Efficiently Map AI and Vision Applications onto Multi-Core AI Processors Using CEVA's Parallel Processing Framework

Rami Drucker

SW Architect

CEVA

2023 embedded VISION SUMMIT

Executive Summary

- Next generation AI and CV applications require higher than ever computing power.

- Edge devices use multi-core processors to deliver high performance.
- However, developers must efficiently map their applications onto the multiple cores, which can be difficult.

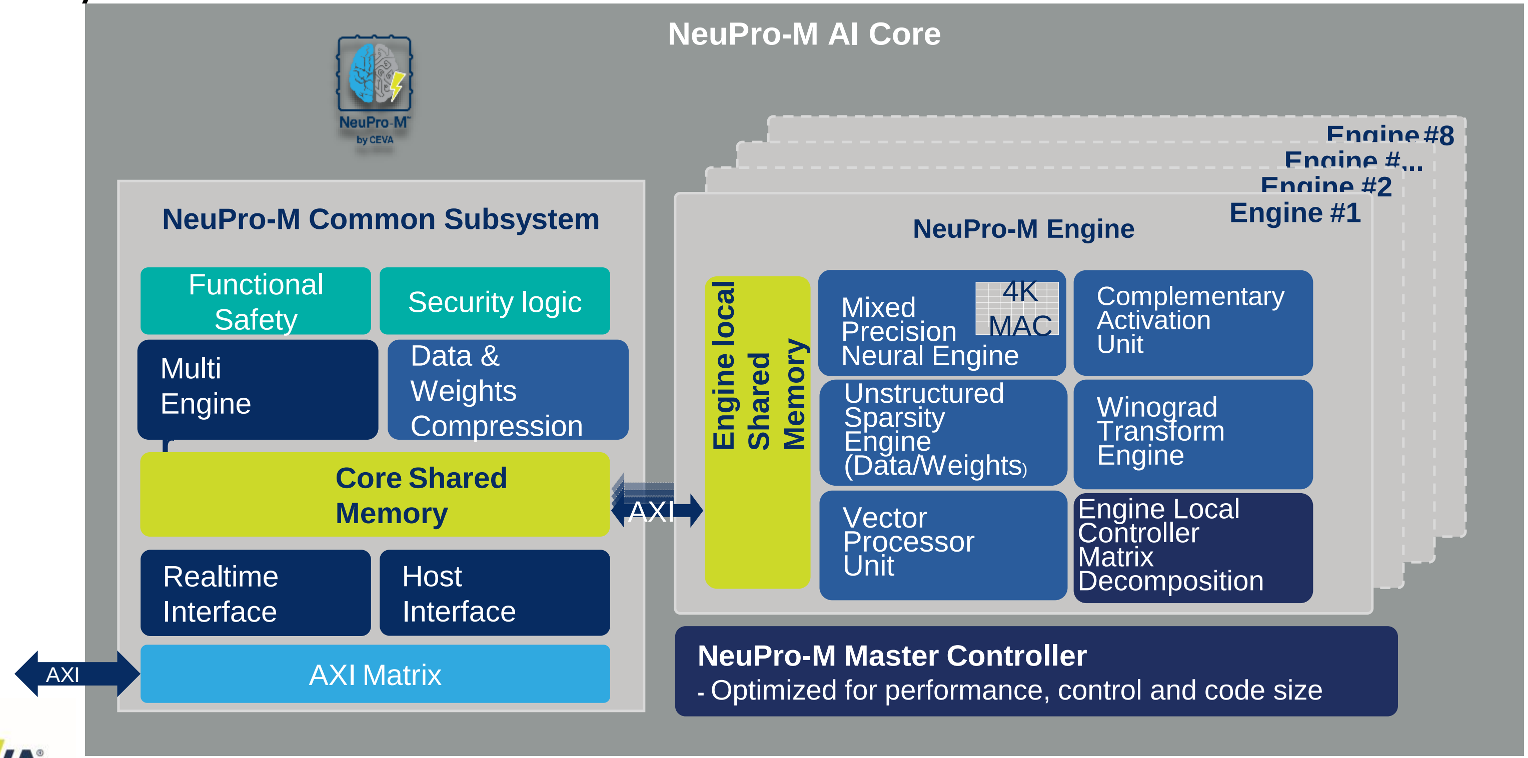
- CEVA has introduced the Architecture Planner tool as a new element of CDNN, CEVA's comprehensive AI SDK.

- In this talk, we'll show how the Architecture Planner tool analyzes the network model and maps the workload onto the multiple cores in an efficient manner.

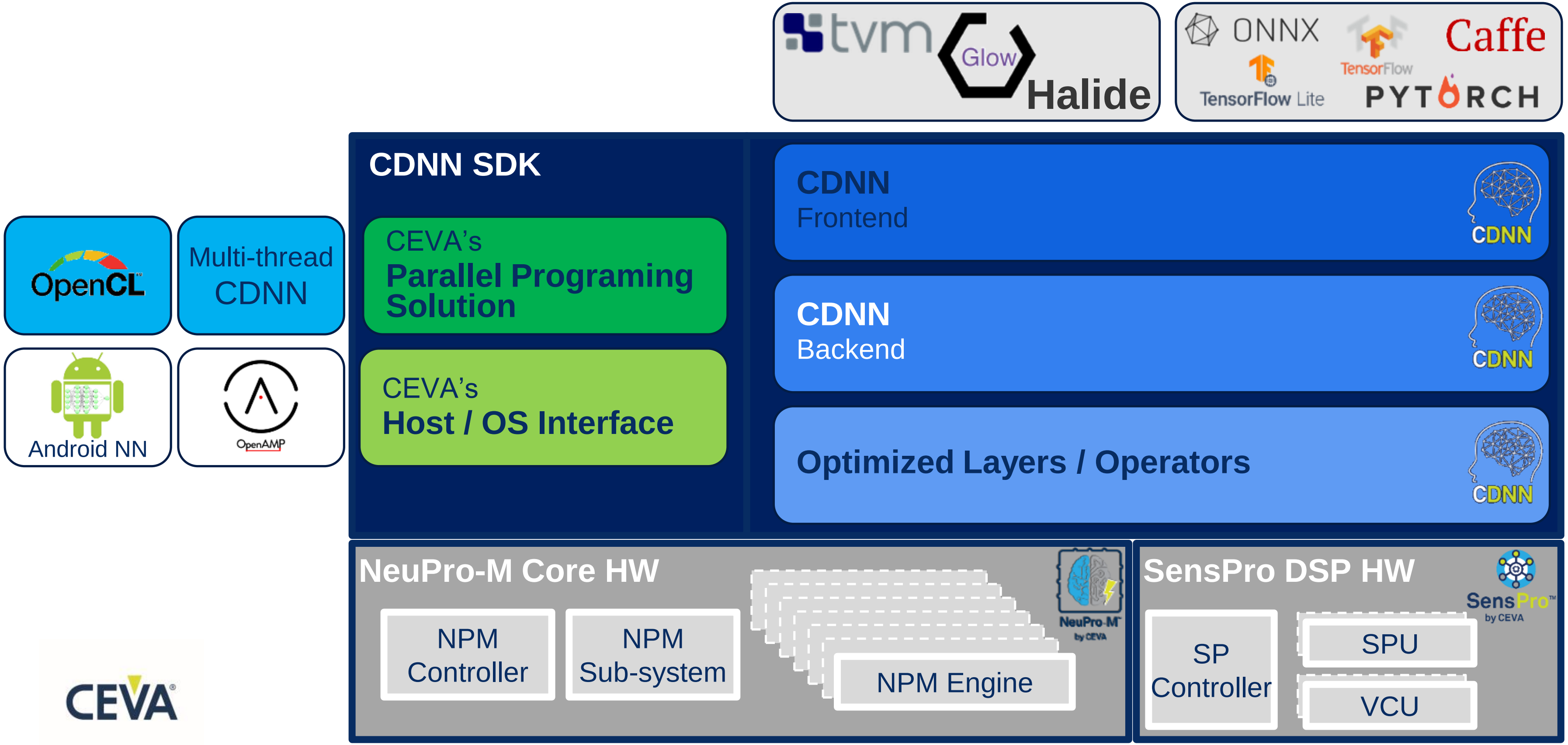
- We'll explain key techniques used by the Tool, including symmetrical and asymmetrical multi-processing paradigms.

NeuPro-M – A Family of AI Processors

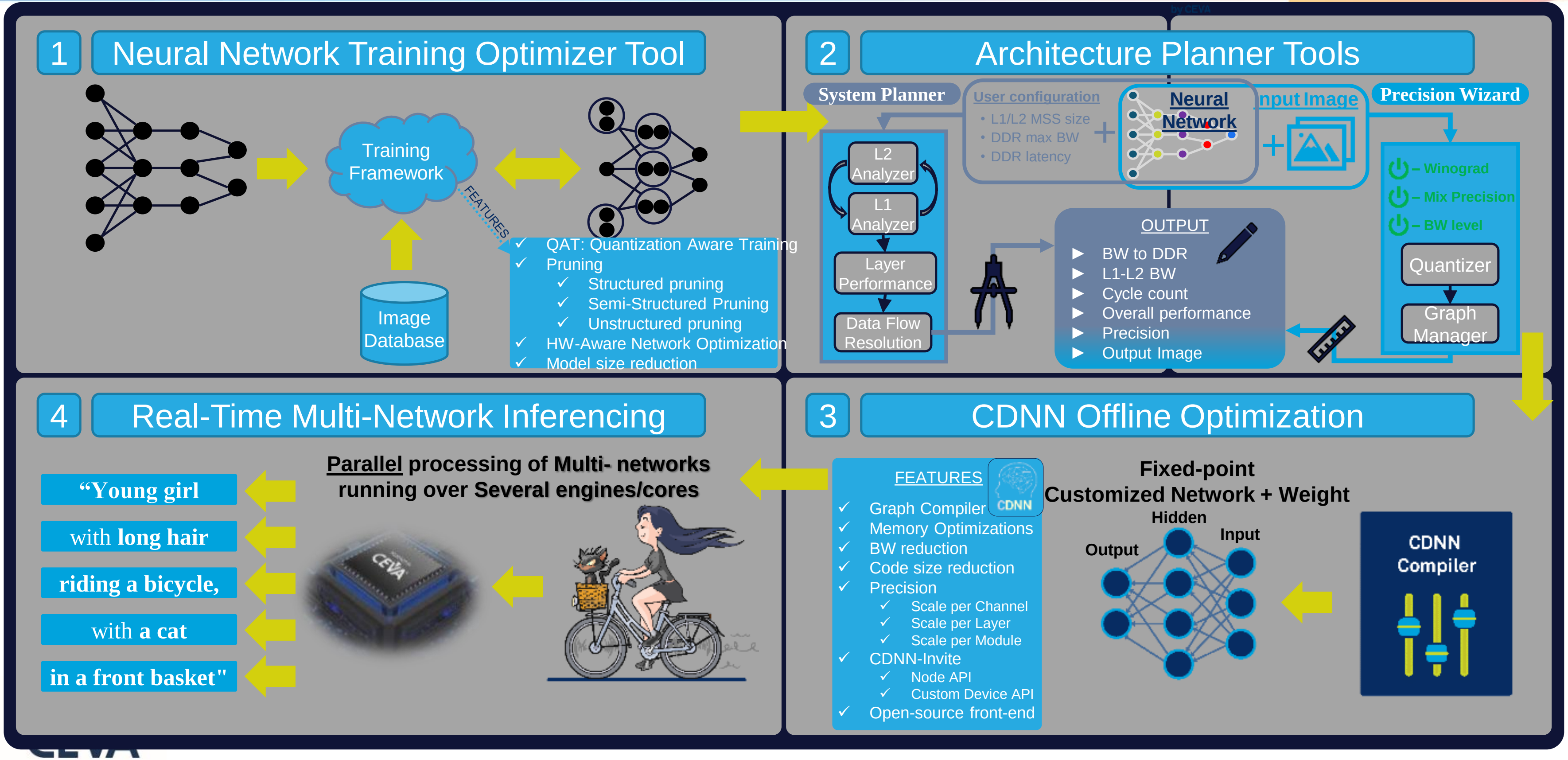
A Full System Solution



CDNN Open Development Platform



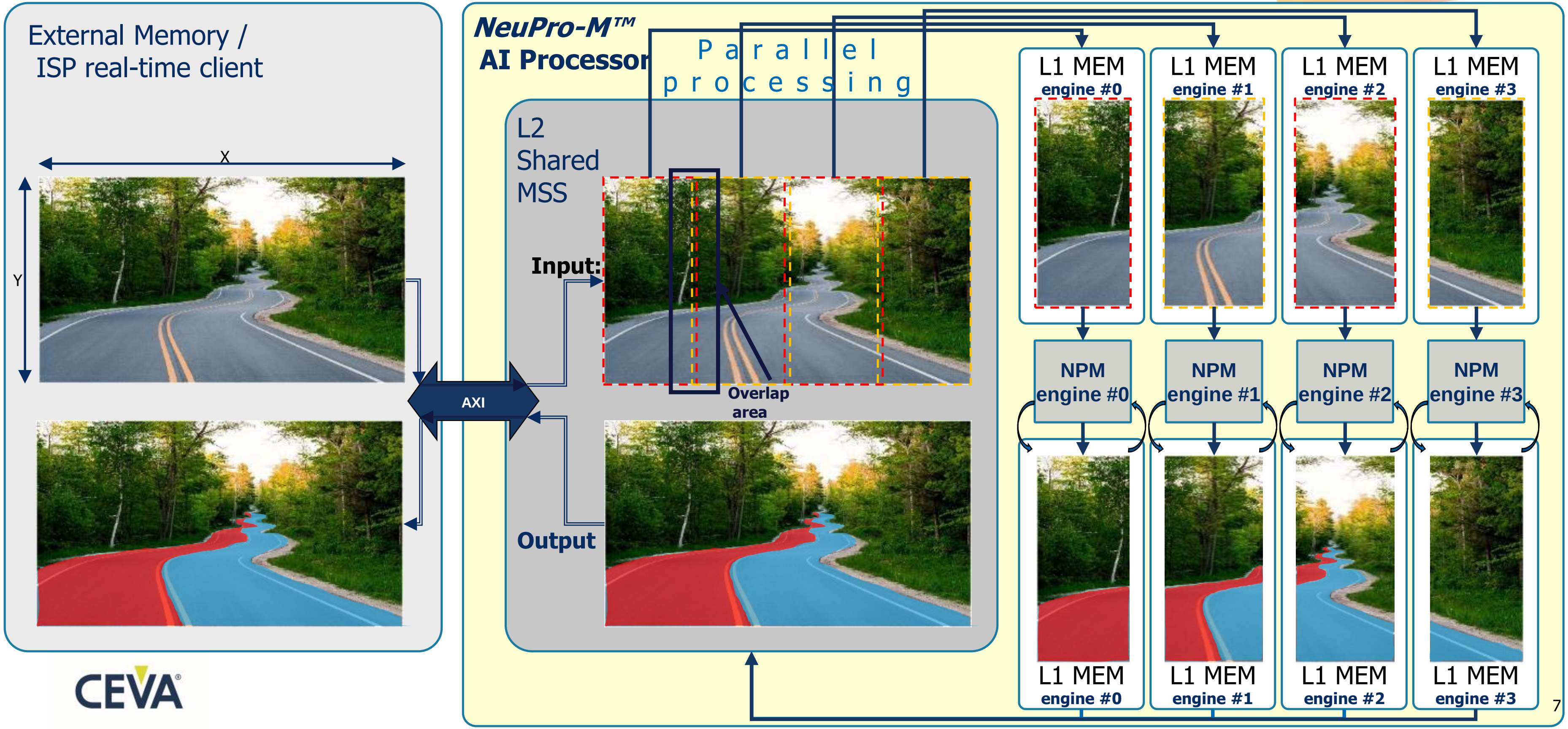
CDNN Toolchain Workflow



CDNN Multi-Engine Features

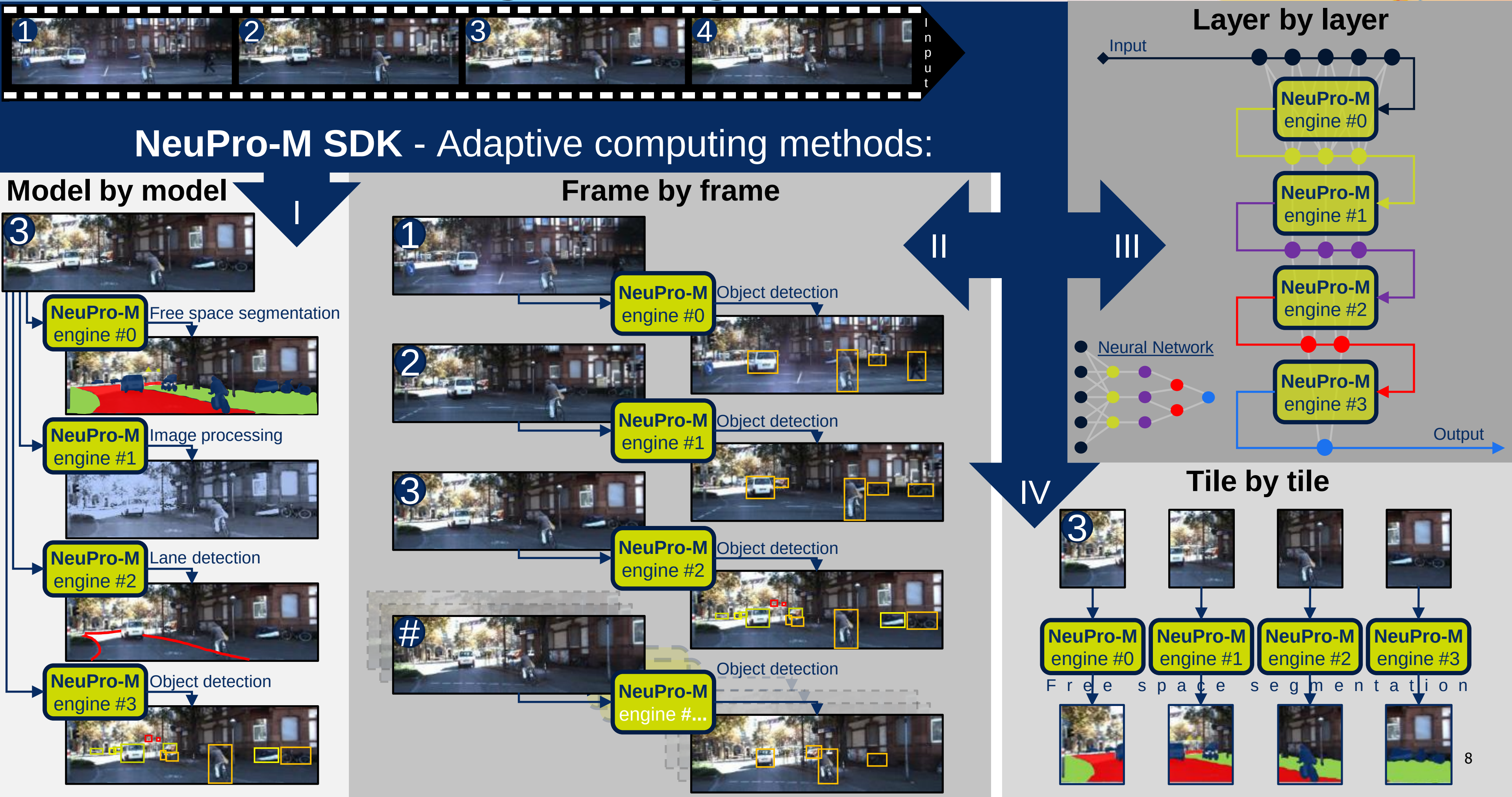
- CDNN supports the following parallel processing paradigms to leverage the compute power of multi-engine device and maximize overall system throughput
  - Partitioning by tiles → different tiles, shared weights. This is efficient in first layers where input maps are large
  - Partitioning by output maps → same tile, different weights
  - Partitioning by sub-graphs → different sub-graphs are assigned to engines
  - Pipeline partitioning → sequence of nodes assigned to engine
  - Batch partitioning → same network, different input image
  - Partitioning by networks → different network per engine

Quad-Engine Core – Segmentation Network Data Flow

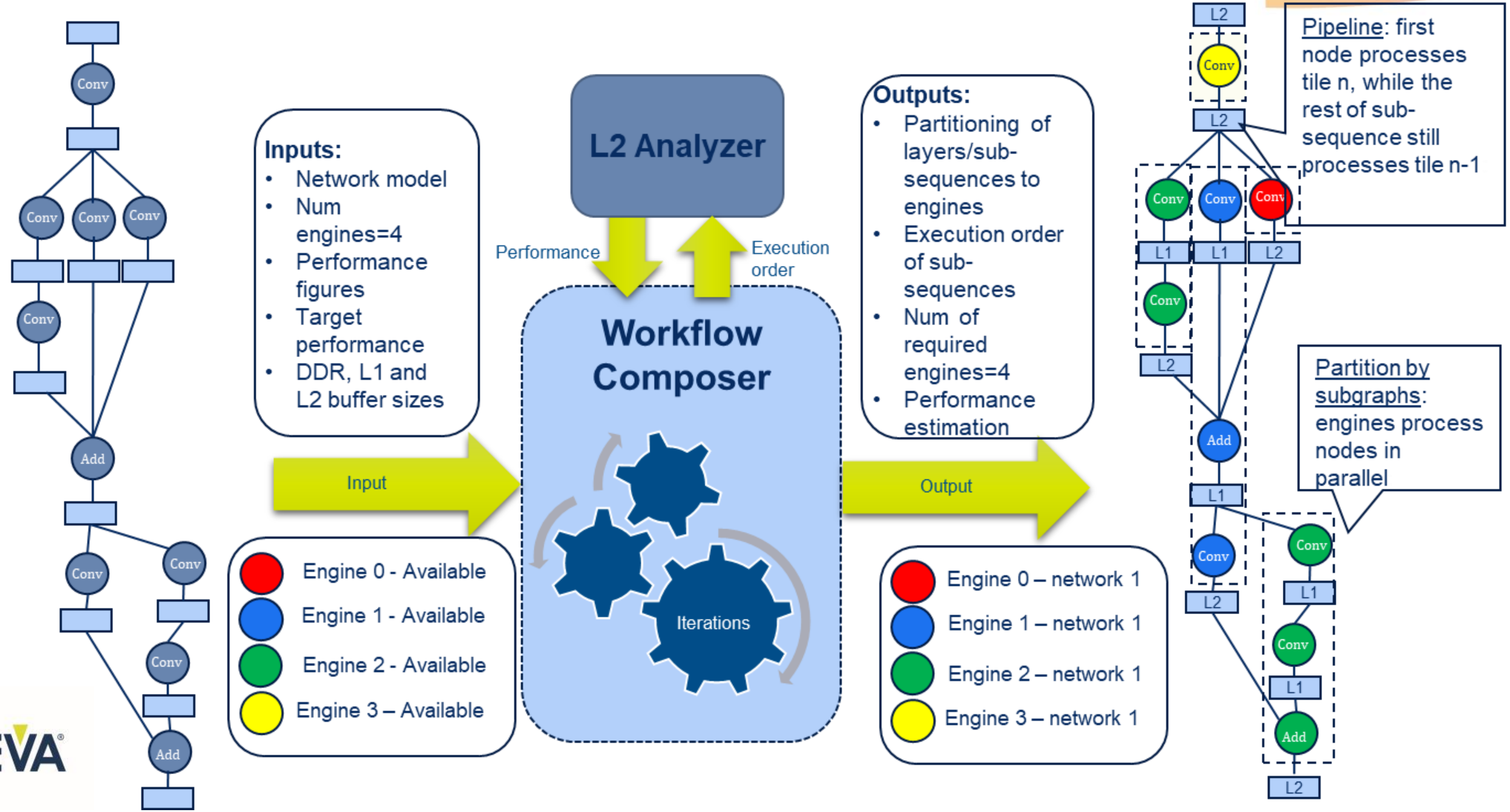


Parallel Processing Paradigms

embedded

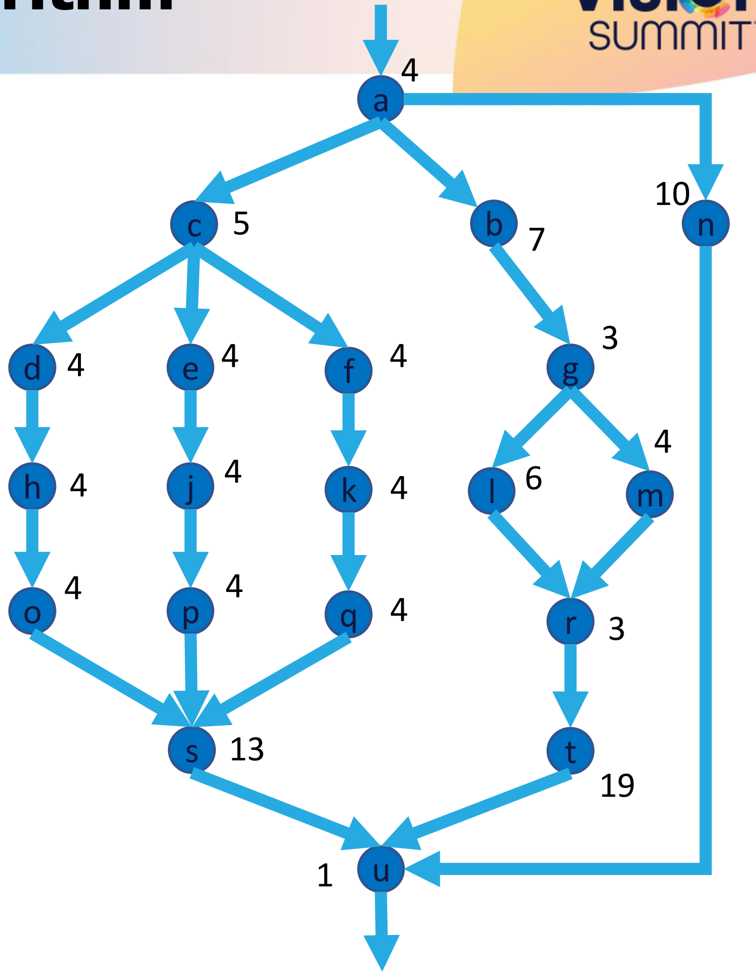


CDNN Architecture Planner



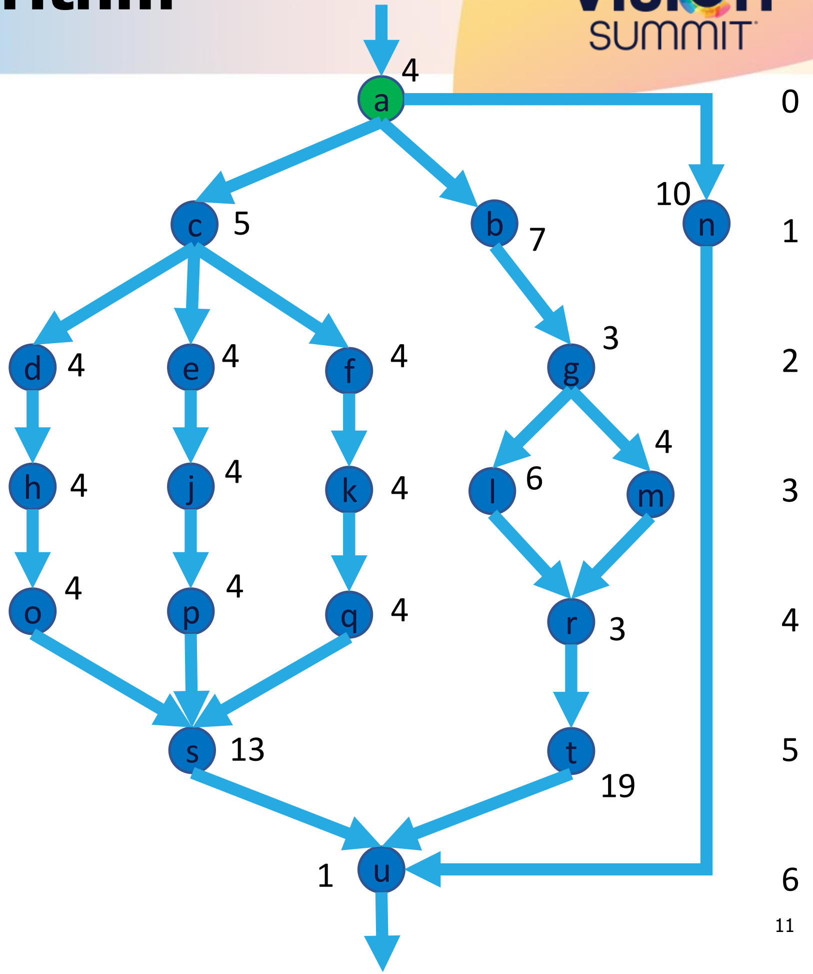
CDNN Multi-Engine Offline Algorithm

- Given graph G=(V,E)
- Given input and output nodes (a,u)
- Given cost function f(v) for each node
- We have multiple paths from a to u
- Parallel execution is appropriate in this case
- We will attempt to parallelize the model across 3 DSPs



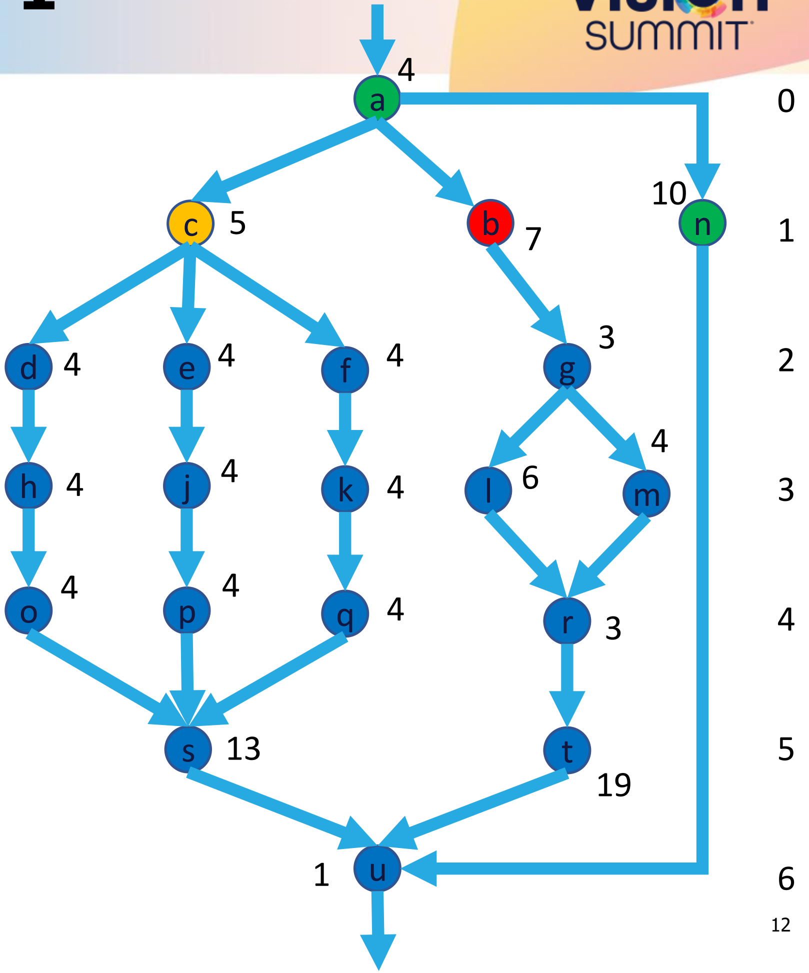
CDNN Multi-Engine Offline Algorithm

Dsp1	Start	finish	Dsp2	Start	finish	Dsp3	Start	finish
a	0	4						



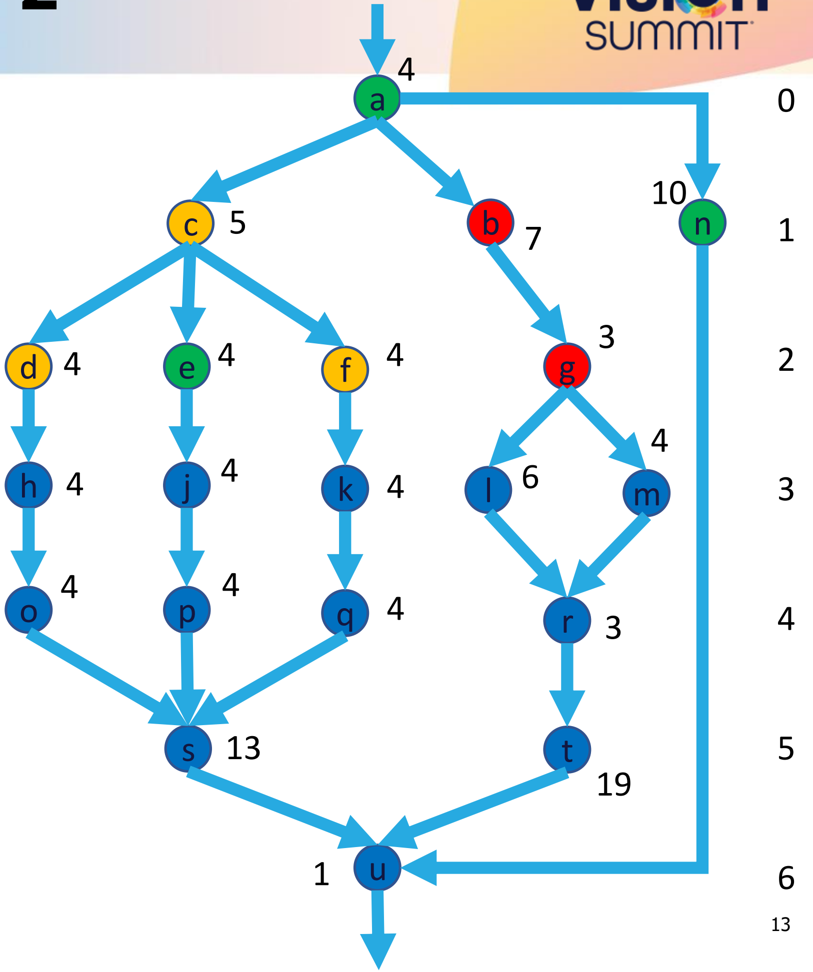
CDNN Offline Algorithm – Layer 1

Dsp1	Start	finish	Dsp2	Start	finish	Dsp3	Start	finish
a	0	4	-----	0	4	-----	0	4
n	4	14	b	4	11	c	4	9



CDNN Offline Algorithm – Layer 2

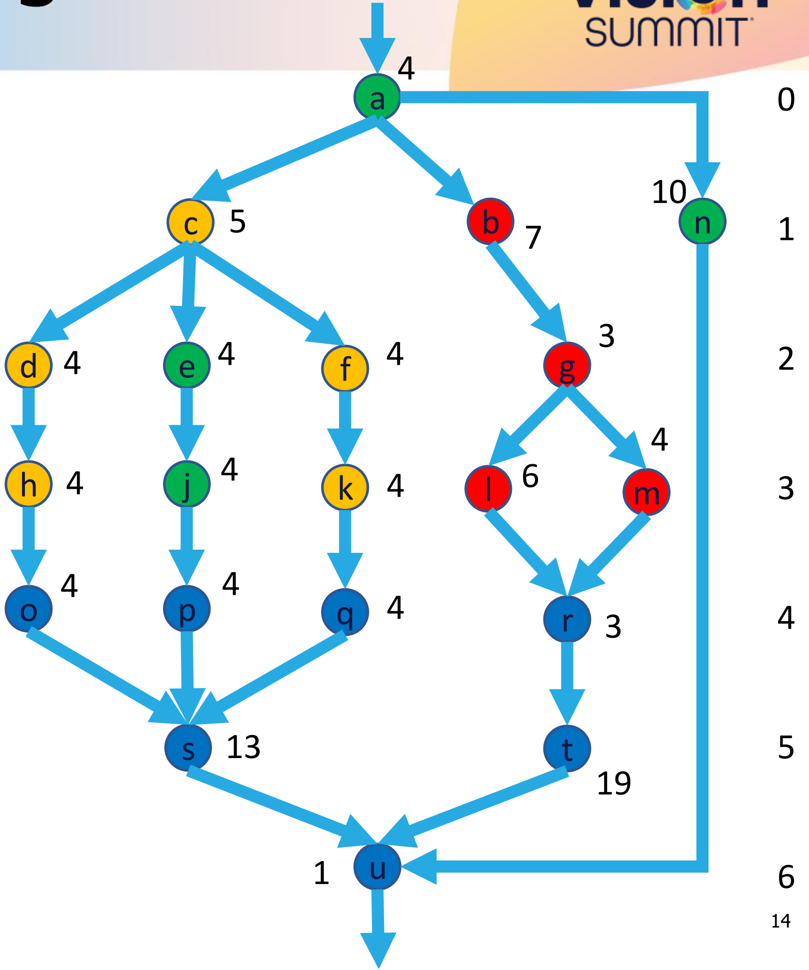
Dsp1	Start	finish	Dsp2	Start	finish	Dsp3	Start	finish
a	0	4	-----	0	4	-----	0	4
n	4	14	b	4	11	c	4	9
e	**14**	**18**	**g**	**11**	**14**	**d**	**9**	**13**
						f	**13**	**17**



CEVA

CDNN Offline Algorithm – Layer 3

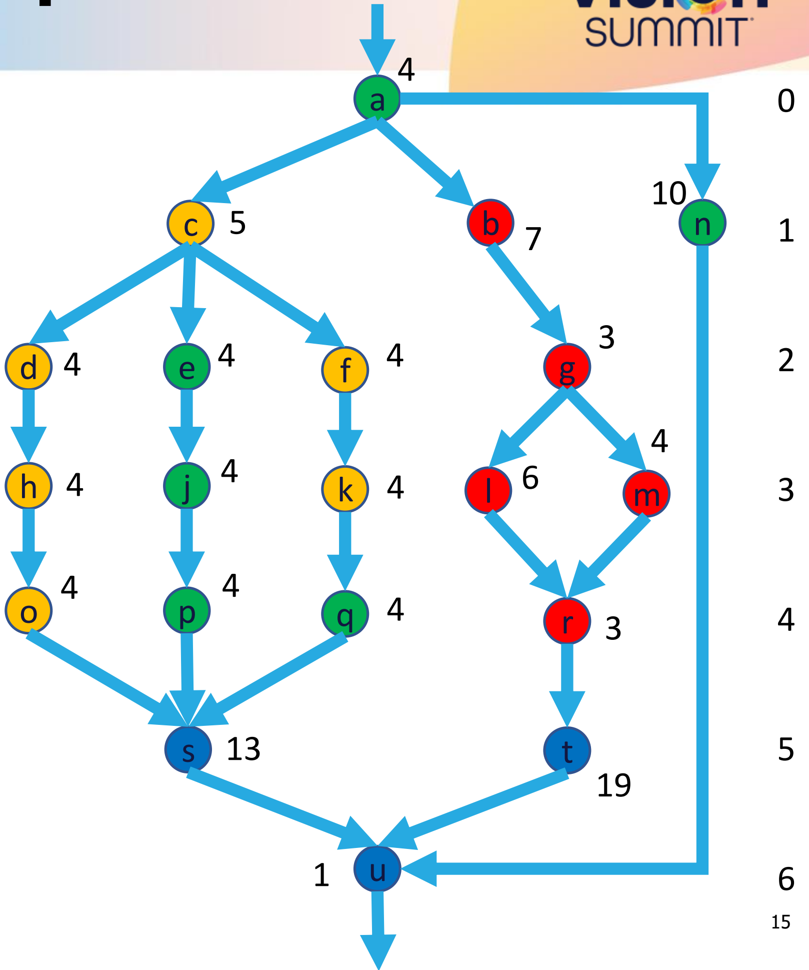
Dsp1	Start	finish	Dsp2	Start	finish	Dsp3	Start	finish
a	0	4	-----	0	4	-----	0	4
n	4	14	b	4	11	c	4	9
e	14	18	g	11	14	d	9	13
j	**18**	**22**	**l**	**14**	**20**	f	13	17
			m	**20**	**24**	**k**	**17**	**21**
						h	**21**	**25**



CEVA

CDNN Offline Algorithm – Layer 4

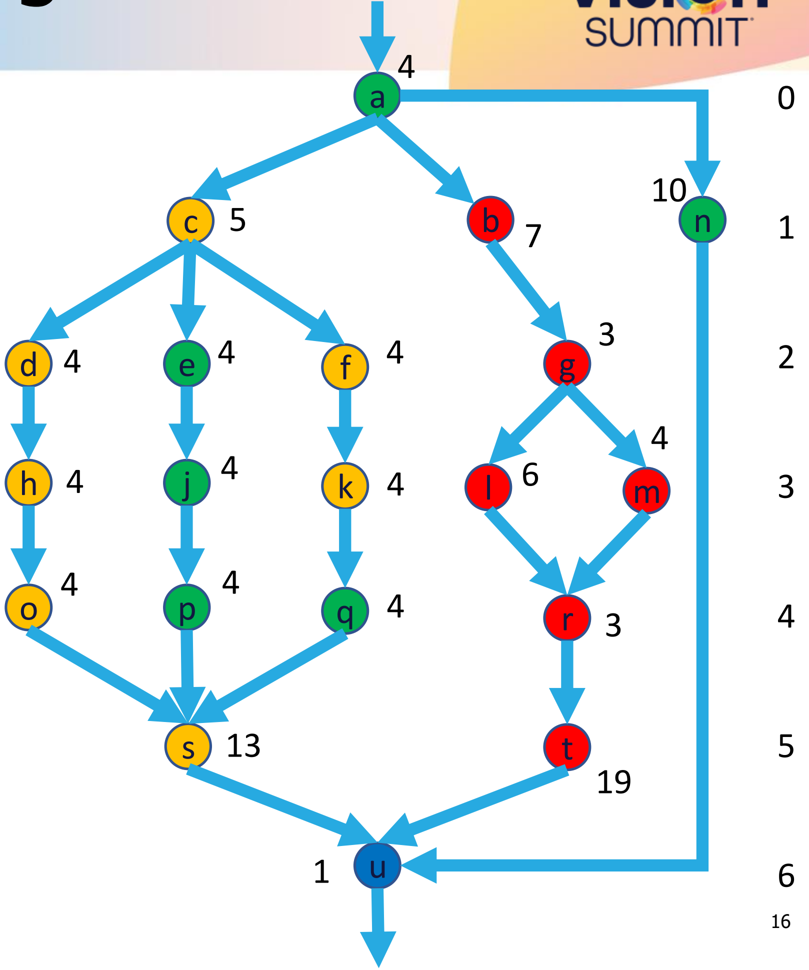
Dsp1	Start	finish	Dsp2	Start	finish	Dsp3	Start	finish
a	0	4	-----	0	4	-----	0	4
n	4	14	b	4	11	c	4	9
e	14	18	g	11	14	d	9	13
j	18	22	l	14	20	f	13	17
p	**22**	**26**	m	20	24	k	17	21
q	**26**	**30**	**r**	**24**	**27**	h	21	25
						o	**25**	**29**



CEVA

CDNN Offline Algorithm – Layer 5

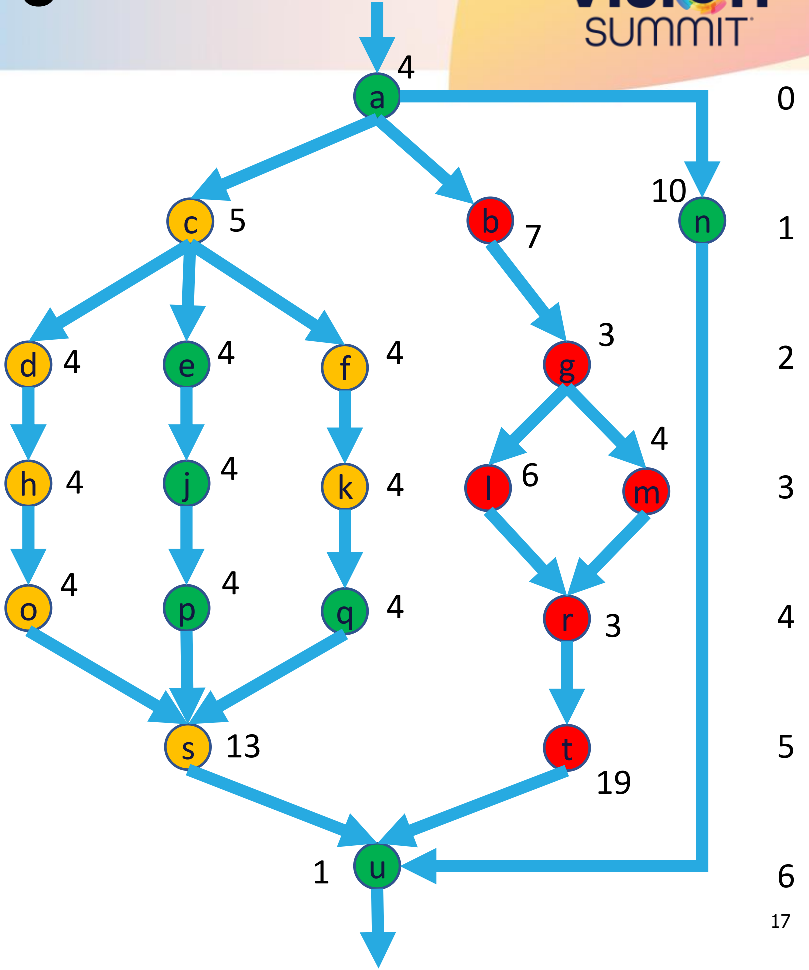
Dsp1	Start	finish	Dsp2	Start	finish	Dsp3	Start	finish
a	0	4	-----	0	4	-----	0	4
n	4	14	b	4	11	c	4	9
e	14	18	g	11	14	d	9	13
j	18	22	l	14	20	f	13	17
p	22	26	m	20	24	k	17	21
q	26	30	r	24	27	h	21	25
			t	**27**	**46**	o	25	29
						s	**29**	**41**



CEVA

CDNN Offline Algorithm – Layer 6

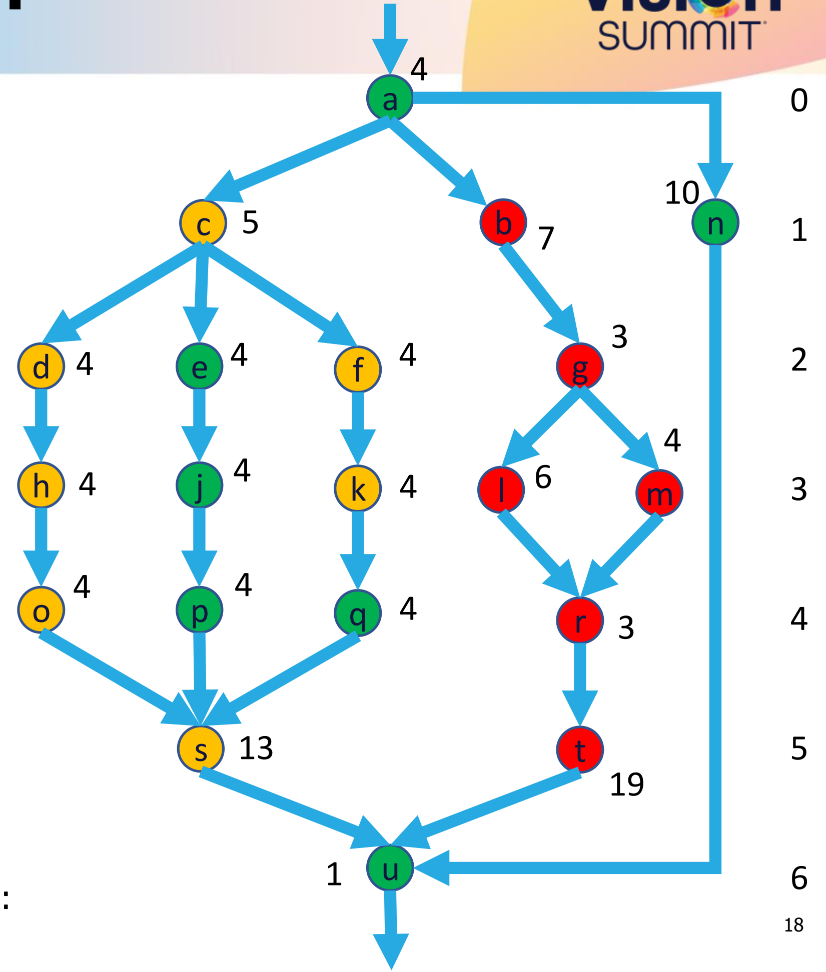
Dsp1	Start	finish	Dsp2	Start	finish	Dsp3	Start	finish
a	0	4	-----	0	4	-----	0	4
n	4	14	b	4	11	c	4	9
e	14	18	g	11	14	d	9	13
j	18	22	l	14	20	f	13	17
p	22	26	m	20	24	k	17	21
q	26	30	r	24	27	h	21	25
u	**46**	**47**	t	27	46	o	25	29
						s	29	41



CEVA

CDNN Offline Algorithm – Finish

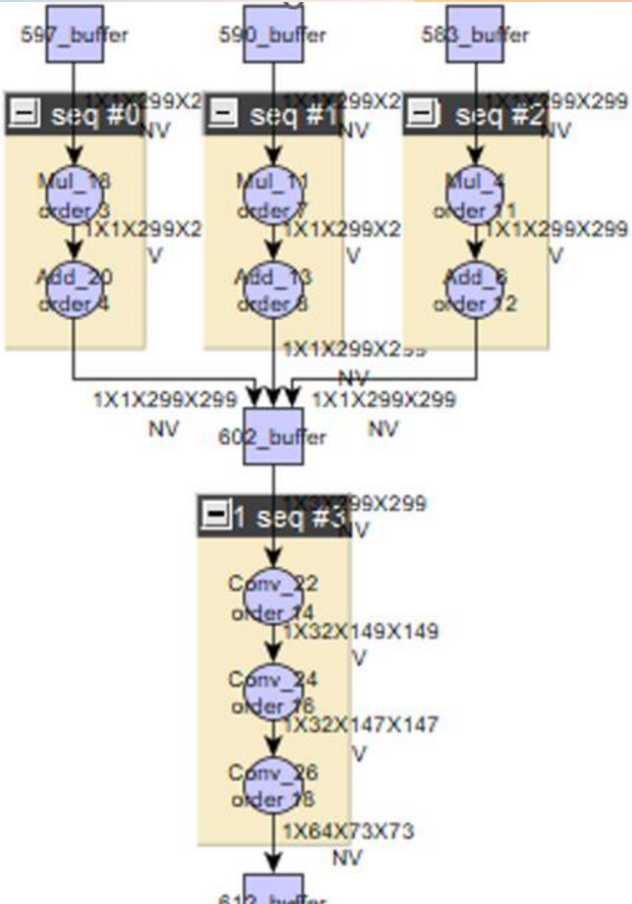
Dsp1	Start	finish	Dsp2	Start	finish	Dsp3	Start	finish
a	0	4	-----	0	4	-----	0	4
n	4	14	b	4	11	c	4	9
e	14	18	g	11	14	d	9	13
j	18	22	l	14	20	f	13	17
p	22	26	m	20	24	k	17	21
q	26	30	r	24	27	h	21	25
-----	30	46	t	27	46	o	25	29
u	**46**	**47**				s	29	41



Total inference time for 3 DSPs: 47
Single DSP inference time is the sum of all nodes:
4+5+7+10+9*4+3+13+13+19+1 = 111

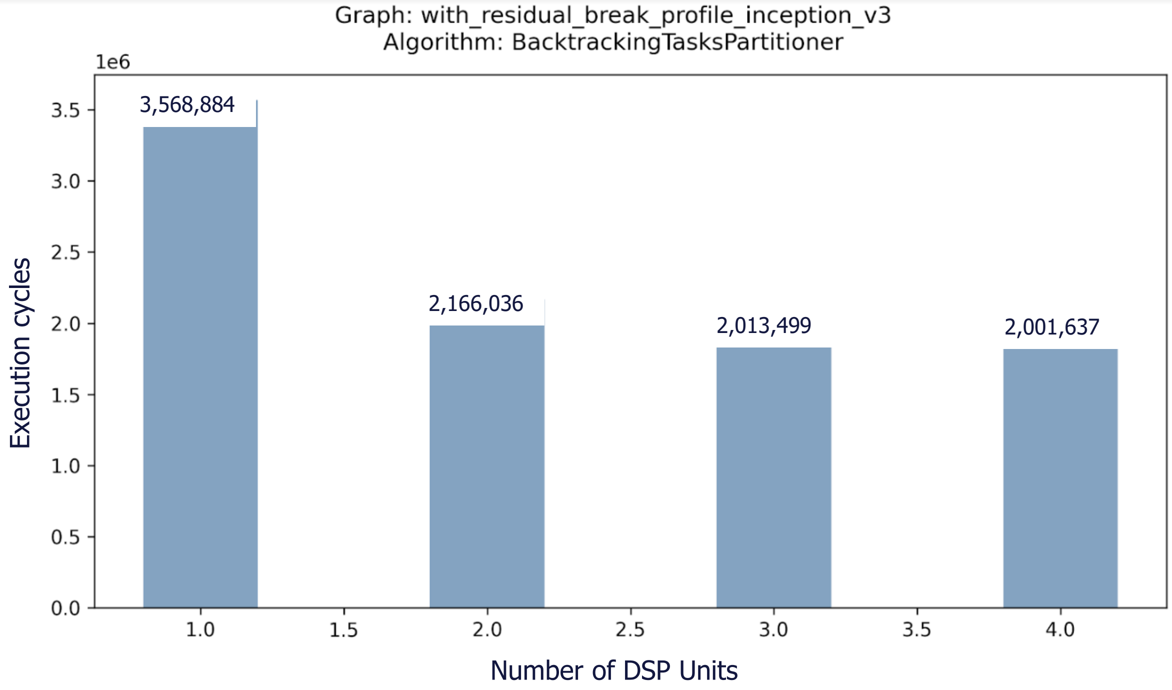
Example: Inception_v3

- Algorithm performs load balancing across sub-graphs to minimize idle time of DSPs
- Parallel execution is appropriate in this case
- We will attempt to parallelize the model across 1, 2, 3, and 4 DSPs and compare the results
- The algorithm implements backtracking traversal of the graph to limit the computational complexity of the process

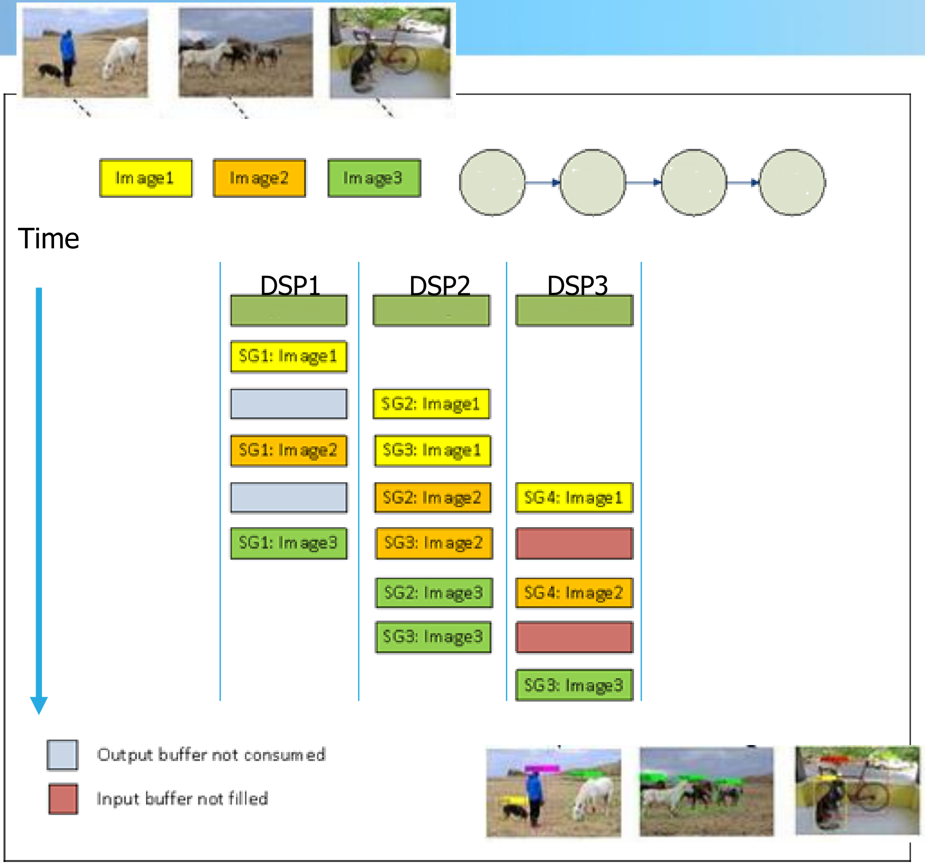


Results: Inception_v3

- Backtracking algorithm produced the best results
- The bar chart compares the performance in cycles of a single DSP vs two, three, and four DSPs



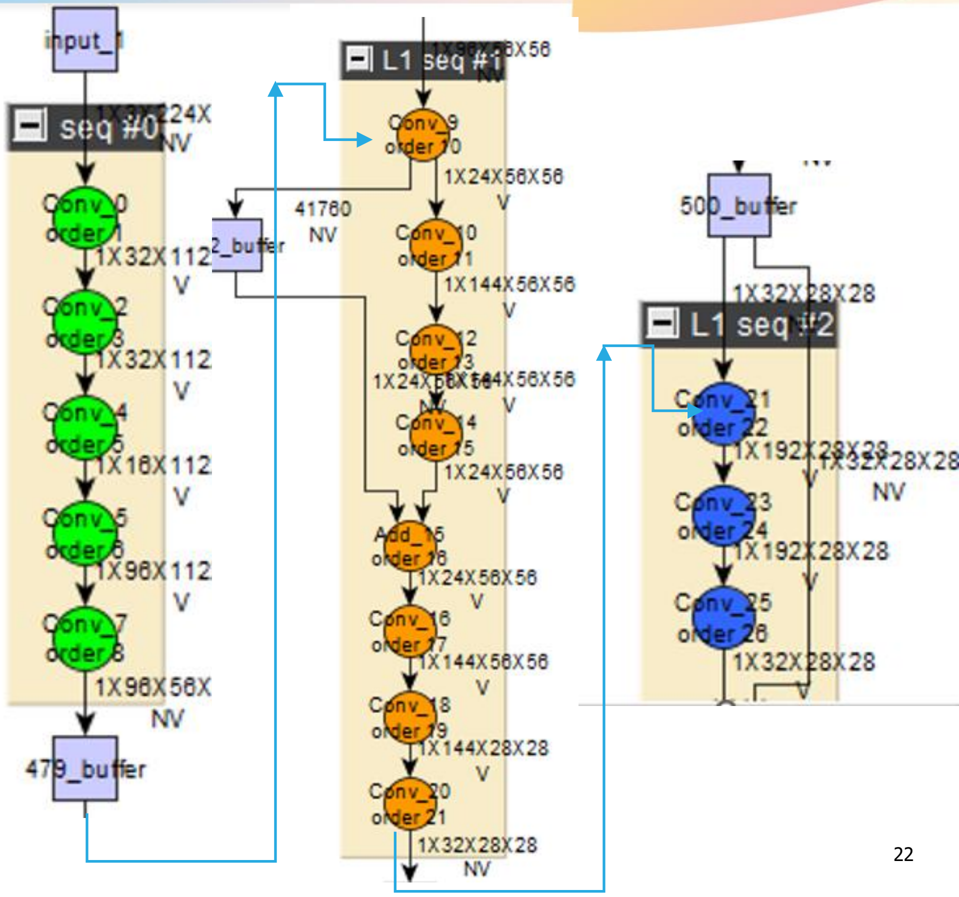
CDNN Multi-engine Pipeline Execution



- Many neural networks comprise a long thread of nodes that are computed one after the other
- Parallelization is difficult because there are no nodes that can run in parallel
- A pipeline approach is adequate in this case
- We will attempt to parallelize the model across 3 DSPs
- To minimize idle time, the algorithm needs to balance the load across DSPs

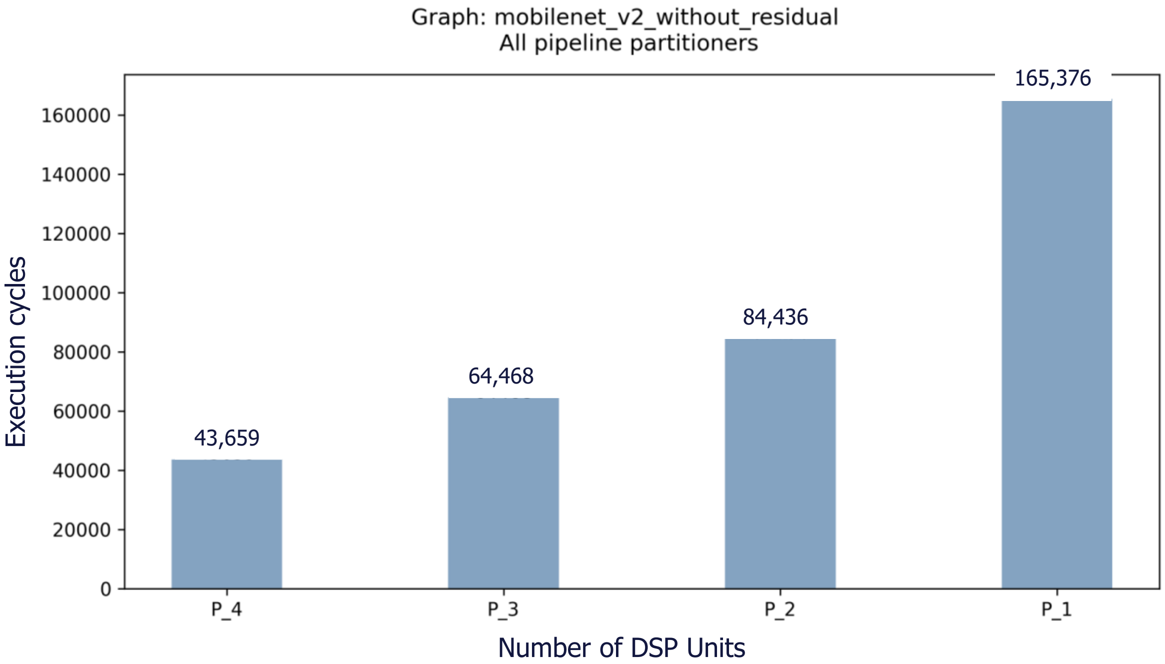
Example: Mobilenet_v2

- Given Graph G=(V,E)
- The resultant frame rate equals to 1/CYCLE_COUNT of the slowest section of the pipeline



Results: Mobilenet_v2

- The pipeline algorithm produced the best result with 4 DSPs



CEVA

Summary and Conclusions

- We introduced CEVA NeuPro-M multi-engine processor for AI and CV applications.
- These cores are complemented by CDNN for multi-engine, a highly optimized graph compiler and runtime framework.
- We presented the results of sub-graph and pipeline algorithms, which were developed at CEVA to distribute the network inference workload across multiple engines.
- Surprisingly, even for Inception_V3, the pipeline approach outperformed the backtracking technique.
- Further testing would need to be carried out to confirm the initial results.

Resources

For more information, please visit our booth, #420

www.ceva-dsp.com