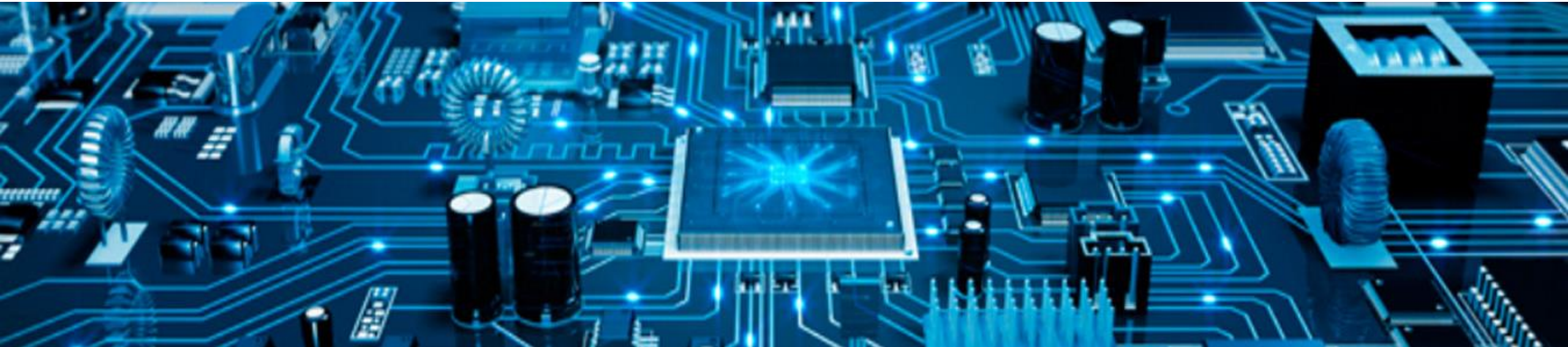
# A HARDWARE DESCRIPTIVE APPROACH TO BEETLE ANTENNAE SEARCH

TEJAS B N (MT2022521)

RAKSHIT BHATIA (MT2022514)

AMAN PRAJAPATI (MT2022501)

HIMANSHU KUMAR RAI (MS2022012)



# OUTLINE OF PRESENTATION

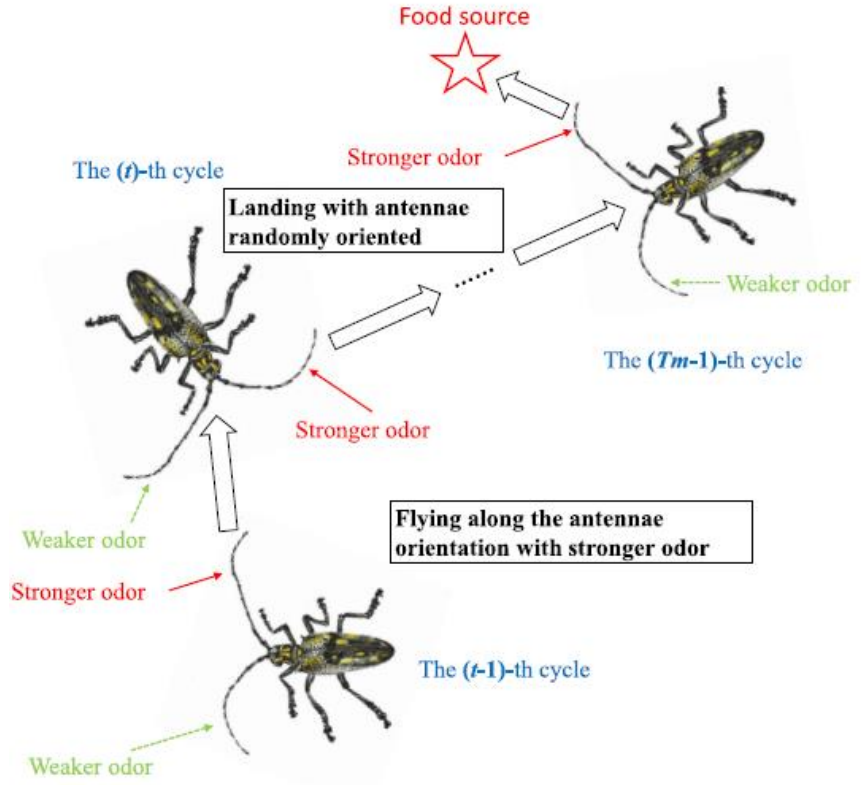
- AIM OF THE PROJECT
- INTRODUCTION
- HARDWARE IMPLEMENTATION OF BAS WITH FIXED POINT REPRESENTATION
- HARDWARE IMPLEMENTATION OF BAS WITH BF16
- IMPLEMENTATION ON ARM PROCESSOR
- SYNTHESIS RESULTS
- SIMULATION RESULTS
- CONCLUSION

# AIM OF THE PROJECT

- The beetle antennae search (BAS) is a newly developed meta-heuristic algorithm, which solves optimization problems with simple structure.
- Meta-heuristic algorithms including the BAS largely relies on programming in a high-level language and executing the code on a computer platform. However, the high-level implementation of the BAS algorithm hinders it from being used in an embedding system, where real-time operations are normally required.
- To address this limitation, we present an approach to implementing the BAS algorithm on a field-programmable gate array (FPGA). We program the BAS function in the Verilog hardware description language (HDL) to reduce the latency of the algorithm.

# INTRODUCTION

- The beetle antennae search (BAS) is a newly developed meta-heuristic algorithm, which solves optimization problems with simple structure. The algorithm simulates the biological trajectory of a beetle
- In a strange environment, the beetle uses the two antennae on its head to conduct a series of flying and landing behaviors for the foraging. In the initial stage of the foraging, the beetle does not know where the food source is, and its two antennae are oriented randomly.
- It guesses the direction of the food through the odorants received by the antennae. The beetle judges which antenna receives the stronger odor, and accordingly uses its orientation as the estimated direction of the food source.
- Then the beetle flies along the estimated direction for a certain distance and lands with the two antennae oriented randomly.
- One cycle of the beetle antennae search consists of a directional flying procedure and a randomly directional landing procedure.
- The cycle is repeated until the final food source is located. Such repetitive procedures are illustrated in Figure on the right.



# MATHEMATICAL MODEL

- **Step–1: Random Antenna Orientations:**

$$\mathbf{d(t)} = \frac{\boldsymbol{r(t)}}{\|\boldsymbol{r(t)}\|}$$

Where, d (t) is a normalised directional vector that describes the orientations of the beetle antennae and r(t) is a random vector.

- **Step–2: Antenna Positions w.r.t Randomly Oriented Landing:**

$$\mathbf{x_r(t)} = \mathbf{x(t)} + p(t)\mathbf{d(t)}$$

$$\mathbf{x_l(t)} = \mathbf{x(t)} - p(t)\mathbf{d(t)}$$

Where, x(t) represent the position of the beetle, $x_r(t)$ and $x_l(t)$ represent the location of a beetle's right and left antennae and p(t) represent the sensing distance at the $t^{th}$ cycle.

Initial value of p(t) is set to a large value to avoid local minima.

# MATHEMATICAL MODEL

- **<u>Step–3</u>: Beetle Position Subject to Directional Flying:**

  A beetle's biological nature dictates that it picks the antenna orientation with the strongest odour as its flight direction. Since we are trying to find minima, we will pick the antenna orientation with weakest odour. Based on this observation, the flight direction is calculated as follows, where f(.) is the fitness function:

$$\mathbf{d_f(t) = d(t)}\text{sign}(f[x_l(t)] - f[x_r(t)])$$

  Then the landing position of the beetle is determined by,

$$\mathbf{x(t+1) = x(t) + d_f(t)}\varepsilon(t)$$

  where $\varepsilon(t)$ denotes the flight distance of the beetle at $t^{th}$ cycle.

# MATHEMATICAL MODEL

- **Step–4: Beetle Position Determination:**

The BAS algorithm makes a comparison between the state {x(t +1), f [x(t +1)]} at the new position and the current state $\{\hat{x}, f(\hat{x})\}$. The process of the comparison is given as follows:

$$\{\hat{x}, f(\hat{x})\} = \begin{cases} \{x(t+1), f[x(t+1)]\}; & f[x(t+1)] < f(\hat{x}) \\ \{\hat{x}, f(\hat{x})\}; & f[x(t+1)] > f(\hat{x}) \end{cases}$$

- **Step–5: Convergence Factor Updates:**

The convergence factors are updated as follows:

$$p(t+1) = 0.95p(t)$$

$$\varepsilon(t+1) = 0.95\ \varepsilon(t)$$

Here, p(t+1) replicates the beetle's sensing distance decreasing as it approaches the food source.

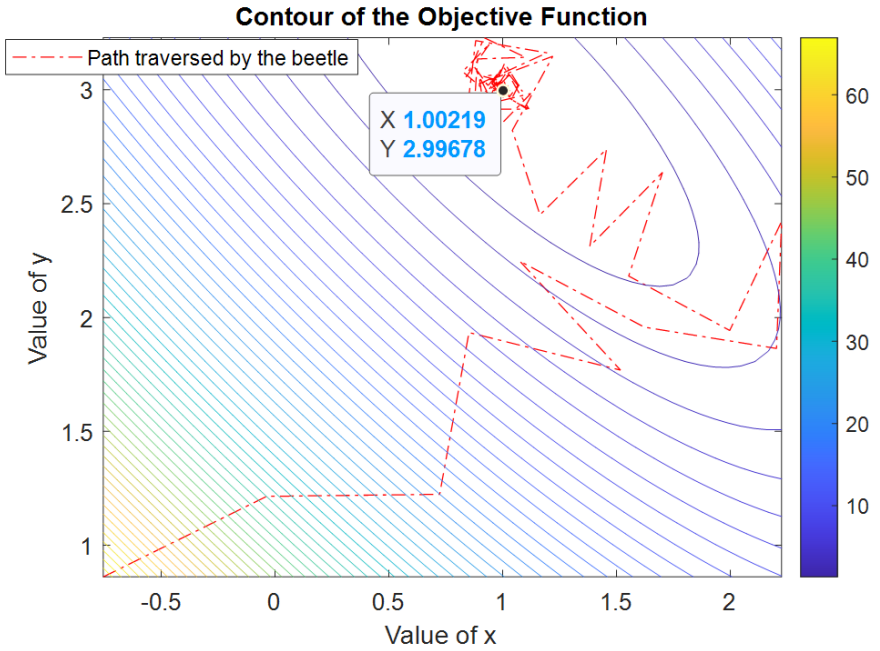
ε(t+1) represents the beetle's flying distance decreasing as it approaches the food source.

# OBJECTIVE FUNCTION

- The objective function is chosen as booth function, which is a widely used benchmark for testing evolutionary algorithms:

$$f(x, y) = (x + 2y - 7)^2 + (2x + y - 5)^2$$

- This function has a minimum value of f = 0 at (x, y) = (1, 3).
- To visualize the working of BAS algorithm, it was implemented on MATLAB the results are plotted below:



# HARDWARE IMPLEMENTATION OF BAS WITH FIXED POINT REPRESENTATION

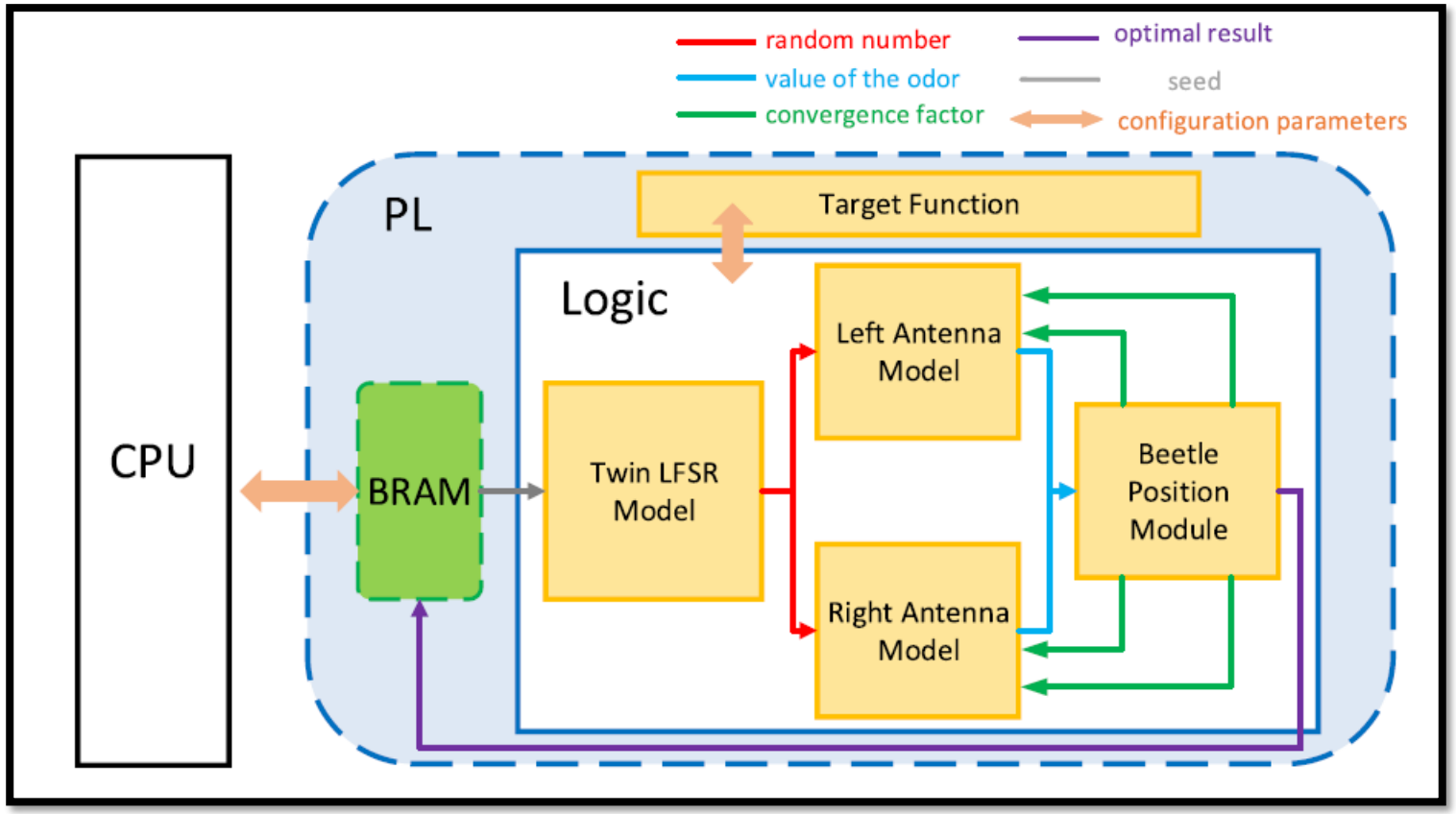
- **Fixed Point Representation:**
- The fixed-point operations consume fewer hardware resources with a guarantee of acceptably accurate results.
- We give an example of 5.555 times 4.444 to explain the algorithmic principle of fixed-point operations.
- We can represent $(5.555)_{10}=(5.8E147AE147AE147AE148)_{16}$ and $(4.444)_{10}=(4.71A9FBE76C8B43958106)_{16}$
- Reg [15:0] a=5.555*$2^8$ // Value of a= 05_8E, Reg [15:0] b=4.444*$2^8$ // Value of b= 04_72
- If we multiply the a and b, we get: 16'h05_8E * 16'h04_71 =32'h0018_ABAE
- To get the product in fixed point representation, we divide the intermediate result by 256 (i.e., 28).
- In the binary operation, the original number is shifted to right by eight bits: 32'h18_ABAE >>> 8 = 32'h000018_AB
- Reg [23:0] product= 32'h18_ABAE >>> 8 // Value of product is 0018_AB= 24.66796875
- Actual product= 24.68642

# HARDWARE IMPLEMENTATION OF BAS WITH FIXED POINT REPRESENTATION

- **Fixed Point BAS Input and Output signals:**
- Initial Position inputs x and y are represented as 16 bit signed wires. 8 bits are reserved for integer and 8 bits are reserved for fractional part. Range from -128 to 127. 99609375
- Number of iterations input is represented as 9 bit unsigned wire. Range from 0 to 511.
- Inputs clock, reset and load signals are 1 bit wires.
- Two 9 bit signed input seeds for twin LFSR to generate random numbers from 0.00390625 to 0.99609375 and -2 to -0.00390625. 8 bits are reserved for fractional part and 1 bit is used as sign bit.
- Input sensing distance and flight distance are represented as 14 bit signed wires. 6 bits are reserved for integer and 8 bits are reserved for fractional part. Range from -32 to 31. 99609375
- Best Output Positions inputs x_best and y_best are represented as 16 bit signed registers. 8 bits are reserved for integer and 8 bits are reserved for fractional part. Range from -128 to 127. 99609375.
- Minimum function value achieved is represented as 40 bit signed registers. 32 bits are reserved for integer and 8 bits are reserved for fractional part.

# HARDWARE IMPLEMENTATION OF BAS WITH FIXED POINT REPRESENTATION

- **Block Diagram of BAS implemented in ZedBoard:**



# HARDWARE IMPLEMENTATION OF BAS WITH BF16

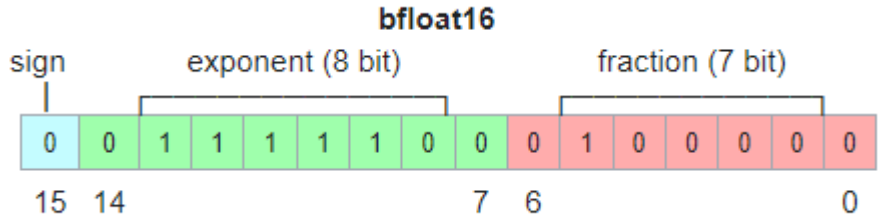
- **Bfloat-16 Representation:**

Bfloat16 has

*i)* ***Sign bit:*** 1bit

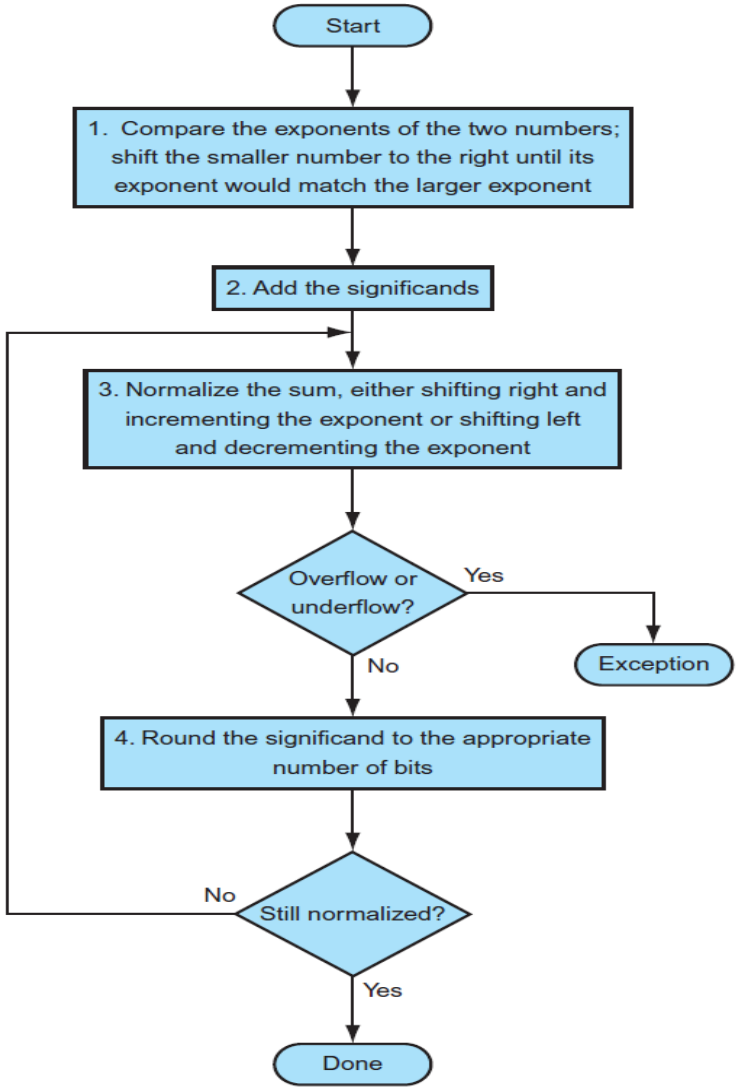
*ii)* ***Exponent width:*** 8-bit

*iii)* ***Significand precision:*** 8-bit of which 7 bits are stored and the MSB 1 is implied.

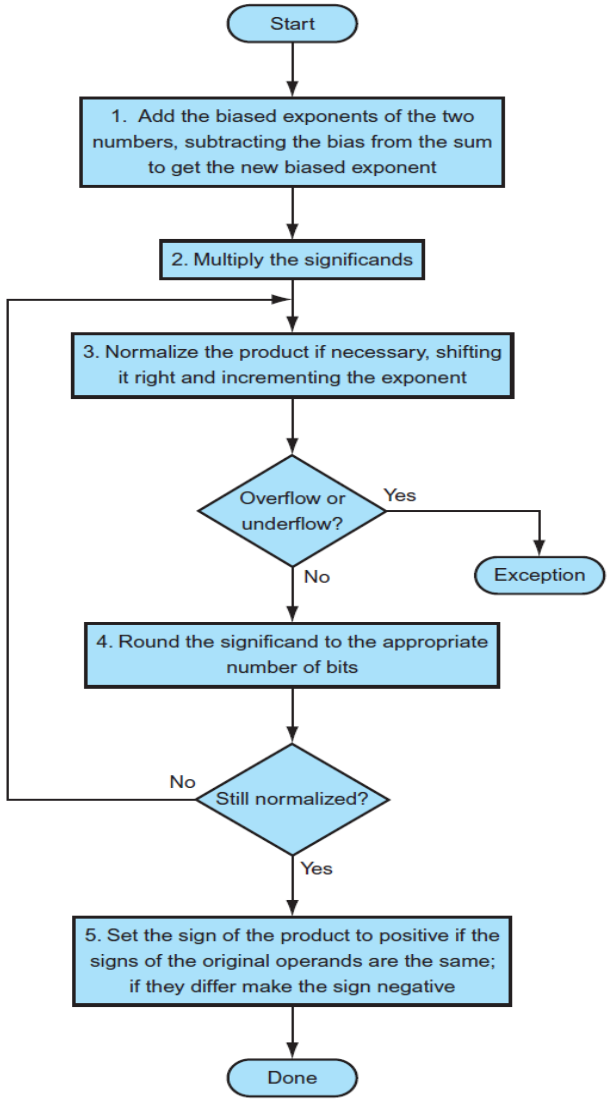



The above image shows the structure of BFloat16 representation. It is basically a truncated IEEE 743 single point precision format.

# BFLOAT16 ADDITION/SUBTRACTION ALGORITHM

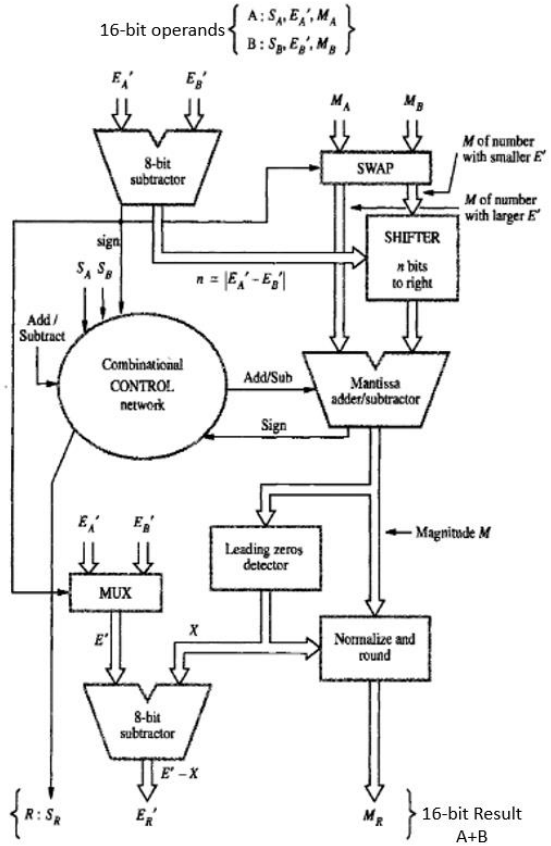


# BFLOAT16 MULTIPLICATION ALGORITHM

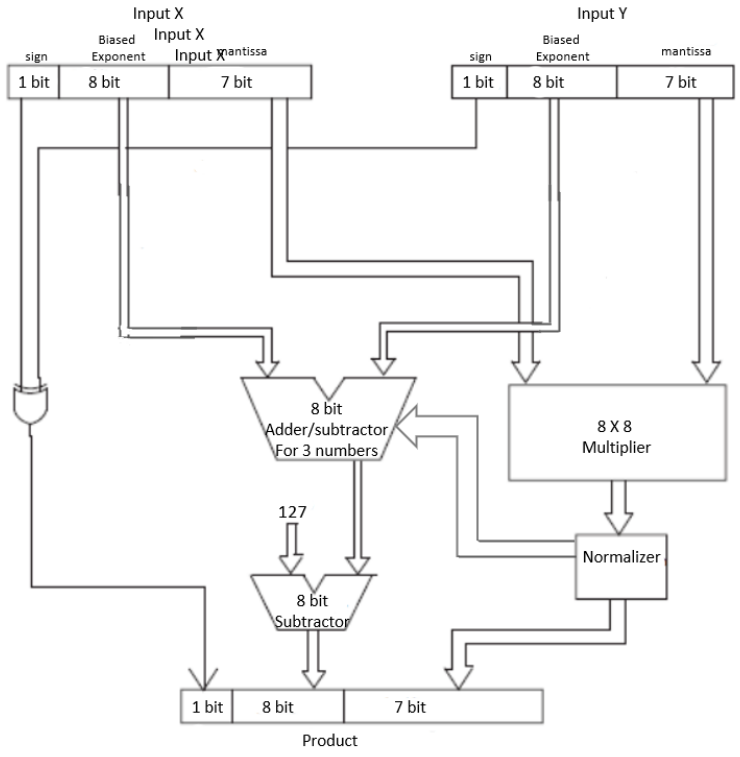


# HARDWARE IMPLEMENTATION OF BAS WITH BF16

- **Bfloat-16 adder and Multiplier Block diagrams:**



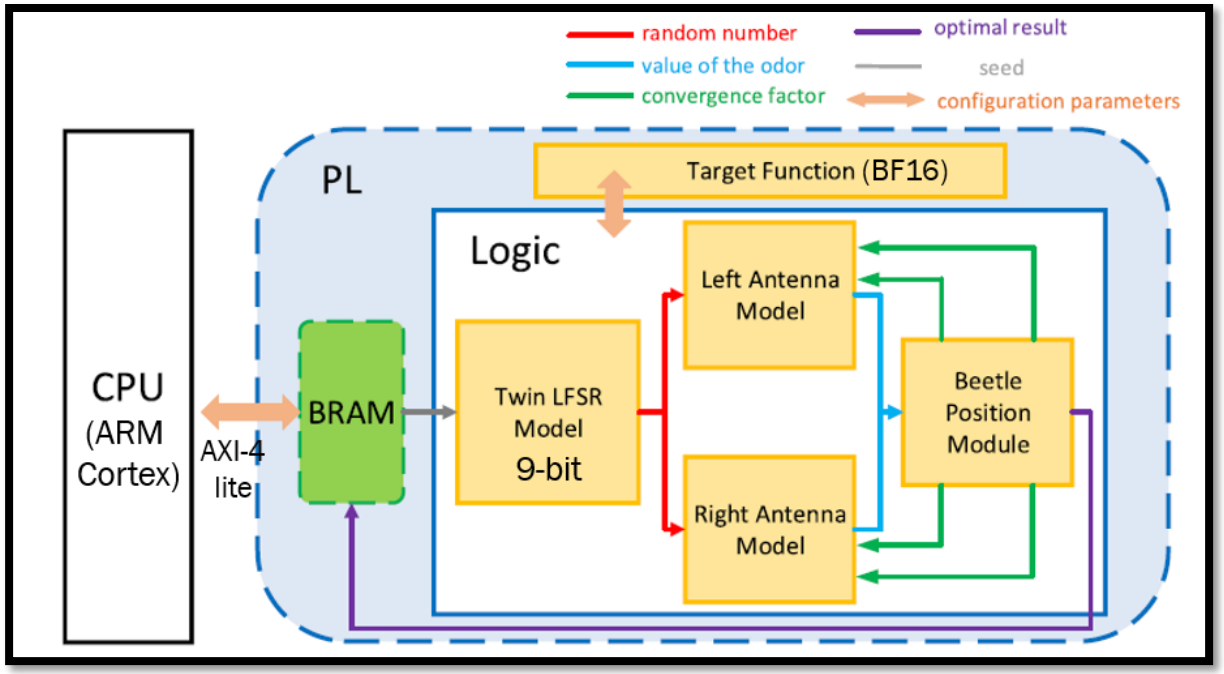

BF16 Adder




BF16 Multiplier

# HARDWARE IMPLEMENTATION OF BAS WITH BF16

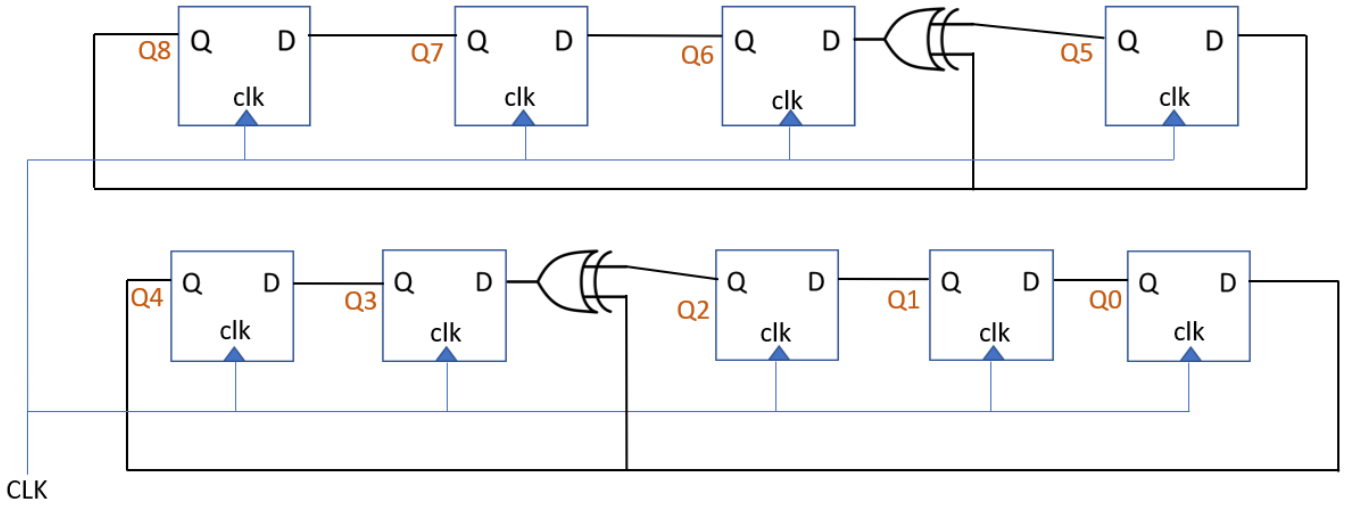
- Below figure shows the block diagram of BAS in BF16 format. Here, all computations are performed in Bfloat-16 format.
- Having Bfloat-16 format increases hardware and decreases clock frequency. But provides a huge increase in the range ($-3.4 \times 10^{38}$ to $-1.17 \times 10^{-38}$ and from $1.17 \times 10^{-38}$ to $3.4 \times 10^{38}$)

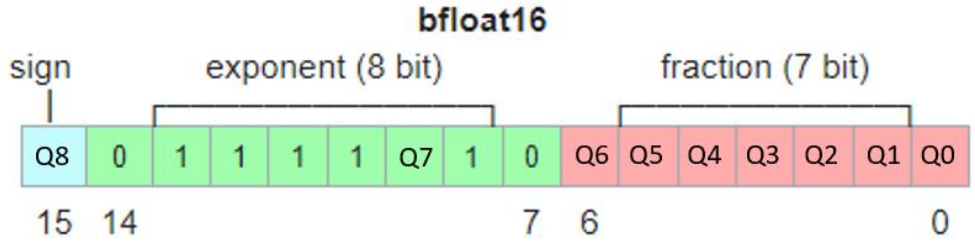


# HARDWARE IMPLEMENTATION OF BAS WITH BF16

- **Structure of LSFR:**

Here, the output of the nine bit LSFR is given to the selected bits of the Bfloat-16 format. As shown below:




Range achieved with this is 0.03125 to 0.99609375 and -0.99609375 to 0.03125

# HARDWARE IMPLEMENTATION OF BAS WITH BF16

- **Configuration Parameters:**

The configuration parameters are

i. Reset Signal

ii. Initial position (xi, yi) in Bfloat-16 format

iii. Seeds (seed1 and seed2 of 9 bit) for the twin lfsr

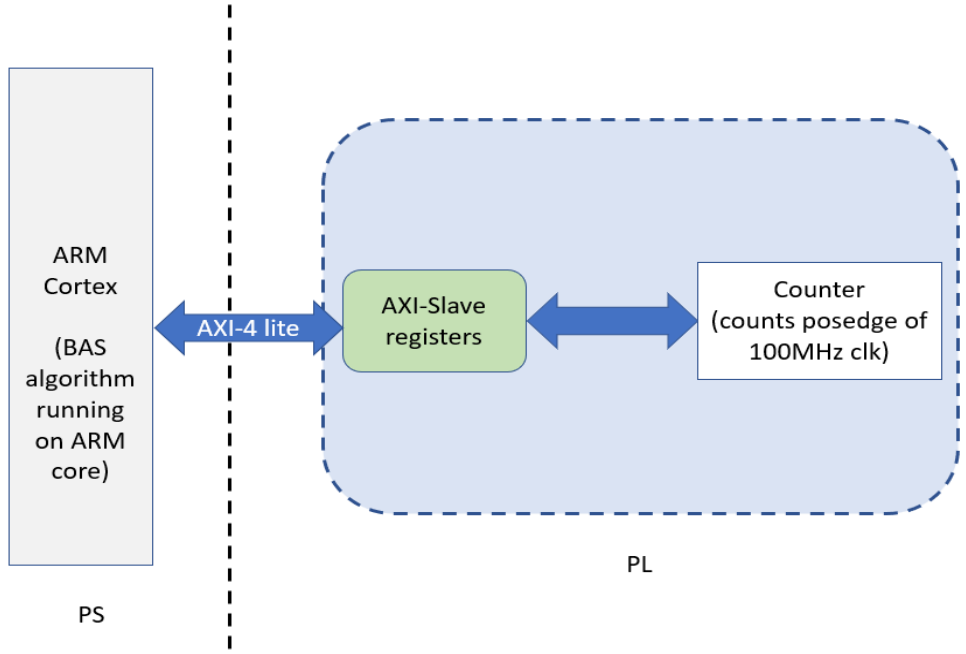
iv. Number of iterations (in 9-bit unsigned format)

v. Sensing Distance (p) in BF16 format

vi. Flight Distance (ε) in BF16 format

# IMPLEMENTATION ON ARM PROCESSOR

- Here, the BAS algorithm is running on the ARM cortex processor on the ZedBoard.
- To calculate the latency (amount of time required to complete the execution of the C-code on the processor), we design a counter that counts up at every positive edge of the clock (100MHz).
- So, the number of counts counted by the counter divided by $10^8$ will give the latency of the system.



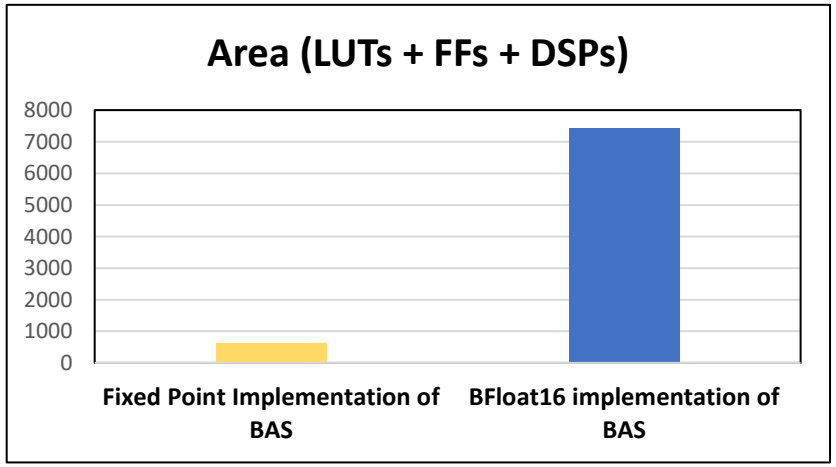
# SYNTHESIS RESULTS

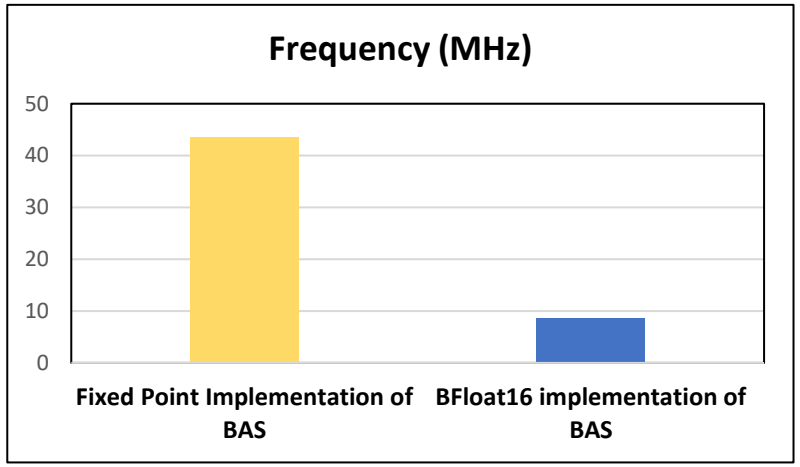
- **Comparison of Fixed-Point and BFloat16 implementation of BAS:**

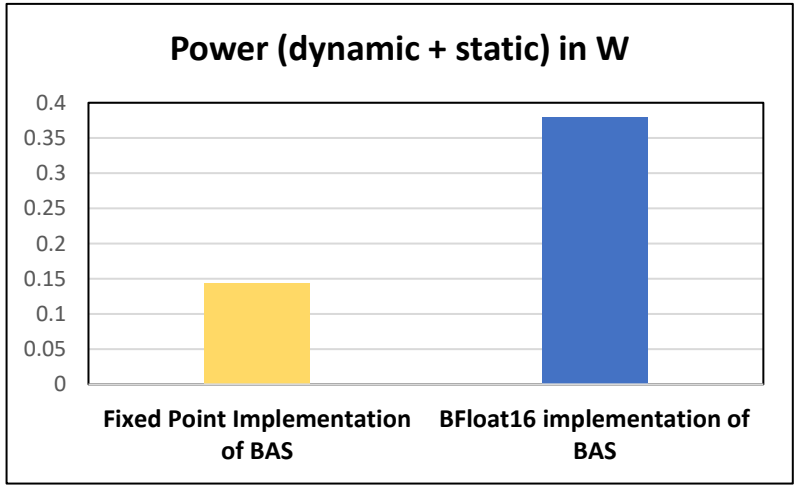
| | | Fixed Point Implementation of BAS | BFloat16 implementation of BAS |
|---|---|---|---|
| **Maximum Operating Frequency** | | 40.90MHz | 8.6MHz |
| **Area** | **LUTs** | 545 | 7288 |
| | **Flip-flops** | 153 | 139 |
| | **DSP** | 30 | 0 |
| **Power** | **Dynamic** | 0.050W | 0.271W |
| | **Static** | 0.105W | 0.108W |

# SYNTHESIS RESULTS

- **Comparison of Fixed-Point and BFloat16 implementation of BAS:**

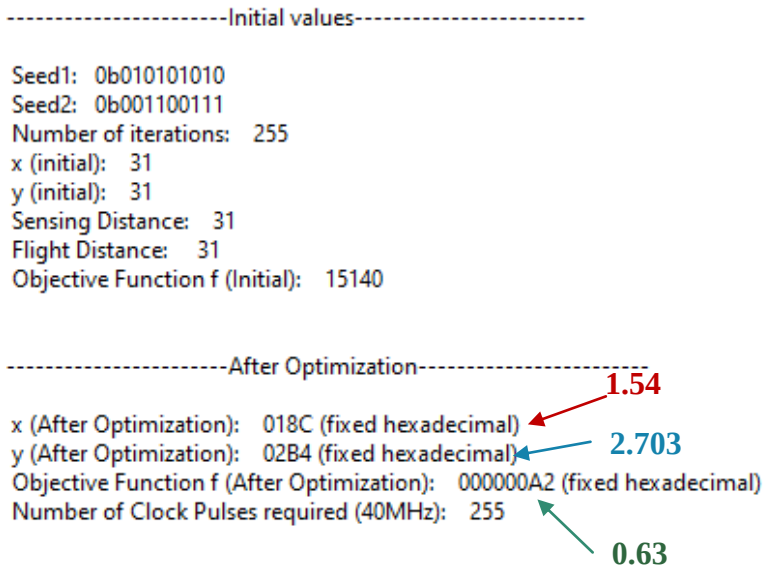


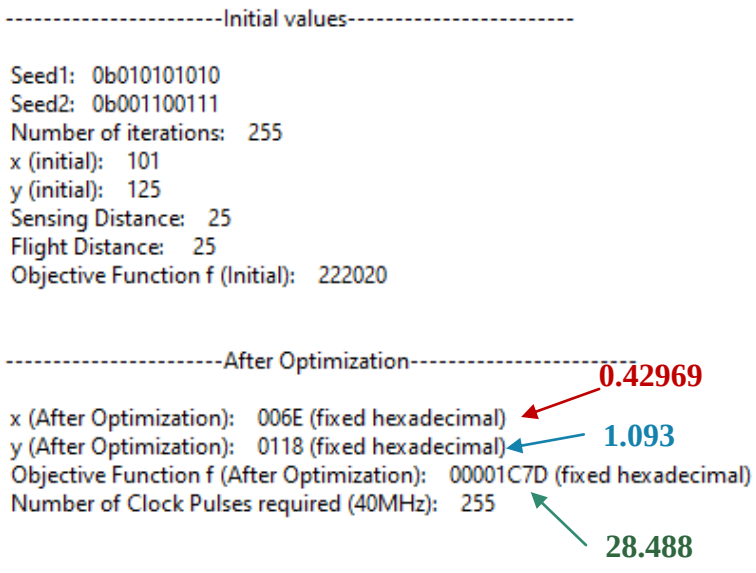



# SIMULATION RESULTS

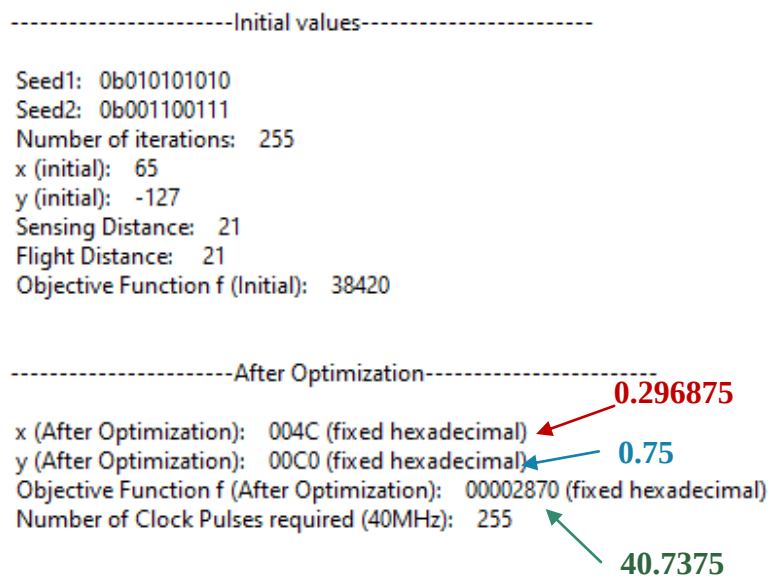
- **Output of BAS algorithm running on Fixed Point Implementation:**




Latency = 6.375us




Latency = 6.375us



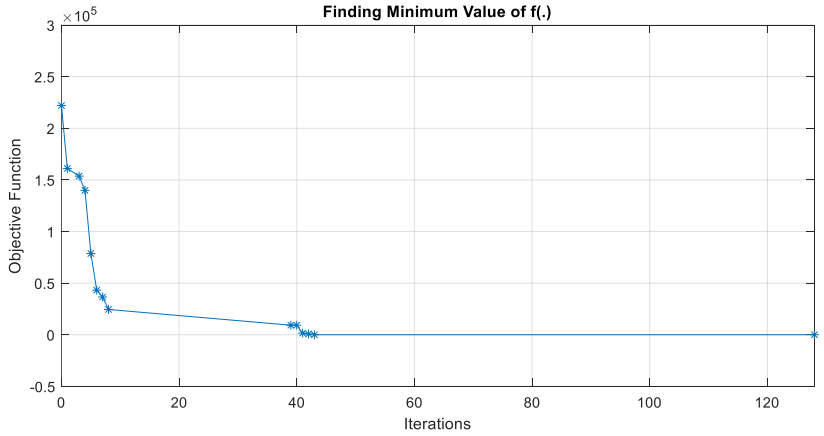

Latency = 6.375us

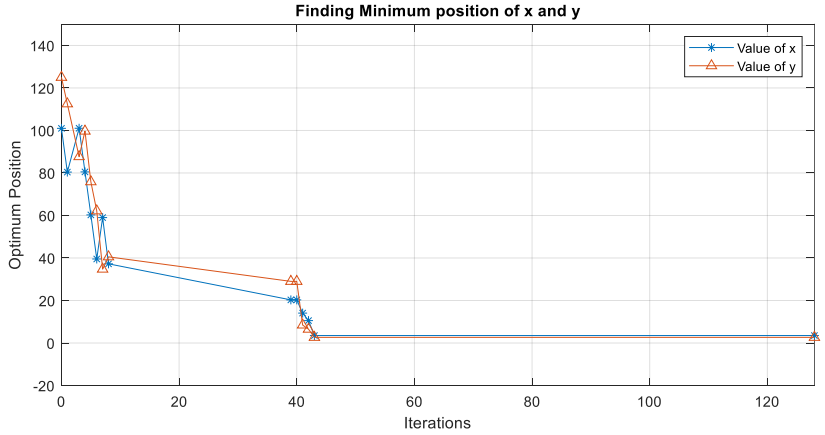
Average Latency = 6.375us

# SIMULATION RESULTS

- **Output of BAS-fixed point interfaced with ARM Cortex:**

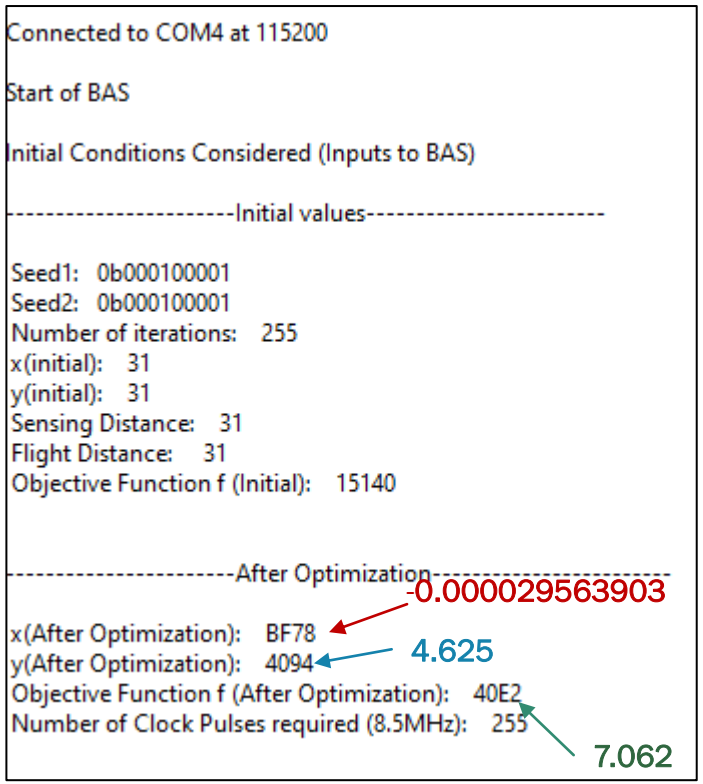
Considered the second example from the simulation where the initial position is (101, 125). The plot of how the optimization works is shown below:



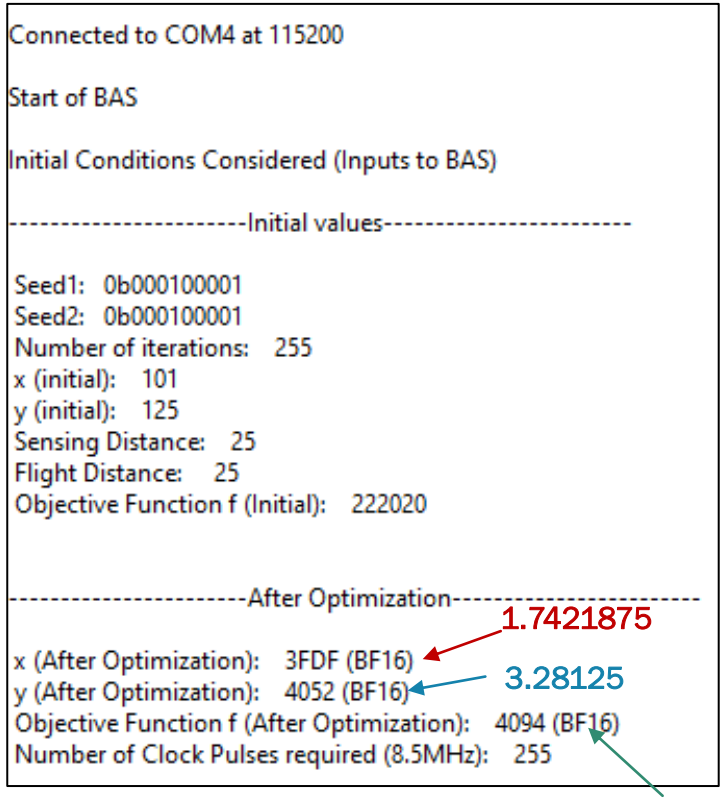



# SIMULATION RESULTS

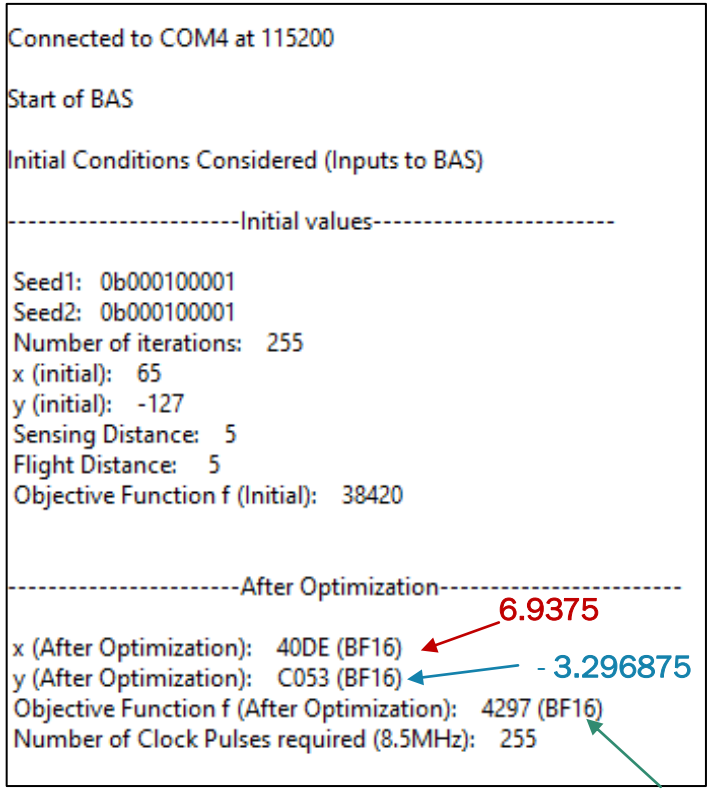
- **Output of BAS-BF16 interfaced with ARM Cortex:**




Latency = 30us




Latency = 30us



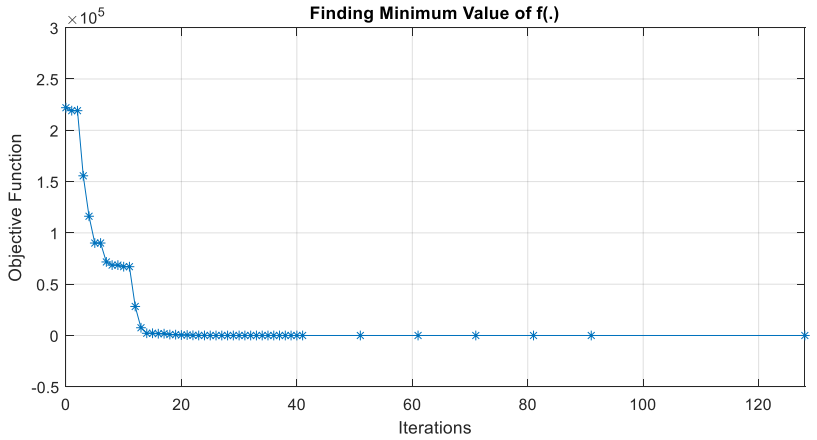

Latency = 30us

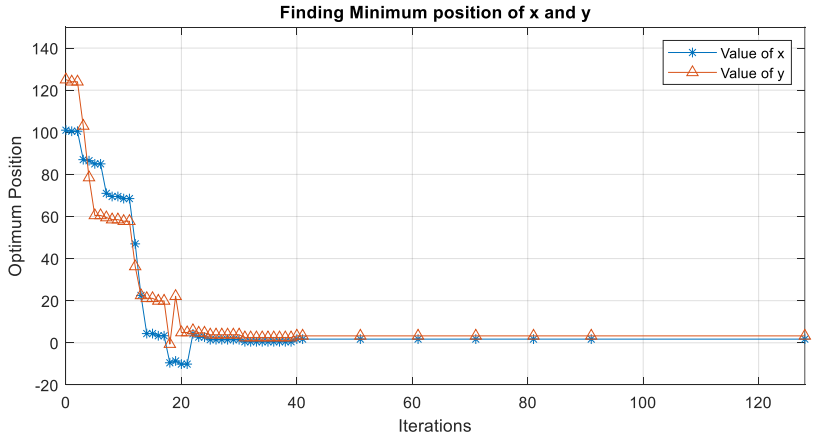
Average Latency = 30us

# SIMULATION RESULTS

- **Output of BAS-BF16 interfaced with ARM Cortex:**

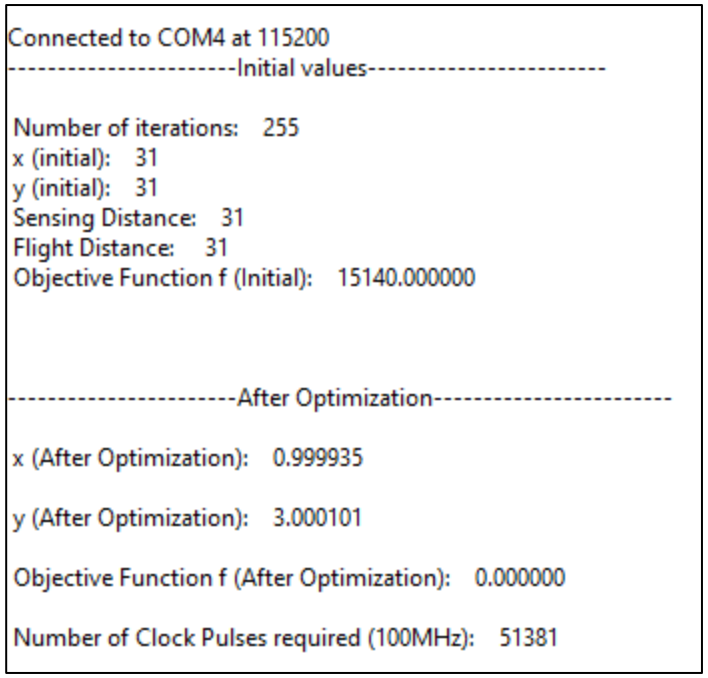
Considered the second example from the simulation where the initial position is (101, 125). The plot of how the optimization works is shown below:



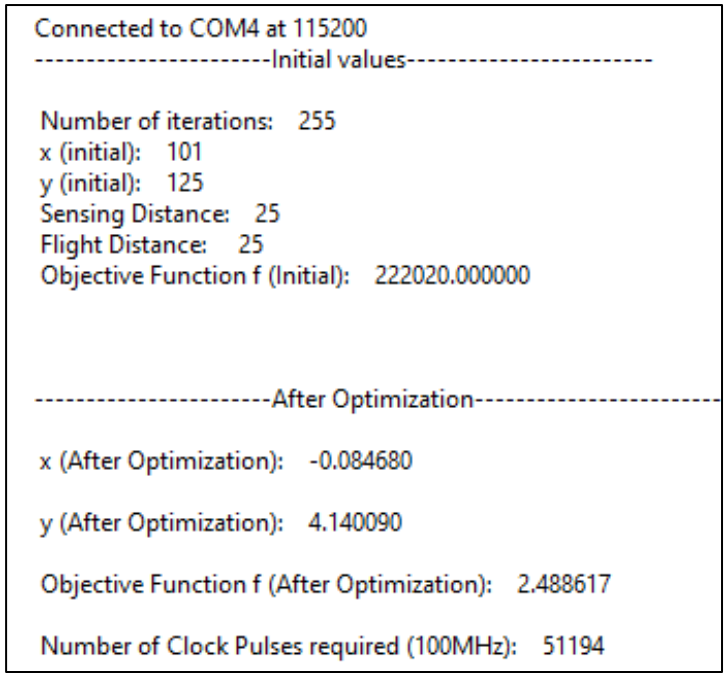



# SIMULATION RESULTS

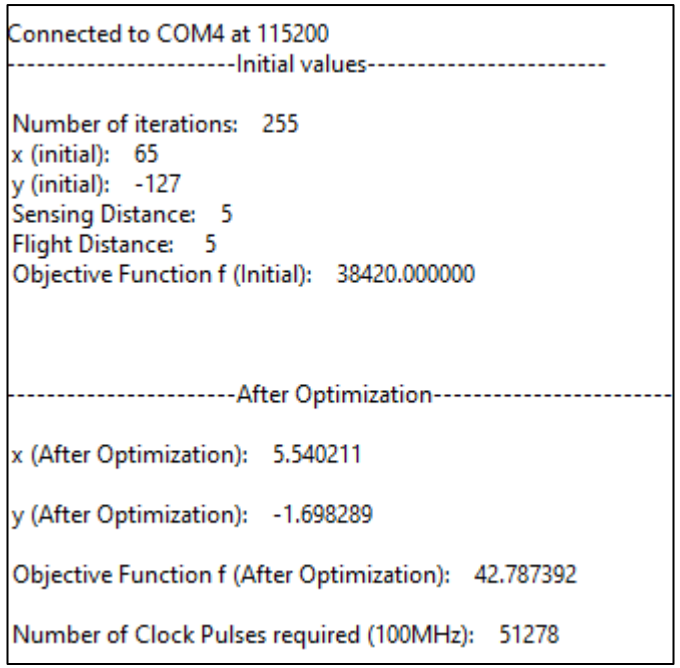
- **Output of BAS algorithm running on ARM Cortex:**




Latency = 513.81us




Latency = 511.94us



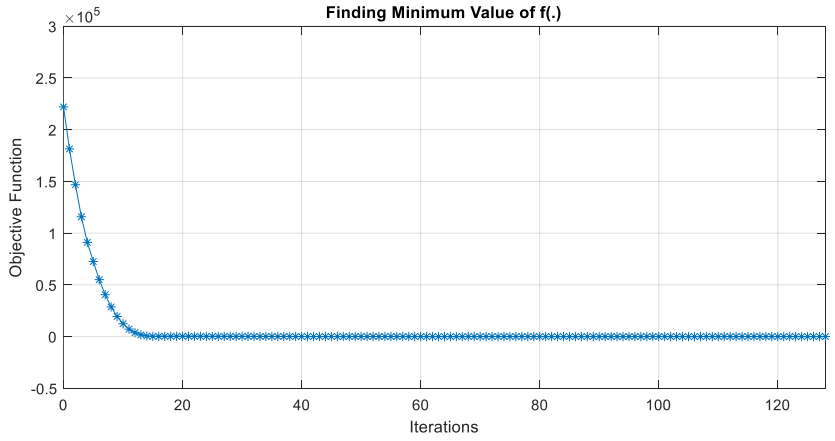

Latency = 512.78us

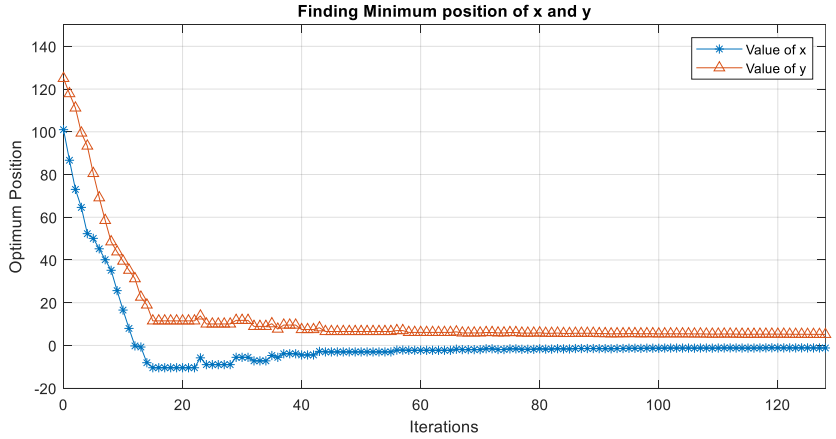
Average Latency = 512us

# SIMULATION RESULTS

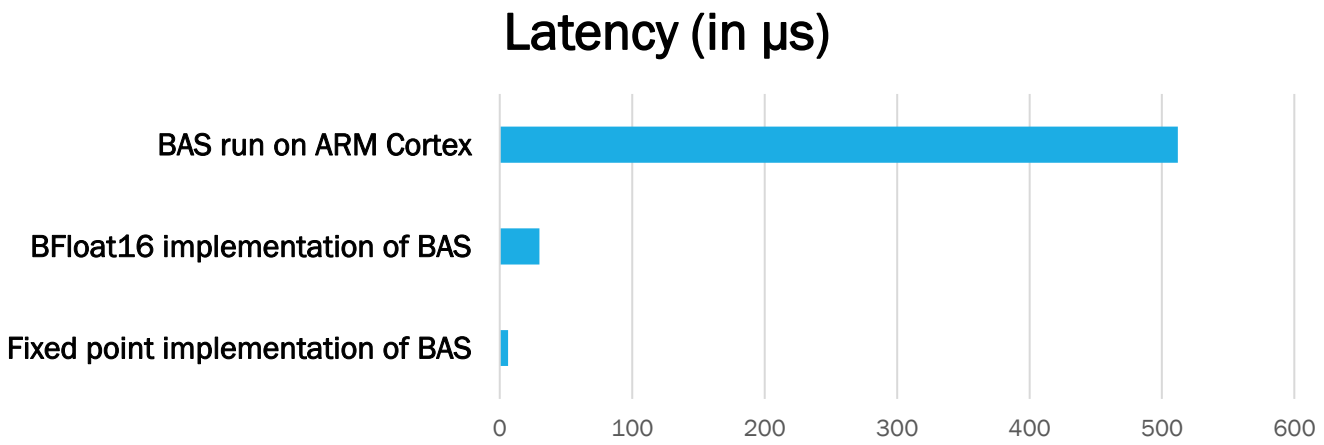
- **Output of BAS algorithm running on ARM Cortex:**

Considered the second example from the simulation where the initial position is (101, 125). The plot of how the optimization works is shown below:



# CONCLUSION




- We observe that there is more than 1000% improvement in the latency when we accelerate the BAS algorithm with the PL (FPGA fabric).
- The PS (ARM Cortex) is working at a frequency of 666.6667MHz clock. When the BAS algorithm is run on the ARM core, the counter counts 51,200 clock cycles of 100MHz clock.
- **So, by hardware acceleration of BAS algorithm, we are saving more than 300,000 clock cycles of the embedded processor which can be used for other crucial operation.**

# QUESTIONS

# REFERENCES


[1] Z. Yue, G. Li, X. Jiang, S. Li, J. Cheng and P. Ren, "A Hardware Descriptive Approach to Beetle Antennae Search," in *IEEE Access*, vol. 8, pp. 89059-89070, 2020, doi: 10.1109/ACCESS.2020.2993600

[2] S. Xie, X. Chu, M. Zheng, and C. Liu, ``Ship predictive collision avoidance method based on an improved beetle antennae search algorithm," *Ocean Eng.*, vol. 192, Nov. 2019, Art. no. 106542.

[3] S. Xie, V. Garofano, X. Chu, and R. R. Negenborn, ``Model predictive ship collision avoidance based on Q-learning beetle swarm antenna search and neural networks," *Ocean Eng.*, vol. 193, Dec. 2019, Art. no. 106609.

[4] S. Xie, X. Chu, C. Liu, and M. Zheng, ``Marine diesel engine speed control based on adaptive state-compensate extended state observer-backstepping method," *Proc. Inst. Mech. Eng., I, J. Syst. Control Eng.*, vol. 233, no. 5, pp. 457471, May 2019.

# THANK YOU