Matrix Multiplication with Data Level Parallelism in Simulink

Submitted by,

Laya Harwin
CB.EN.P2EBS17014

1. Abstract:

The objective of my project is to implement data level parallelism. Data level parallelism is also known as loop level parallelism is a form of parallel computing for multiple processors by distributing the data across different parallel processor nodes. The main aim of my project is to compare the execution time of data level parallelism to sequential execution of the tasks of a system. MATLAB Simulink is the tool used to implement this. This can be used for multi core systems where speed of execution, time of execution and performance needs to be calculated and improved.

2. Introduction:

Multicore refers to an architecture in which a single physical processor tries to incorporates the core logic of more than one processor. A single integrated circuit is generally used to package or hold these processors which are known as die. Multicore architecture places multiple processor cores and combines them as a single physical processor. The objective is to create a system that can complete more tasks at the same time and better speed, thereby gaining better overall system performance.

The concept of multicore technology is mainly focusing on the possibility of parallel computing, which can significantly boost computer speed and efficiency by including two or more central processing units (CPUs) in a single chip. This can reduce the system's heat and power consumption. This means much better performance with less or the same amount of energy and time.

The architecture of a multicore processor helps in communication between all available cores to ensure that the processing tasks are divided and assigned accurately and correctly. At the time of task completion, the processed data from each core is delivered back to the required destination. This technique significantly improves the performance compared to a single-core processor of similar speed.

Multicore technology is very effective and successful in challenging tasks and applications, such as encoding, 3-D gaming and video editing.

Data level parallelism is a form of parallelism across multiple processors in parallel computing environment. It focuses on distributing the data across different nodes, which operate on the data in parallel and execute parallelly. It is applied on regular data structures like arrays and matrices by working on each element parallelly. In a multiprocessor system executing a single set of instructions (SIMD), data parallelism is achieved when each processor performs the same task on different pieces of data. In some situations, a single execution thread controls operation on all the pieces of data across different nodes. In others, different threads control the operation, but they execute the same code with different data.

3. Literature Survey:

In the past few years, many papers have been published based on data level parallelism in multicore processors.

In the paper written by Tim Schmidt, Guantao Liu, and Rainer Dömer, they are trying to focus on the major limitations of most parallel SystemC. They have also tried to exploit opportunities for data-level parallelization. There are two factors N and M representing thread and data-level. Speedup of NxM has been nearly linear according to the experimental results. Here, a 4-core multi-processor achieves a speedup of up to 8.8x, and a 60-core Xeon Phi processor reaches up to 212x according to the observation.

In the paper written by Jurgen Kadidlo and Alfred Strey, they present a new method for a high-performance thread- and data-parallel computation of normalized cross correlation in the spatial domain. They showed that computing the two-dimensional cross correlation in spatial domain can be as fast or even faster than in the frequency domain. By exploiting the possibilities of the SSE2 instruction set and optimizing the algorithms their data-parallel implementation achieved a speedup of 4.7 on a single CPU core. The performance was can easily

be further improved through thread parallel computation on multi-core CPUs or other parallel high-performance computing systems.

In this paper written by Upul Senanayake, Rahal Prabuddha and Roshan Ragel, they presented how to utilize a multi-core environment to introduce Data Level Parallelism (DLP) to the Auto Dock Vina software. It is a widely used in molecular docking software. Auto dock Vina exploits Instruction Level Parallelism (ILP) already in a multi-core environment and hence it is optimized for such environments. However, with their results it can be clearly understood that their approach has enhanced the throughput.

4. Block Diagram:

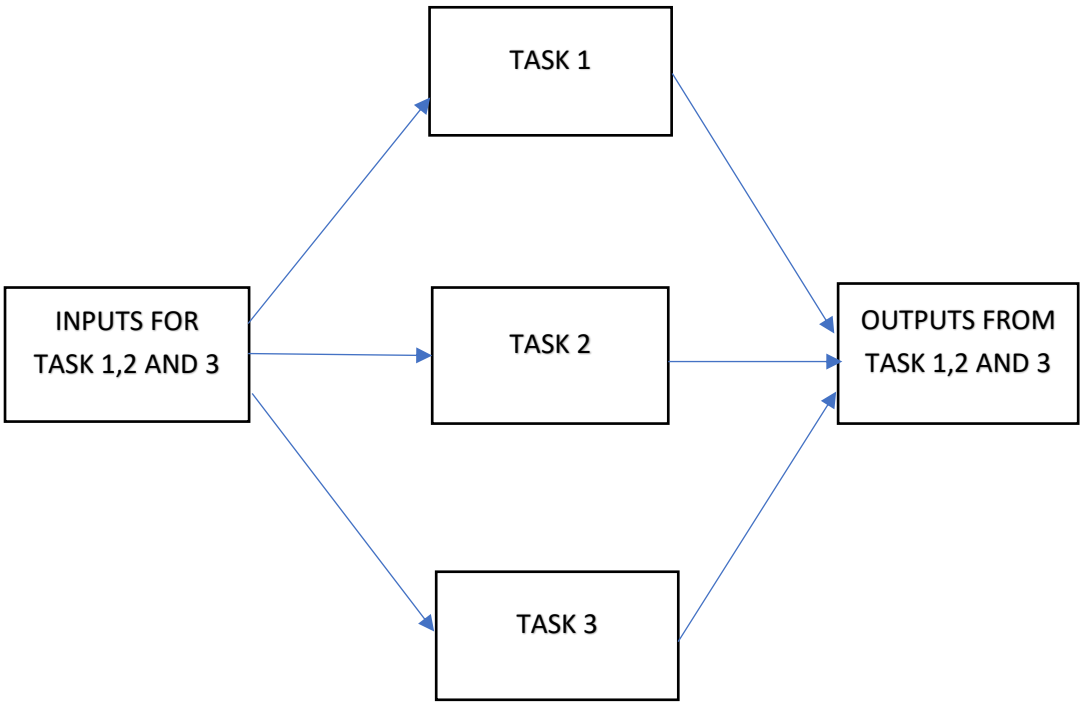

Figure1: Block Diagram

Here, different inputs are given to the same task. There are three tasks with the same operation and variation in the output is due to the input and is observed in the scope at the output section.

5. Methodology:

STEP 1: Three 3x3 matrix multiplication is implemented with the blocks available in Simulink library browser.

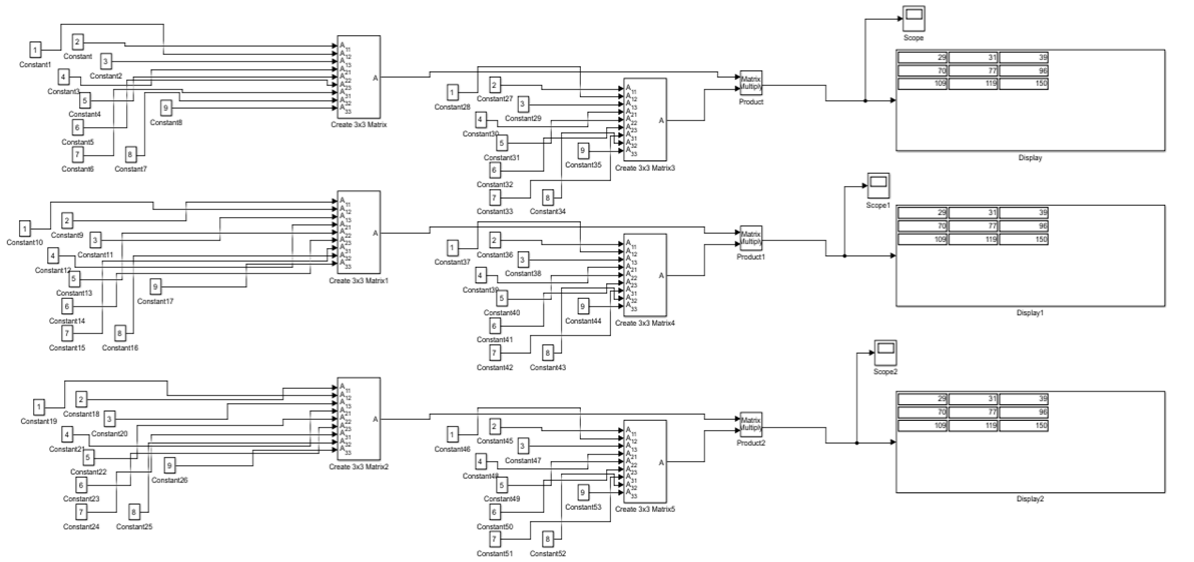

Figure2: Initial Model

STEP 2: Separate areas are created for the input, tasks and output section The model now consists of an input, a functional component that applies to each input, and a concatenated output.

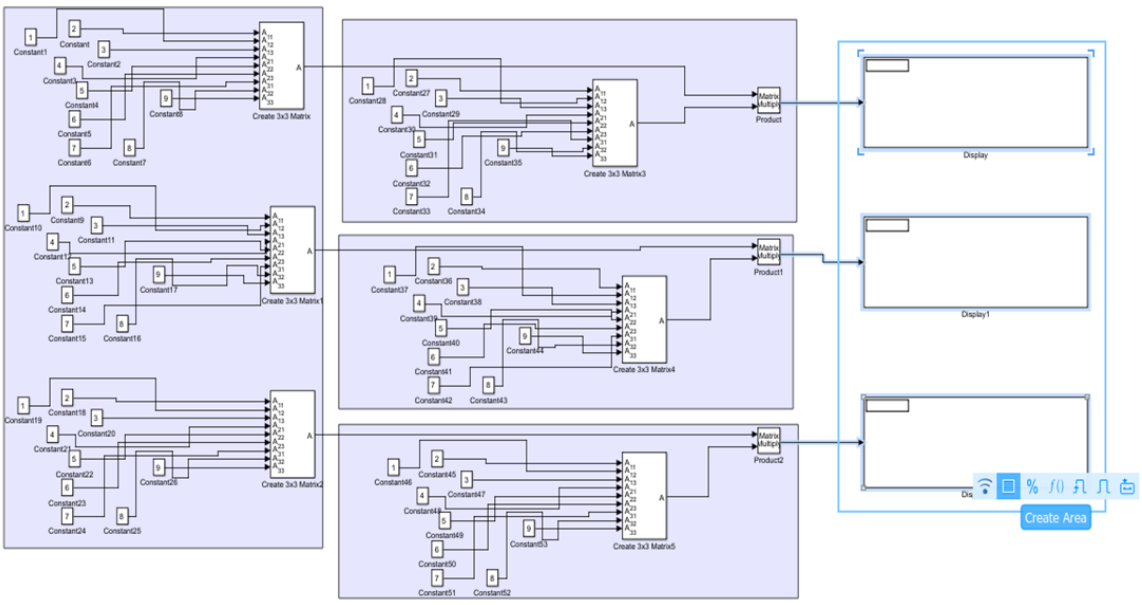

Figure3: Separate area created

STEP 3: Subsystems are made for the input, the 3 tasks and the output section and the model is completed.

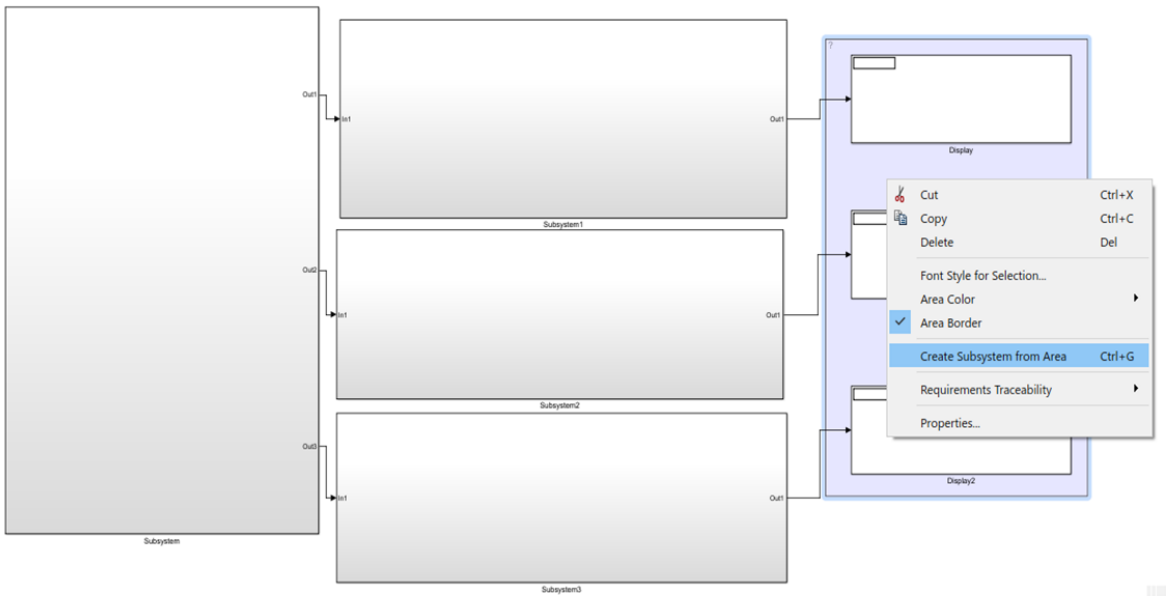

Figure4: Model Result

Now, the input section contains:

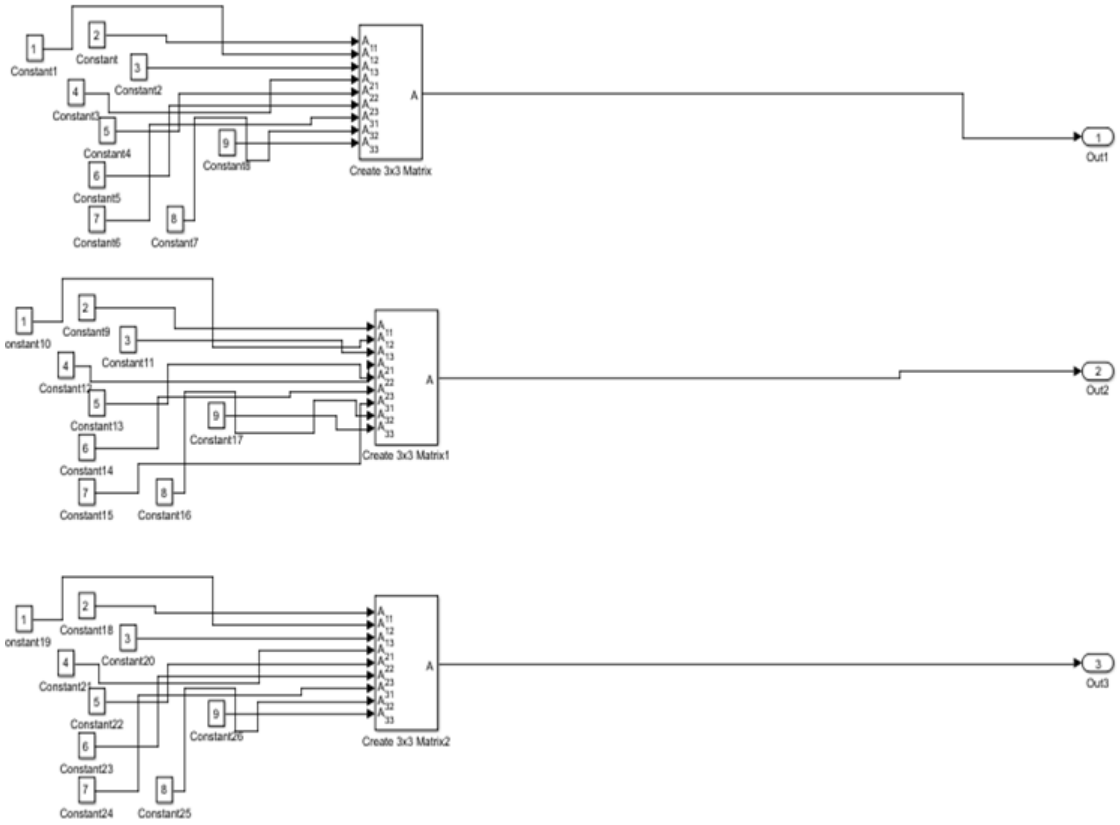

Figure5: Input Section

The three tasks are the same and the task subsystem contains:

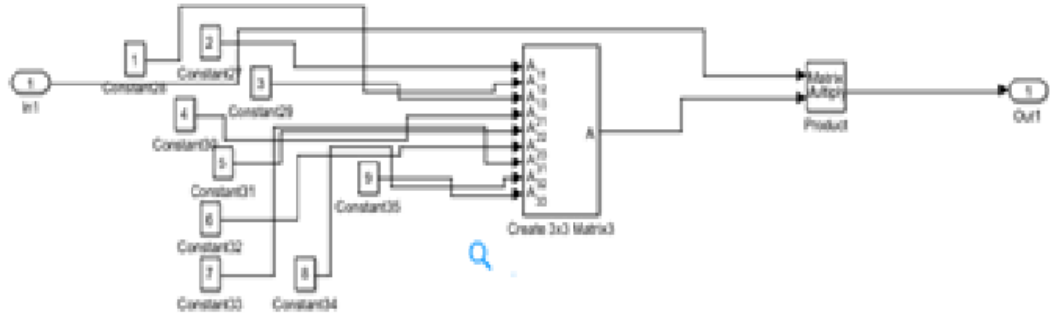

Figure6: Task Section

The output subsystem contains:

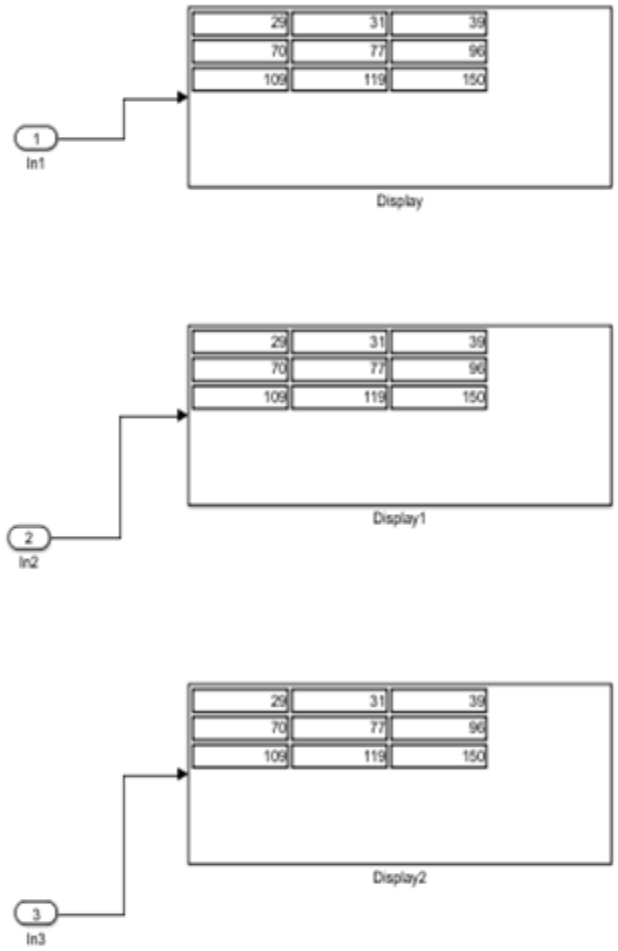

Figure7: Output Section

6. Result:

The execution time of data level parallelism block is compared with single task execution. Single task is shown in figure 8.

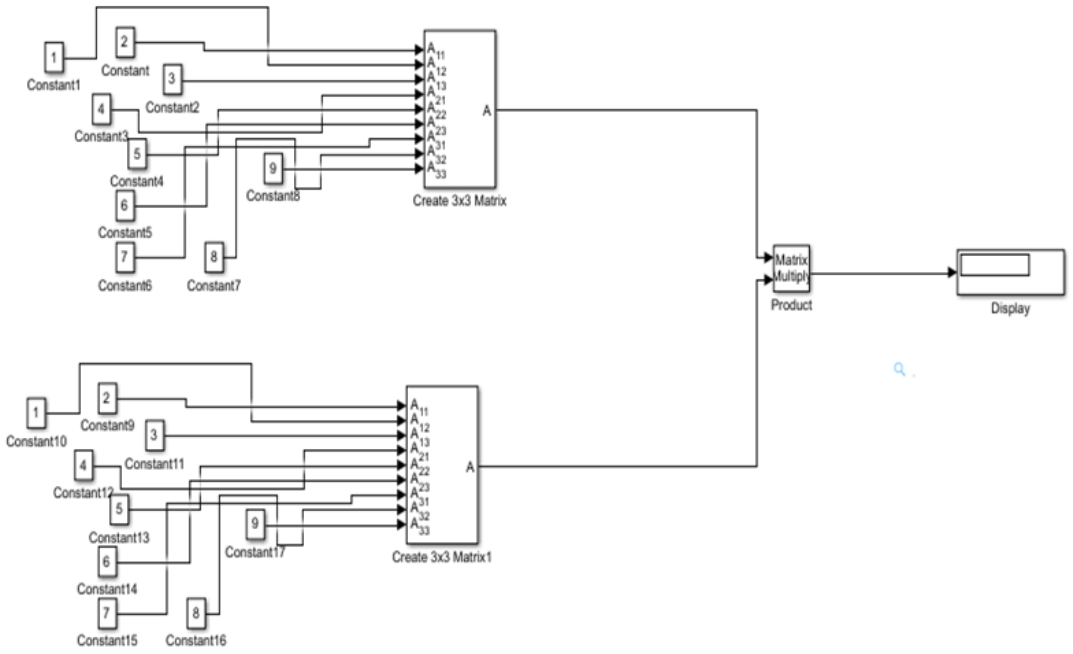

Figure8: Single Task

The execution time of data level parallelism and single task executions are calculated and given below:

Table 1: Performance comparison

MODEL	EXECUTION TIME (units of time)
Final model	1.5587
Single task execution	1.5182

7. Conclusion:

From the results obtained it is evident that data level parallelism is much faster execution time compared to sequential task execution. This can be used for multi core

systems where speed of execution, time of execution and performance needs to be calculated and improved.

8. References:

1. Tim Schmidt, Guantao Liu, and Rainer Dömer," Exploiting Thread and Data Level Parallelism for Ultimate Parallel SystemC Simulation", DAC-2017.
2. Jurgen Kadidlo and Alfred Strey," Exploiting data- and thread-level parallelism for image correlation",PDP 2008
3. Upul Senanayake, Rahal Prabuddha and Roshan Ragel," High Throughput Virtual Screening with Data Level Parallelism in Multi-Core Processors", ICIAfS – 2012
4. Roger Espasa and Mateo, "Expoiting instruction and data level parallelism",IEEE Micro 1997