Think Fast: A Tensor Streaming Processor (TSP) for Accelerating Deep Learning Workloads

Yash Shah (ys562)

December 2024

Two key observations

$$\frac{\text{Time}}{\text{Program}} = \underbrace{\frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock Cycles}}{\text{Instruction}}}_{\text{ISA-dependent}} \times \underbrace{\frac{\text{Time}}{\text{Clock Cycles}}}_{\text{Process-dependent}}$$

1. Abundant data parallelism present in machine learning workloads
   — Exploit Single Instruction Multiple Data (SIMD)
2. Lots of control hardware costs power-efficiency, performance, and computational density
   — Limited control hardware required if non-deterministic components like tiered caches removed
   — Determinism allows pushing instruction scheduling complexities into compiler

Architecture Overview

Functional slices

- In a conventional multicore, each tile (core) is independent, with interconnect to transfer data between cores
- In TSP, each tile is a functional unit (FU), with vertically stacked tiles forming a slice
- Each slice performs a specific subset of the ISA in a SIMD fashion
- **Instruction fetch and decode logic separated into separate tile**

Parallel lanes achieve SIMD data parallelism [1]

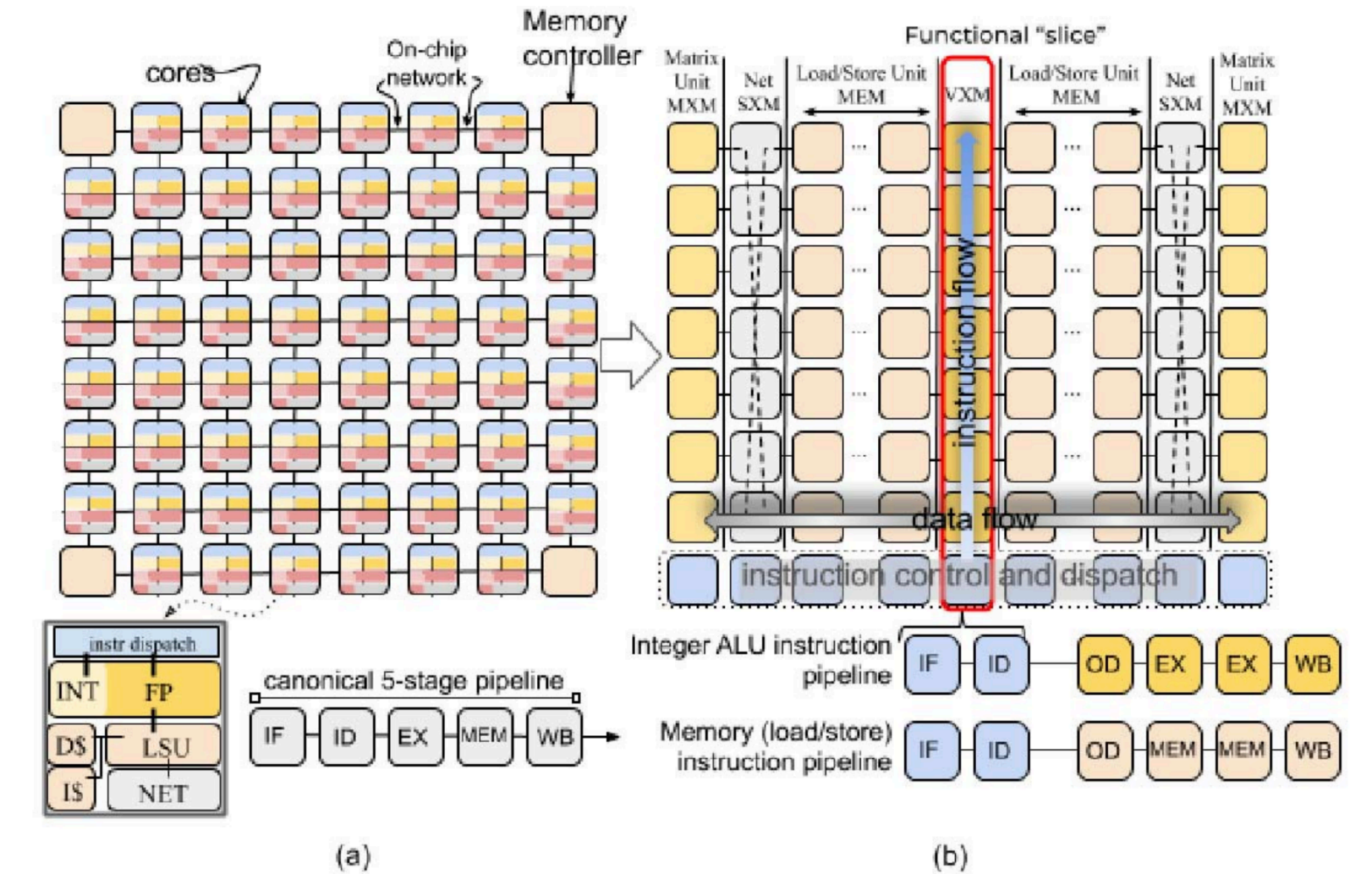


Fig. 1. Conventional 2D mesh of cores (a) reorganized into a functionally sliced arrangement of tiles (b).

Streams

- **Horizontally flowing streams carry operands, which intercept vertically travelling instructions on FUs, producing another stream(s) of outputs**
- 64 logical streams per lane (32 east and 32 west)
- Streaming processing model allows for chaining of operations through common stream register, without explicit write-back exploiting dataflow locality

Streams implemented by a streaming register file [1]

Fig. 4. Stream registers are numbered to show their locations between the functional slices within a superlane.

Instructions operate on streams instead of registers [1]

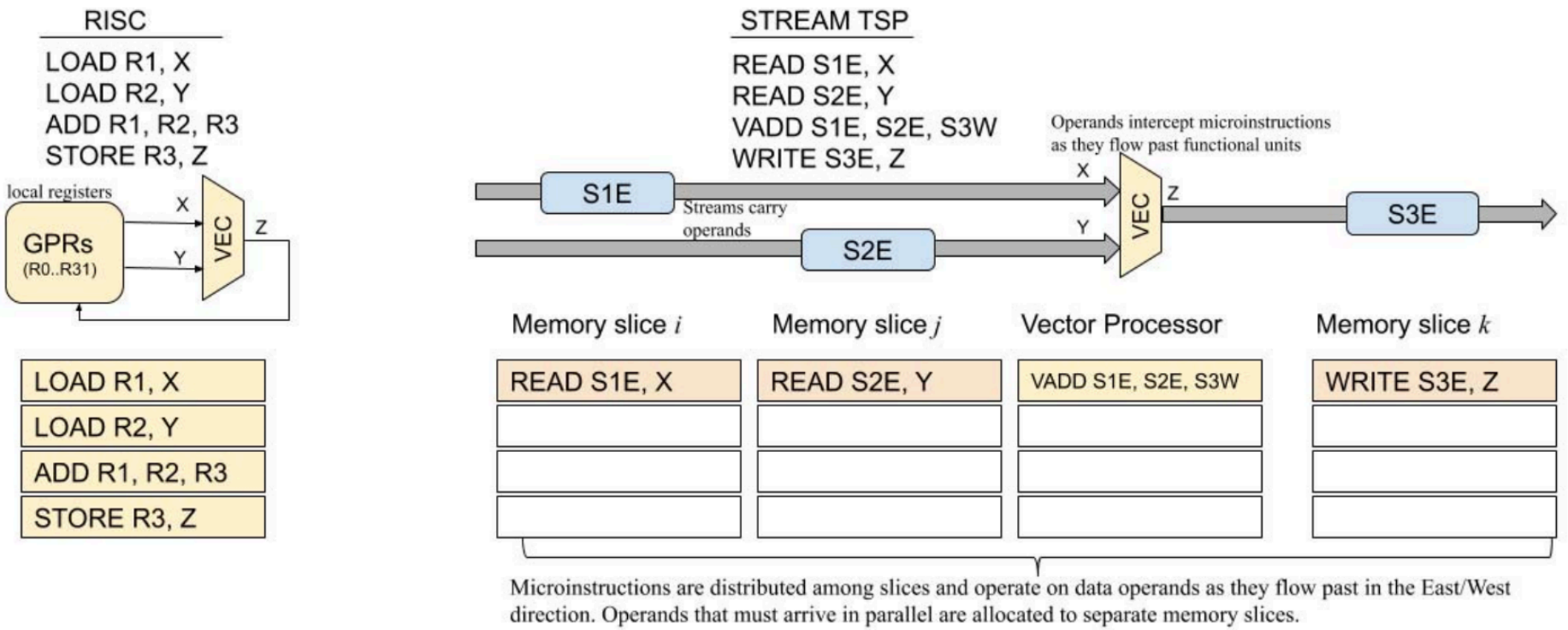


Fig. 3. Conventional RISC execution contrasted with producer-consumer streams in the TSP.

Staggered instruction execution [1]

- SIMD operation of up to 320-element vector is actually pipelined vertically across 20 superlanes in a slice
- Minimum vector length of 16 elements (otherwise masking required)
- MXM can fill all four arrays in < 40 cycles including SRAM and on-chip network transit delay.

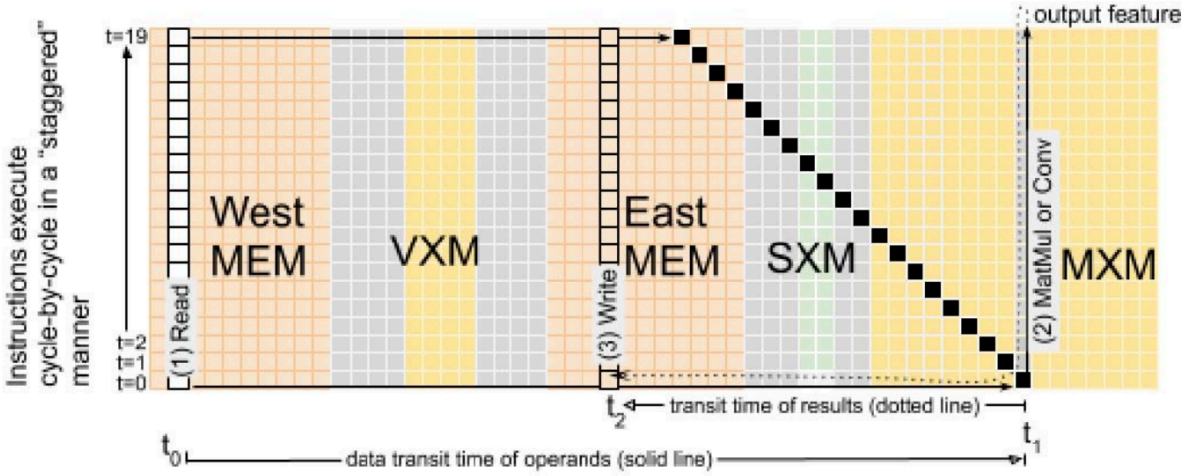


Fig. 6. Staggered instruction execution and dataflow within a superlane.

Error handling and reliability

- Error Correcting Code (ECC) used to protect data in SRAM and stream registers
- Single-error correction with double-error detection (SECDED) scheme used
- Soft Error Upsets (SEUs) automatically corrected and recorded in Control and Status Register (SR) to record early signs of chip-wearout

Programming model

Instruction set exposes temporal information

- Compiler has precise control over instruction dispatch time through determinism and knowing temporal information regarding execution latency
  - d_{func} is the functional delay (number of cycles) to produce stream output
  - d_{skew} is the instruction-operand skew exposing the timing relationship between instruction dispatch time and when operands are required

$$\underbrace{T}_{\text{Execution Time}} = \underbrace{N}_{\text{Tiles per Functional Slice}} + \underbrace{d_{\text{func}}}_{\text{Functional Delay}} + \underbrace{\delta(j,i)}_{\text{Stream Propagation Delay}}$$

- **Compiler solves 2D scheduling problem to intersect instructions with their operands**

Instruction Control Unit (ICU)

- `NOP` and `Repeat` for temporal separation between instructions
  - Clock enables turned off when `NOP`ing for more than a few cycles
- `Sync` and `Notify` for barrier synchronization once after chip reset (35 clock cycles), then synchronization-free operation due to determinism
- Explicit `Ifetch` instruction fetching with compiler performing omniscient prefetching

Other instructions

- Memory (MEM) offers pseudo-dual-ported `read` and `write`
  - Also `gather` and `scatter` for indirect addressing using contents of stream
- Vector (VXM) processor offers point-wise arithmetic operations e.g. to implement activation functions
- Matrix Execution Module (MXM) offers multiply-accumulate and convolution operations to implement neural-network layers
- Switch Execution Module (SXM) offers transposition, permutation, shifting, and rotation for tensor reshape operations

Determinism

On-chip determinism

— Large 220 MiB SRAM with single-cycle random access to any address eliminates non-determinism e.g. through tiered caches

— Instruction execution latency precisely known allowing for static reasoning

System-level determinism [2]

- Links are synchronous within some tolerance e.g. 50 (cycles) ppm
- Hardware and software aligned counters detect skew
- `Deskew` instruction compensates for global drift
- No hardware flow control (avoid deep queues and dynamic contention resolution)
- Forward Error Correction (FEC) and reply-only communication possible due to static semantics
- A partitioned global address space allows deterministic memory semantics even across TSP nodes in a system over a network
- **Allows to view every link in the system as a fixed-latency link**

Lessons from ResNet50 implementation

Lessons from ResNet50 implementation

- Explicit memory management
  - Compiler can schedule on memory bank-level (each MEM slice represents a bank) to maximize stream concurrency
  - `Read` from one bank and `write` to the opposite bank to exploit pseudo-dual-ported SRAM
  - Need to carefully allocate memory, interleaving banks, to avoid memory slice contention leading to pipeline latency bubbles
- Most expensive resources are MXM's 4 320x320 MACC arrays and MEM slices feeding them
  - Misalignment of 320x320 matrix multiply capacity and 256x256 dimensions of weights under-utilizes MXM

Results

Number of deep learning operations per transistor

- First generation TSP can perform 820 TeraOps/sec from 26.8B transistors, yielding **30K deep learning ops/sec/transistor**
- NVIDIA's V100 can perform 130 TeraFlops of mixed-precision arithmetic from 21.1B transistors, yielding **6.2K deep learning ops/sec/transistor**
- $\sim 5 \times$ improvement in computation density over GPU
- Nearly $4 \times$ speedup in batch-size-1 throughput and nearly $4 \times$ reduction in inference latency in comparison to TPU, GPU, and Habana Lab's GOYA [3] chip

ResNet50 performance

- $\sim 5 \times$ lower inference latency on single image compared to GOYA [3] ($49\mu s$ compared to $240\mu s$)
- $2.5 \times$ throughput compared to Google's TPUv3 large batch inference when each image sample is a separate query

Operating regimes [1]

- Initially memory-bandwidth limited, then arithmetically limited
- Also demonstrates that compiler is better able to allocate memory for tensors

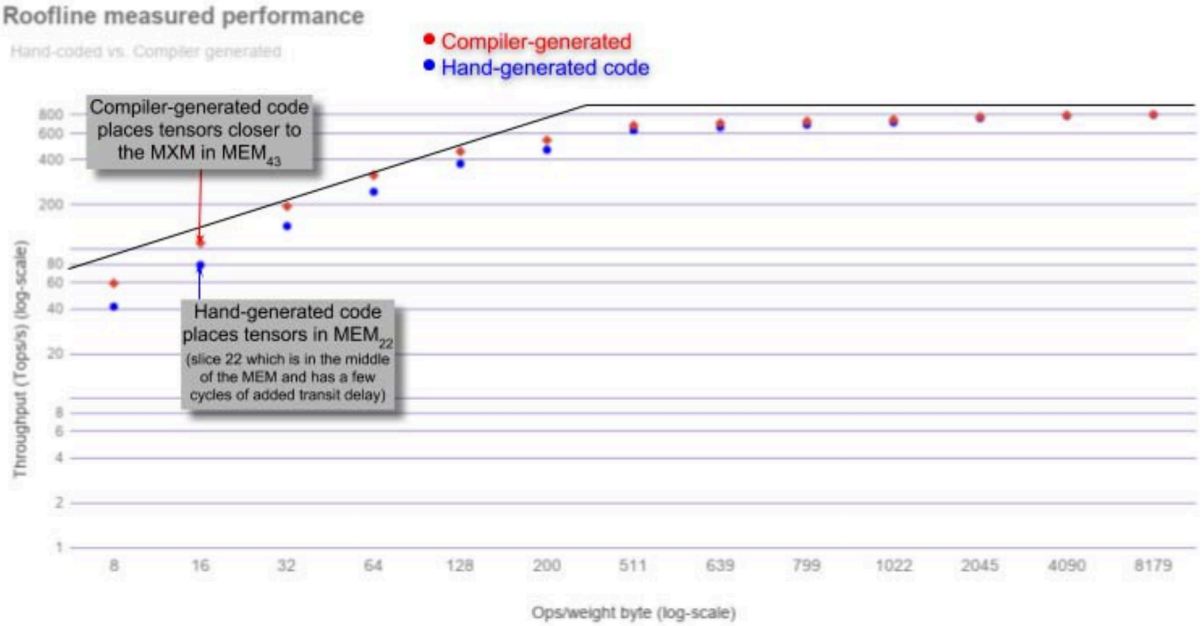


Fig. 9. Roofline diagram showing arithmetic throughput (at 1 GHz core clock) varying with offered load.

Critique and future development

Strengths

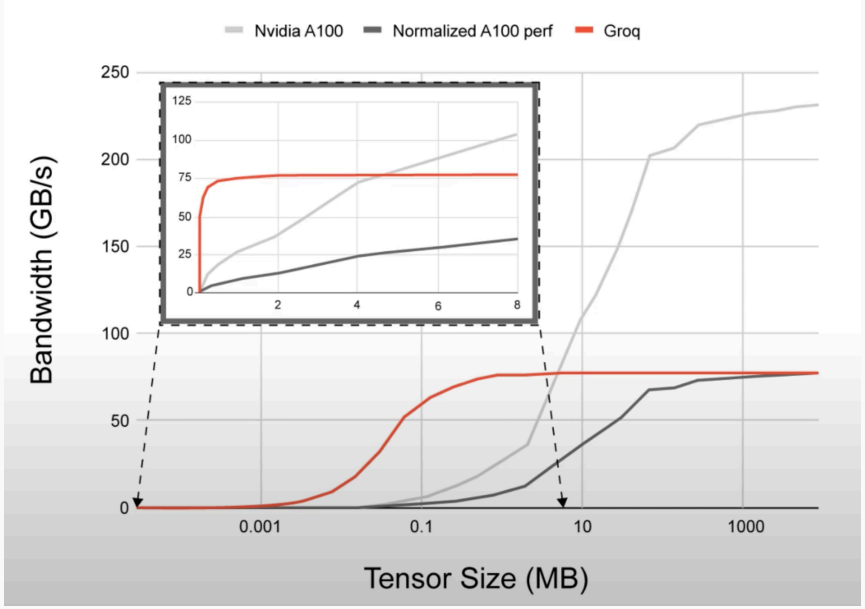
- Die area and transistors spent on control truly minimized (ICU < 3% of area)
- Amortization of instruction fetch and decode
- Streams are a natural programming abstraction for working with vector data
- Significant latency improvements over existing GPU/TPU/accelerator hardware great for real-time applications
- Deterministic design and drift-correction allows for novel software-based scheduling and networking removing need for time-consuming resynchronization
- Minimum vector length of 16 is not too large
- Large SRAM has adequate bandwidth and fixed latency characteristics to continuously feed FUs, minimizing idle time

Weaknesses

- 220 MiB SRAM may be limiting
- How much does low latency matter?
- Hard to use for training models

220 MiB SRAM may be limiting [2]

- Larger SRAM size can reduce likelihood of memory slice contention
- No commercial model can fit on a single Groq accelerator, or even a Groq node (collection of 8 accelerators)
- Inter-node traffic has significantly lower bandwidth [2] at 50 GB/s compared to 240 GB/s intra-node
- Network channel bandwidth constraint may impair TSP's performance



220 MiB SRAM may be limiting—Possible mitigation

- Increase amount of SRAM per accelerator
- Could utilize 3D-stacked SRAM like AMD's 3D V-Cache [4]
- This could even reduce power and latency by $\sim 30\%$ [5]
- Currently using 14nm node—switching to a more modern node can boost SRAM (and transistor) density, improving power consumption, but costs more

How much does low latency matter?

- TSP focuses on batch-1 performance to emphasize its lower latency
- GPUs with large VRAM buffers mean that at large batch sizes, fewer GPUs are required to host the model, while matching or exceeding TSP's throughput at the cost of latency
- However this lower latency does not benefit many real-time applications like self-driving cars as we need a rack to have enough SRAM to host the model

Hard to use for training models

- Backpropagation is a crucial part of training models as it is the method used to calculate gradients
- Limited SRAM means there is little space to store the computation graph for backpropagation
- Since the entire model cannot fit onto a single accelerator, the computation graph is also split and gradients must be accumulated across nodes
- However, TSP is not designed to support distributed graph-like computation structures
- This weakens the financial case for TSPs if they can only be used for inference

Conclusion

TSP, with its deterministic, software-defined architecture offers tremendous speed for batch-1 inference, but may not be a replacement for GPUs since it cannot be used for training, and it's low latency may not justify the cost for some applications

Bibliography

[1] D. Abts *et al.*, “Think Fast: A Tensor Streaming Processor (TSP) for Accelerating Deep Learning Workloads,” in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, 2020, pp. 145–158. doi: 10.1109/ISCA45697.2020.00023.

[2] D. Abts, “ISC 2020 ML Hardware Workshop.” [Online]. Available: https://www.youtube.com/watch?v=uJbTEXFAsLs

[3] E. Medina and E. Dagan, “Habana labs purpose-built ai inference and training processor architectures: Scaling ai training systems using standard ethernet with gaudi processor,” *IEEE Micro*, vol. 40, no. 2, pp. 17–24, 2020.

[4] R. Agarwal *et al.*, “3D packaging for heterogeneous integration,” in *2022 IEEE 72nd Electronic Components and Technology Conference (ECTC)*, 2022, pp. 1103–1107.

[5] C.-L. Hsu and C.-F. Wu, “High-performance 3D-SRAM architecture design,” in *2010 IEEE Asia Pacific Conference on Circuits and Systems*, 2010, pp. 907–910.