Einfochips | The Solutions People

Next-Generation Streaming Multiprocessor (SMX) and Function Units in Kepler Architecture

Application Note

Contents

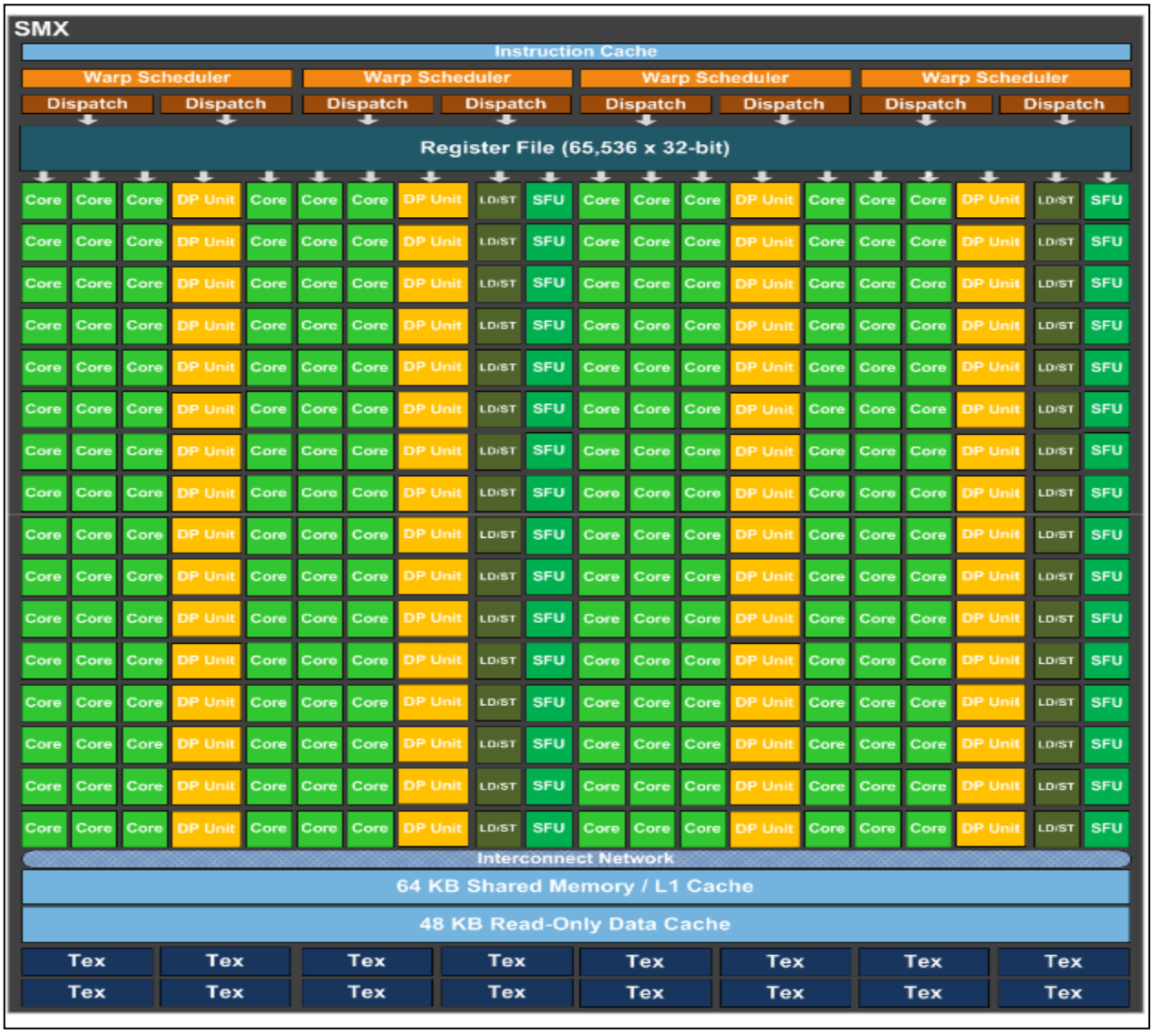
About Kepler GK110 architecture – an introduction

The Kepler GK110 is one of the innovative and complex microprocessors ever built by NVIDIA, consisting over 7 billion transistors. The microprocessor is designed specially to fulfill the needs of the emerging HPC market. Besides, it delivers more compute performance than its Fermi counterpart.

The New SMX and the Function Units

The GK110 architecture now features a new Next-generation Streaming Multiprocessor Architecture called SMX.

SMX delivers more processing performance (delivering up to 3x the performance per watt of Fermi). The number of cores has been increased to 192, which provides more space for computation. The figure below shows a detailed diagrammatic representation of SMX.

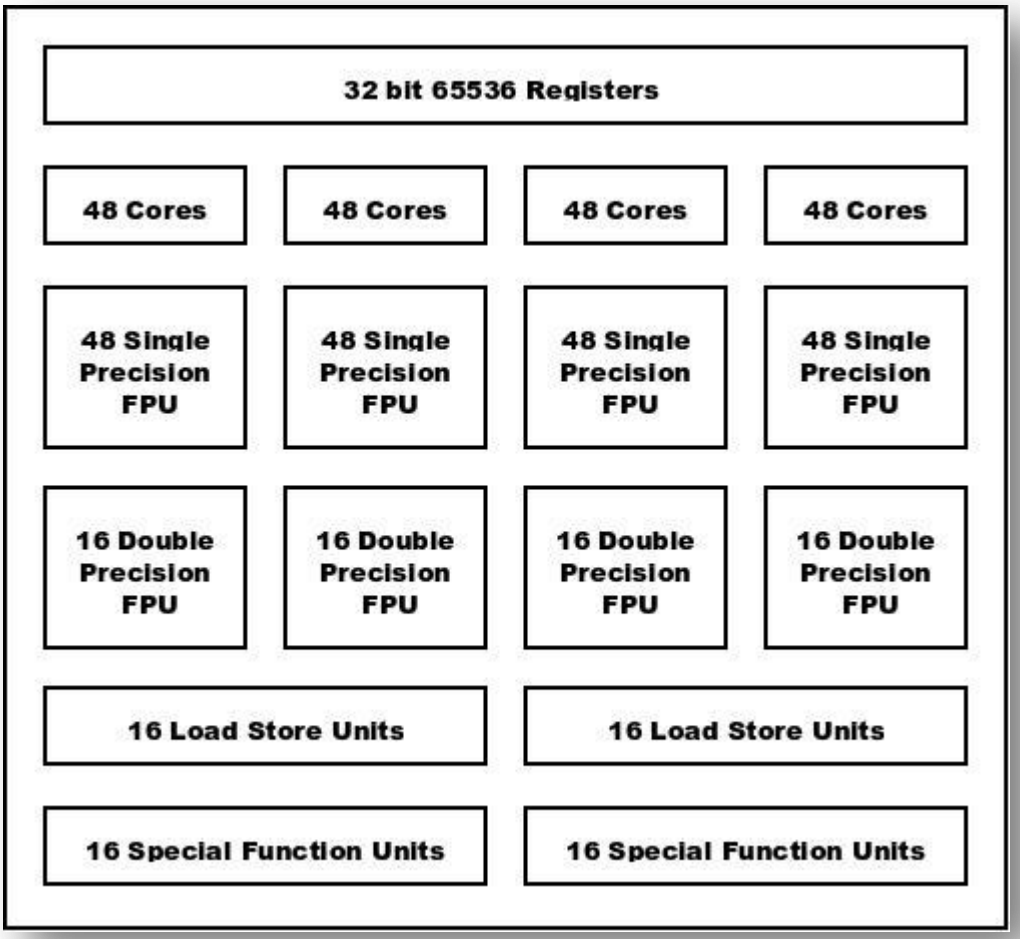


The front end consists of Instruction Scheduling units viz. Instruction cache, Warp scheduler and Dispatch units. Compared to the Fermi architecture, there are 4 more Warp schedulers. Each warp

scheduler has 2 Dispatch units. In Kepler, each SMX can issue 8 warp-instructions per cycle, but due to limitations imposed by resources and dependencies

- 7 is the sustainable peak.
- 4-5 is a good amount for instruction-limited codes.
- <4 in memory- or latency-bound codes.

The Back end consists of Instruction Execution units, which consists of 65536 32-bit registers and 512 Functional units. They are dedicated as displayed in the diagram below:

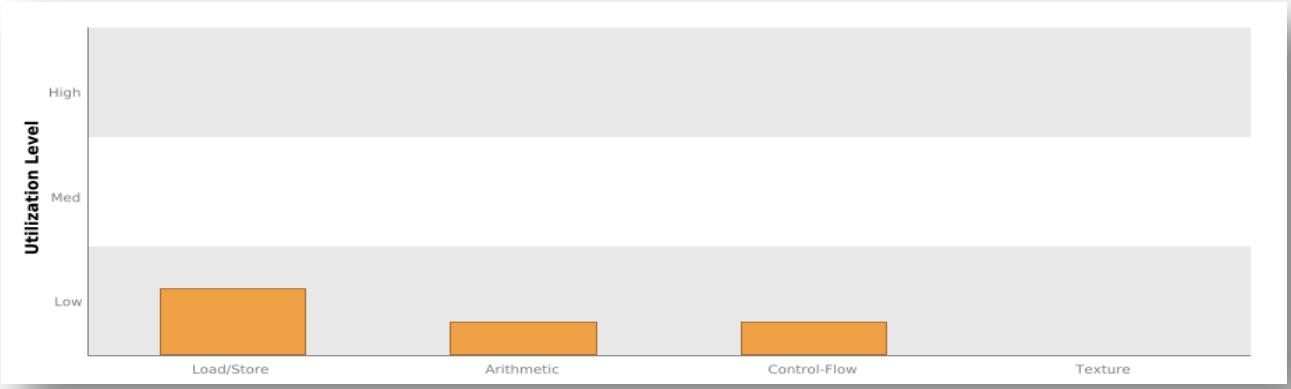


The NVIDIA Visual Profiler Tool

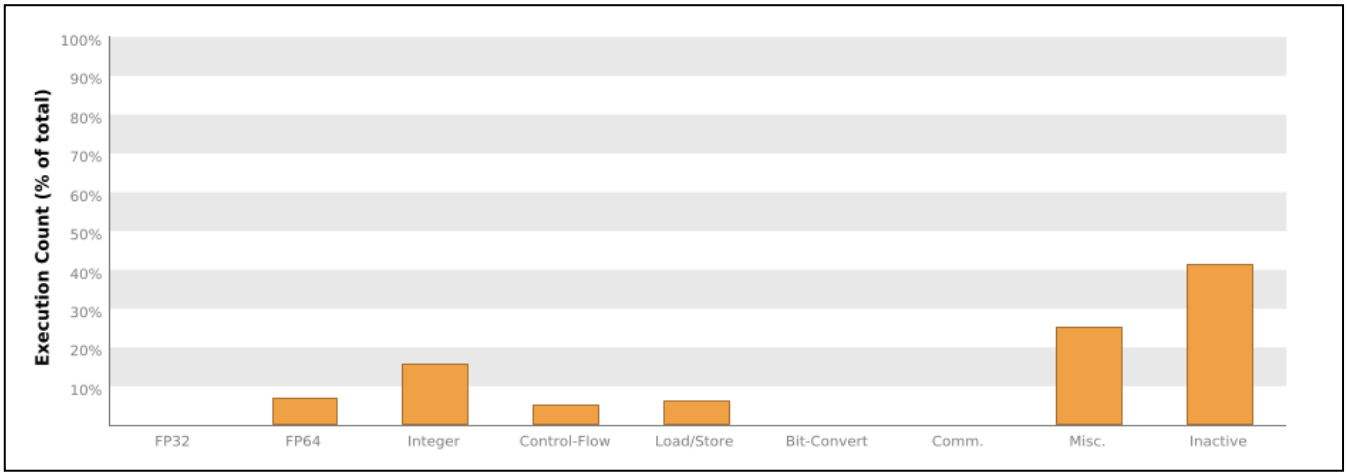
The NVIDIA Visual Profiler tool helps the user to know how a CUDA Application can utilize functional units.

As different instructions are executed on different function units within each SMX, the computed analysis of the kernel shows the function unit, which is utilized.

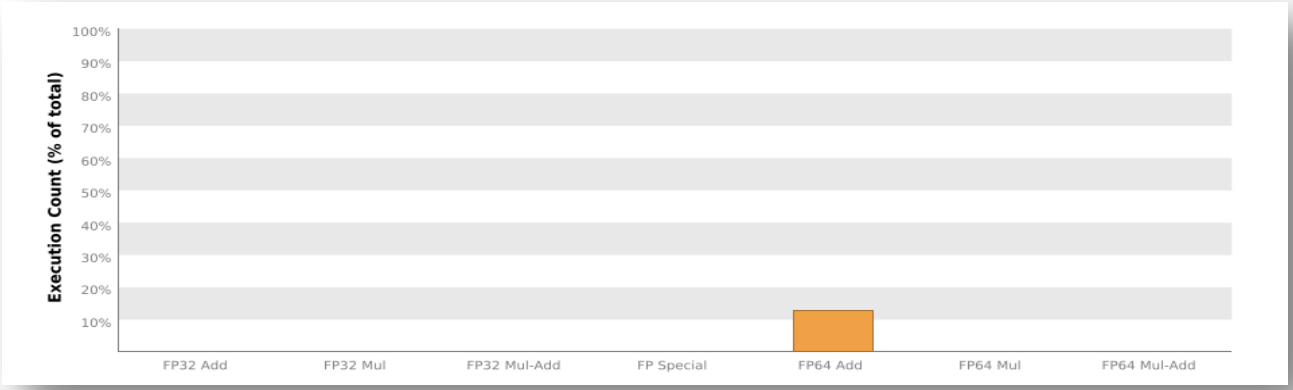
As shown in the chart, the Load/Store Unit is used for local, shared, global, constant memory. Arithmetic Unit is used for arithmetic instructions including integer and floating-point add and multiply, logical and binary operations, etc. Control-Flow Units is utilized for direct and indirect branches, jumps, and calls while the Texture Unit is used for texture operations.



The following chart shows the comparison of instructions executed by the kernel. The instructions are grouped into classes and for each class, the chart shows the percentage of thread execution cycles that were devoted to executing instructions in that class.

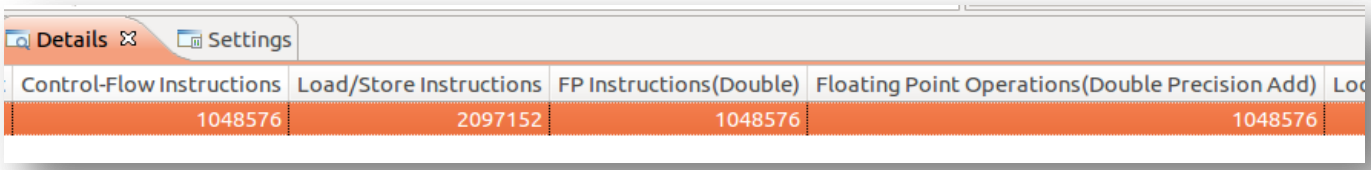


The following chart shows the floating-point operations executed by the kernel. The operations are grouped into classes and for each class the chart shows the percentage of thread execution cycles



that were devoted to executing operations in that class.

The number of floating point instructions can be viewed in the “detail” view during the analysis of the application in NVIDIA Visual Profiler.



Here is the sample code used for the analysis of the Functional Units and its utilization.

```
#include <stdio.h>
#include <cuda.h>

#define MY_VAR double

__global__ void my_kernel1(MY_VAR *d_a) {
	int idx = threadIdx.x + blockDim.x * blockIdx.x;
	d_a[idx] = d_a[idx] + 1;
}

__global__ void my_kernel2(MY_VAR *d_a) {
	int idx = threadIdx.x + blockDim.x * blockIdx.x;
	d_a[idx] = d_a[idx] + 1;
}

int main() {
	cudaError_t err;
	MY_VAR *d_a, *h_a;
	int SIZE = 1024 * 1024;
	h_a = (MY_VAR *)malloc(sizeof(MY_VAR) * SIZE);		// Host Memory Allocation

	if(h_a == NULL) {
```

```
        printf("Error in malloc \n");
        exit(1);
    }

    err = cudaMalloc((void **)&d_a, sizeof(MY_VAR) * SIZE);// Device Memory Allocation
    if(err != cudaSuccess) {
        printf("Error in cudaMalloc: d_a \n");
        exit(1);
    }

        for(int i=0; i<(SIZE); i++)
        h_a[i] = 1;

    err = cudaMemcpy(d_a, h_a, sizeof(MY_VAR) * SIZE, cudaMemcpyHostToDevice);
        // Host to Device Memcpy
    if(err != cudaSuccess) {
        printf("Error in cudaMemcpy : Host To Device\n");
        exit(1);
    }

    if(sizeof(MY_VAR) == 4)
        my_kernel1<<<1024, 1024>>>(d_a);          // Kernel call

    if(sizeof(MY_VAR) == 8)
        my_kernel2<<<1024, 1024>>>(d_a);          // Kernel call

    err = cudaMemcpy(h_a, d_a, sizeof(MY_VAR) * SIZE, cudaMemcpyDeviceToHost);
        // Device to Host Memcpy
    if(err != cudaSuccess) {
        printf("Error in cudaMemcpy : Device to Host\n");
        exit(1);
    }

    cudaFree(d_a);
    free(h_a);

    return 0;
}
```

The GFLOP/S for the operation of Single Precision Instructions is found to be double than that of GFLOP/S for the Double Precision Instructions (Maximum being 3.522 TeraFLOP/s for Single Precision and 1.174 TeraFLOP/s for Double Precision for the Tesla K20c device).

Conclusion

The new SMX has increased the capacity of instruction execution by increasing the number of cores and function units. The Single Precision FPUs are more as compared to the Double Precision FPUs that allows the peak bandwidth to perform much better than the latter. It now depends on the user to choose between performance speed or the accuracy of the result.

About eInfochips

eInfochips is a speciality OEM partner to NVIDIA, to develop client solutions based on various GPU platforms for applications in Aerospace, Defense, Medical Imaging, Automotive, Industrial and High Performance Computing (HPC).

eInfochips dedicated team of CUDA programmers, is trained on the latest parallel programming techniques. The design team is updated on the new features available on NVIDIA Tesla™, Tegra™ and Quadro™ platforms. They also have direct access to NVIDIA product teams for support.

eInfochips expertise in reference board designs, image processing, camera solutions, system software, computer vision accelerate design and development of client products based on NVIDIA GPUs. In order to achieve the above, eInfochips Design team will collaborate with NVIDIA technical team to get firsthand experience on latest Tegra™ and Tesla™.

Author of the Article

Lalit Chandivade, Sr. Technical Lead, Embedded Systems, eInfochips