Increasing Energy Efficiency of GPUs Through Hardware Resource Partitioning and Masking

Mika Shanela Carodan, Marcus Chow, Kiran Ranganath, Dr. Daniel Wong
Department of Computer Science and Engineering, University of California - Riverside

Abstract

Known for its high computational efficiency, graphics processing units (GPUs) are becoming increasingly prominent as today's leading accelerator for applications that require parallel processing of massively complex workloads. However, its limited power maintenance and overall energy inefficiencies has been a restraining factor in efficiently sustaining the resource demands from parallel programs that all compete for the high performance capabilities of GPUs. Thus, it becomes imperative to question the necessity of sharing all of the gpu resources versus meagerly sharing certain parts of the hardware as a possible solution to the lack of the existing power saving mechanisms in GPUs. In an attempt to optimize these aforementioned power management issues, a feature in AMD's HIP programming, called CU Masking, is exploited in order to assess the reliance of manipulating hardware allocation policies in procuring a more sustainable and energy-efficient parallel computing system. Firstly, A mapping of the CU Mask Vector bits to SMid was collected as a reference to assist with manual activation of each individual hardware resource (CU) one at a time. After learning how the individual CUs can be set on its own, power and energy profilings were conducted to evaluate the overall sustainability of the program as active resources were scaled. In consequence of these experiments, it was found that less than half of the hardware resources in a shader engine could be maximally activated before power and energy becomes negatively impacted. From this information, we can optimistically progress towards an energy-efficient GPU by developing an optimized runtime modeling system that can direct power-saving allocation policies and workload manipulation -- which are models that can be further extended to other parallel computing systems and enforce the paradigm of hardware sustainability.

I. Background

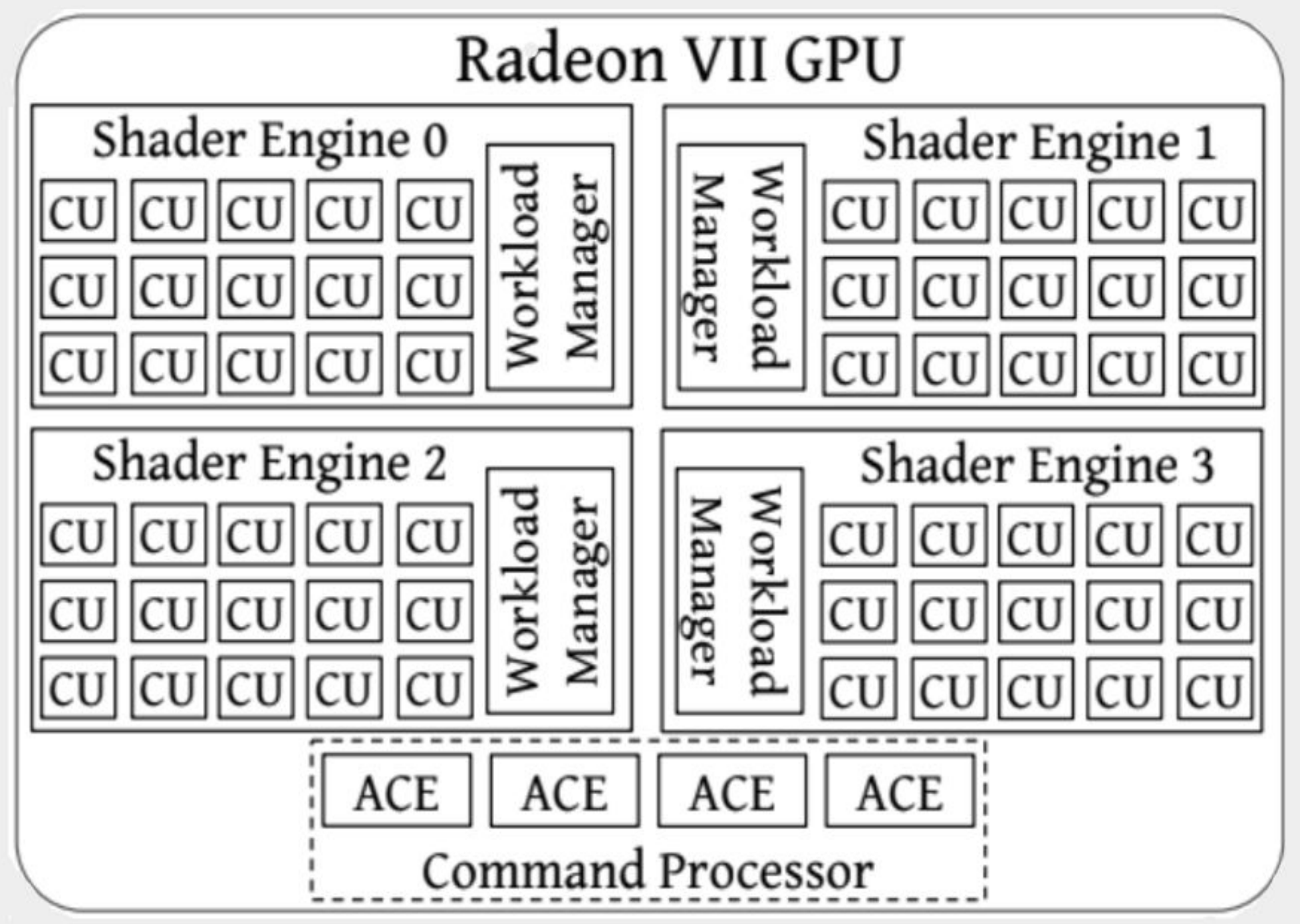


Figure 1: The Radeon VII's compute-related components

GPUs in *High Performance Computing* Applications

The skyrocketing demands for high computing power has increased the reliance of GPUs and their parallel nature to accelerate workloads in HPC applications that require the execution of multiple parallel programs. However, GPUs aren't optimized for power management in a resource-sharing environment where different applications are all competing for the same hardware resources.

- **Compute Units (CU)**
  The GPU's hardware resources; each block of CU contains a core. In total, there are 60 CUs that represent 60 GPU cores.
- **Shader Engine (SE)**
  Shelters 15 CUs per SE; dedicated units for different types of operations in the rendering pipeline of information processing.

II. Resource Partitioning Through CU Masking

Figure 2a: *CU Mask-SM_id Bit Mappings: Profiling individual CUs to its corresponding active CU Identification*

CU Mask Bit	SM_ID	CU Mask Bit	SM_ID	CU Mask Bit	SM_ID	CU Mask Bit	SM_ID	CU Mask Bit	SM_ID	CU Mask Bit	SM_ID
1	1	11	34	21	6	31	39	41	11	51	45
2	17	12	51	22	22	32	56	42	27	52	61
3	32	13	4	23	37	33	9	43	43	53	14
4	49	14	20	24	54	34	25	44	59	54	30
5	2	15	35	25	7	35	40	45	12	55	46
6	18	16	52	26	23	36	57	46	28	56	62
7	33	17	5	27	38	37	10	47	44	57	15
8	50	18	21	28	55	38	26	48	60	58	31
9	3	19	36	29	8	39	41	49	13	59	47
10	19	20	53	30	24	40	58	50	29	60	63

CU Masking: Partitioning desired hardware resources (CU) that parallel applications will use during its runtime execution.

- The CU Mask Vector holds 60 bit positions that corresponds to the 60 CUs in the GPU. Each bit position in the vector is a "key" to activate each block of GPU core (CU).
- Knowing the mapping of CU Mask vector bits to its SMid (CU identification) counterpart allows for direct activation of individual CUs and SEs, which in turn manipulates workload partitioning in the hardware.
- With the CU Masking technique, specifying masks for a fixed number of CUs dictates the GPU resources that will be expended to execute the required computations needed to accommodate competing parallel programs. The specified number of activated CUs could have consequences on power efficiency and energy consumption.

Are CU Masking techniques a viable solutions in managing the GPU's power efficiency?

Figure 2b: GPU Representation of CU Mask-SMid Bit Mappings

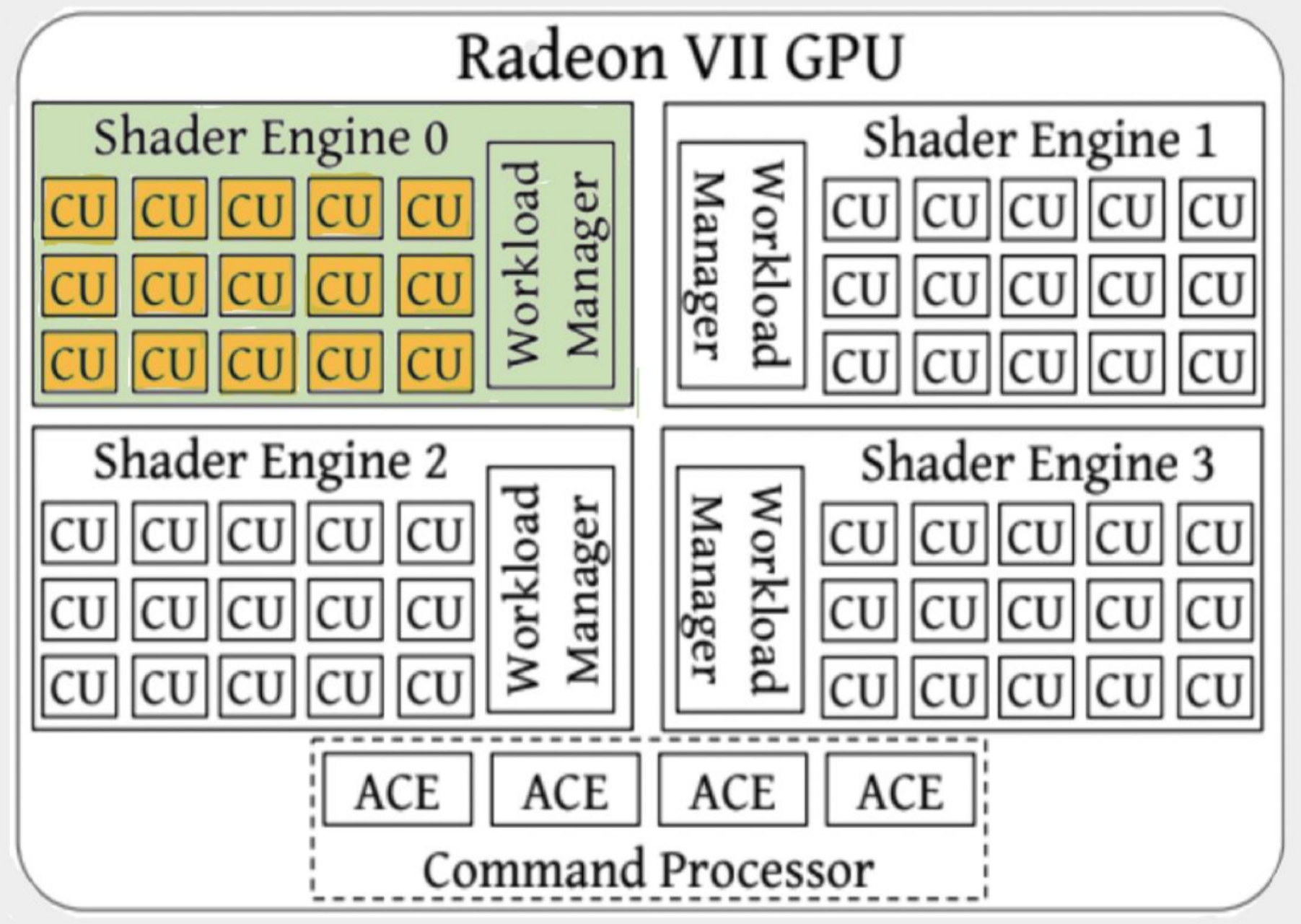
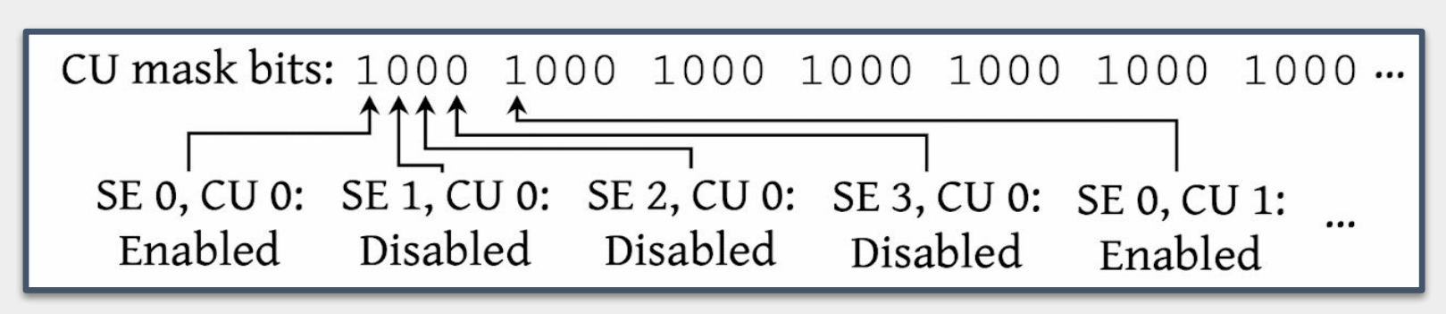


Figure 2c: Mapping of CU Mask bits to SE



III. Power and Energy Characterization

- **Power Characterization:** Measures the power usage (watts) of parallel programs as the number of active CUs are restricted. Consumption is assessed by conducting tests on the average power exhausted from incrementally setting all 60 CUs and 4 SEs.
- **Energy Characterization:** Measures the overall energy usage (joules) of the program by multiplying the average power (watts) and the total execution time (seconds) of each activated CUs and SEs.

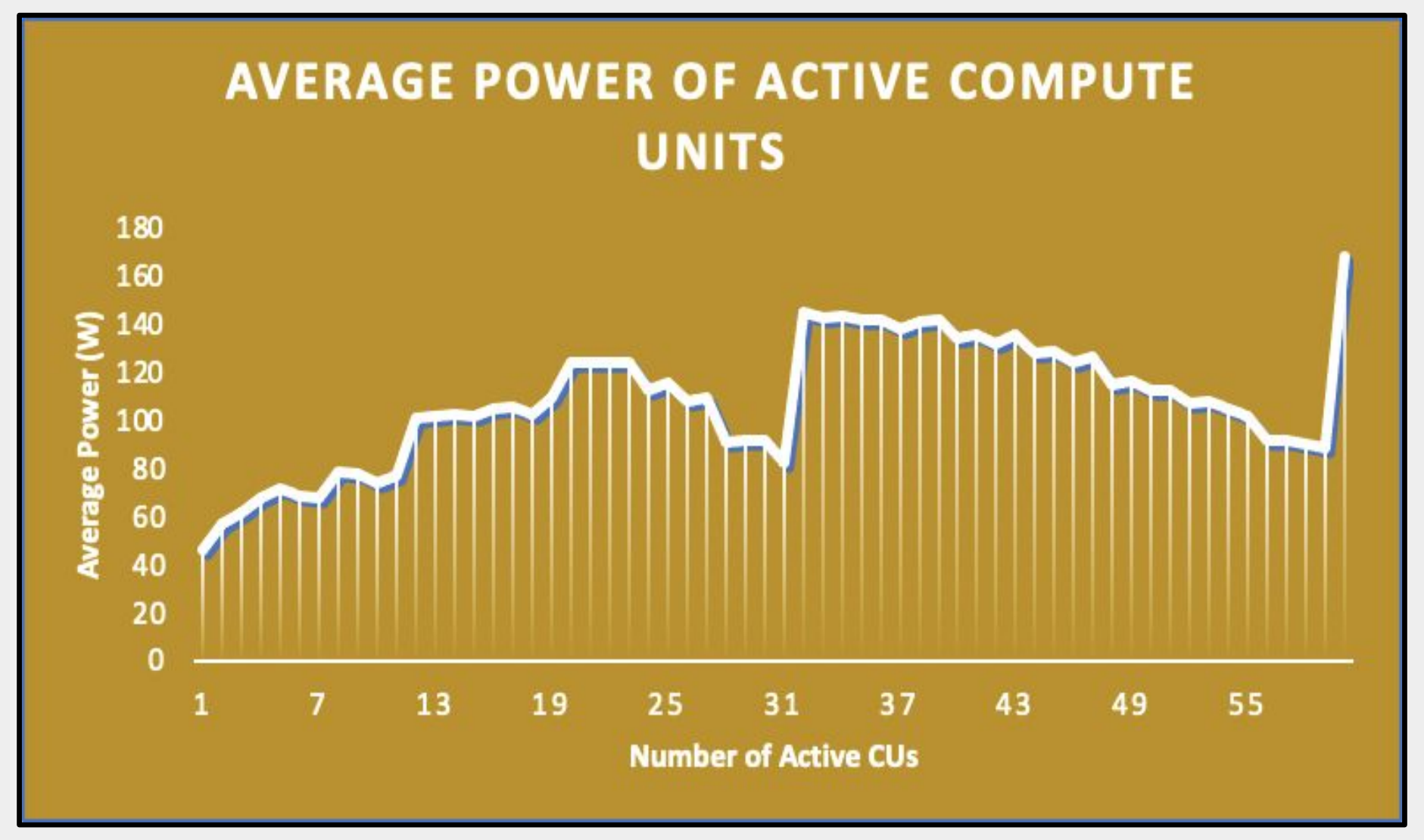


Figure 3a: Average Power of Active Compute Unite

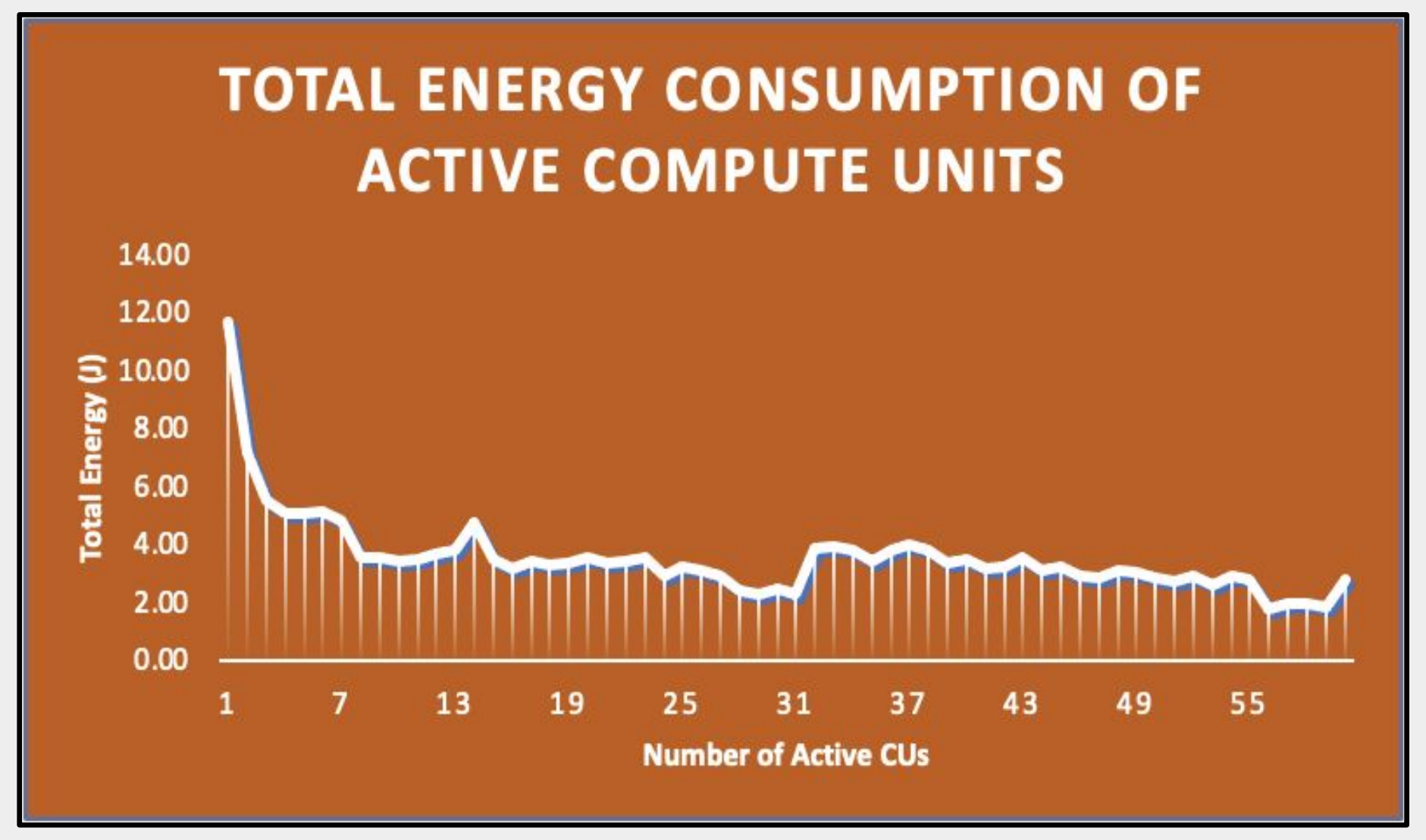


Figure 3b: Energy Consumption of Active Compute Unite

While the GPU's power usage seemingly increases in relation to the rising number of active CUs, the notable "dips" in its trendline depicts the feasibility of partitioning considerable resources that the GPU is willing to share while still preserving hardware sustainability. In particular, the average power and energy consumption reaching its lowest marks at 31 and 59 CUs are the ideal numbers for specifying the necessary CUs to employ during runtime since it most optimizes the GPU's expended energy (*Figure 3a* and *Figure 3b*). This observed pattern can also be seen when dissecting the consumption beyond the internal round-robin style activation of CUs within GPUs. Looking closely at the sequential setting of CUs in the order of SE, there's a sizeable power jump after moving from one SE to another (*Figures 3c* and *3d*) -- suggesting that activating less than half of the CUs in the SE is most desirable, while using more than recommended will suboptimally inflate the power and energy usage.

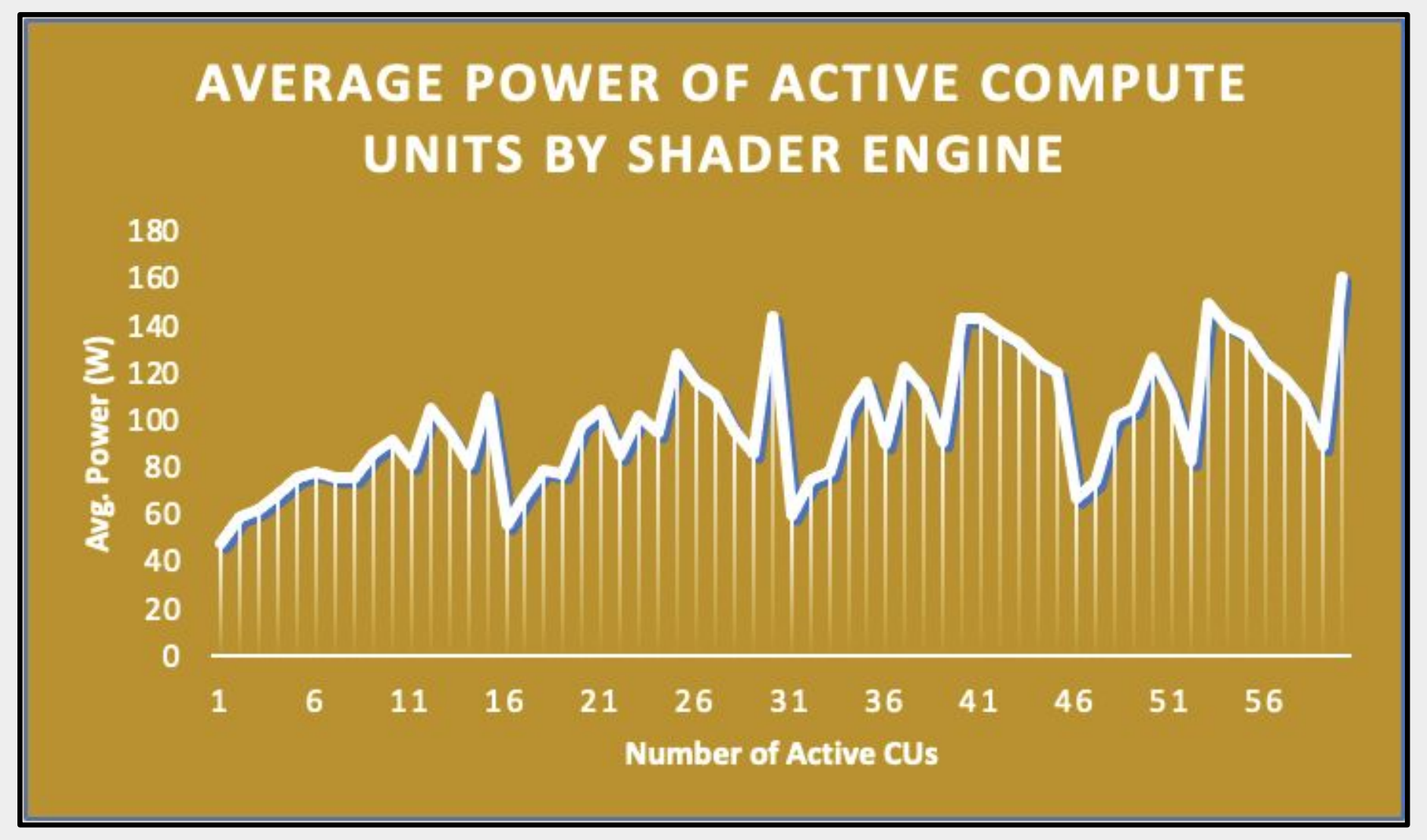


Figure 3c: Average Power of Active Compute Unite by Shader Engine

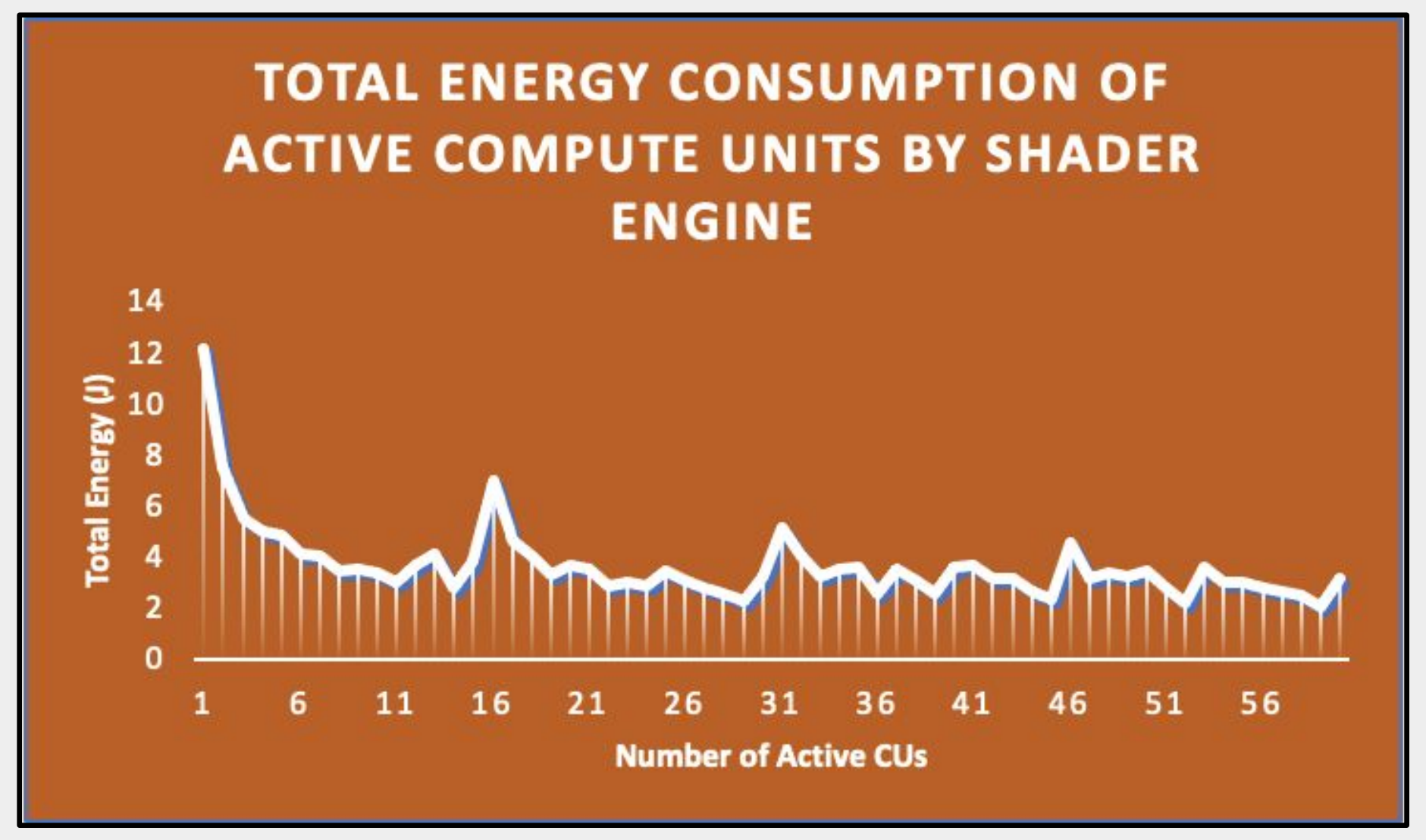


Figure 3d: Energy Consumption of Active Compute Unite by Shader Engine

References and Acknowledgements

Anderson, James H. Otterness, Nathan. Exploring AMD GPU Scheduling Details by Experimenting With "Worst Practices". *International Conference on Real-Time Networks and Systems (RTNS).*

Special gratitude towards NSF, MacREU, UCR, the SOCAL Research Lab, and most deservingly to Iva Petrovic and the 2021 MacREU Director of Experience, Marissa Moreno

MacREU R'side '21

LinkedIn