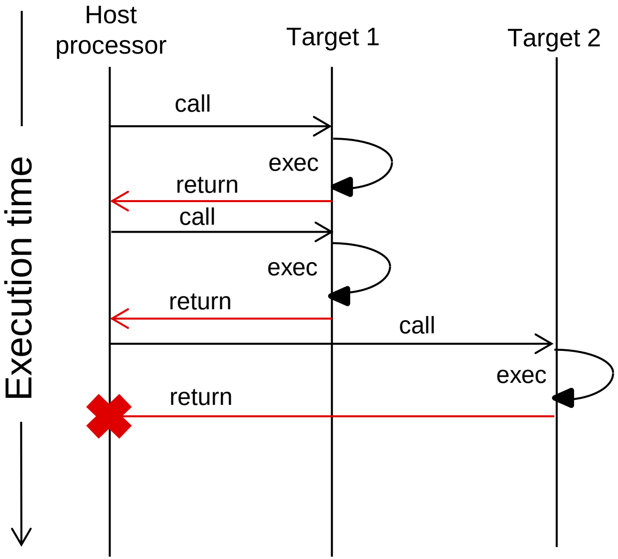
TI OpenVX™ (TIOVX): Framework optimizations

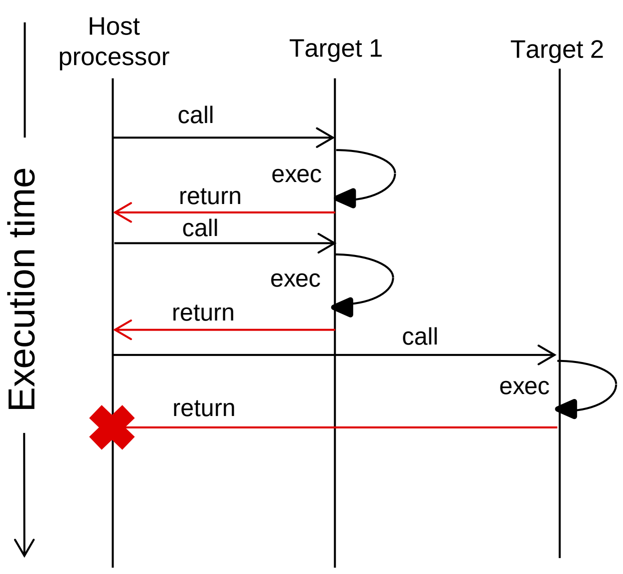
IPC aggregation



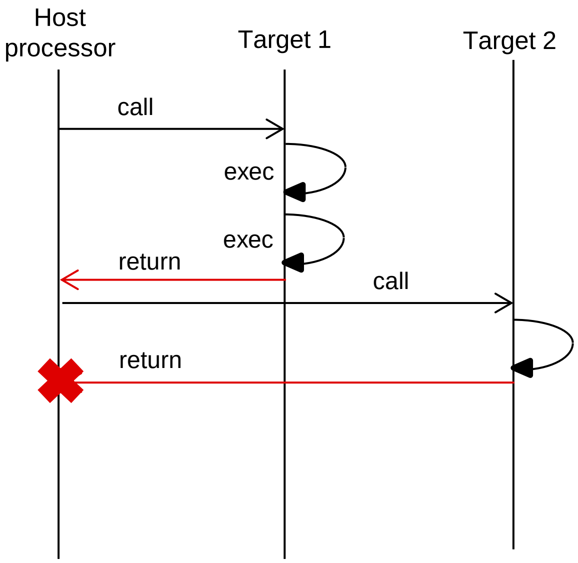
No aggregation

Highest overhead

IPC aggregation

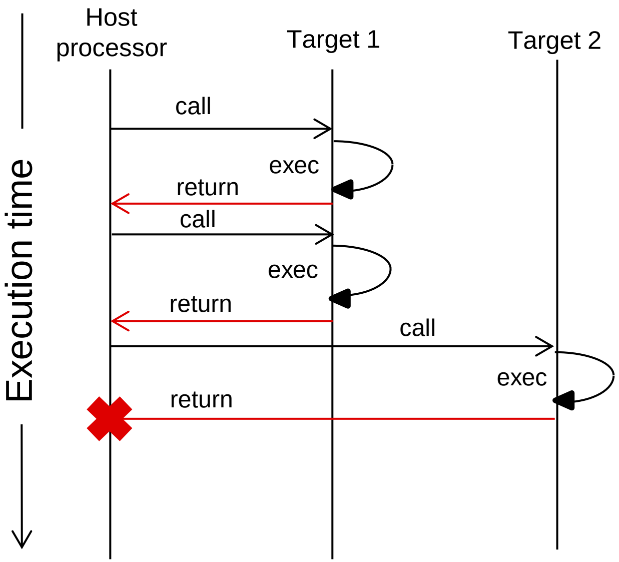


No aggregation
Highest overhead

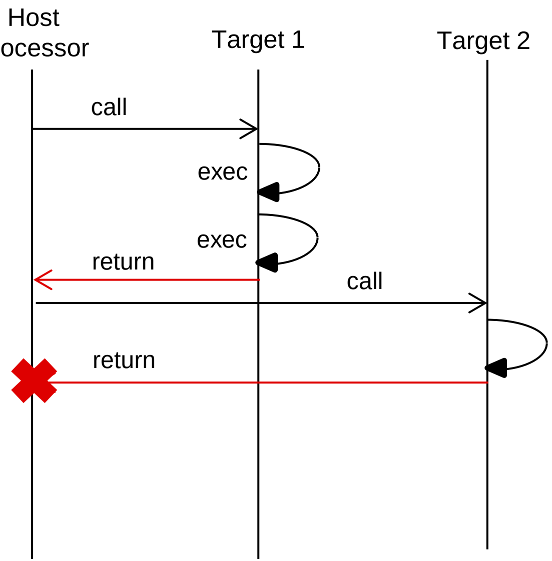


Aggregation
(star topology)
Medium overhead

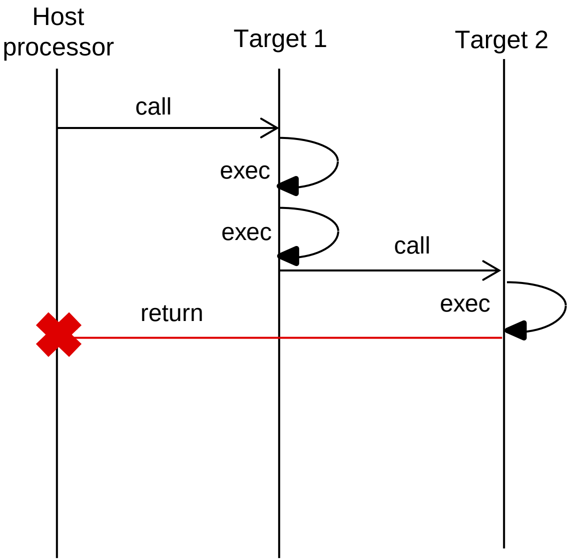
IPC aggregation



No aggregation
Highest overhead

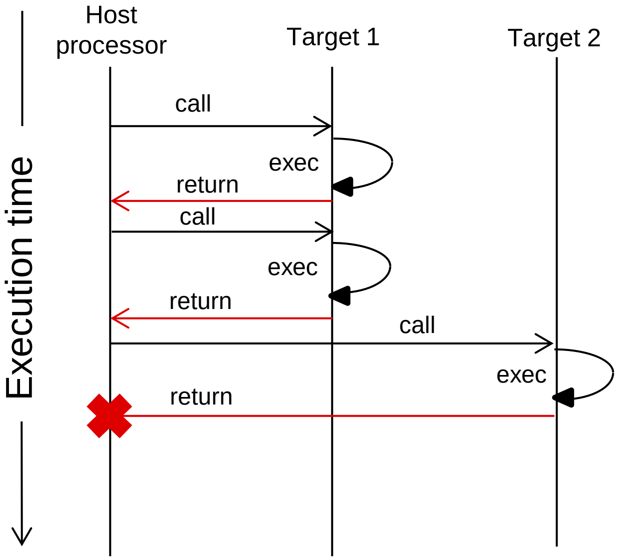


Aggregation
(star topology)
Medium overhead

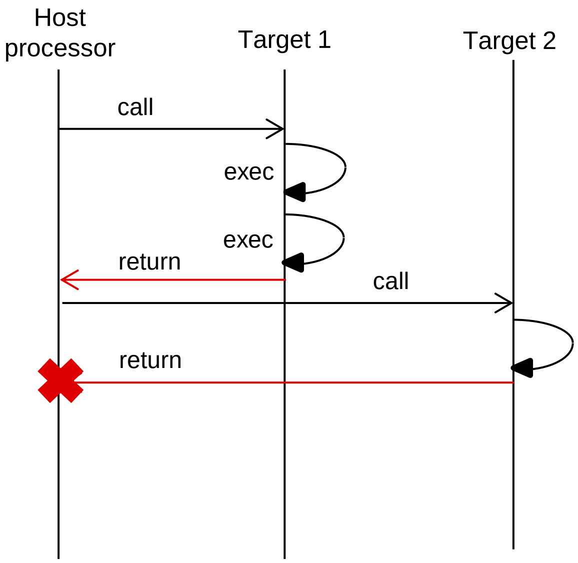


Aggregation
(mesh topology/
Peer to peer)
Least overhead

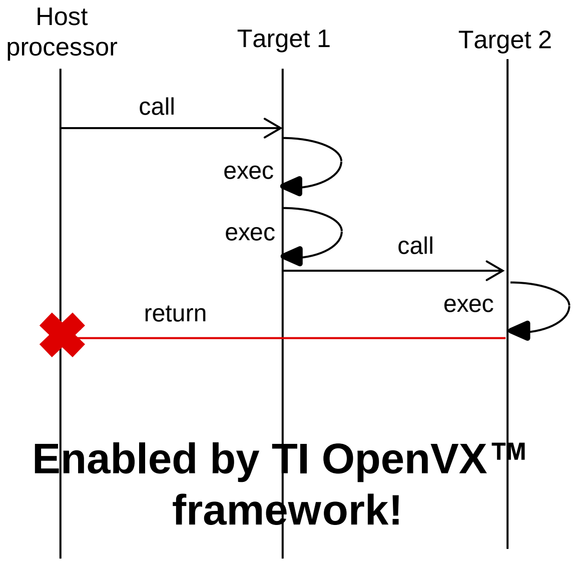
IPC aggregation



No aggregation
Highest overhead



Aggregation
(star topology)
Medium overhead

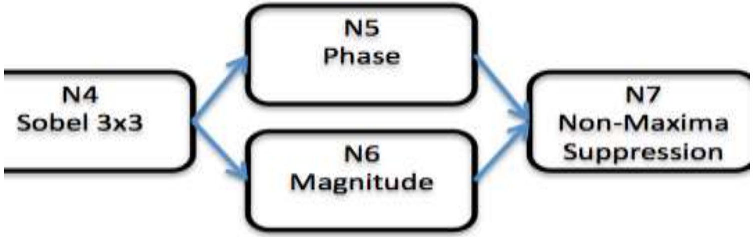


Enabled by TI OpenVX™ framework!

Aggregation
(mesh topology/
Peer to peer)
Least overhead

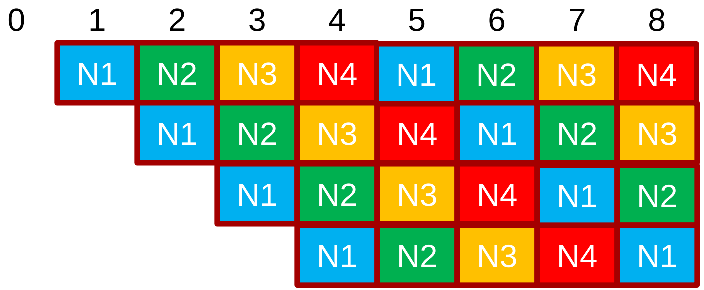
Parallelism

- Enabled by graph model
- Independent nodes can potentially be executed in parallel (using different execution units/cores)



Graph Pipelining

- Pipelining multiple invocations of a graph across targets:
  - As images are fed to the system, the graph execution of these input images can be staged across targets in a pipeline fashion.
  - Enables full performance entitlement on SoC’s with multiple cores



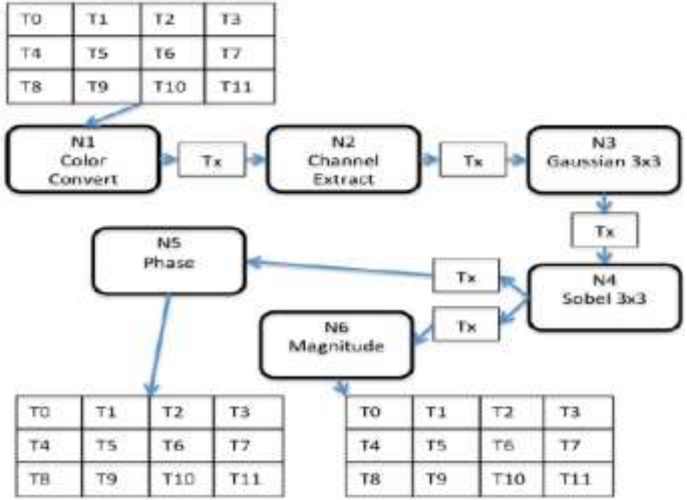
0	1	2	3	4	5	6	7	8
	N1	N2	N3	N4	N1	N2	N3	N4
		N1	N2	N3	N4	N1	N2	N3
			N1	N2	N3	N4	N1	N2
				N1	N2	N3	N4	N1

Example of 4 nodes pipelined across 4 target cores

Additional optimizations

- Block/tile processing
  - When output depends only on subset of input (not entire input), the data to be processed can be broken into tiles, and fed into the graph.
  - Each node processes 1 tile of data at a time and the graph executes *N* times for 1 image (*N* is number of tiles in the image)
  - Removes intermediate round-trips to memory
  - Requires a custom DMA framework to be implemented

- Abstraction of cache operations/address translation
  - TI provides internal memory API's for optimized cache maintenance and address translation



T0	T1	T2	T3
T4	T5	T6	T7
T8	T9	T10	T11

N1 Color Convert

Tx

N2 Channel Extract

Tx

N3 Gaussian 3x3

Tx

N4 Sobel 3x3

Tx

Tx

N5 Phase

N6 Magnitude

T0	T1	T2	T3
T4	T5	T6	T7
T8	T9	T10	T11

T0	T1	T2	T3
T4	T5	T6	T7
T8	T9	T10	T11

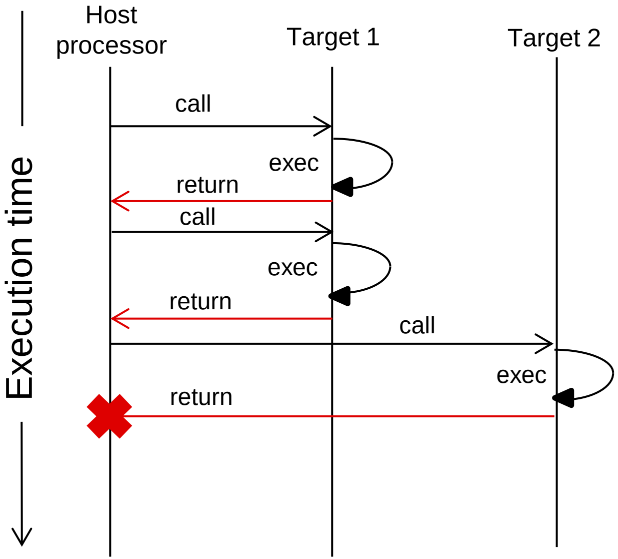
For more information

- Jacinto 7 Processor SDK Automotive download: http://www.ti.com/tool/PROCESSOR-SDK-JACINTO-DRA8X-TDA4X
- Processor SDK Linux Automotive (PSDKLA) user guide: ${PSDKLA_INSTALL_PATH}/docs/linux/index.html
- Processor SDK RTOS Automotive (PSDKRA) user guide: ${PSDKRA_INSTALL_PATH}/index.html
- For additional questions, refer to the E2E community forums: https://e2e.ti.com/support/processors/f/791
- "Addressing System-Level Optimization with OpenVX Graphs" https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6910050

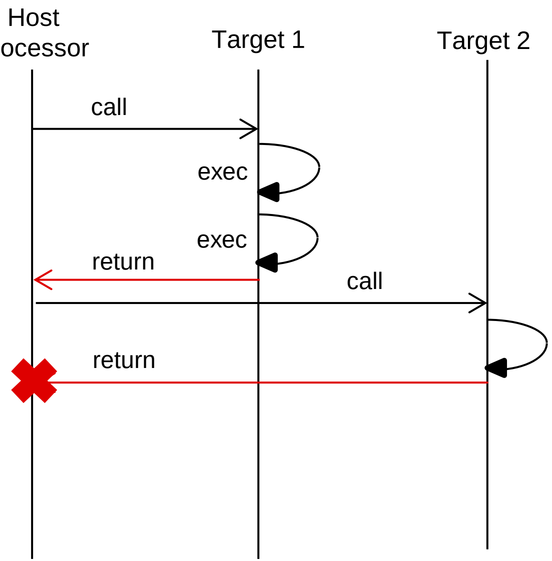
Texas Instruments

The material is provided strictly "as-is" for informational purposes only and without any warranty.
Use of this material is subject to TI's **Terms of Use**, viewable at TI.com

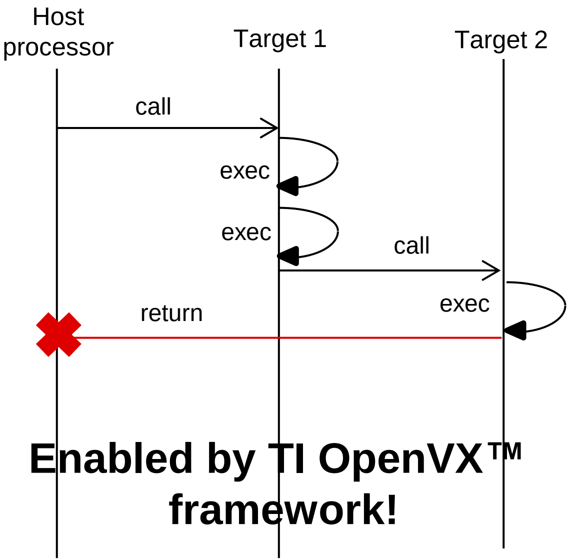
IPC aggregation



No aggregation
Highest overhead



Aggregation
(star topology)
Medium overhead

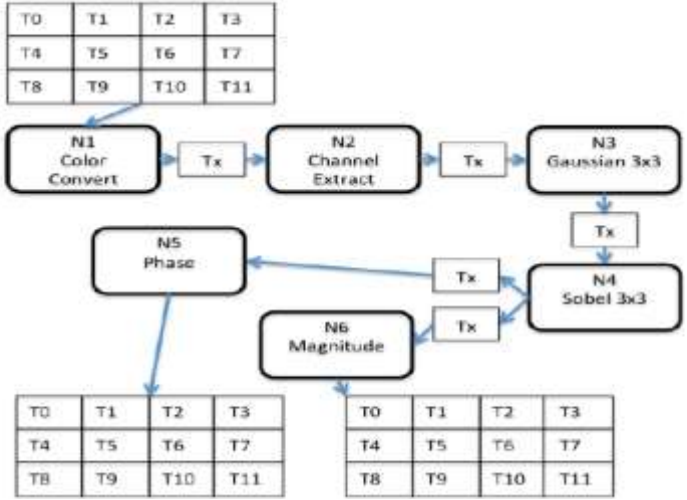


Enabled by TI OpenVX™ framework!

Aggregation
(mesh topology/
Peer to peer)
Least overhead

Block/tile processing

- When output depends only on subset of input (not entire input), the data to be processed can be broken into tiles, and fed into the graph.
- Each node processes 1 tile of data at a time and the graph executes *N* times for 1 image (*N* is number of tiles in the image)
- Removes intermediate round-trips to memory



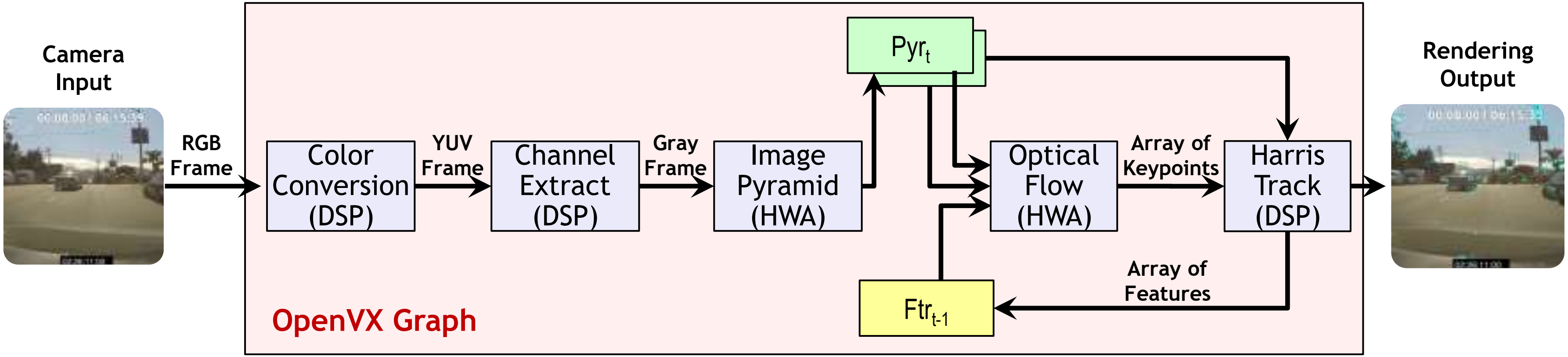
T0	T1	T2	T3
T4	T5	T6	T7
T8	T9	T10	T11

T0	T1	T2	T3
T4	T5	T6	T7
T8	T9	T10	T11

T0	T1	T2	T3
T4	T5	T6	T7
T8	T9	T10	T11

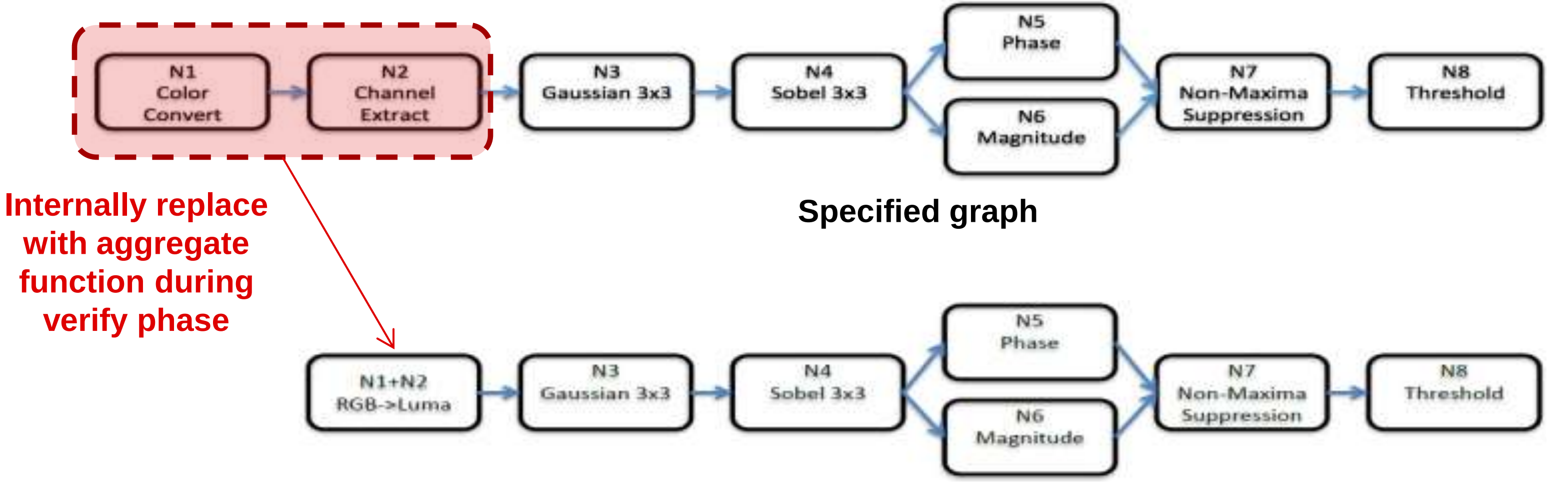
Abstraction of cache operations/address translation

- TI provides internal memory API's for optimized cache maintenance and address translation



Graph model enables … kernel fusion

- Aggregate function replacement :
  - Identifying a sub-set of nodes which can be replaced by a single equivalent node:



Internally replace with aggregate function during verify phase

Specified graph

Executed graph