Digital Signal Processing in VLSI Design

Final Project Presentation

Sparse CNN Hardware Design

GROUP 12
R09921132 劉彥甫
R09921129 黃意堯

Motivation and Prior Works

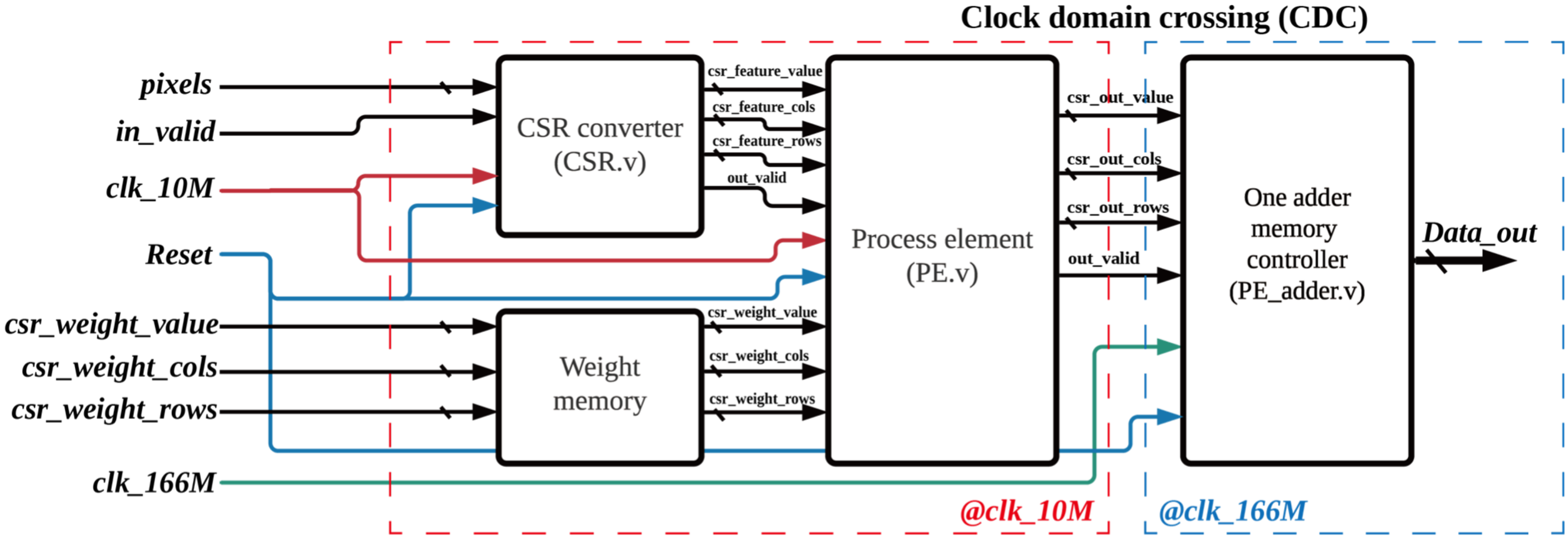
- Two additional benefit of sparse matrix convolution are saving computing power and saving memory usage. Like we talked about FFT Radix-n design, does sparse convolution really save cost in hardware perspective?
- The irregular sparse patterns introduced by both weights and activations are much more challenging for efficient computation. For example, due to the issues of access contention, workload imbalance, and tile fragmentation, the state-of-the-art sparse accelerator SCNN fails to fully leverage the benefits of sparsity, leading to nonoptimal results for both speedup and energy efficiency.

System Architecture and Design Features

- Use streaming data flow to reduce memory traffic/usage.
- Use one adder working at 16 times clock frequency to solve memory access contention which mentioned at prior works.
- Use parallel adder to solve the problem on power consuming of one adder.
- Use clock gated to save some power.

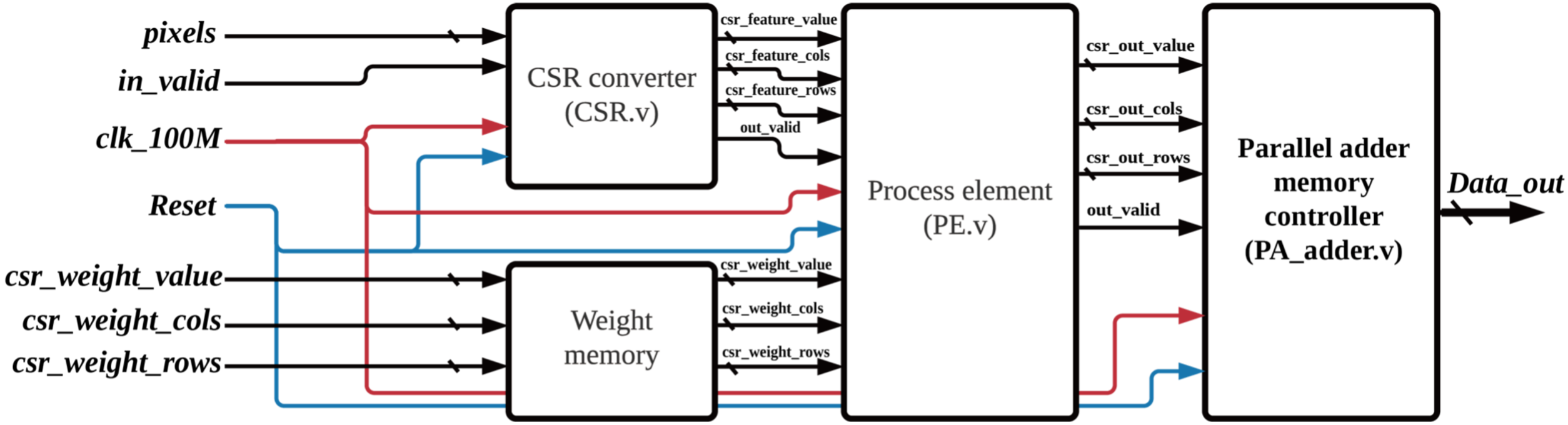
System Architecture and Design Features (cont.)

Sparse Convolution



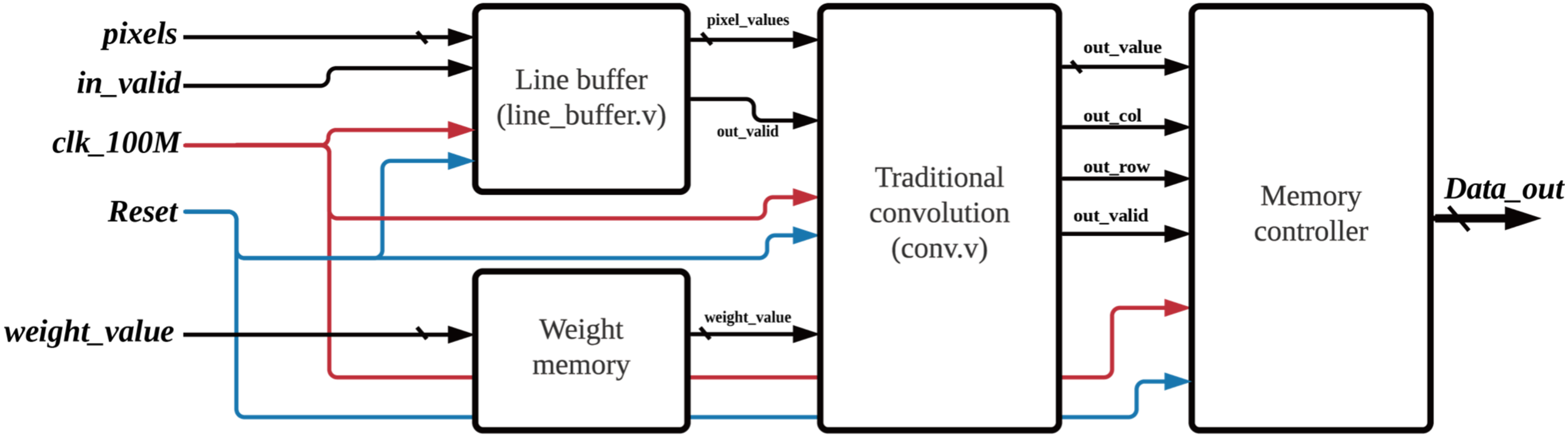
System Architecture and Design Features (cont.)

Sparse Convolution with parallel adder



System Architecture and Design Features (cont.)

Traditional Convolution



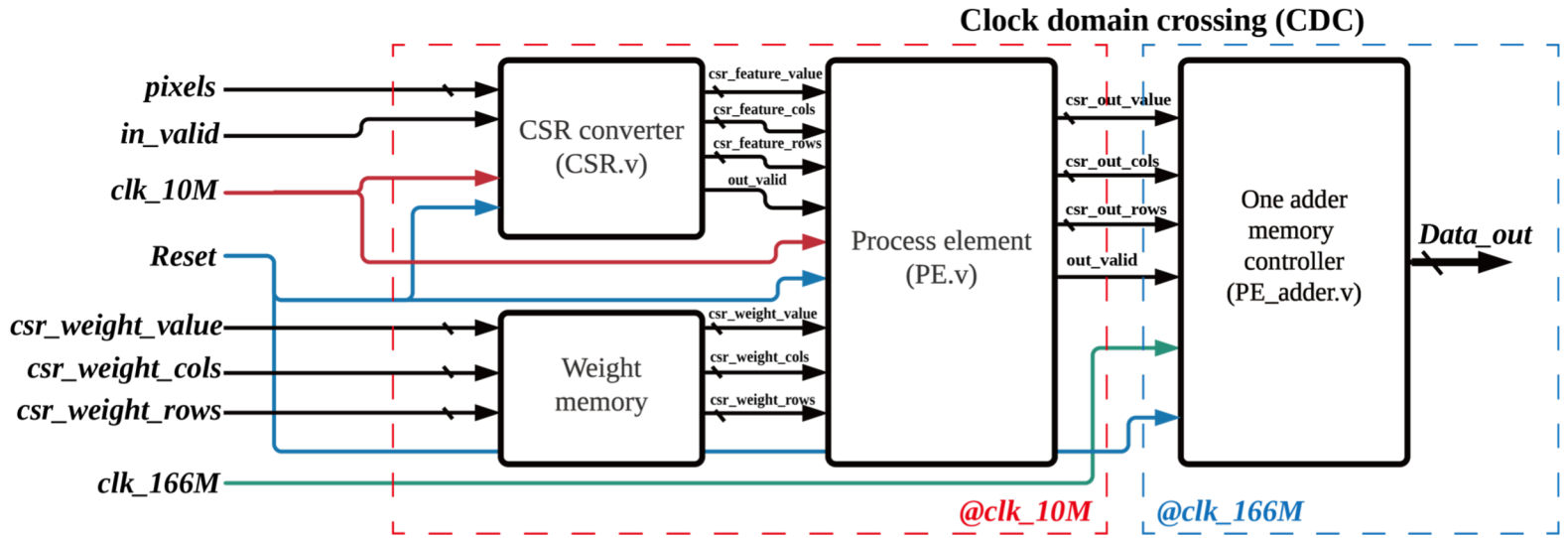
Design Space Exploration (and Methodology)

- CSR module: data-stream v.s data-batch (in report)
- CSR/adder module: reference design v.s 2x parallel design (in report)
- PE module: reference design v.s 2x parallel v.s 2x pipeline (in report)
- Write back module: one adder in 16x fclk v.s fully parallel (in report)
- Traditional convolution v.s Sparse convolution @ different sparsity

Implementation Results

◆ Sparse convolution@10/166.67MHz (PE_CSR_clk/Memory_clk)

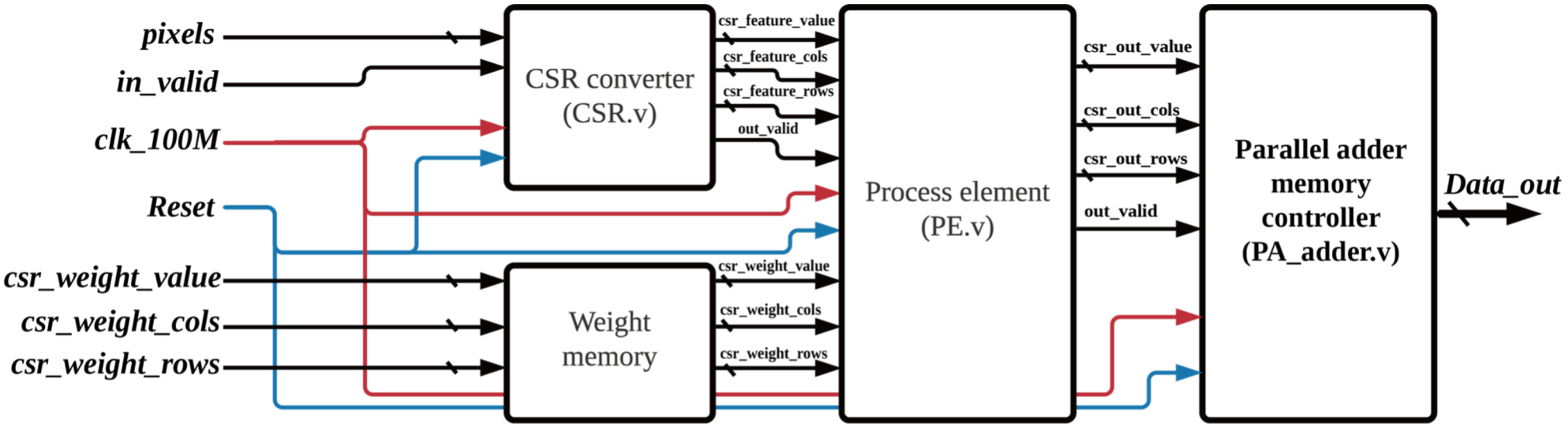
	Area[um^2]	Power[W](prime_time)	Power*Area	Latency [ns] (Time to finish one frame)
Input@0%sparsity	**2,417,592 um^2**	**0.154 W**	**372,309**	**203,955 ns**



Implementation Results

◆ Sparse convolution with parallel adder@100MHz

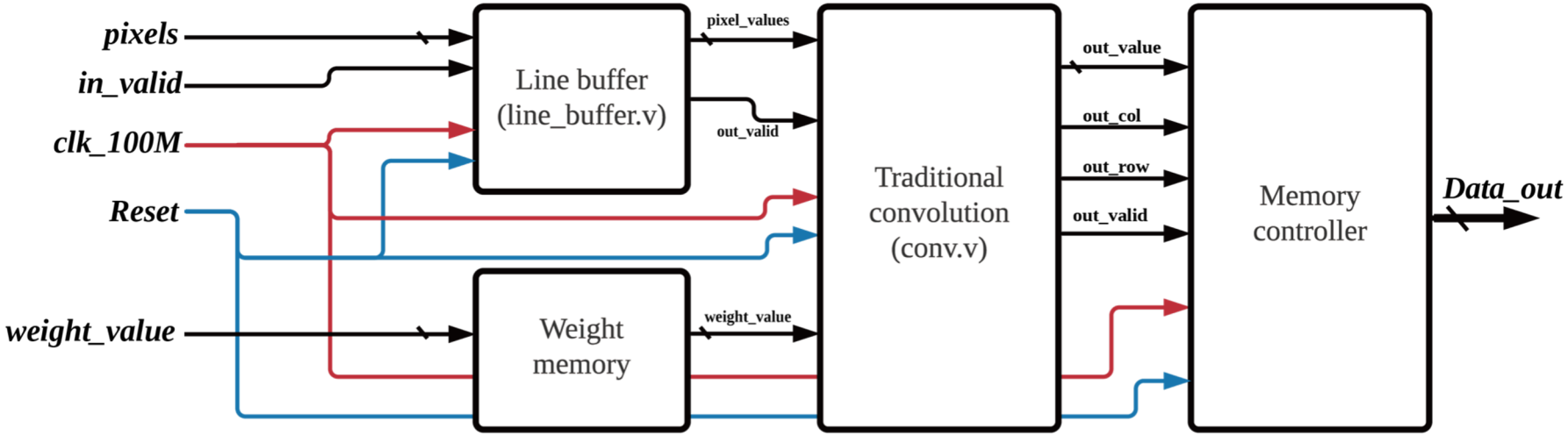
	Area[um^2]	Power[W](prime_time)	Power*Area	Latency [ns] (Time to finish one frame)
Input@0%sparsity	**21,267,287 um^2**	**0.482 W**	**10,250,832**	**21,265.3 ns**



Implementation Results

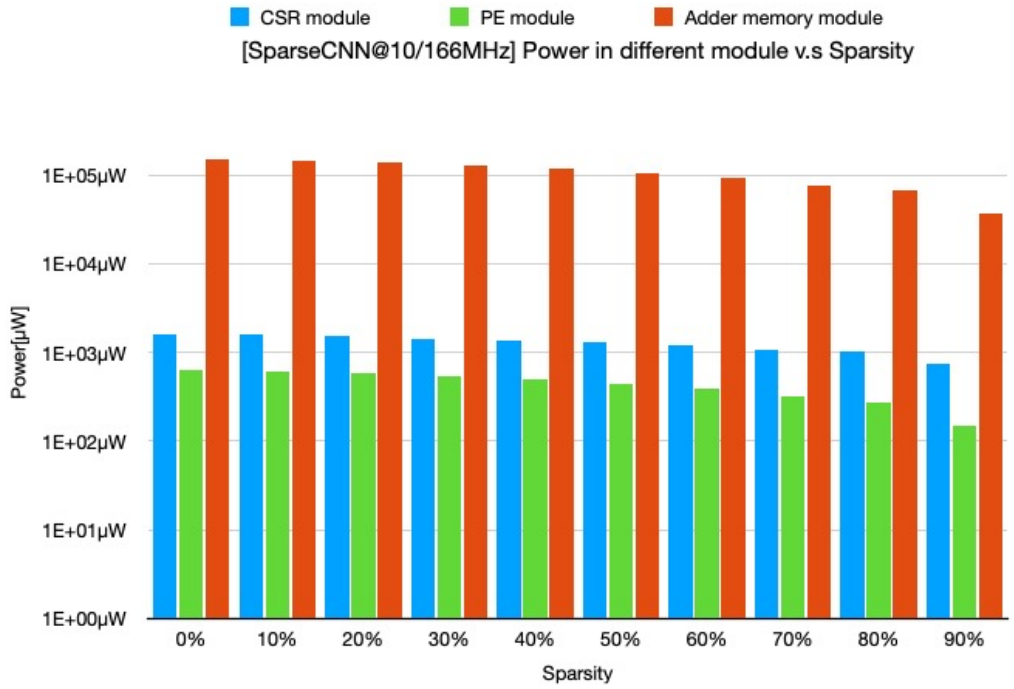
◆ Traditional convolution@100MHz

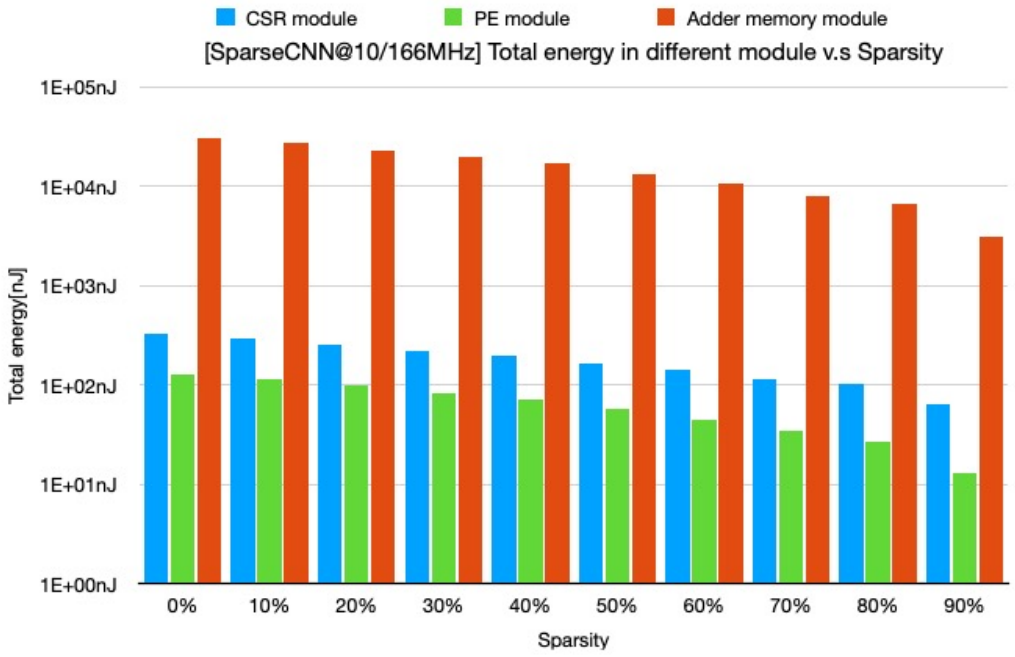
Traditional	Area[um^2]	Power[W](prime_time)	Power*Area	Latency [ns] (Time to finish one frame)
Input@0%sparsity	**1,569,046 um^2**	**0.1572 W**	**246,654**	**12,996.6ns**



Performance Comparison

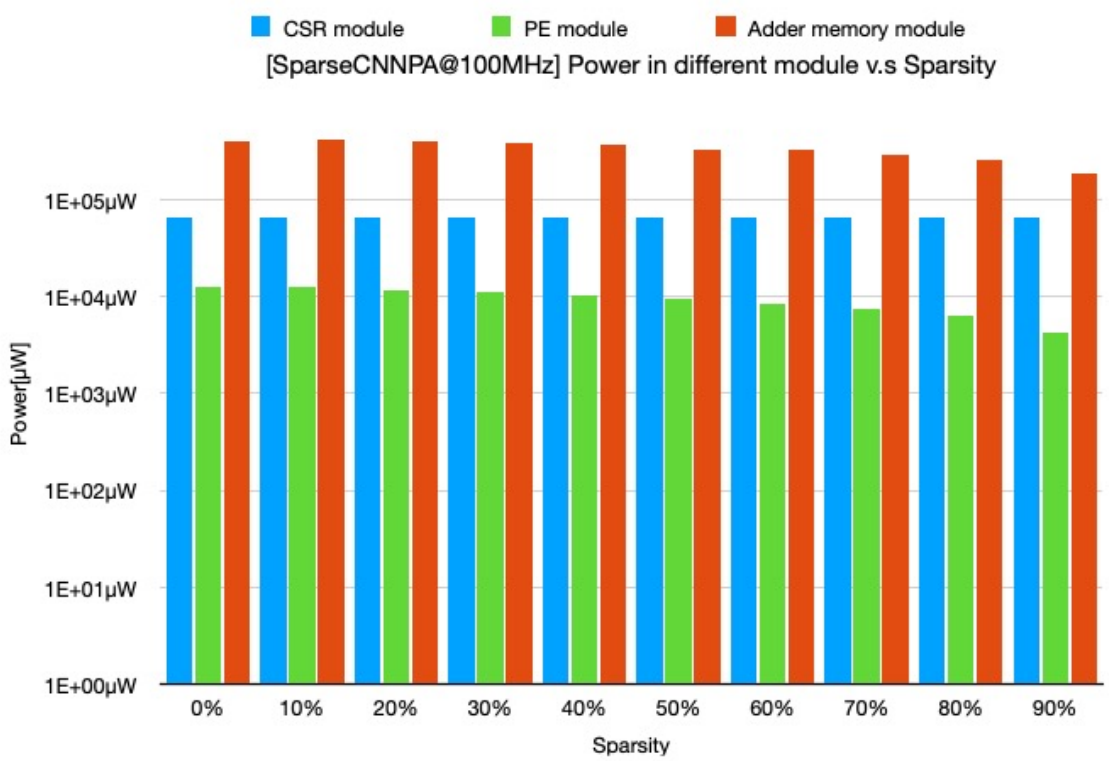
- **Difference between sub-modules in SparseCNN**

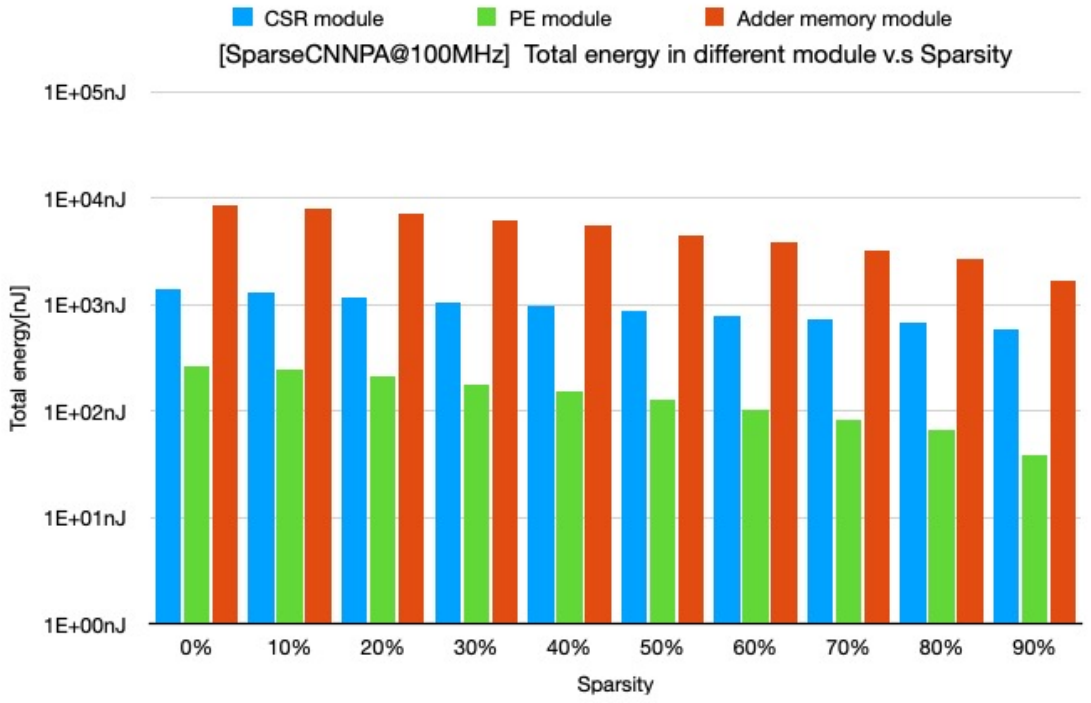




Performance Comparison

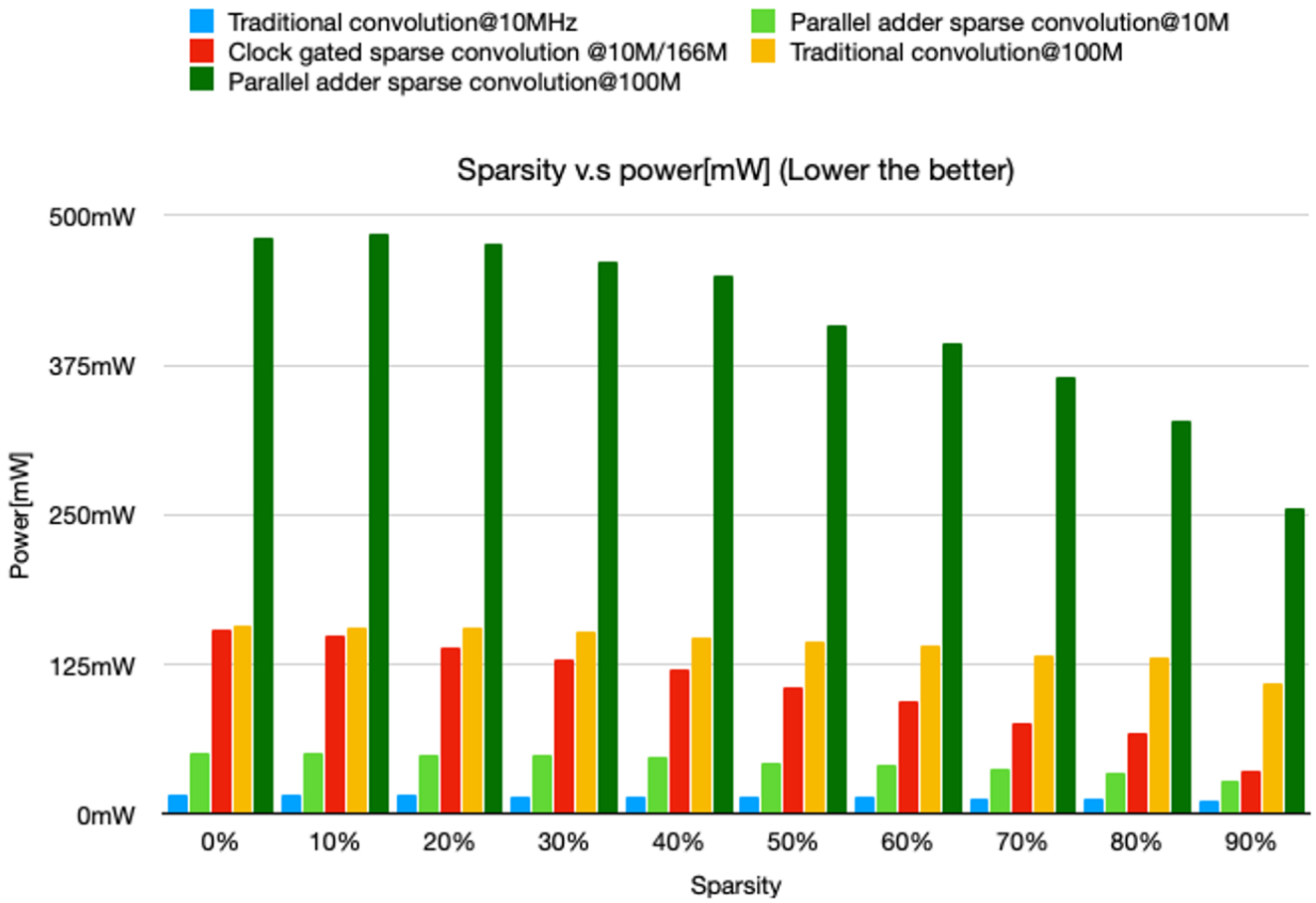
◆ Difference between sub-modules in SparseCNNPA





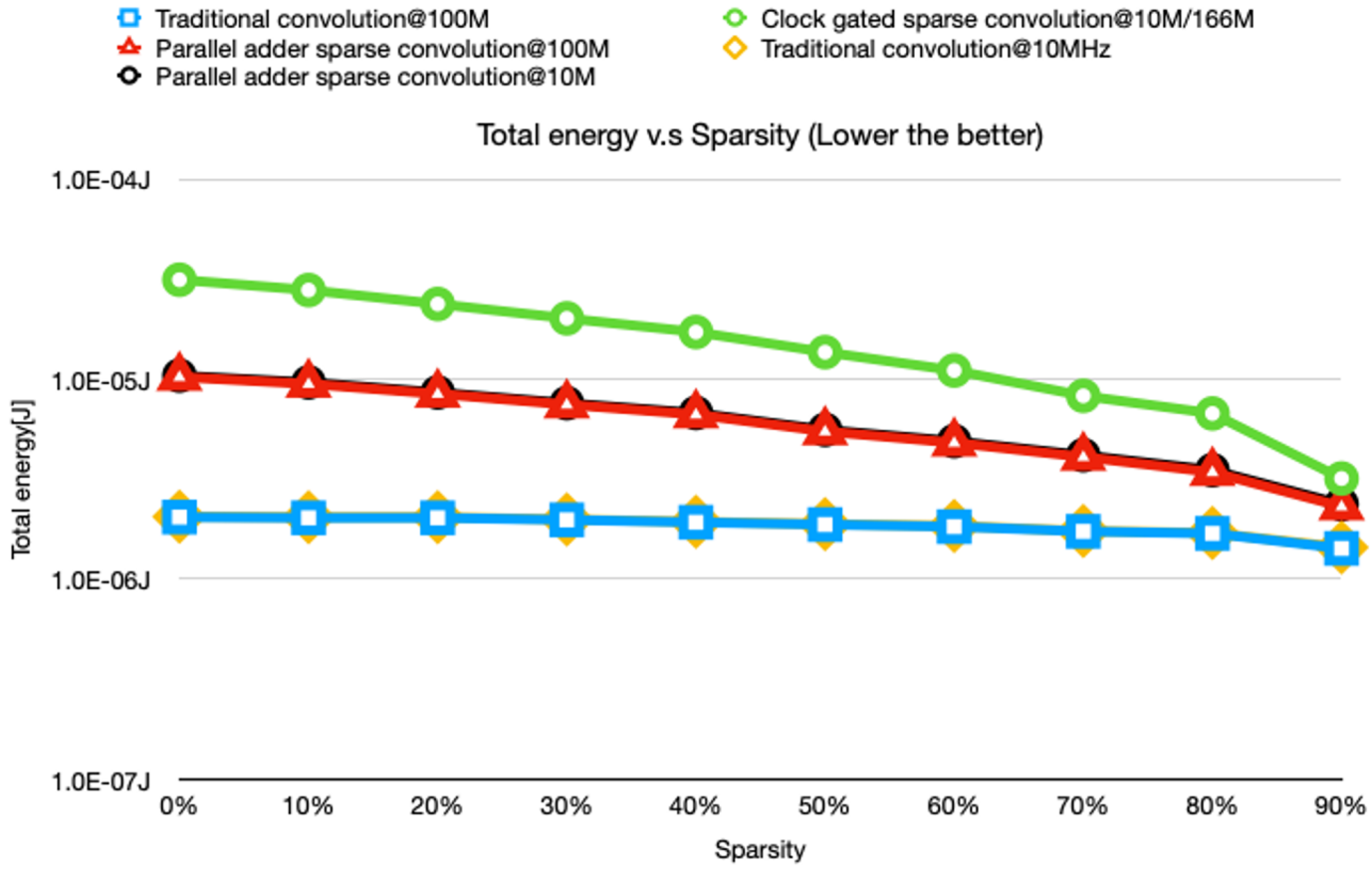
Performance Comparison

◆ Traditional convolution v.s Sparse convolution



Performance Comparison

◆ Traditional convolution v.s Sparse convolution



Performance Comparison

◆ Traditional convolution v.s Sparse convolution@0% sparsity

	Area[um^2]	Power[W](prime_time)	Power*Area	Latency [ns] (Time to finish one frame)	Total Energy [J]
Taditional convolution@0% sparsity	**1,569,046 um^2** 	**0.1572 W**	**244,614** 	**12,996.6ns** 	**2.04E-6** 
Sparse CNN@0%sparsity	**2,417,592 um^2**	**0.154 W** 	**372,309**	**203,955 ns**	**3.14E-5**
Sparse CNN parallel adder@0%sparsity	**21,267,287 um^2**	**0.482 W**	**10,250,832**	**21,265.3 ns**	**1.02E-5**

Performance Comparison

◆ Traditional convolution v.s Sparse convolution@90% sparsity

	Area[um^2]	Power[W](prime_time)	Power*Area	Latency [ns] (Time to finish one frame)	Total Energy [J]
Taditional convolution@90% sparsity	**1,569,046 um^2** 	**0.110 W**	**171,967**	**12,996.6ns**	**1.43E-06 J** 
Sparse CNN@90%sparsity	**2,417,592 um^2**	**0.0369 W** 	**89,209** 	**86355.3 ns**	**3.19E-06 J**
Sparse CNN parallel adder@90%sparsity	**21,267,287 um^2**	**0.256 W**	**5,444,425**	**9015.3 ns** 	**2.31E-06 J**

Summary and Conclusions

- In algorithm perspective, sparse convolution should save computing effort which is also true in hardware design. In hardware perspective, sparse convolution need more memory access than traditional convolution which results in large area and power/time consuming memory blocks.

- Sparse CNN needs extra memory to save CSR data and needs to deal with output memory access contention problem.

References

[1] F. Li, G. Li, Z. Mo, X. He and J. Cheng, "FSA: A Fine-Grained Systolic Accelerator for Sparse CNNs," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 39, no. 11, pp. 3589-3600, Nov. 2020, doi: 10.1109/TCAD.2020.3012212.