Efficient Hardware Acceleration on SoC- FPGA using OpenCL

Advisor : Dr. Benjamin Carrion Schafer

Susmitha Gogineni

30th August '17

Design Automation and Reconfigurable Computing

UT DALLAS

Presentation Overview

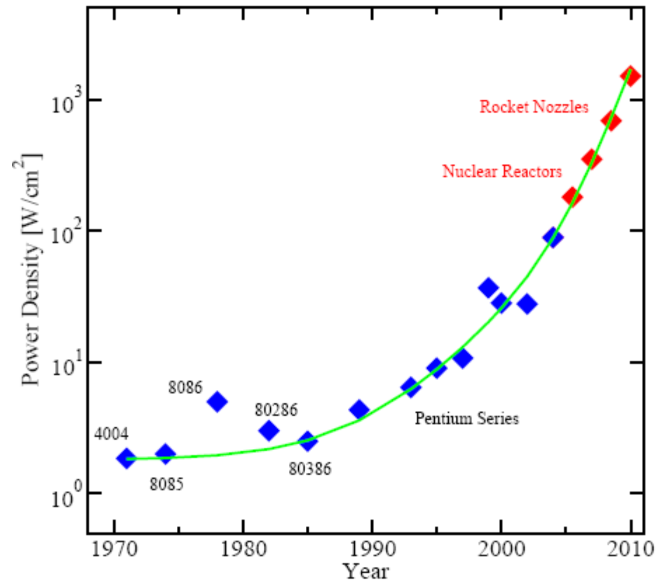
1. Objective & Motivation
2. Configurable SoC -FPGA
3. High Level Synthesis (HLS) & OpenCL
4. Hardware Acceleration on FPGA
5. Design Space Exploration (DSE)
6. Conclusions & Future work

Motivation

Quest for Performance & Power efficient hardware platforms to meet processing needs

Trends in Hardware

- Moore's Law led to increase in Transistor density, Frequency and Power.
- By Breakdown of Dennard's scaling, it is no more feasible to increase frequency due to power constraints



What Next ?

SoC – FPGA Platforms

Performance improved by offloading a CPU from computationally intensive parts to an FPGA.

Acquisition of Altera by Intel is also mainly motivated by this.
Intel has recently announced that it is targeting the production of around 30% of the servers with FPGAs in data-centers by 2020

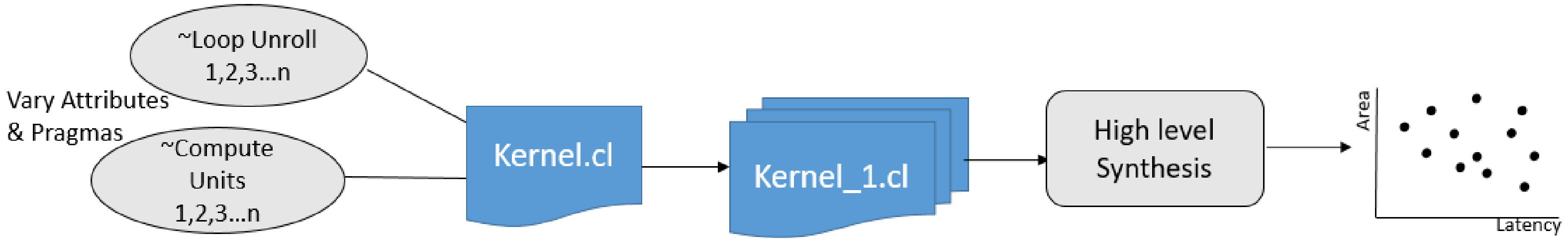
Objective

1. Hardware Acceleration using OpenCL

- Accelerate Computationally Intensive Applications on SoC-FPGA
- Study effects on acceleration with different attributes.
- Perform DSE using HLS.

2. Automated & Faster Design Space Exploration (DSE) method

- DSE is exploiting the different micro-architectures based on parameters of interest.
- DSE is multi objective optimization Problem.
- The search space is extremely large and Time constrained.
- Genetic Algorithm based meta heuristic to automate DSE is implemented.

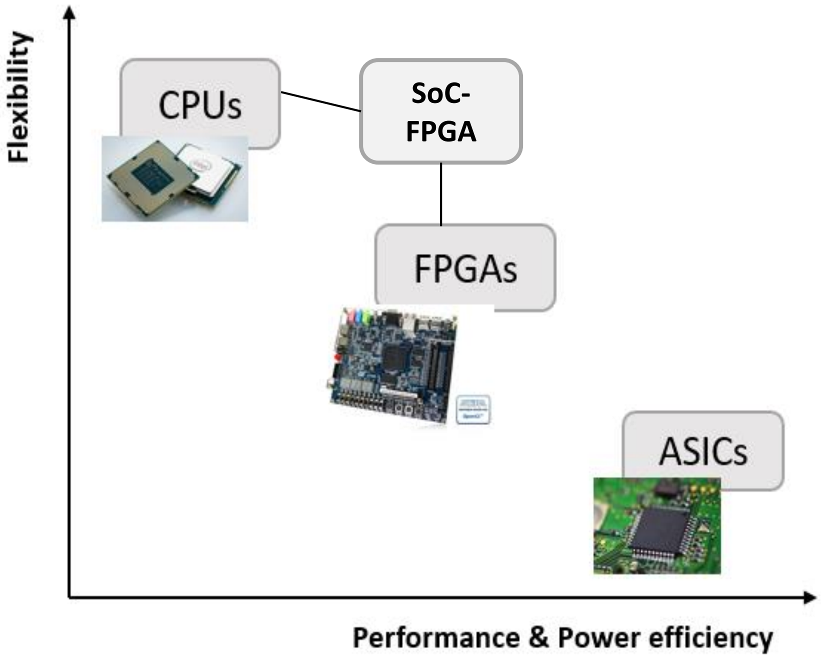


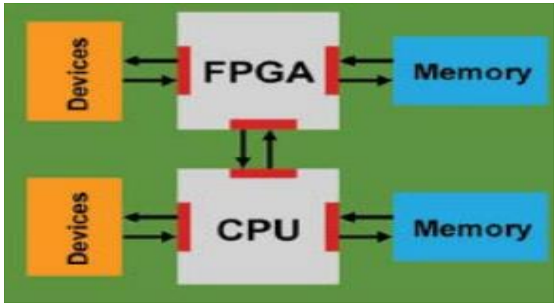
Design Space Exploration

Configurable SoC -FPGA

SoC-FPGA devices integrate both processor and FPGA architectures into a single chip.

- higher bandwidth for communication between Processor & FPGA
- Performance & Power efficiency





Off-chip Integration

On-chip Integration

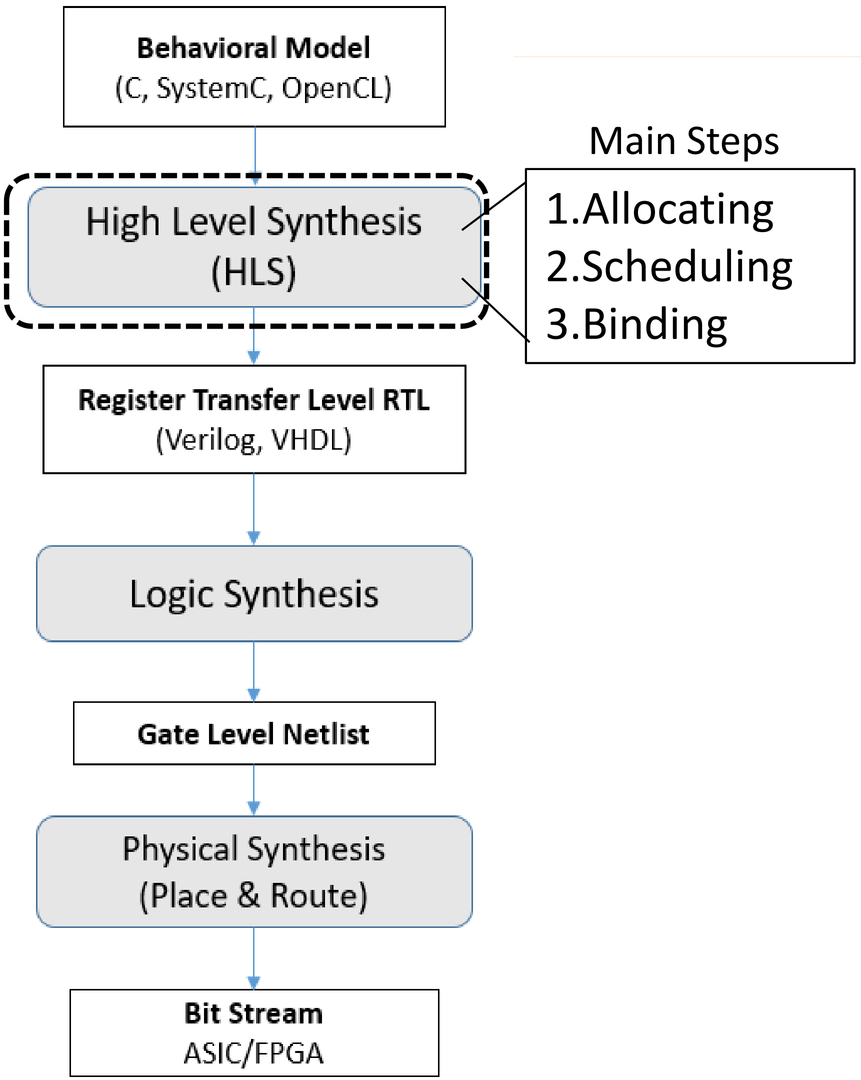
High Level Synthesis (HLS)

What is HLS?

- Automatic conversion of behavioral, untimed descriptions into hardware that implements the behavior.

Why HLS?

- Raises abstraction level of Languages for design.
- Less Coding , less verification, less bugs
- Design Productivity
- Meet Time to Market
- Design Space Exploration



ASIC/FPGA Design Flow

OpenCL – Open Computing Language

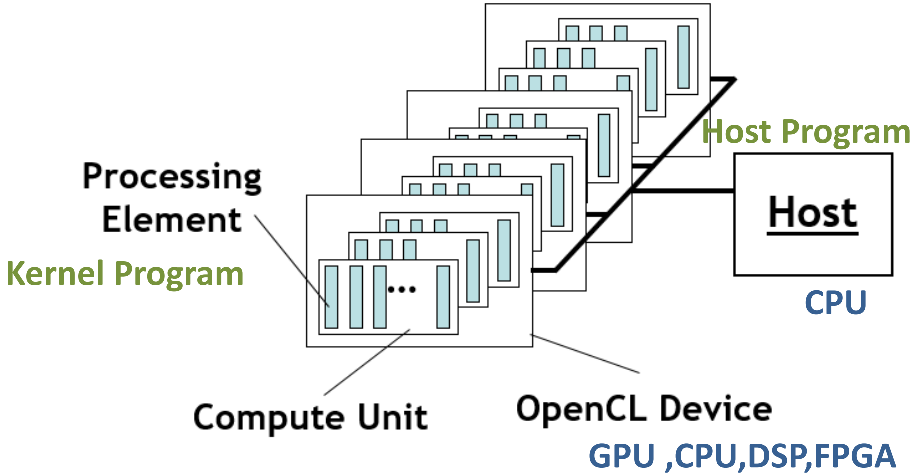
OpenCL (Open Computing Language) is a standard framework which allows parallel programming of heterogeneous systems.

- Programming the devices on the heterogeneous platform
- Application Programming Interface (API) to control the communication between the compute devices.

OpenCL Platform Model

Hardware abstraction Layers

- Host
- OpenCL Device
- Compute Units
- Processing Elements



OpenCL Execution Units

1. Host Program

   Manages Workload division and Communication among the Compute Units.

2. Kernel program

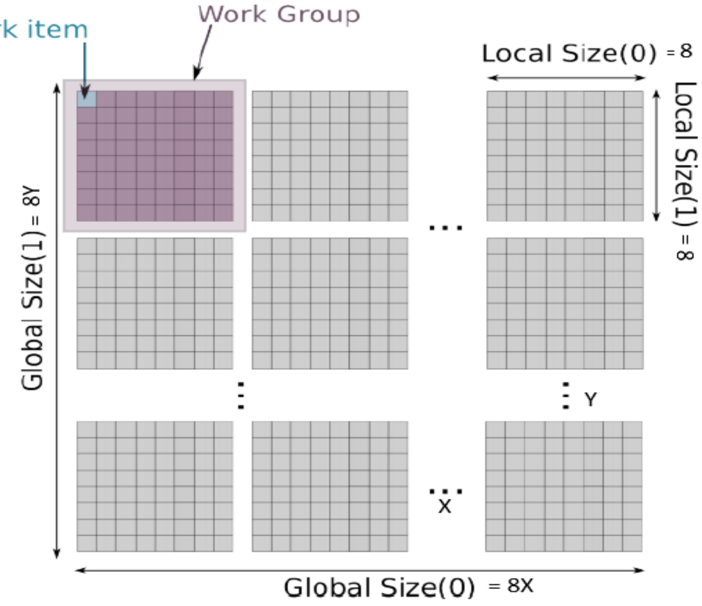
   Execute the computational part of the OpenCL application.

OpenCL Execution Model

Workload Division : Work-groups & Work-items

The entire work space to be executed is decomposed into Work-groups and Work-items

1. Work-item : Each independent element of execution in the entire workload is called a work-item.
2. Work-Group: A set of work-items are grouped to into a Work-Group



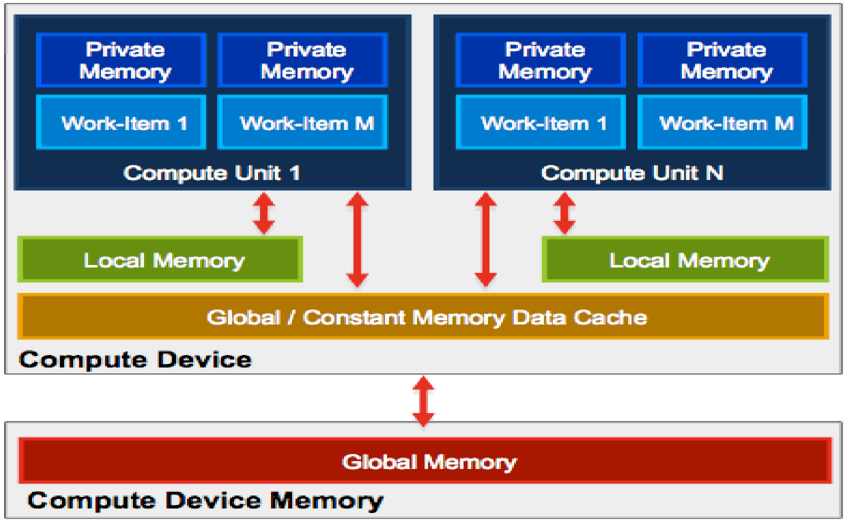
- Each Work-group is assigned to a Compute units .
- Work-items in a work group are executed by the same Compute unit.

OpenCL Memory Hierarchy

Memory hierarchy is structured to support data sharing ,Communication and synchronization of the work items.

Memory Type	*Scope*	*Accessibility*
Global Memory	Host Compute Units Processing Elements	All work groups All workitems
Local Memory	Compute Units Processing Elements	All Work-items in a workgroup Not shared to other workgroups
Private Memory	Processing Elements	Exclusive to Work-item Not shared to other work-item

Capacity decreases
Latency decreases



Hardware Acceleration on FPGA using OpenCL

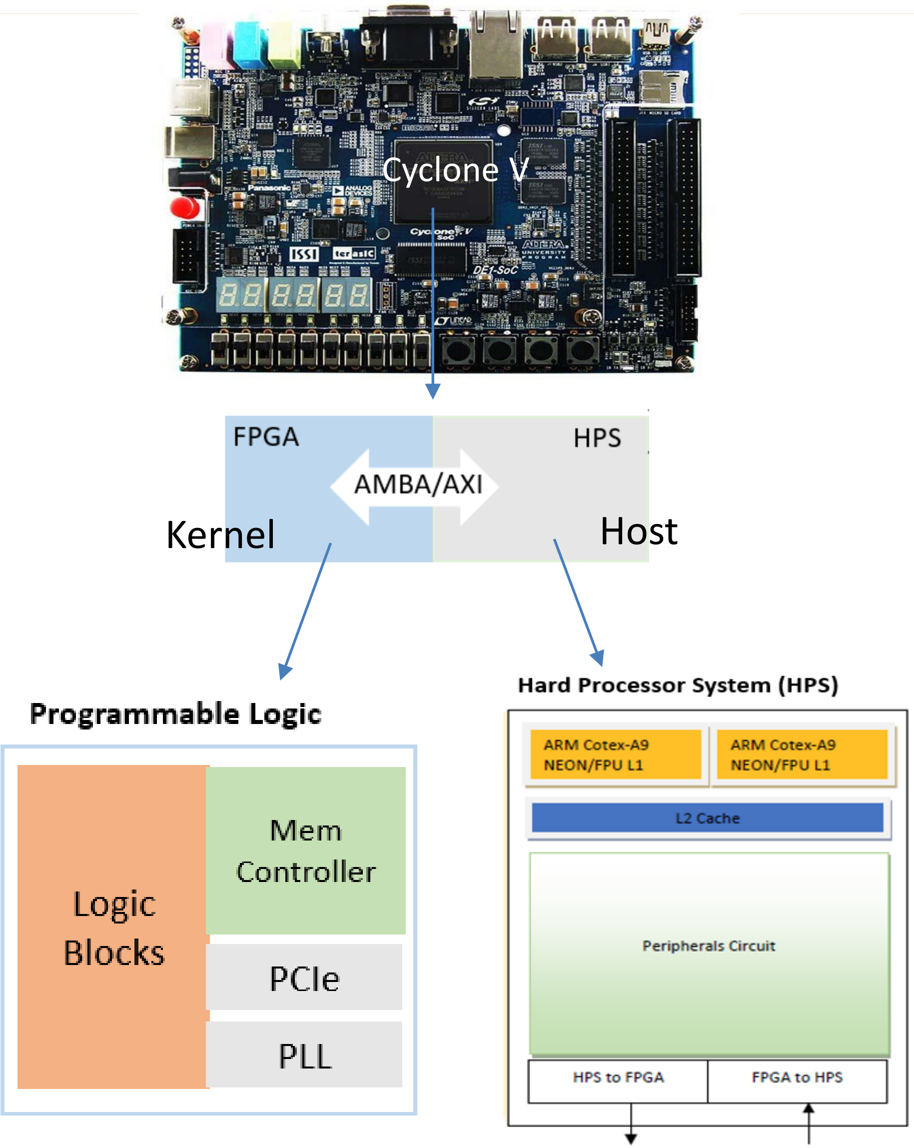
System Description

1. System Hardware

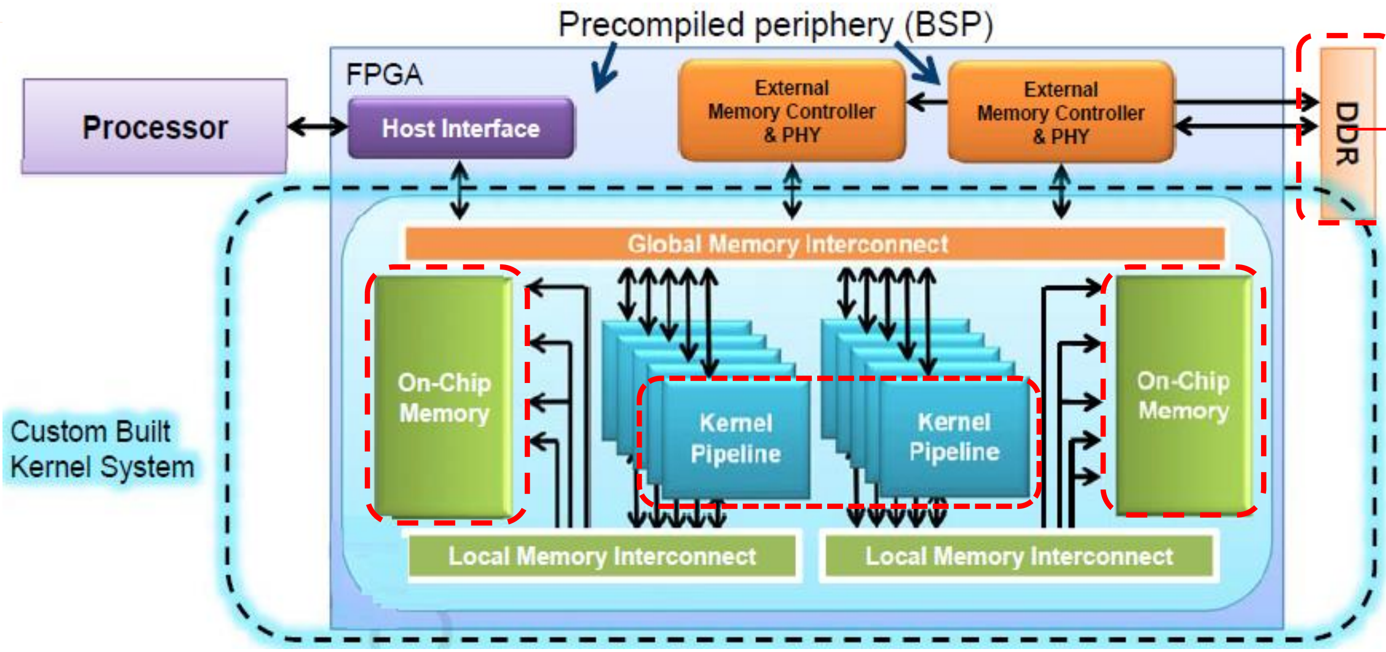
- Terasic DE1-SoC Board
  Altera Cyclone V FPGA

2. Software Tools

- Intel FPGA SDK for OpenCL
- Altera Quartus II
- Intel® SoC-FPGA Embedded
  Development Suite (SoC EDS)



3. System Memory Model



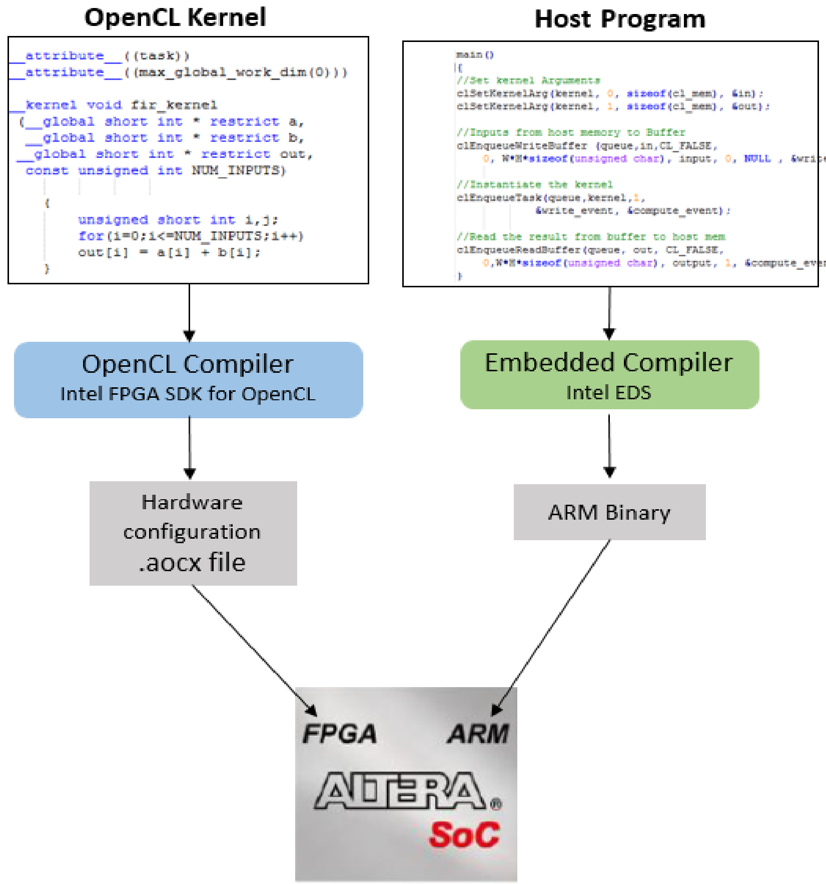
Communication Host & Kernel

Read buffers
Write Buffers

- Global Memory :
  Off-Chip DDR , High Capacity and High latency , Host to Kernel Interface
- Local Memory:
  On-Chip memory ,Higher Bandwidth & lower latency than Global Memory.
- Private memory:
  Registers & Block RAM on FPGA, Lowest latency at cost of area utilization.

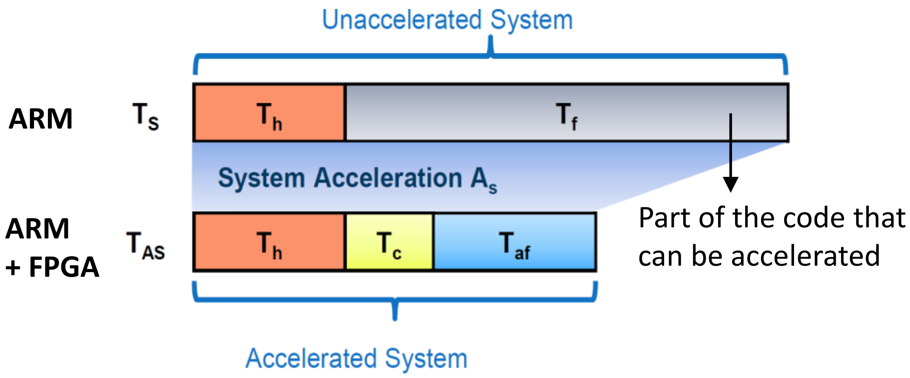
4.Programming the System

Kernel and Host Programs



Computationally intensive processing is off-loaded from ARM processor to FPGA

Optimization of the Design for Acceleration



T_h	Time spent for part of the main code running on Host
T_f	Time spent on the Unaccelerated Function
T_c	Time spent for communication between Host and Accelerator
T_{af}	Time spent on the accelerated Function
A_S	System Acceleration = Ts/ Tas

Timing Metrics for Acceleration

1. Reduce T_{af} : Increase Number of Parallel Operations

- Data level Parallelism : Duplicating Processing Units and launch in parallel
- Instruction level Parallelism: Pipelining Instructions and loop Iterations
- Task Level Parallelism : Independent tasks run parallel on Individual Kernels

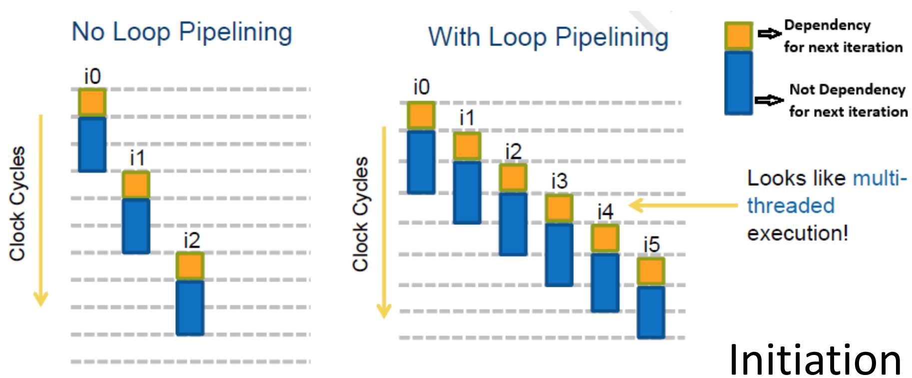
2. Reduce T_c : Reduce Communication Time

- Caching frequently used data to Local memory from Global Memory .
- Coalesce Memory access patterns when possible.
- Using Channels and pipes to communicate between kernels instead of Global memory

Kernel Architectures: Single Task Kernel & ND Range Kernel

1. Single Task Kernel

- The entire kernel is executed as single thread on single Compute unit
- Loop Pipelining : Loop iterations are Pipelined
- Data dependencies are handled using logic cells and registers on FPGA
- Used when there is dependency between work-items , No parallelism possible.
- Ex. Decimation, FIR etc.

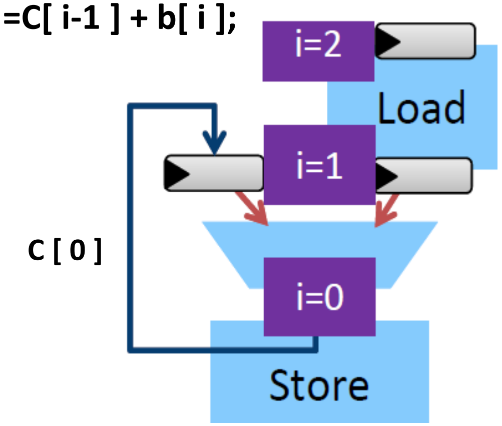


No Overlap of Iterations

Faster Because Iterations are Overlapped

Initiation Interval
ii =1

for (i=1; i<n; i++)
C[i]=C[i-1] + b[i];



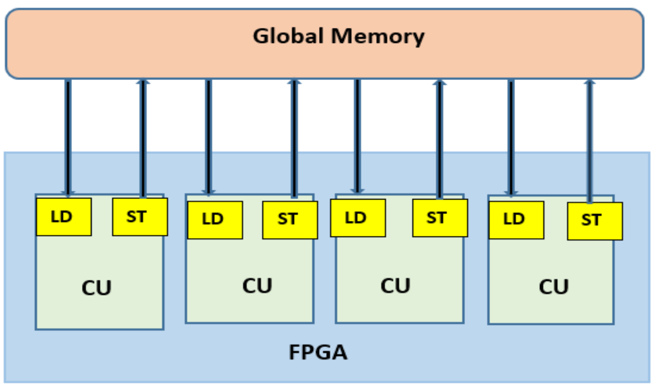
*Exec Time = ((Num_Iterations * ii) + Loop_latency)* Time_period*
ii = Initiation Interval

Dependencies are handled through Data-feedbacks

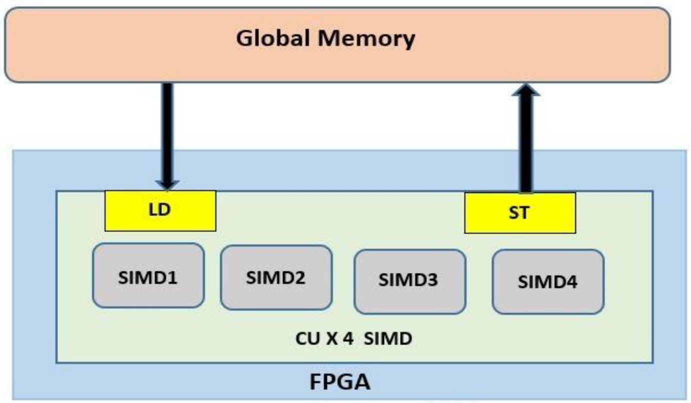
2. ND Range Kernel

- Throughput achieved by Data level Parallelism
  Work-items are executed in Parallel, by replicating Hardware units on the Kernel
  (ie multiple Compute Units (CUs) , Single Instruction Multiple Data units (SIMDs))
- Increases Memory Bandwidth and Logic Utilization
- Used when there are few or no data, memory dependencies between the Work-items.
- Ex. AES , Matrix multiplication etc

CUs vs SIMDs



num_compute_units(4)



num_compute_units (1)
num_simd_work_item(4)

- CUs work on different Work-Groups
- Replicates Data Paths and Control path.
- Memory accesses patterns are scattered.

- SIMDs work on different Work-items of same Work-Groups
- Replicates Data Paths only . Control path is shared
- Memory accesses can be Coalesced.
- Cannot be used when Work-items have different Control Paths

Optimization Attributes & Pragmas

1. num_compute_units(N)
2. num_simd_work(N)
3. #pragma unroll < N >
4. max work group size(N)
5. reqd work group size(x; y; z)

```
__attribute__((max_work_group_size(512)))
__attribute__((num_compute_units(1)))
__attribute__((num_simd_work_items(1)))
__kernel void aes_kernel(__global const int * restrict exp_key,
                        __global const unsigned char * restrict in_data,
                        __global unsigned char * restrict out,)
    {
    unsigned int n = ((get_global_id(0)* workgroup_size) + get_local_id(0));

    #pragma unroll 8
    for(short int i = 0; i< (NB * (NR + 1)) ;i++  )
```

OpenCL Benchmarks

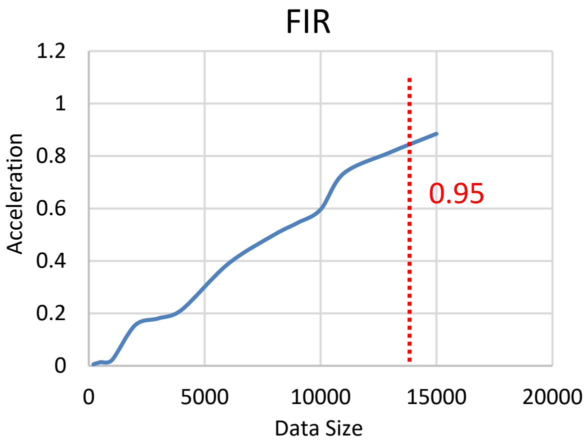
- 6 OpenCL applications
- Kernel Type chosen based on application
- Experimented on both Unaccelerated System(ARM) and Accelerated System (ARM+FPGA) to compare performance

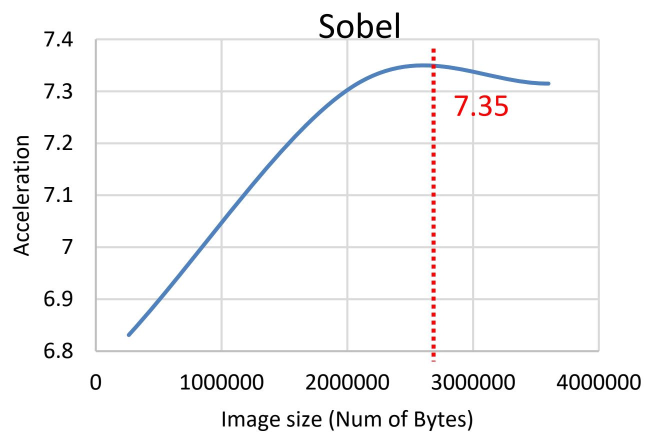
BENCHMARK	Kernel Type	Pipeline Initiation Interval	Logic (%)
Sobel	Single Task	2	20
FIR	Single Task	1	20
ADPCM	Single Task	40	20
Decimation	Single Task	1	81
Interpolation	Single Task	1	28
AES	NDRange CU=2,SIMD=2	Not Pipelined	84

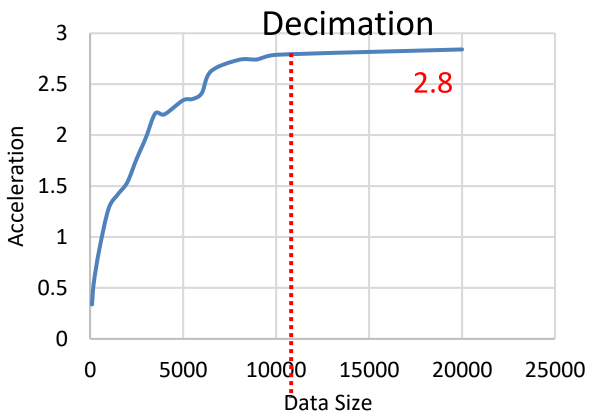
Results: Acceleration

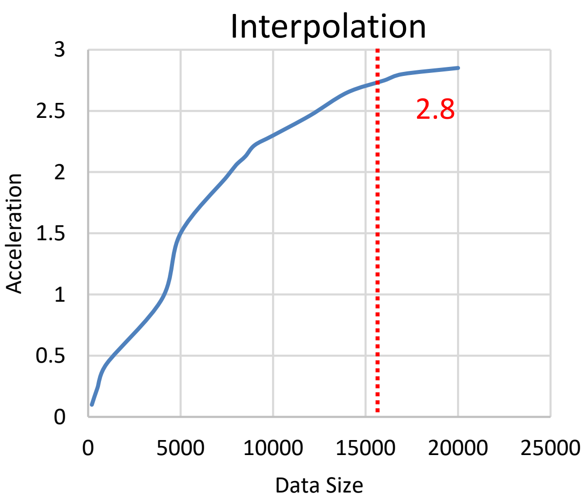
Acceleration tends to saturate at high data sizes

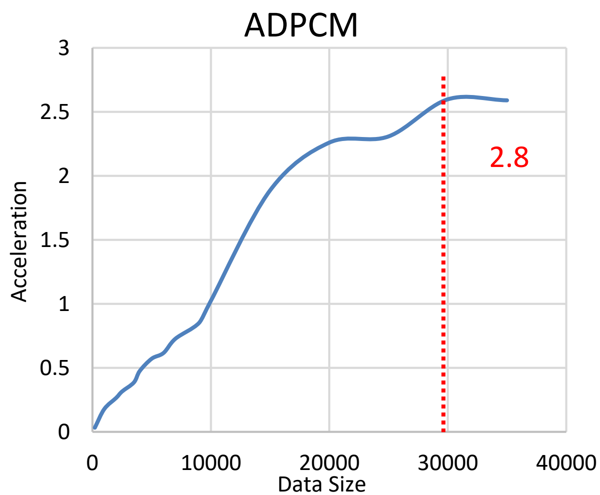
Plots of Acceleration vs data size

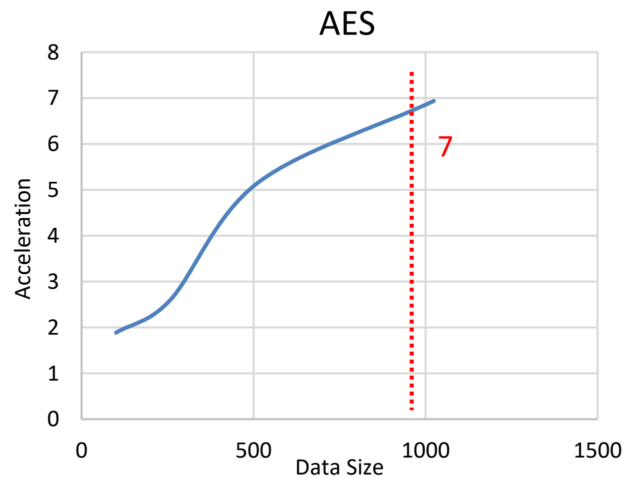




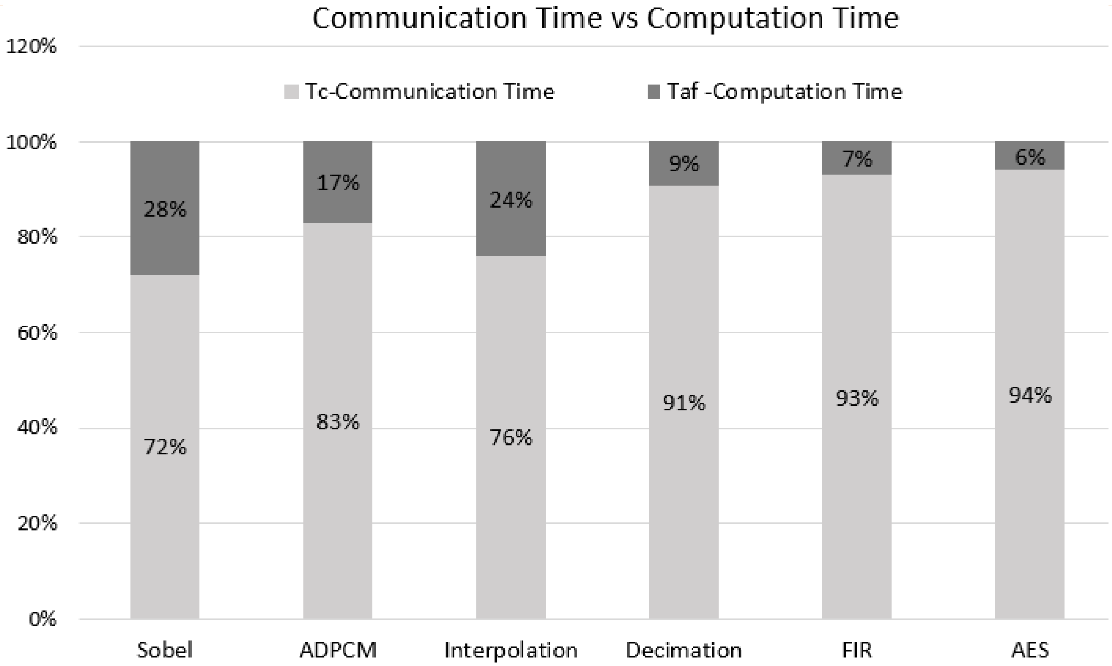








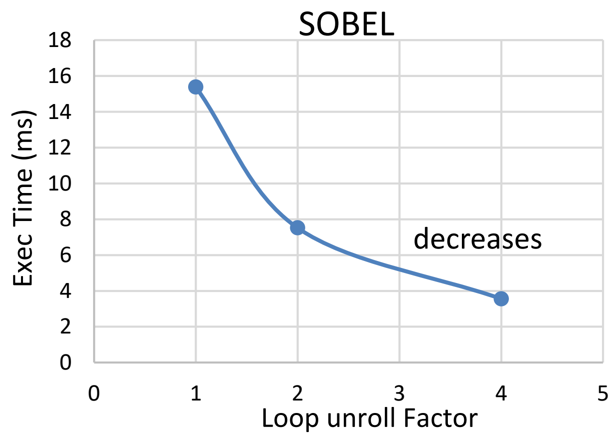
Results: Communication Overhead

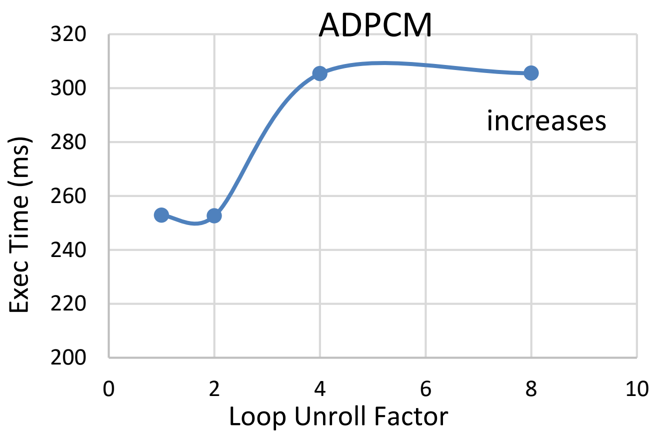


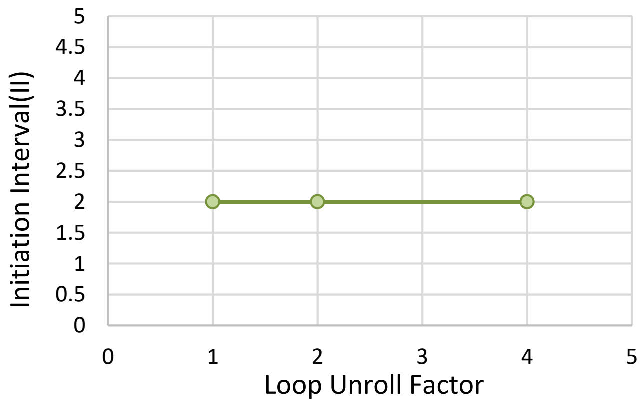
- Communication Overhead:
  Most of the time on the accelerated system is spent for data Communication between the host and Kernel
- Initially, the acceleration tends to increase with data size due to growing computation complexity.
- The acceleration ceases beyond a point because of no immediate data is available for processing due to communication overhead and limited Communication Data buffer size between Host and Kernel.

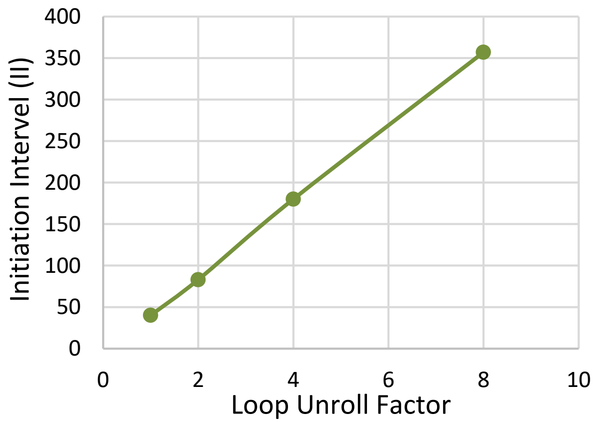
Observations: Acceleration effects due to attributes

2. Plots of Execution Time vs Loop Unroll Factor









In ADPCM, Execution Time Increased with Loop Unrolling as a result of increase in Initiation Interval (II) Which is caused due unrolling iterations with dependencies or memory stalls.

Observations: Acceleration effects due to attributes

1. Acceleration of AES by varying the number of CU and SIMD attributes across different data sizes
Number of Workgroups = 2 , Number of Workitems = (Num_inputs)/2

Compute Units	SIMD Units	Acceleration		
		256 Inputs	512 Inputs	1024 Inputs
1	1	2.5	4.6	6.6
1	2	2.7	5.4	6.6
1	4	2.4	5.4	6.9
2	1	4	4.6	6.9
2	2	2.6	5.2	6.9

Decreased

Increased

trade off between Data processing efficiency and Bandwidth requirement

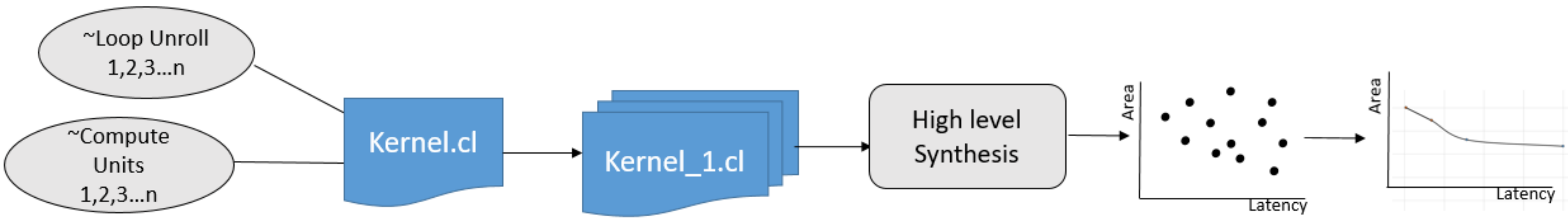
Each Attributes has various trade-off affects on the performance ,memory access ,Bandwidth requirement ,Logic utilization etc.

Design Space Exploration

Exhaustive Search vs Fast Heuristic

DSE by Exhaustive Search methodology

- Exhaustive search DSE involves analyzing of all possible search combinations.
- Pareto Optimal Solutions is the set of dominant solutions, for which no parameter can be improved without sacrificing at least one other Parameter.
- Area and Execution time parameters are used.



1. Add Tunable attributes

2. Generate possible Solutions

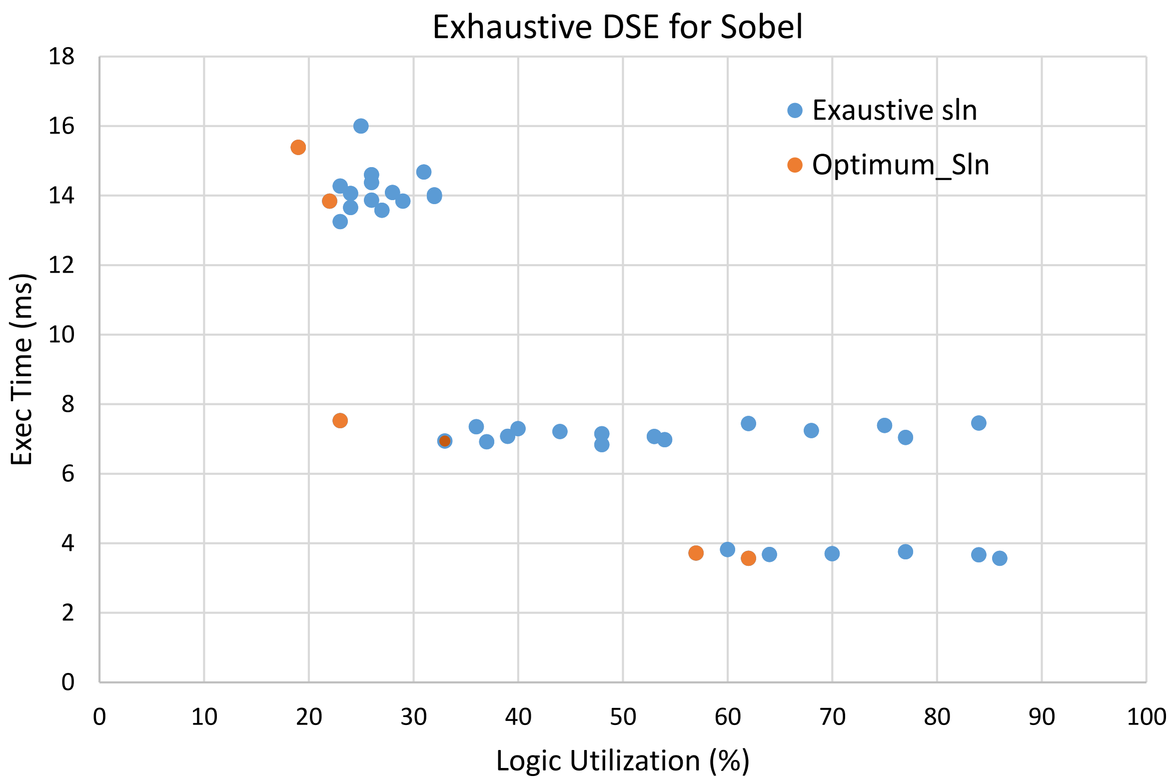
3. Compile Solutions

4. Design Space

5. Pareto Optimal Solutions

The main disadvantage is that the design space is typically large and grows exponentially with the number of exploration options.

Example of Design Space and Pareto Optimal Solution

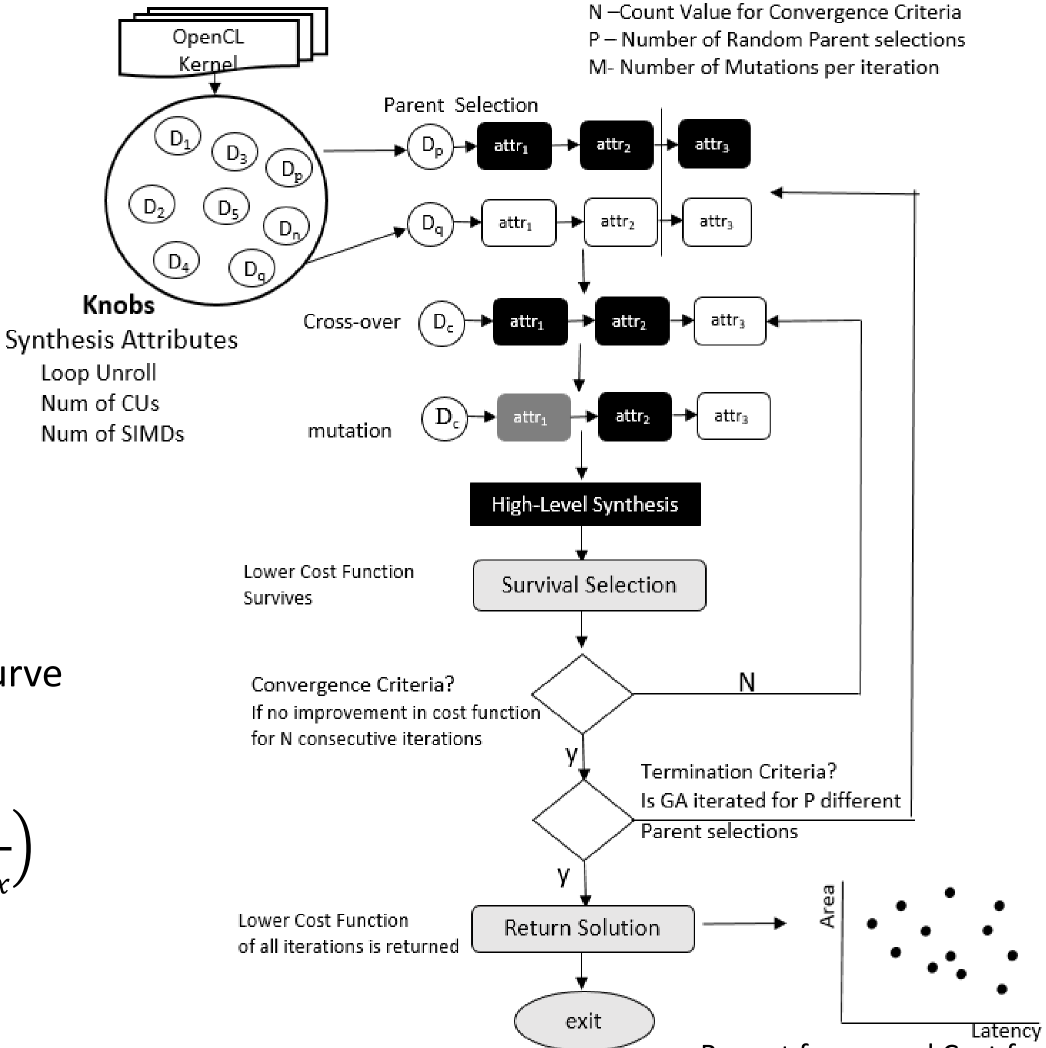


DSE by Genetic Algorithm

1. Parent Attributes selection
2. Random Crossover

   & Mutation (M)
3. Cost Function based Survival
4. Convergence Criteria (N)
5. Termination Criteria (P)
6. Repeat for various cost factors (a,b)
7. Find the Pareto dominant trade off curve

$$\text{cost} = a * \left(\frac{area}{area_{max}}\right) + b * \left(\frac{time}{time_{max}}\right)$$



Repeat for several Cost factors (a,b)

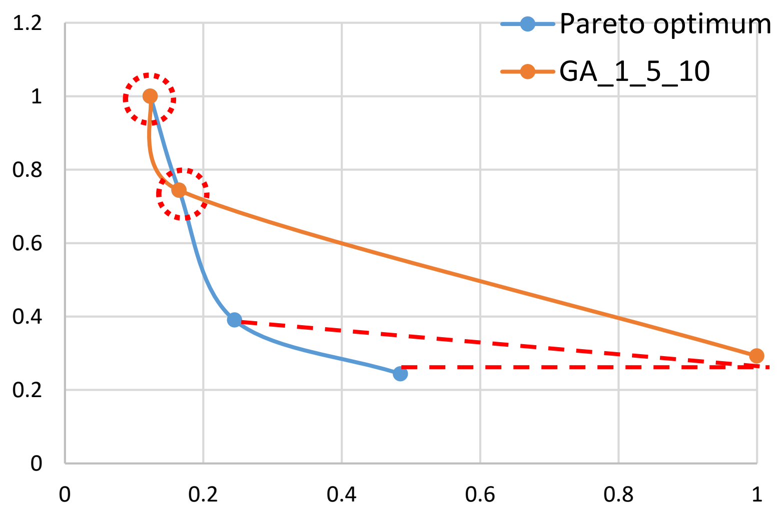
Efficiency Metrics

Metrics to measure the quality of solutions in Genetic Algorithm based Heuristic :

1. Pareto Dominance (Dom) :
   The fraction of total number of solutions in the Pareto set being evaluated ,also present in the reference Pareto set .

2. Average Distance from Reference Set (ADRS):
   ADRS measures the average distance between the Heuristic approximated front and the Pareto Optimal Front.

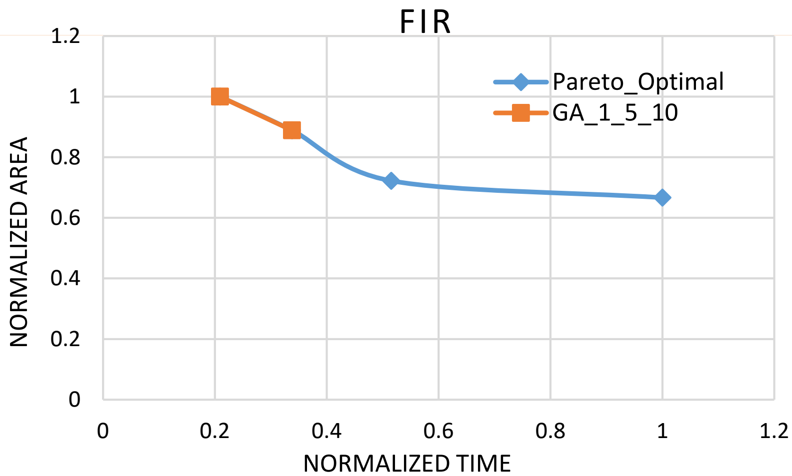
3. Speed up:
   Determines speed up in the compilation time to find the Pareto dominant Front compared to the Exhaustive search.

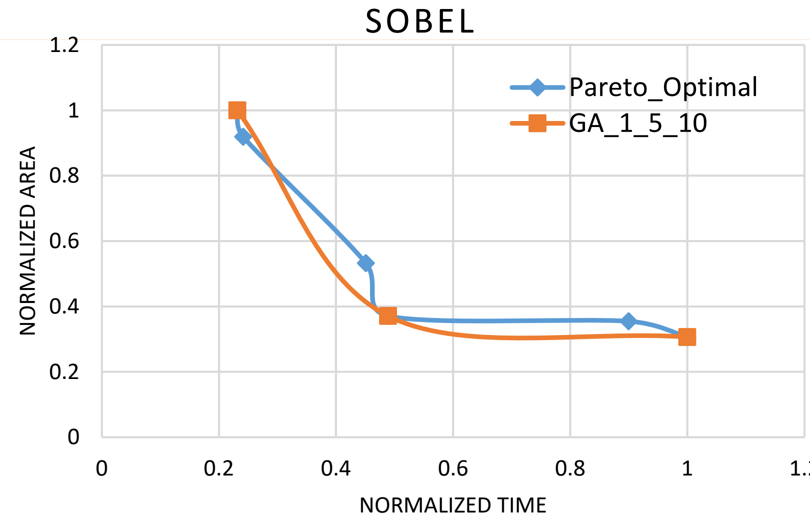


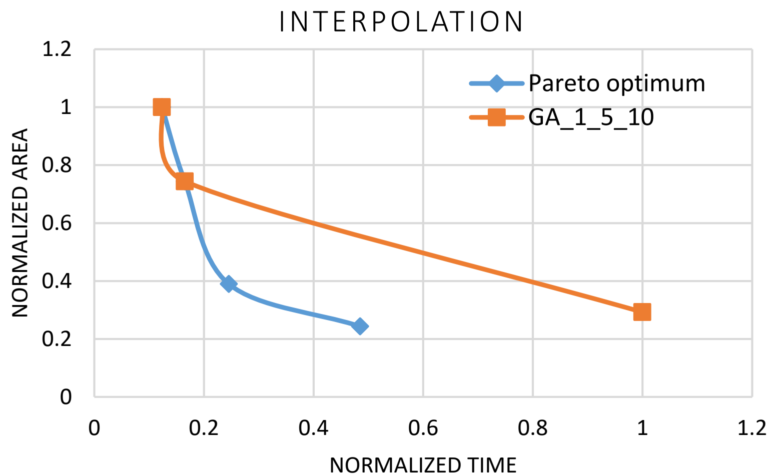
Dominance = 2/4 = 0.5

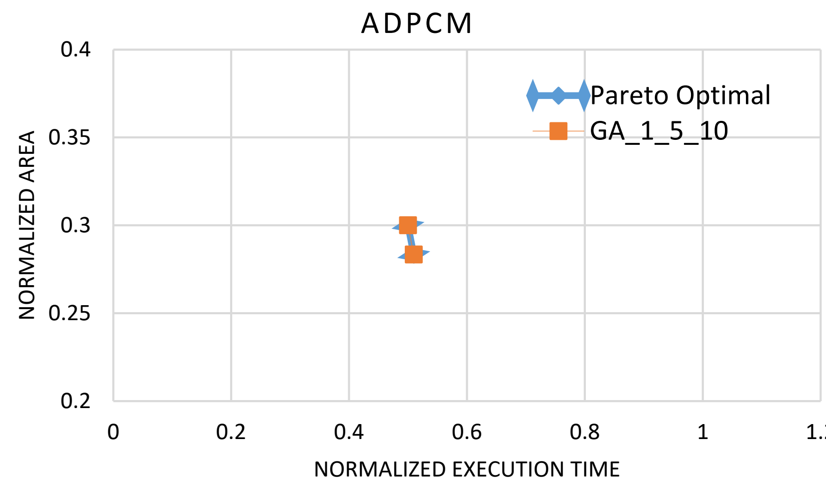
ADRS = 0.64

Comparison of Results : Exhaustive Search vs Genetic Algorithm









System Exploration Trade-off Curves

Pareto optimal Front of Exhaustive DSE vs Pareto Dominant Front of Genetic Algorithm

DARClab Design Automation and Reconfigurable Computing

UT DALLAS

Results : Genetic Algorithm Efficiency Metrics

GA Parameters	Performance Metrics	Benchmarks				Average Metrics For GA
		FIR	Interpolation	ADPCM	Sobel	
Mutations=1 Parent=5 Count =10	DOM	0.5	0.5	0.5	0.6	0.52
	ADRS	0.47	0.33	0.06	0.1	0.24
	Spd Up	4.6	4	1.2	12.6	5.6
Mutations=2 Parent=5 Count =10	DOM	0.75	0.5	0.5	0.6	0.59
	ADRS	0.48	0.18	0.06	0.16	0.22
	Spd Up	3	2.8	1.17	12	4.74
Mutations=2 Parent=5 Count =5	DOM	0.75	0.75	1	0.6	0.77
	ADRS	0.48	0.23	0	0.1	0.20
	Spd Up	5.2	4	2	12.5	5.92
Mutations=1 Parent=3 Count =15	DOM	0.75	1	1	0.33	0.77
	ADRS	0.48	0	0	0.32	0.2
	Spd Up	4.6	4	1.6	21.8	8
Mutations=2 Parent=3 Count =10	DOM	0.5	0.75	1	0.6	0.71
	ADRS	0.47	0.83	0	0.2	0.37
	Spd Up	4	3.4	1.56	22	7.74

Results Summary :

Average Dominance = 0.7 Average ADRS = 0.2 Average speedup = 6

- Genetic algorithm heuristic can determine about 70% of the optimal dominant solutions
- Solutions can be within a 20% range in design space around the optimal solutions

Conclusions

- We developed set of OpenCL benchmarks to study the trend in acceleration as a result of various attributes.
- A fast and heuristic method to explore the design space is implemented. Its performance is analyzed & compared with the reference solution set.
- Based on the experiments, an average dominance of 0.7, average ADRS of 0.2 at average speed up of 6 times compared to the exhaustive DSE search is observed.

Future Work

- Experimenting with wider range of benchmarks.
- Upgrade benchmarks to multiple FPGA Platforms.
- Other fast Heuristic methods like Simulation Annealing or Machine Learning algorithms can be used for exploration of design space.

Thank You !

Questions ?