Harris Hawk Optimisation for Digital Circuits

Digital VLSI project

Shanmukh Alle
Naga Sreevatsava

M.V.S.Sravan
B.Sreyamsh

Objective

- Aim : To reduce the leakages and delays by optimizing the widths and lengths of transistor in a digital circuit
- Challenges
  - Optimization problem is multi-Objective while the optimization algorithm is single objective
- Optimize
  - 6 delays
  - 8 leakages

Goals

- Implement Harris Hawk in matlab and python
- Find minima of mathematical function using the above implementation
- Optimise adder circuit using the above implementation

Status:- Done with all the three.

Harris Hawk Optimization Algorithm

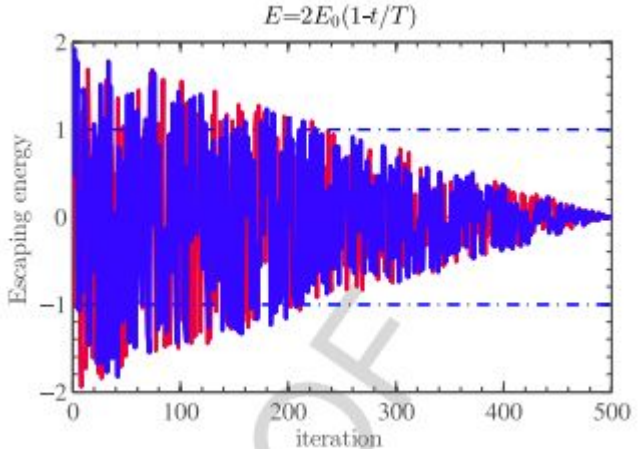
Overview

- HHO is inspired by the exploring a prey, surprise pounce, and different attacking strategies of Harris hawks.
- HHO is a population-based, gradient-free optimization technique; hence, it can be applied to any optimization problem subject to a proper formulation.
- We use this nature inspired harris hawk optimisation for minimising leakage current and transmission delays. By varying the transistor widths and lengths.

Algorithm

- The Algorithm is based on how Harris hawks hunt its prey.
- It has two main phases
  - Exploration Phase
  - Exploitation phase
- Transition between exploration and exploitation is done based on the below equation

$$E = 2E_0(1 - \frac{t}{T})$$



Exploration Phase

- In HHO, the Harris' hawks perch randomly on some locations and wait to detect a prey based on two strategies.
- If we consider an equal chance q for each perching strategy, which is modeled in Equation below.
- for the condition of q < 0.5, or perch on random tall trees (random locations inside the group's home range), which is modeled in Eq. (1) for condition of q ≥ 0.5

$$X(t+1) = \begin{cases} X_{rand}(t) - r_1 \left| X_{rand}(t) - 2r_2 X(t) \right| & q \geq 0.5 \\ (X_{rabbit}(t) - X_m(t)) - r_3(LB + r_4(UB - LB)) & q < 0.5 \end{cases}$$

Exploitation Phase

- Exploitation phase has four update types.
  - Soft besiege
    - Rabbit has enough energy to move so harris hawks move softly towards the rabbit.
  - Hard besiege
    - Weak rabbit rabbit so hawks perform surprize attack
  - Soft besiege with progressive dives
  - Hard besiege with progressive dives

Soft besiege

- When r ≥ 0.5 and |E| ≥ 0.5, the rabbit still has enough energy, and try to escape by some random misleading jumps but finally it cannot.
- During these attempts, the Harris' hawks encircle it softly to make the rabbit more exhausted and then perform the surprise pounce.
- This behavior is modeled by the following rules

$$X(t+1) = \Delta X(t) - E\,|JX_{rabbit}(t) - X(t)|$$
$$\Delta X(t) = X_{rabbit}(t) - X(t)$$

Hard Beseige

- When r ≥0.5 and |E| <0.5, the prey is so exhausted and it has a low escaping energy.
- In addition, the Harris' hawks hardly encircle the intended prey to finally perform the surprise pounce.
- In this situation, the current positions are updated using the following equation

$$X(t+1) = X_{rabbit}(t) - E\,|\Delta X(t)|$$

Levy Flight function

- The dive is based on Levy-Flight based patterns that can be generated using the following rule

$$Z = Y + S \times LF(D)$$

- The levy flight function calculated using.

$$LF(x) = 0.01 \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}}, \sigma = \left(\frac{\Gamma(1+\beta) \times sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}) \times \beta \times 2^{(\frac{\beta-1}{2})}} \right)^{\frac{1}{\beta}}$$

where u, v are random values in (0,1).

Soft Besiege with rapid dives

- To perform a soft besiege, we supposed that the hawks can evaluate (decide) their next move based on the following rule

$$Y = X_{rabbit}(t) - E\left|JX_{rabbit}(t) - X(t)\right|$$

$$Z = Y + S \times LF(D)$$

$$X(t+1) = \begin{cases} Y & if F(Y) < F(X(t)) \\ Z & if F(Z) < F(X(t)) \end{cases}$$

Hard Besiege with rapid dives

- When |E| <0.5 and r <0.5, the rabbit has not enough energy to escape and a hard besiege is constructed before the surprise pounce to catch and kill the prey.
- The situation of this step in the prey side is similar to that in the soft besiege, but this time, the hawks try to decrease the distance of their average location with the escaping prey.

$$X(t+1) = \begin{cases} Y & if F(Y) < F(X(t)) \\ Z & if F(Z) < F(X(t)) \end{cases}$$

where Y and Z are obtained using:

$$Y = X_{rabbit}(t) - E\,|JX_{rabbit}(t) - X_m(t)|$$

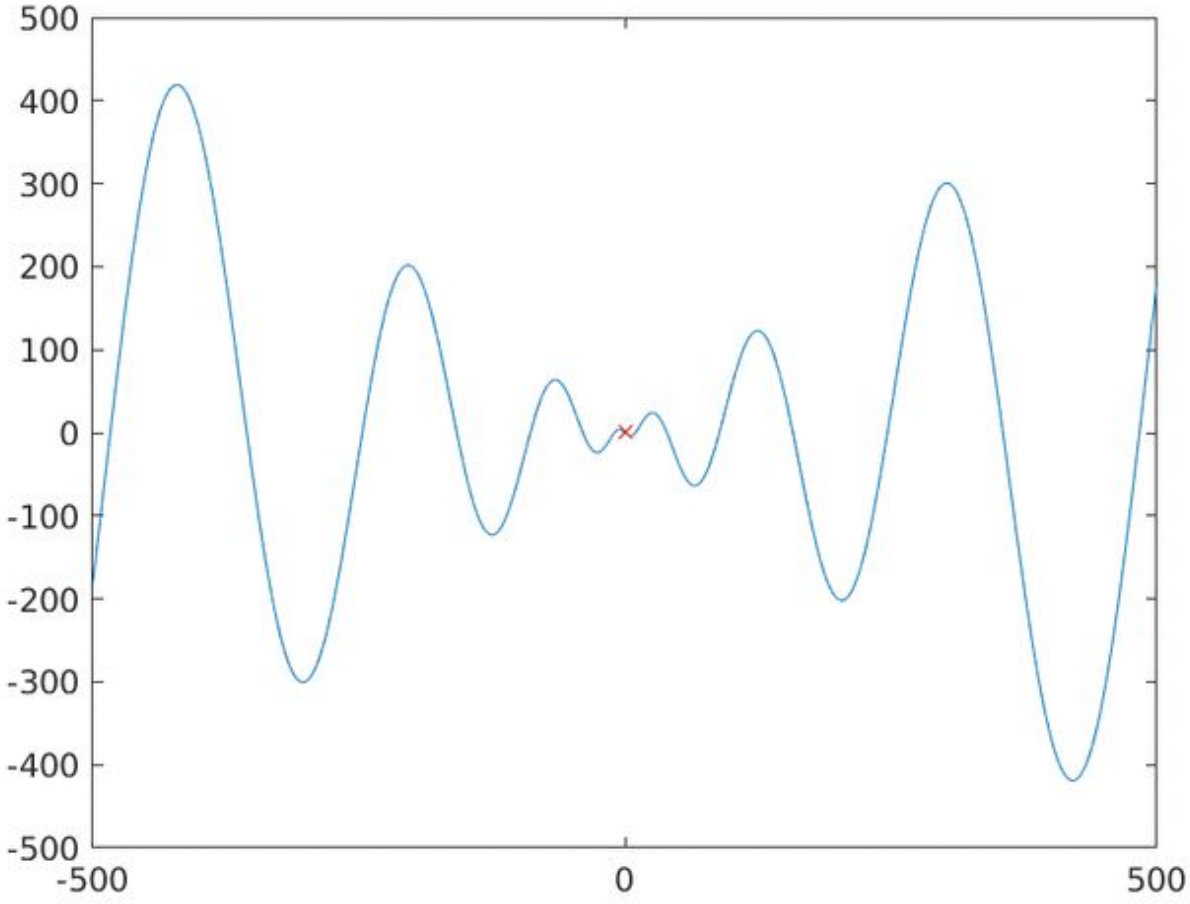
$$Z = Y + S \times LF(D)$$

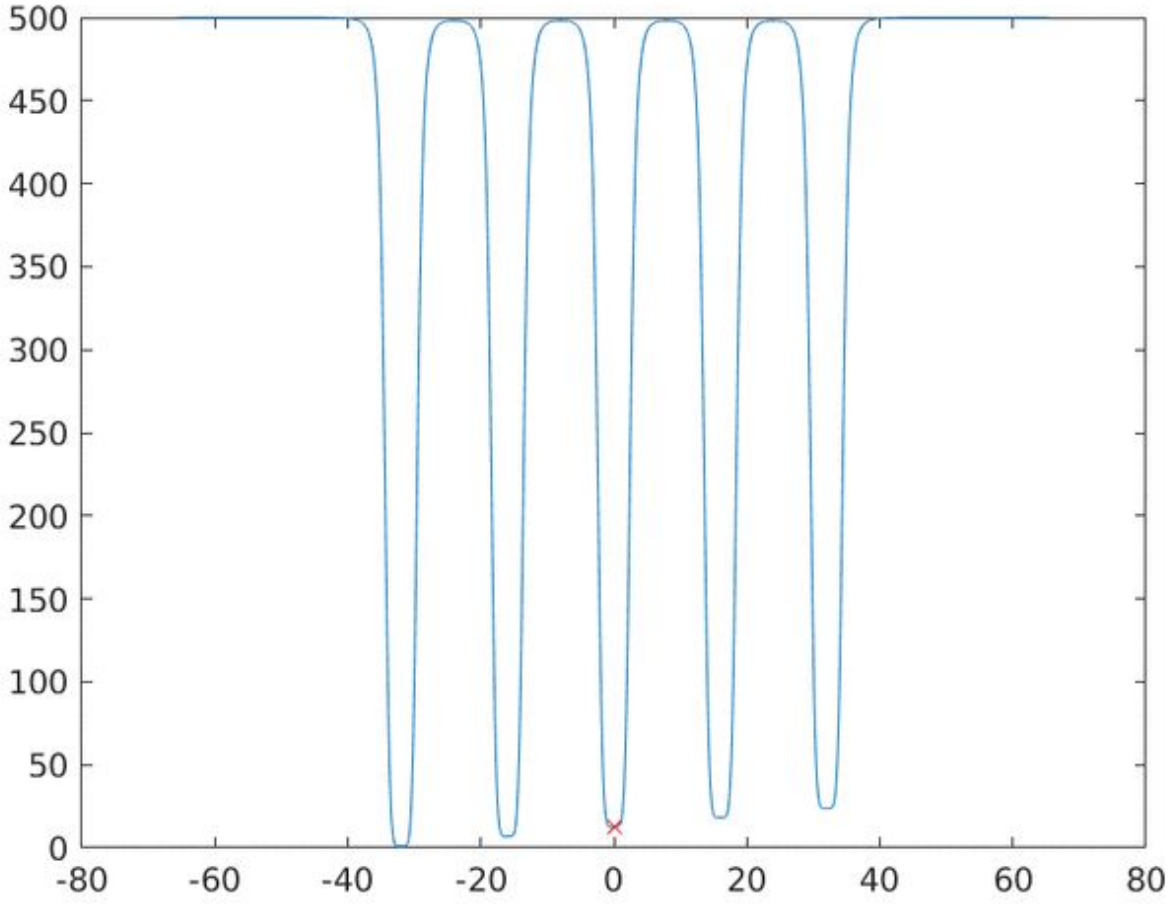
Pseudocode

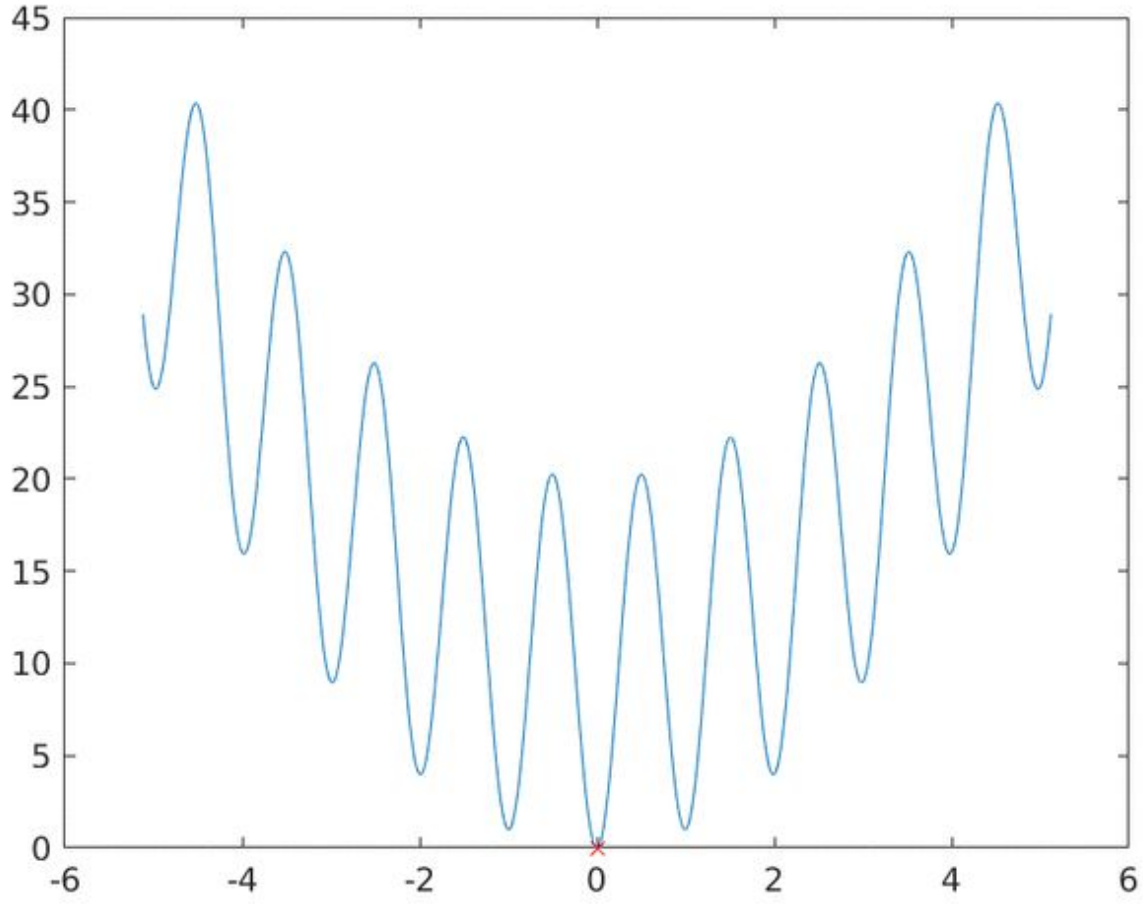
Algorithm 1 Pseudo-code of HHO algorithm

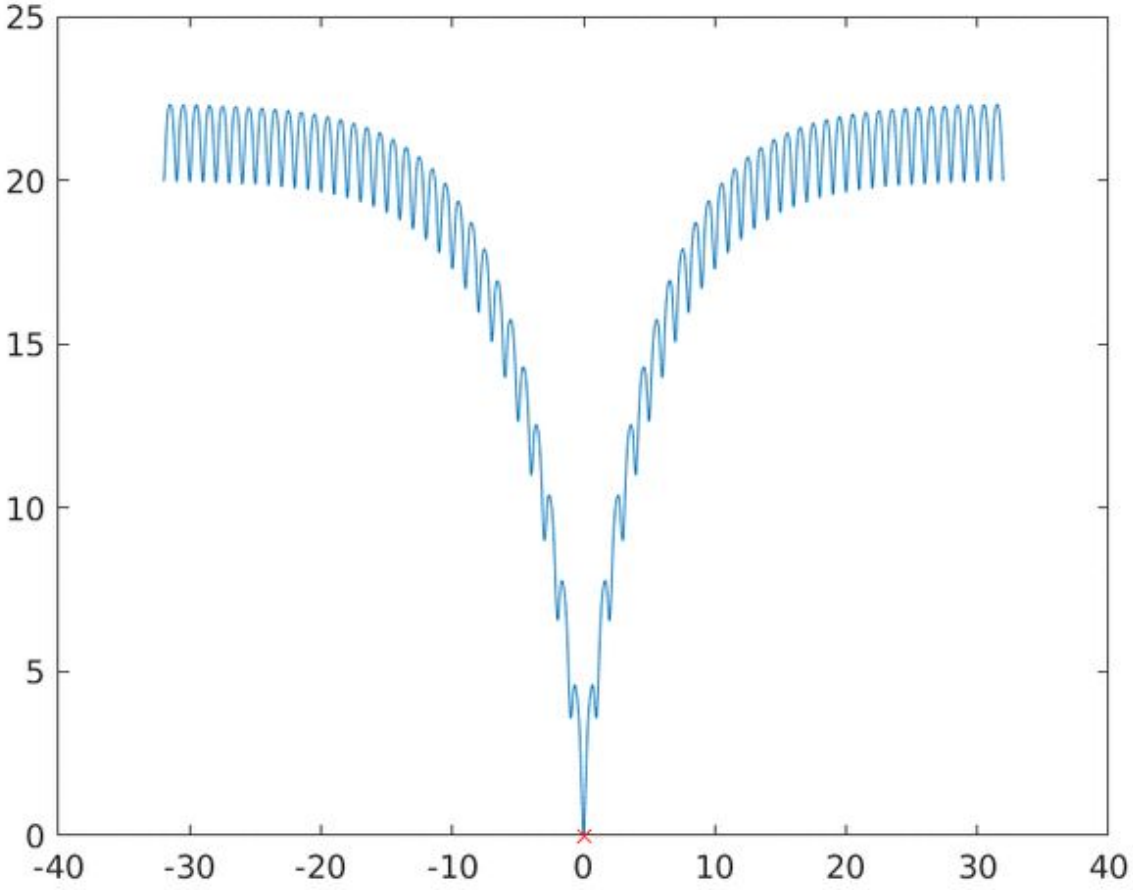
Inputs: The population size N and maximum number of iterations T
Outputs: The location of rabbit and its fitness value
Initialize the random population $X_i (i = 1, 2, \ldots, N)$
while (stopping condition is not met) **do**
  Calculate the fitness values of hawks
  Set X_{rabbit} as the location of rabbit (best location)
  for (each hawk (X_i)) **do**
    Update the initial energy E_0 and jump strength J ▷ E_0 =2*rand()-1, J=2(1-rand())
    Update the E using Eq. (3)
    if $(|E| \geq 1)$ **then** ▷ Exploration phase
      Update the location vector using Eq. (1)
    if $(|E| < 1)$ **then** ▷ Exploitation phase
      if $(r \geq 0.5$ and $|E| \geq 0.5)$ **then** ▷ Soft besiege
        Update the location vector using Eq. (4)
      else if $(r \geq 0.5$ and $|E| < 0.5)$ **then** ▷ Hard besiege
        Update the location vector using Eq. (5)
      else if $(r < 0.5$ and $|E| \geq 0.5)$ **then** ▷ Soft besiege with progressive rapid dives
        Update the location vector using Eq. (10)
      else if $(r < 0.5$ and $|E| < 0.5)$ **then** ▷ Hard besiege with progressive rapid dives
        Update the location vector using Eq. (11)
Return X_{rabbit}

Results


500
400
300
200
100
0
-100
-200
-300
-400
-500
-500
0
500

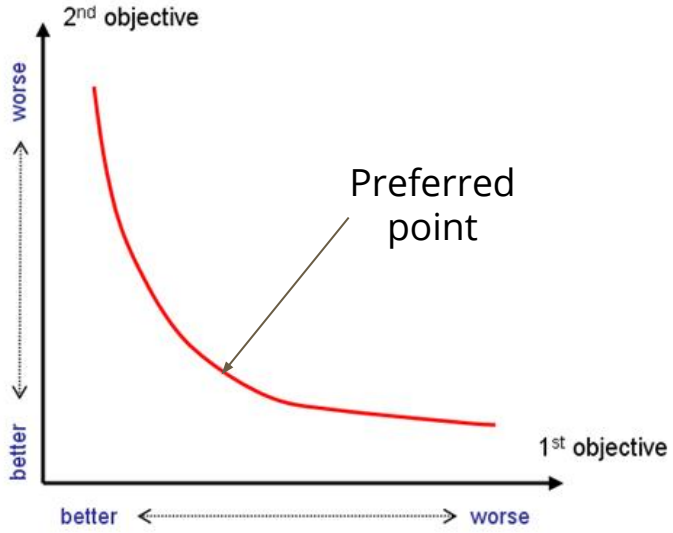

500
450
400
350
300
250
200
150
100
50
0
-80
-60
-40
-20
0
20
40
60
80


45
40
35
30
25
20
15
10
5
0
-6
-4
-2
0
2
4
6


25
20
15
10
5
0
-40
-30
-20
-10
0
10
20
30
40

Pareto Optimization Curve

- Extremes represent single objectives while the function is multi objective
- Points on the curve are obtained by varying weightage for the individual objectives
- Points above the curve are feasible and below are infeasible
  - 1st Objective : Leakage
  - 2nd Objective : Delay
- Since the optimization is randomized
  - The pareto curve cannot be reproduced
  - Hence multiple simulations were run and the lower Skyline is used as pareto curve



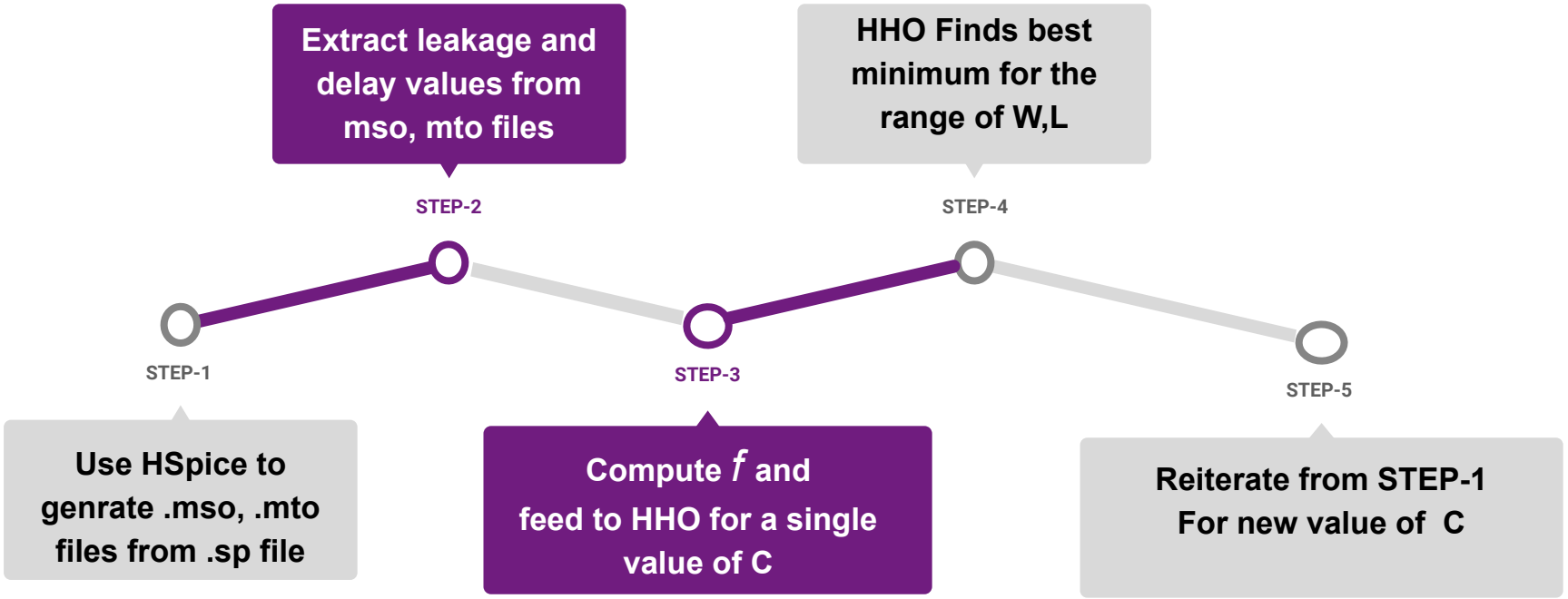
Objective Function

- Delays represented by max(6-delays).
- Leakages represented by sum(8-leakages)
- Objective function to convert

$$f = c*max(delays)+(1-C)*sum(leakages)$$

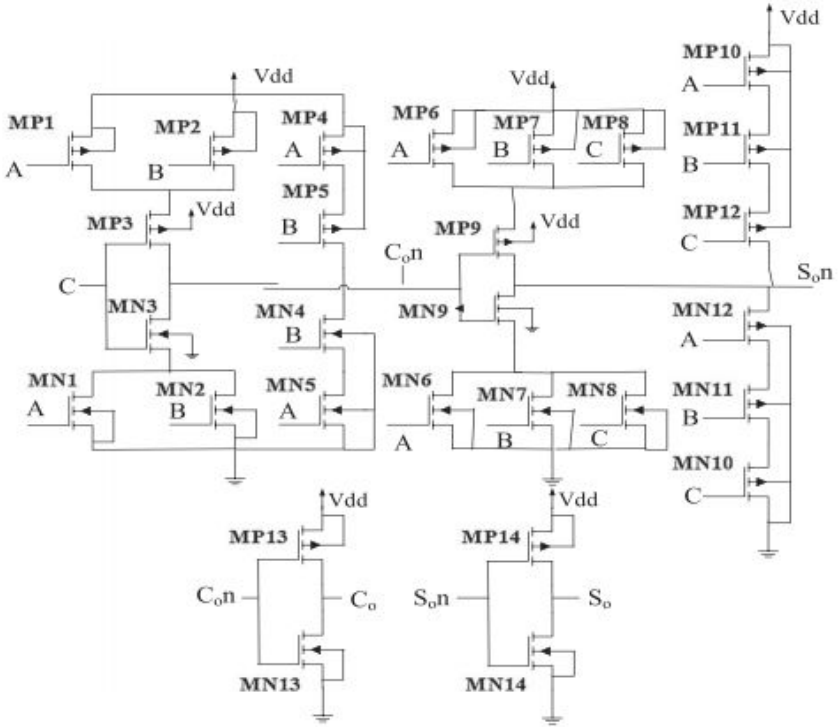
- When C=0, delays are ignored and while C=1, leakages are ignored.
- Trade off for both occurs at some C which the algorithm helps to find.

Pipeline


Extract leakage and delay values from mso, mto files
HHO Finds best minimum for the range of W,L
STEP-2
STEP-4
STEP-1
STEP-3
STEP-5
Use HSpice to genrate .mso, .mto files from .sp file
Compute f and feed to HHO for a single value of C
Reiterate from STEP-1 For new value of C

Full Adder

- 28 Transistors
  - 14 pmos
  - 14 nmos
- Each transistor has 2 parameters
- Hence 56 parameters to search for.

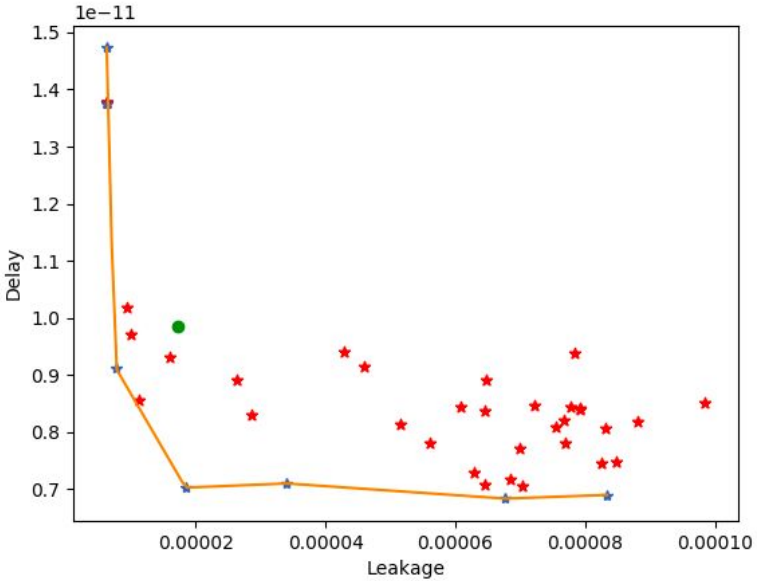


Implementation Details

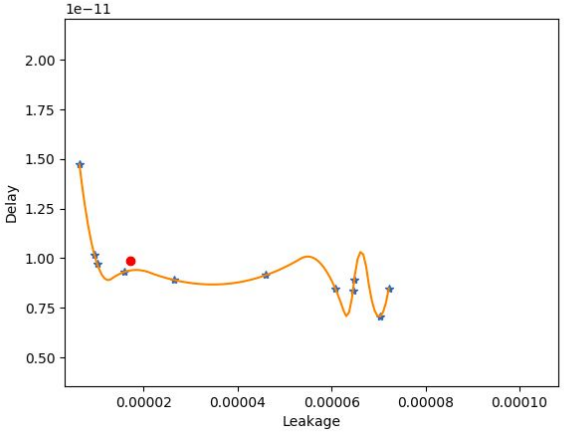
- Take logarithm of objective function as the values of the objectives is very small
- Reason : Leakages are in the order of 10^{-6} while delays are in the range of 10^{-11}
- Taking a logarithm normalizes the values
- Linear variation of weightage value leads to points biased towards one side.
- Use exponentially varying weightage values to get even distribution of points.

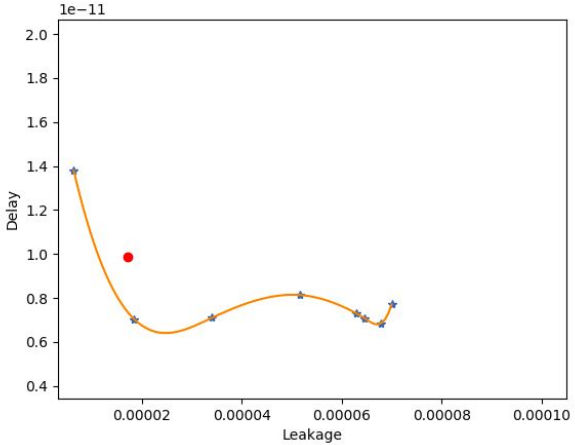
Pareto optimality curve

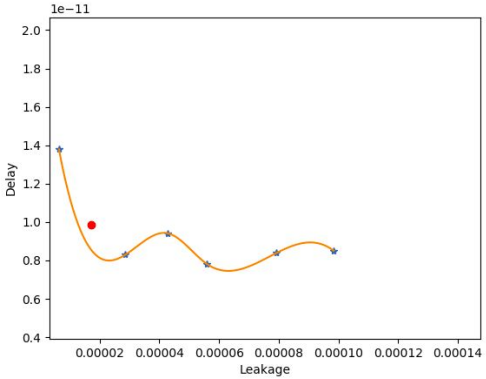
- The skyline is plotted as shown
- When the point with same delay but optimised leakage than default has 41.79% improvement.
- When point with same leakage but optimised delay has 28.73% improvement



Other simulation results

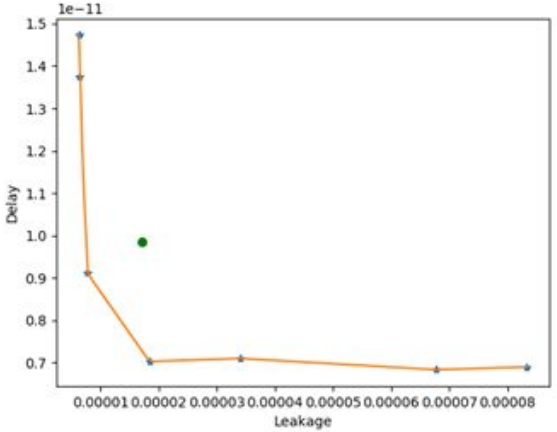




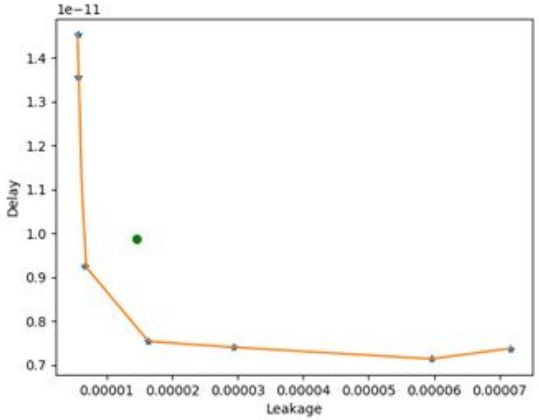


Scaling with temperature variation

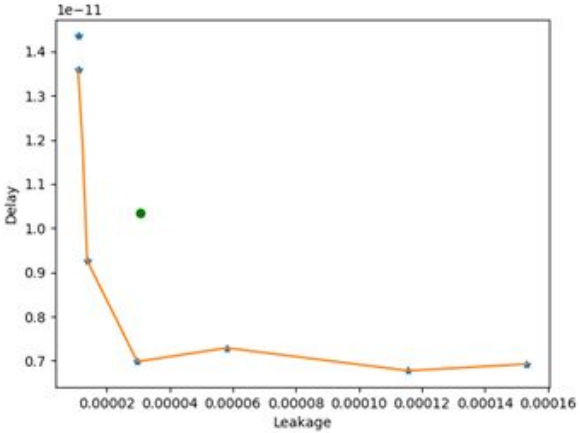
25



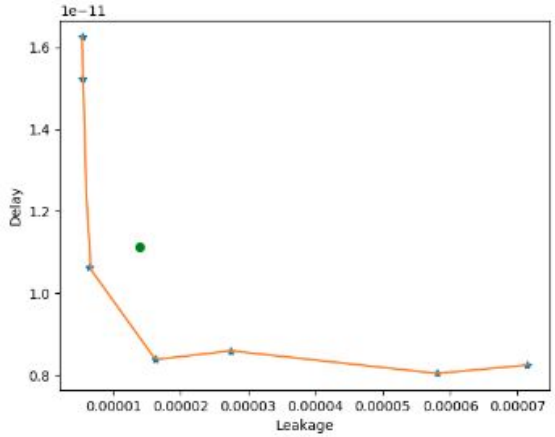
-55



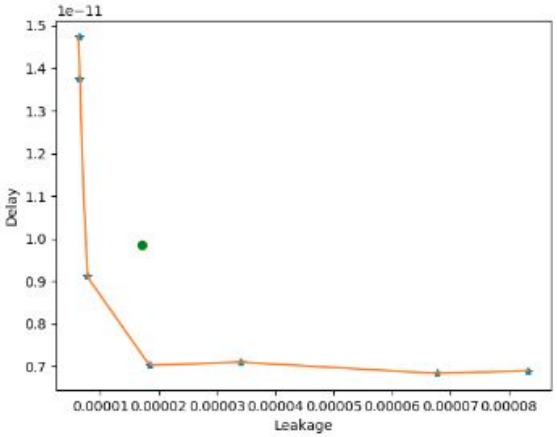
125



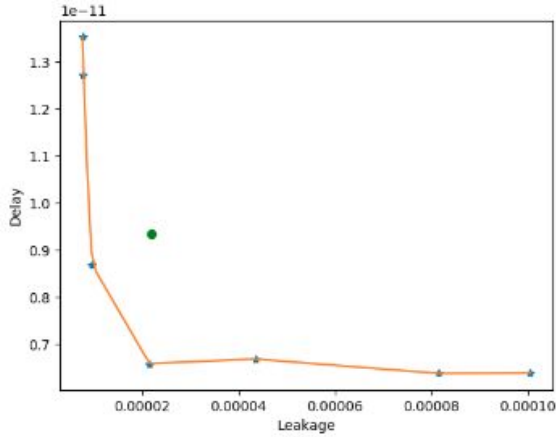
Supply Voltage Scaling -/+10%



0.72 V

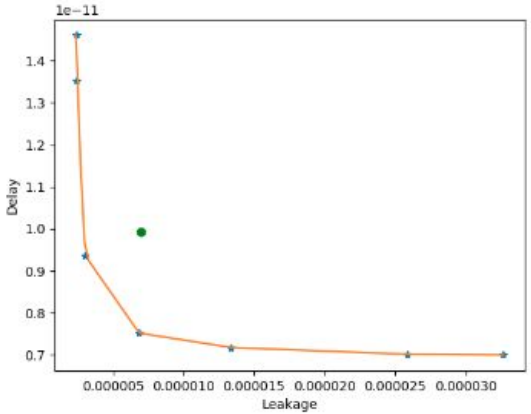


0.8 V

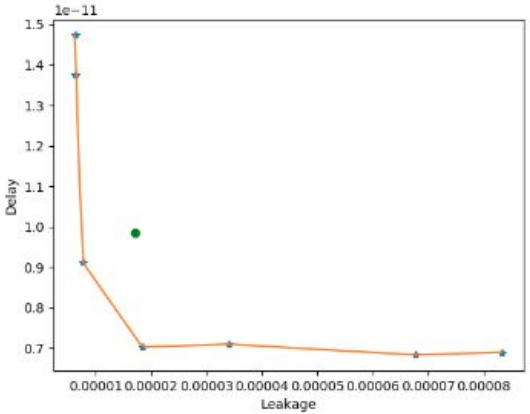


0.88 V

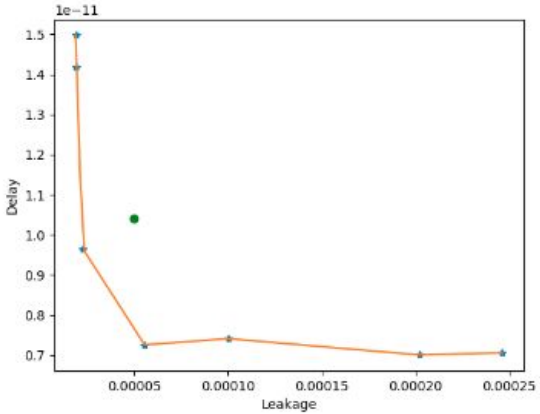
Process Variation +/-10%



+10%



0



-10%