID: 02.01.01.00

New In This Release

- Fixed bug in `BAM_contextRestoreMemRec()`

Instead of restoring the context, the function was saving the context.

- Fixed bug in `BAM_execute()`

If the graph objects was restored into a location different than the original location where it was created then the graph execution would fail, as each node's kernel context pointer would be incorrectly calculated. The fix is to keep track of the last location where the graph object was restored to and relocate each node's kernel context pointer accordingly inside BAM_execute().

- Added new applet *apps/imagePyramid_u8*

Please use this applet as development template for future applet based on BAM as it shows a more streamlined code flow.

- Updated BAM user's guide p.3-22. and p.6-40 with some important information on a previous undocumented limitation of how an edge's list must be formatted.

Build ID: 02.01.00.00

New In This Release

- Added support for ARP32 node.

Compute kernels that are implemented with `BAM_KernelInfo::coreType= BAM_EVE_ARP32` can now be associated to a compute node in any graph.
BAM is able to schedule ARP32 only kernels along with VCOP kernels. These ARP32 kernels can read/write into image buffers and hence take the output of VCOP node, process it and pass it to a next VCOP or ARP32 node. BAM takes care of the buffer switching and create a copy of the context structure that contains pointers to image buffer B since ARP32 does not have any ALIAS view capability. BAM will toggle between the two contexts depending on which image buffer is passed to the ARP32 node. The only limitation is that an ARP32 node which accesses image buffers cannot be ran concurrently with EDMA and VCOP. BAM will automatically wait for vcop completion and EDMA completion before running such a node. It is recommeneded to place such ARP32 node accessing image buffer to the end of the graph as a post processing step. On the other hand if the ARP32 doesn't access any image buffer then BAM is able to schedule such a node concurrently.
Note that the stack size had to be increased in the linker.cmd file from 0x1000h to 0x1200h due to higher memory requirement of BAM_execute() to support ARP32 node.

Validation Information

Release Configuration	Description	Validation Platform
BAM	BAM framework and example algorithms using it: ISP, resize, array_op, array_scalar_op, filter	Vision28 Super (Vayu)

- This release is validated using following components
  - Code Composer Studio (CCS) version 5.4.0.00091
  - ARP32 Code Generation Tools version 1.0.2
  - C6000 Code Generation Tools version 7.4.2

Fixed in this release

Defect ID	Description	Applicable Release Configuration	Defect found in release	Defect Fixed in release
1	BAM_contextRestoreMemRec() failing	All	2.01	2.01.01
2	BAM_execute() failing if graph object restored to a different location than where it was created.	All	2.01	2.01.01

Known Issues

Defect ID	Description	Applicable Release Configuration	Defect found in release	Defect Fixed in release