

# “Low Power Design”

2023-2024

Prof. FRUSTACI

## Energy-Delay Optimization of a Three-Stages CMOS Buffer

Date	<18/10/2023>
Document	Final Document

Full Name	ID	E-mail Address
Giorgio Ubbriaco	247284	bbrgrg00h11d086x@studeni.unical.it

# Contents

<b>DEFINITIONS .....</b>	<b>4</b>
DIRECTORIES PROJECT PATHS.....	4
GLOBAL VARIABLES.....	4
GLOBAL FUNCTIONS.....	4
<b>I. DESIGN CONSTRAINTS.....</b>	<b>5</b>
<b>II. TASKS TO BE PERFORMED .....</b>	<b>6</b>
<b>III. ABSTRACT .....</b>	<b>7</b>
<b>1. INTRODUCTION .....</b>	<b>8</b>
1.1. ENERGY MODEL.....	8
1.2. DELAY MODEL.....	9
1.3. OPTIMIZATION MODEL.....	9
<b>2. ANALYSIS AND DESIGN OF A MINIMUM CMOS INVERTER .....</b>	<b>11</b>
2.1. MINIMUM SIZING .....	11
2.2. MINIMUM INVERTER .....	12
2.2.1. <i>Minimum Inverter Disconnected</i> .....	12
2.2.2. <i>Minimum Inverter Connected</i> .....	13
<b>3. ANALYSIS AND DESIGN OF A THREE-STAGE CMOS BUFFER .....</b>	<b>15</b>
3.1. PARAMETERS ANALYSIS .....	16
3.2. PARETO ENERGY-DELAY CURVE EMPIRICALLY BY MONTE CARLO METHOD.....	17
3.3. PARETO ENERGY-DELAY OPTIMIZED BY PYTHON SCIPY ALGORITHM .....	17
3.3.1. <i>Optimal Values Generation</i> .....	17
3.3.2. <i>Optimized Buffer with Optimal Values</i> .....	19
3.4. PARETO ENERGY-DELAY OPTIMIZED BY MATLAB FMINCON ALGORITHM .....	20
3.4.1. <i>Optimal Values Generation</i> .....	20
3.4.2. <i>Optimized Buffer with Optimal Values</i> .....	21
3.5. COMPARISON BETWEEN PYTHON SCIPY ALGORITHM AND MATLAB FMINCON ALGORITHM.....	22
3.5.1. <i>Factor Sizing Generation</i> .....	22
3.5.2. <i>Pareto Energy-Delay Curve</i> .....	23
<b>4. CONCLUSIONS .....</b>	<b>25</b>



# Definitions

Qui sono contenute diverse definizioni utilizzate all'interno del report per semplificare la fluidità delle analisi.

## Directories Project Paths

- data = “./python/buffer-analysis/data”
- ltspice = “./ltspice/models/”
- matlab = “./matlab/”
- rit\_models\_for\_ltspice\_file\_path = “./ltspice/utils/utils-montecarlo-experiments/RIT\_Models\_For\_LTSPICE”

## Global Variables

- nr\_runs = 10000
- l\_min\_pmos = ‘0.1u’
- l\_min\_nmos = ‘0.1u’
- w\_min\_nmos = ‘0.12u’
- S1 = ‘mc(4,0.75)’
- S2 = ‘mc(16,0.75)’
- S\_LOAD = ‘50’
- tran = ‘.tran 0 65n 0 10p’
- rit\_models = f’.inc {rit\_models\_for\_ltspice\_file\_path}’

## Global Functions

```
# energy-connected
def energy_connected(from_ns, to_ns):
    return f".measure tran energy_connected INTEG (v(supply)*i(Vsupply)) from={from_ns}n to={to_ns}n"

# energy-disconnected
def energy_disconnected(from_ns, to_ns):
    return f".measure tran energy_disconnected INTEG (v(supply)*i(Vsupply)) from={from_ns}n to={to_ns}n"

# rise-delay-connected
def rise_delay_connected():
    return ".measure tran rise_delay_connected trig v(IN) val=0.5 fall=1 targ v(OUT) val=0.5 rise=1"

# rise-delay-disconnected
def rise_delay_disconnected():
    return ".measure tran rise_delay_disconnected trig v(IN) val=0.5 fall=1 targ v(OUT) val=0.5 rise=1"

# fall-delay-connected
def fall_delay_connected():
    return ".measure tran fall_delay_connected trig v(IN) val=0.5 rise=1 targ v(OUT) val=0.5 fall=1"

# fall-delay-disconnected
def fall_delay_disconnected():
    return ".measure tran fall_delay_disconnected trig v(IN) val=0.5 rise=1 targ v(OUT) val=0.5 fall=1"
```

## I. Design constraints

- a) Il buffer deve avere tre stadi;
- b) Il primo stadio è un inverter dimensionato minimo;
- c) L'ultimo stadio ha come carico capacitivo un inverter dimensionato 50x l'inverter minimo;
- d) Le variabili indipendenti che possono essere usate per l'ottimizzazione sono il rapporto d'aspetto del secondo e del terzo inverter rispetto alle dimensioni dell'inverter minimo;

## II. Tasks to be performed

- a) Trovare il dimensionamento dell'inverter minimo;
- b) Calibrare il modello di energia e ritardo del buffer sulla tecnologia utilizzata. In particolare, trovare le seguenti costanti: gamma\_delay, gamma\_energia, tau\_0, capacità di ingresso dell'inverter minimo;
- c) Considerando il rapporto d'aspetto del secondo e del terzo inverter rispetto alle dimensioni dell'inverter minimo come variabili da poter settare nel processo di ottimizzazione, ricavare la curva di Pareto energy-delay in maniera empirica, cioè ricavando l'inviluppo dei punti di design nello spazio Energy-Delay ricavati tramite simulazione Monte Carlo.
- d) Ricavare la stessa curva di Pareto utilizzando la metodologia della sensitivity analysis: attraverso un tool di ottimizzazione numerica e i modelli di energia e ritardo di cui sopra, ricavare, per un set opportuno di constraint di ritardo, i rapporti di aspetto che minimizzano la dissipazione di energia dinamica. In questa maniera, si otterrà, per ognuno dei constraint di delay utilizzati, una coppia di dimensionamenti;
- e) Per ognuna delle coppie trovate al punto precedente, simulare il buffer con LTspice per ricavare i reali punti di design corrispondenti nello spazio Energy-Delay. L'insieme di tali punti costituisce la curva di Pareto ottima ricavata tramite la sensitivity analysis. Verificare che tale curva sia pressoché coincidente con la curva ricavata per via empirica.

### III. Abstract

La parte iniziale del progetto ha previsto la progettazione di un'inverter chain (buffer) a 3 stadi: il primo stadio composto da un inverter di dimensioni minime, il secondo ed il terzo stadio composti da un inverter ciascuno e rispettivamente dimensionati S1 ed S2 volte l'inverter minimo. Nello specifico, tali parametri di dimensionamento sono stati ottenuti considerando il Metodo di Monte Carlo così da farci ottenere una certa randomicità durante le N\_RUNS simulazioni condotte. In uscita al buffer è stato considerato un carico rappresentato da un inverter dimensionato S\_Load volte l'inverter minimo. Nello specifico, è stato analizzato il comportamento del buffer considerando una certa tensione V\_IN, di tipologia pulse avente determinati parametri, una tensione V\_Supply per i 3 stadi di inverter ed una V\_Supply\_L per l'inverter di carico. Dopo aver effettuato delle simulazioni riguardo il circuito in questione, è stata calcolata l'energia associata all'inverter chain considerando le due transizioni possibili: 0->1 (da 13 ns a 18 ns) e 1->0 (da 18 ns a 22 ns). Inoltre, sono stati calcolati i ritardi di rise e di fall considerando rispettivamente fronti di salita e di discesa del segnale. Bisogna specificare che nelle analisi è stato considerato l'inverter minimo isolato così da poter calcolare la capacità totale (collegando l'inverter minimo al carico), la capacità di uscita intrinseca (scollegando l'inverter minimo dal carico) e, infine, la capacità di ingresso (minima) ottenuta come differenza tra le due appena citate. Pertanto, dopo aver calcolato i parametri caratteristici del buffer, quali gamma\_e, gamma\_d, tau\_nom e c\_min, è stato possibile passare alla successiva fase del progetto: l'ottimizzazione dei fattori di dimensionati S1 ed S2 del circuito. Nello specifico, considerando un algoritmo di ottimizzazione, presente nella libreria SciPy in Python, e tenendo conto di una funzione obiettivo rappresentata dal modello dell'energia di un'inverter chain, e di constraint di uguaglianza e diseguaglianza ricavati dal modello del delay, è stato possibile ottenere valori ottimi considerando un delay range tra d\_max e d\_min a step di 10. Successivamente, i valori ottimi, ottenuti dall'algoritmo appena citato, sono stati utilizzati all'interno dello schematic buffer\_optimized per simulare nuovamente il circuito dell'inverter chain e ricavarne i parametri di delay ed energia. Pertanto, infine, è stato possibile effettuare un'analisi comparativa tra gli esperimenti condotti durante la parte iniziale del progetto, ottenuti tramite Metodo Monte Carlo, e la curva di Pareto ottima, ottenuta tramite processo di ottimizzazione e simulazione del circuito. Tutte queste analisi appena citate sono state effettuate in LTspice (per quello che riguarda la descrizione hardware degli schematic di riferimento) e in Python (per quello che riguarda l'esecuzione delle simulazioni degli schematic contenuti in LTspice e i relativi calcoli richiesti dalla traccia) tramite la libreria PyLTspice. Nello specifico, quest'ultima libreria appena citata ha permesso di impostare tramite linguaggio Python i parametri caratteristici degli schematic, effettuare le simulazioni corrispondenti e generare i file .mout in seguito all'applicazione di comandi .meas all'interno delle descrizioni dei relativi schematic.

Inoltre, verranno affrontate le stesse procedure di ottimizzazione considerando l'algoritmo di ottimizzazione fmincon presente in MATLAB e successivamente si effettuerà un confronto riguardo l'accuratezza relativa all'algoritmo di ottimizzazione presente in Python e quello presente in MATLAB.

# 1. Introduction

In questo capitolo verranno introdotti alcuni concetti teorici utili ad affrontare le tematiche e le analisi svolte all'interno del report del progetto.

## 1.1. Energy Model

Un buffer ad  $N$  stadi presenta  $N$  inverter ed ognuno di quest'ultimi è caratterizzato da un sizing factor  $S_i$ . Quest'ultimo parametro rappresenta il fattore di dimensionamento dello stadio di inverter  $i$ -esimo rispetto all'inverter minimo, cioè un sizing factor  $S_i = m$  vuol dire che lo stadio  $i$ -esimo risulta essere  $m$  volte l'inverter minimo, cioè il parametro di larghezza del canale associato al PMOS e all'NMOS risulta essere  $w_i = m \cdot w_{min}$ . Solitamente il primo stadio di un'inverter chain corrisponde all'inverter minimo, cioè tale stadio presenta un fattore di dimensionamento pari a 1 dal momento che  $w_i = w_{min}$ , cioè  $S_0 = 1$ .

Considerando lo stadio  $i$ -esimo dell'inverter chain e considerando ciò che è stato appena citato, si può arrivare alla conclusione che all'aumentare del fattore di dimensionamento dello stadio, l'inverter associato allo stadio in questione sarebbe più veloce perché all'aumentare di  $w$  la corrente di drain aumenta essendo dipendente quest'ultima da  $w$  e, inoltre, tale stadio dissipava anche una maggiore energia dal momento che la corrente risulta essere maggiore.

Pertanto, modificando opportunamente i fattori di dimensionamento  $S_i$  è possibile modellizzare la relazione di energia derivante dal buffer in questione.

Considerando che ad ogni stadio  $i$ -esimo dell'inverter chain è associato un quantitativo di energia  $E_i$ , allora l'energia totale di un buffer ad  $N$  stadi sarà:

$$E_{tot} = E_0 + E_1 + \dots + E_N$$

L'energia dissipata dall' $i$ -esimo inverter  $E_i$  dipende dalla capacità associata  $C_i$ . Nello specifico, la capacità che vede questo inverter è dovuta alla capacità del gate dell'inverter successivo ma anche dalla capacità parassita dell'inverter stesso.

Inoltre, la capacità dell'inverter  $i$ -esimo, invece, dipende dalla dimensione dell'inverter stesso, cioè se questo è dimensionato minimo allora la sua capacità sarà pari ad un "tot di  $fF$ " mentre se è dimensionato il doppio allora la sua capacità sarà il doppio del "tot di  $fF$ ".

In aggiunta bisogna specificare che la capacità intrinseca di uscita di ogni inverter dipende dalla capacità di ingresso dello stesso secondo la seguente relazione:

$$C_{out} = \gamma_E C_{in}$$

Pertanto, la capacità associata ad ogni singolo stadio sarà:

$$C_i = \gamma_E C_{min} S_i + C_{min} S_{i+1} = \gamma_E C_{in,i} + C_{in,i+1} = C_{out,i} + C_{in,i+1}$$

Quindi, l'energia associata all' $i$ -esimo stadio di un inverter chain sarà:

$$E_i = V_{DD}^2 \cdot C_i = V_{DD}^2 \cdot (\gamma_E C_{min} S_i + C_{min} S_{i+1})$$

E l'energia totale associata ad un buffer ad  $N$  stadi sarà:

$$E_{tot} = E_0 + \dots + E_N = V_{DD}^2 \cdot (\gamma_E C_{min} S_0 + C_{min} S_1) + \dots + V_{DD}^2 \cdot (\gamma_E C_{min} S_N + C_{min} S_{LOAD})$$

dove  $S_{LOAD}$  è il fattore di dimensionamento relativo ad un inverter di carico, rappresentante l'equivalente di carico dell'inverter chain, collegato all'OUT del buffer in questione.

Pertanto, se si riuscisse a trovare un dimensionamento ottimo per i sizing factor associati ad ogni stadio e per il sizing factor associato all'inverter di carico, si otterrebbe un'energia dissipata ottima.

Considerando un numero di stadi  $N = 3$ , si ottiene la relazione dell'energia associata al buffer del progetto in questione:

$$\begin{aligned} E_{tot} &= E_0 + E_1 + E_2 \\ &= V_{DD}^2 \cdot (\gamma_E C_{min} S_0 + C_{min} S_1) + V_{DD}^2 \cdot (\gamma_E C_{min} S_1 + C_{min} S_2) + V_{DD}^2 \\ &\quad \cdot (\gamma_E C_{min} S_2 + C_{min} S_{LOAD}) \\ &= V_{DD}^2 \cdot (\gamma_E C_{min} + C_{min} S_1) + V_{DD}^2 \cdot (\gamma_E C_{min} S_1 + C_{min} S_2) + V_{DD}^2 \\ &\quad \cdot (\gamma_E C_{min} S_2 + C_{min} S_{LOAD}) \end{aligned}$$

dove  $S_0 = 1$  dal momento che l'inverter del primo stadio corrisponde all'inverter minimo.

## 1.2. Delay Model

Facendo considerazioni analoghe al modello dell'energia appena ricavato, è possibile ottenere un modello del delay associato al buffer in questione.

Il ritardo  $i$ -esimo di un inverter della chain in funzione delle capacità sarà:

$$\tau_i = \tau_{nom} \left( 1 + \frac{1}{\gamma} \frac{C_{i+1}}{C_i} \right)$$

dove  $\tau_{nom}$  è il ritardo intrinseco dell'inverter minimo, cioè un ritardo di un inverter senza nessuna capacità di carico. Quindi, quest'ultima quantità corrisponde al ritardo di un inverter minimo dove la capacità di carico è soltanto la capacità parassita del nodo di uscita. Mentre il rapporto  $\frac{C_{i+1}}{C_i}$  rappresenta il rapporto tra la capacità di ingresso dell'inverter e la sua capacità di carico.

Però, il nostro obiettivo è quello di esprimere il ritardo in funzione del sizing. Banalmente si può sostituire la capacità con il parametro di dimensionamento nel seguente modo:

$$\tau_i = \tau_{nom} \left( 1 + \frac{1}{\gamma_D} \frac{S_{i+1}}{S_i} \right)$$

Pertanto, il ritardo totale del buffer sarà:

$$\tau_{tot} = \tau_0 + \tau_1 + \dots + \tau_N = \tau_{nom} \left( 1 + \frac{1}{\gamma_D} \frac{S_1}{S_0} \right) + \tau_{nom} \left( 1 + \frac{1}{\gamma_D} \frac{S_2}{S_1} \right) + \dots + \tau_{nom} \left( 1 + \frac{1}{\gamma_D} \frac{S_{LOAD}}{S_N} \right)$$

Considerando un numero di stadi  $N = 3$ , si ottiene la relazione del delay associata al buffer del progetto in questione:

$$\begin{aligned} \tau_{tot} &= \tau_0 + \tau_1 + \tau_2 = \tau_{nom} \left( 1 + \frac{1}{\gamma_D} \frac{S_1}{S_0} \right) + \tau_{nom} \left( 1 + \frac{1}{\gamma_D} \frac{S_2}{S_1} \right) + \tau_{nom} \left( 1 + \frac{1}{\gamma_D} \frac{S_{LOAD}}{S_2} \right) \\ &= \tau_{nom} \left( 1 + \frac{1}{\gamma_D} S_1 \right) + \tau_{nom} \left( 1 + \frac{1}{\gamma_D} \frac{S_2}{S_1} \right) + \tau_{nom} \left( 1 + \frac{1}{\gamma_D} \frac{S_{LOAD}}{S_2} \right) \end{aligned}$$

## 1.3. Optimization Model

Quindi, dopo aver trovato due relazioni per il buffer,  $E(S_2, S_3)$  e  $D(S_2, S_3)$ , affinché venga effettuata un'ottimizzazione dell'inverter chain in questione, bisognerà risolvere il seguente problema di ottimizzazione:

$$\begin{cases} \min E(S_2, S_3) \\ D(S_2, S_3) = D_0 \end{cases}$$

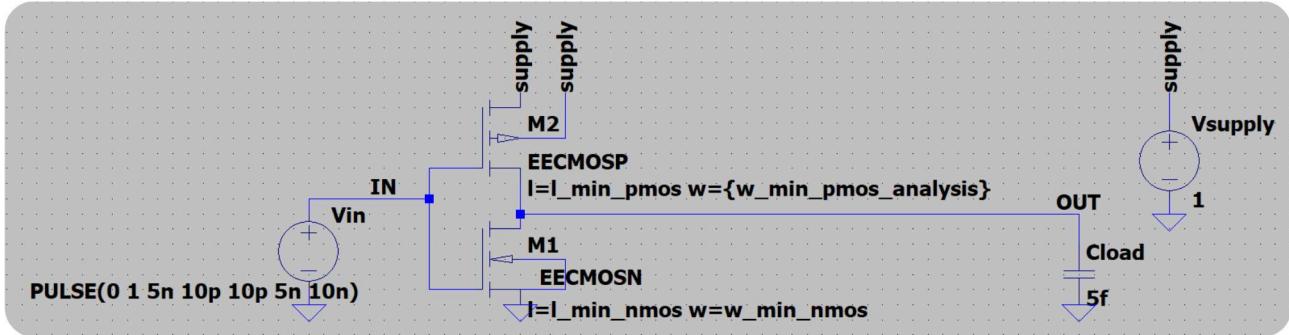
da cui ricavare una coppia ottima  $(S_2, S_3)$ . Nello specifico,  $E(S_2, S_3)$  è la funzione obiettivo rappresentata dal modello di energia ricavato mentre  $D(S_2, S_3) = D_0$  è il constraint relativo al problema in questione.

Se volessi trovare una curva ottima dovrei considerare un determinato range di delay e per ogni valore di delay  $D_0 \in [D_{min}, D_{max}]$ , utilizzato per il calcolo del constraint di delay, otterrei una coppia  $(S_2, S_3)$  che corrisponderà al punto di lavoro associato a quel delay considerato. Ovviamente tale curva ottima si otterrebbe solo in seguito ad aver considerato tali coppie ottime all'interno del circuito di riferimento. Successivamente, dopo aver ottenuto la curva ottima, bisognerebbe chiedersi se effettivamente tale curva è quella ottima. Per farlo bisognerebbe graficare nello stesso plot tale curva ottima con i punti randomici calcolati tramite, ad esempio, Metodo Monte Carlo. Quindi, considerando un elevato numero di simulazioni e, pertanto, un elevato numero di punti, bisognerà verificare se tale curva ottima risulta essere l'inviluppo di tutti questi punti generati. Pertanto, effettuare un'analisi di sensitività equivale a risolvere un problema di minimizzazione vincolata.

## 2. Analysis and Design of a Minimum CMOS Inverter

### 2.1. Minimum Sizing

Il requisito, affinché si possa parlare di un inverter CMOS minimo, è che quest'ultimo presenti tempo di salita, *rise\_delay*, e tempo di discesa, *fall\_delay*, uguale o comunque approssimabile. Pertanto, considerando una lunghezza del canale per il PMOS e l'NMOS pari a  $0.1 \mu\text{m}$  e la larghezza di canale per l'NMOS pari  $0.12 \mu\text{m}$ , l'obiettivo è ricavare la larghezza del canale minima per il PMOS. Quindi, si consideri lo schematico contenuto nella directory “Itspice/minimum-inverter/get\_sizing/minimum\_inverter\_sizing\_analysis.asc”



Inoltre, si consideri come range di analisi per la larghezza del canale del PMOS il seguente intervallo di valori  $[0.12 \mu\text{m}, 0.36 \mu\text{m}]$  e come *.meas*, rispettivamente per *rise\_delay* e *fall\_delay*, le seguenti stringhe contenute nelle funzioni python presenti nel file ops.py

```
".measure tran rise_delay_connected trig v(IN) val=0.5 fall=1 targ v(OUT) val=0.5 rise=1"
".measure tran rise_delay_disconnected trig v(IN) val=0.5 fall=1 targ v(OUT) val=0.5 rise=1"
```

Pertanto, per effettuare l'analisi riguardo il dimensionamento minimo basterà eseguire lo script *minimum\_inverter\_sizing\_analysis.py* che permetterà di ottenere la W minima del PMOS in questione e di salvare quest'ultima nel file “data/out/minimum-inverter/get\_sizing/w\_min\_pmos.txt” così da essere utilizzata in script esterni per successive analisi. Nello specifico, lo script permetterà di effettuare la simulazione dello schematico sopra citato e successivamente calcolare la differenza minima tra *rise\_delay* e *fall\_delay* alla quale corrisponderà la w minima del PMOS.

```
...
diff_delay = [
    abs(rise_delay - fall_delay)
    for rise_delay, fall_delay in
    zip(rise_delay_minimum_inverter_sizing_analysis, fall_delay_minimum_inverter_sizing_analysis)
]

min_diff_delay = min(diff_delay)
i_min_diff_delay = diff_delay.index(min_diff_delay)
w_min_pmos_value = w_min_pmos_analysis[i_min_diff_delay]
...
```

W\_PMOA Sizing Analysis

rise_delay [s]	fall_delay [s]	rise_delay-fall_delay [s]	w_pmos [m]
1.80376e-10	1.13062e-10	6.7314e-11	1.2e-07
1.64419e-10	1.13422e-10	5.099700000000005e-11	1.3e-07
1.51378e-10	1.13781e-10	3.759700000000005e-11	1.4e-07
1.39945e-10	1.14138e-10	2.5807e-11	1.5e-07
1.30472e-10	1.14494e-10	1.5978e-11	1.6e-07
1.22016e-10	1.14849e-10	7.167e-12	1.7e-07
1.14767e-10	1.15203e-10	4.36000000000005e-13	1.8e-07
1.08353e-10	1.15556e-10	7.20299999999998e-12	1.9e-07
1.02605e-10	1.15909e-10	1.330399999999991e-11	2e-07
9.73211e-11	1.16259e-10	1.893790000000004e-11	2.1e-07
9.27746e-11	1.16609e-10	2.383439999999999e-11	2.2e-07
8.85776e-11	1.16957e-10	2.83794e-11	2.3e-07
8.48159e-11	1.17304e-10	3.248810000000001e-11	2.4e-07
8.13779e-11	1.17668e-10	3.629010000000004e-11	2.5e-07
7.81652e-11	1.18322e-10	4.015680000000001e-11	2.6e-07
7.52808e-11	1.18671e-10	4.339020000000004e-11	2.7e-07
7.26217e-11	1.19018e-10	4.63963e-11	2.8e-07
7.01208e-11	1.19365e-10	4.924420000000001e-11	2.9e-07
6.77563e-11	1.19711e-10	5.19547e-11	3e-07
6.56434e-11	1.20055e-10	5.44116e-11	3.1e-07
6.36717e-11	1.20399e-10	5.672730000000001e-11	3.2e-07
6.18041e-11	1.20742e-10	5.893790000000001e-11	3.3e-07
6.00287e-11	1.21084e-10	6.10553e-11	3.4e-07
5.83347e-11	1.21424e-10	6.30892999999998e-11	3.5e-07
5.67684e-11	1.21764e-10	6.49956e-11	3.6e-07

Dai risultati generati si può notare che la differenza minima si ha in corrispondenza di una w pari a  $0.18 \times 10^{-6} m$ , cioè  $0.18 \mu m$ .

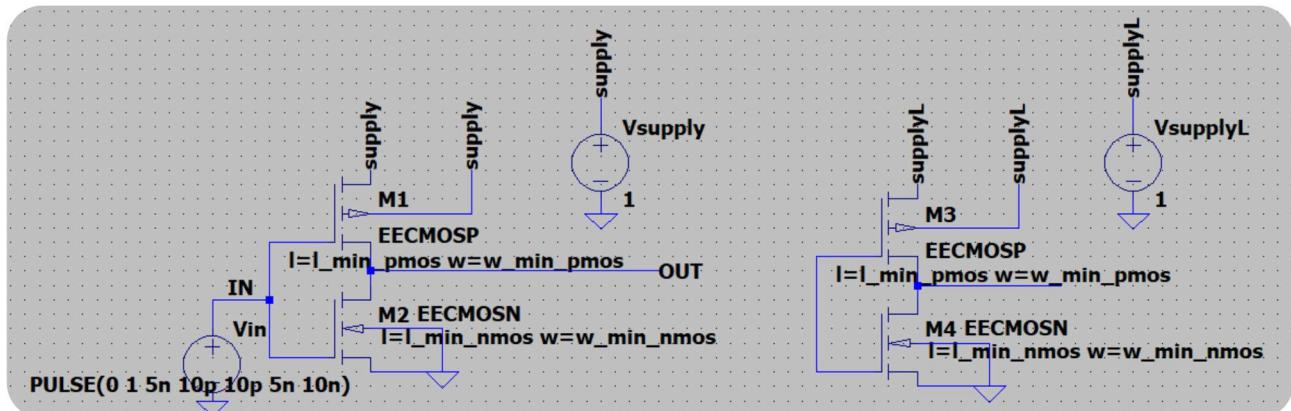
Da questo momento in poi verrà considerata la seguente definizione  $w_{min\_pmos} = '0.18u'$ .

## 2.2. Minimum Inverter

L'inverter minimo considerato si può presentare in due forme: disconnesso e connesso. Nello specifico, il primo fa riferimento ad uno schematic in cui l'inverter minimo non è collegato ad alcun carico mentre il secondo fa riferimento ad uno schematic in un cui tale inverter presenta il nodo OUT collegato ad un inverter minimo con identiche caratteristiche.

### 2.2.1. Minimum Inverter Disconnected

L'analisi dell'inverter minimo non collegato ad alcun carico è utile per calcolare l'energia ed il delay associato.



In particolare, l'energia è stata calcolata considerando la definizione  $energy\_disconnected(18, 22)$ , cioè considerando la singola transizione compresa tra 18 ns e 22 ns mentre il delay è stato calcolato come media tra il  $fall\_delay$ , calcolato tramite la definizione  $fall\_delay\_disconnected()$ , e il  $rise\_delay$ , calcolato tramite la definizione  $rise\_delay\_disconnected()$ . Nello specifico:

```

...
energy_disconnected_minimum_inverter = [float(line.split('\t')[1]) for line in energy_disconnected_minimum_inverter_lines]
energy_disconnected_minimum_inverter = [abs(energy) for energy in energy_disconnected_minimum_inverter]
...
rise_delay_disconnected_minimum_inverter = [float(line.split('\t')[1]) for line in rise_delay_disconnected_minimum_inverter_lines]
...
fall_delay_disconnected_minimum_inverter = [float(line.split('\t')[1]) for line in fall_delay_disconnected_minimum_inverter_lines]
...
delay_disconnected_minimum_inverter = [
    (rise_delay + fall_delay) / 2 for rise_delay, fall_delay in
    zip(rise_delay_disconnected_minimum_inverter, fall_delay_disconnected_minimum_inverter)
]
...

```

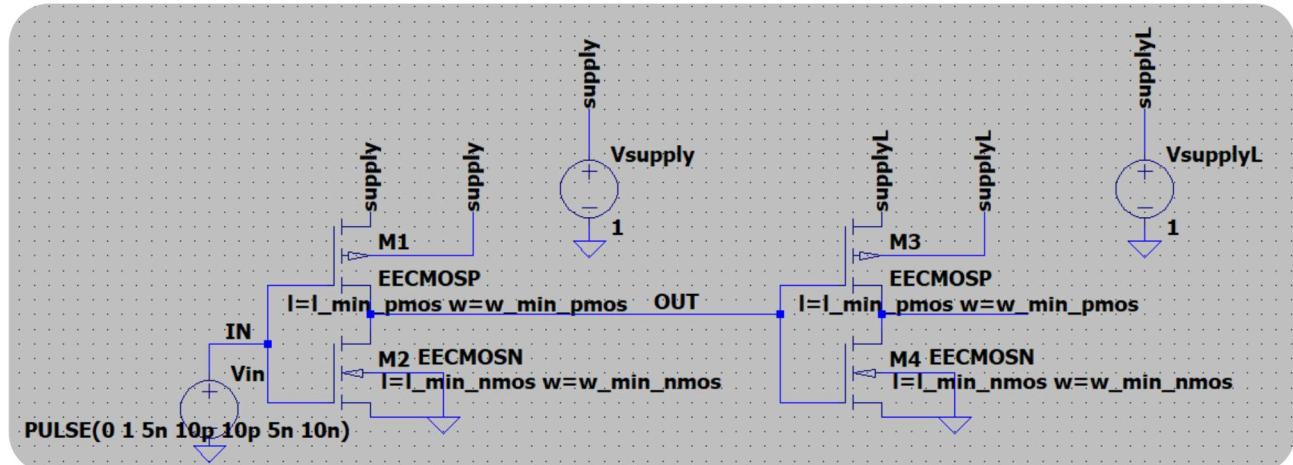
Pertanto, eseguendo lo script *minimum\_inverter\_disconnected\_analysis.py* sarà possibile ottenere i seguenti valori:

energy_disconnected	rise_delay_disconnected	fall_delay_disconnected	delay_disconnected
4.72623e-16	8.14506e-12	7.52738e-12	7.836220000000001e-12

Ovviamente, i valori di energia e delay sono stati opportunamente salvati in file .txt contenuti nella directory "data/out/ minimum-inverter/disconnected/" così da essere utilizzati nelle successive analisi.

### 2.2.2. Minimum Inverter Connected

L'analisi dell'inverter minimo collegato ad un carico corrispondente ad un inverter minimo con uguali caratteristiche è utile per calcolare l'energia ed il rise delay ed il fall delay associato.



In particolare, l'energia è stata calcolata considerando la definizione `energy_connected(18, 22)`, cioè considerando la singola transizione compresa tra 18 ns e 22 ns, il fall delay tramite la definizione `fall_delay_connected()` ed il rise delay tramite la definizione `rise_delay_connected()`. Nello specifico:

```

...
energy_connected_minimum_inverter = [float(line.split('\t')[1]) for line in energy_connected_minimum_inverter_lines]
energy_connected_minimum_inverter = [abs(energy) for energy in energy_connected_minimum_inverter]
...
rise_delay_connected_minimum_inverter = [float(line.split('\t')[1]) for line in rise_delay_connected_minimum_inverter_lines]
...
fall_delay_connected_minimum_inverter = [float(line.split('\t')[1]) for line in fall_delay_connected_minimum_inverter_lines]
...

```

Pertanto, eseguendo lo script *minimum\_inverter\_connected\_analysis.py* sarà possibile ottenere i seguenti valori:

energy_connected	rise_delay_connected	fall_delay_connected
1.07767e-15	2.0827e-11	1.85558e-11

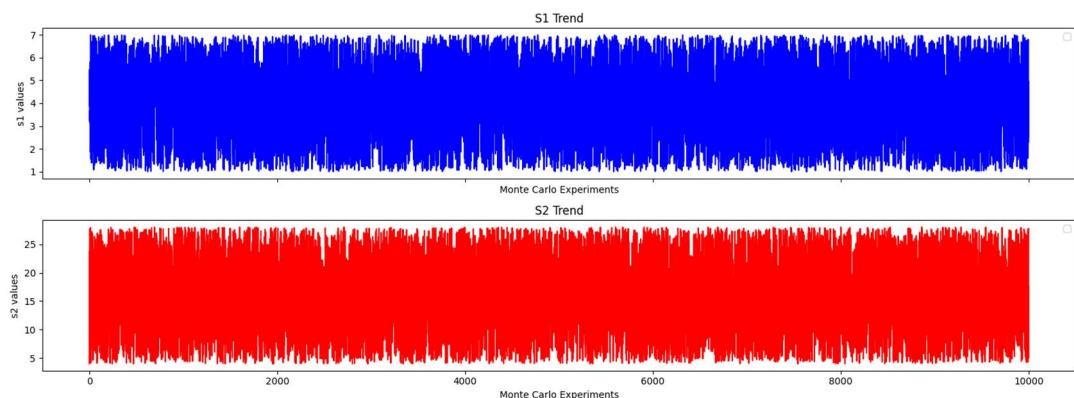
Ovviamente, i valori di energia e delay sono stati opportunamente salvati in file .txt contenuti nella directory "data/out/minimum-inverter/connected/" così da essere utilizzati nelle successive analisi.

### 3. Analysis and design of a three-stage CMOS buffer

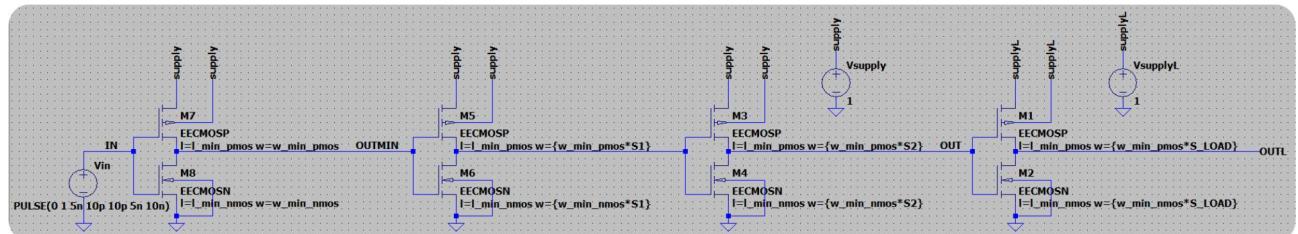
Il buffer considerato per l'analisi di questo progetto è un inverter chain a tre stadi. Nel primo stadio è presente un inverter dimensionato minimo secondo i parametri descritti nel capitolo precedente, nel secondo stadio è presente un inverter dimensionato  $S_1$  volte l'inverter minimo e nel terzo stadio è presente un inverter dimensionato  $S_2$  volte l'inverter minimo. Bisogna specificare che i parametri  $S_1$  ed  $S_2$  sono i fattori di dimensionamento ottenuti tramite Metodo Monte Carlo tale che  $S_1 < S_2$ . Questi valori sono stati generati considerando le seguenti definizioni:

$$S_1 = ' mc(4,0.75)'$$

$$S_2 = ' mc(16,0.75)'$$



Inoltre, bisogna precisare che il nodo OUT del buffer in questione è collegato ad un carico rappresentato da un inverter dimensionato  $S\_LOAD$  volte l'inverter minimo, dove  $S\_LOAD$  è stata considerata pari a 50.



In particolare, l'energia è stata calcolata considerando la definizione `energy_connected(13, 22)`, cioè considerando la doppia transizione e, quindi, quella compresa tra 13 ns e 18 ns e quella compresa tra 18 e 22 ns, mentre il delay è stato calcolato come media tra il `fall_delay`, calcolato tramite la definizione `fall_delay_connected()`, e il `rise_delay`, calcolato tramite la definizione `rise_delay_connected()`. Nello specifico:

```
buffer_analysis.py

...
energy_connected_buffer = [float(line.split('\t')[1]) for line in energy_connected_buffer_lines]
energy_connected_buffer = [abs(energy) for energy in energy_connected_buffer]
...
rise_delay_connected_buffer = [float(line.split('\t')[1]) for line in rise_delay_connected_buffer_lines]
...
fall_delay_connected_buffer = [float(line.split('\t')[1]) for line in fall_delay_connected_buffer_lines]
...
delay_connected_buffer = [
    (rise_delay + fall_delay) / 2 for rise_delay, fall_delay in
    zip(rise_delay_connected_buffer, fall_delay_connected_buffer)
]
...
```

Pertanto, eseguendo lo script *buffer\_analysis.py* sarà possibile ottenere i seguenti valori:

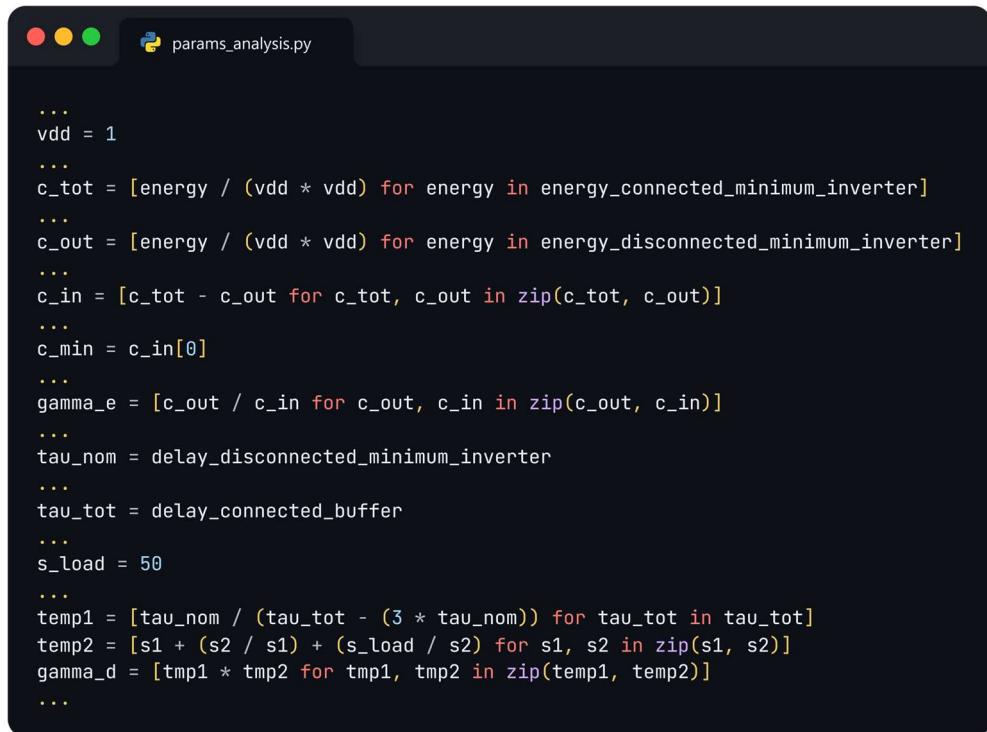
<b>energy_connected</b>	<b>rise_delay_connected</b>	<b>fall_delay_connected</b>	<b>delay_connected</b>
4.72623e-16	8.14506e-12	7.52738e-12	7.836220000000001e-12

Ovviamente, i valori di energia, delay e dei fattori di sizing S1 ed S2 sono stati opportunamente salvati in file .txt contenuti nella directory “data/out/buffer/standard/” così da essere utilizzati nelle successive analisi.

Prima di mettere al vaglio altri aspetti e risultati ottenuti tramite il progetto in questione, è necessario analizzare il modello di energia e delay dipendente dai fattori di sizing S1 ed S2 relativi al buffer a tre stadi appena citato. Affinché si possa effettuare un’analisi corretta si suppone che ogni inverter abbia la medesima tensione di alimentazione e, inoltre, si trascura la capacità delle linee rispetto alle capacità parassite degli inverter stessi.

### 3.1. Parameters Analysis

Pertanto, si possono ora calcolare i parametri caratteristici relativi all’analisi del buffer:



```

...
vdd = 1
...
c_tot = [energy / (vdd * vdd) for energy in energy_connected_minimum_inverter]
...
c_out = [energy / (vdd * vdd) for energy in energy_disconnected_minimum_inverter]
...
c_in = [c_tot - c_out for c_tot, c_out in zip(c_tot, c_out)]
...
c_min = c_in[0]
...
gamma_e = [c_out / c_in for c_out, c_in in zip(c_out, c_in)]
...
tau_nom = delay_disconnected_minimum_inverter
...
tau_tot = delay_connected_buffer
...
s_load = 50
...
temp1 = [tau_nom / (tau_tot - (3 * tau_nom)) for tau_tot in tau_tot]
temp2 = [s1 + (s2 / s1) + (s_load / s2) for s1, s2 in zip(s1, s2)]
gamma_d = [tmp1 * tmp2 for tmp1, tmp2 in zip(temp1, temp2)]
...

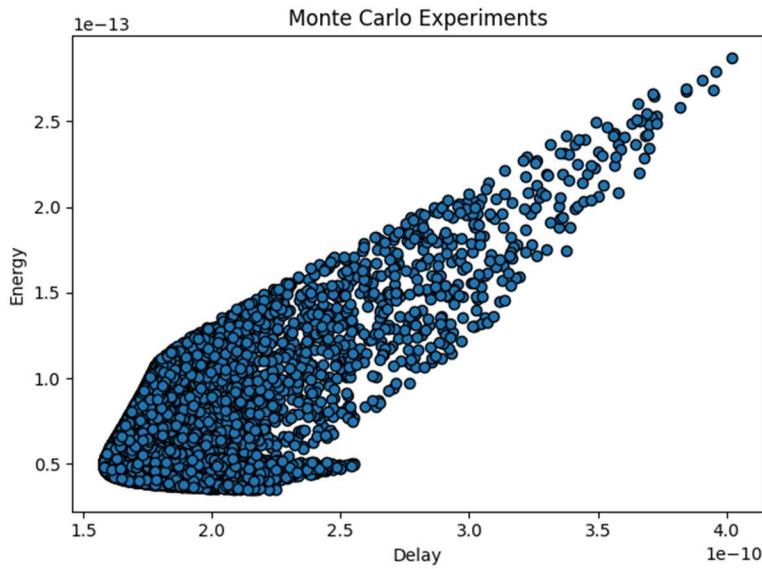
```

Effettuando i calcoli sopra allegati, cioè eseguendo lo script *params\_analysis.py*, si ottengono i seguenti risultati:

<b>c_min</b>	<b>gamma_e</b>	<b>tau_nom</b>	<b>gamma_d</b>
6.05047e-16	0.7811343581573003	7.836220000000001e-12	0.6419216350606208

### 3.2. Pareto Energy-Delay Curve Empirically by Monte Carlo Method

La curva di Pareto Energy-Delay empirica si ottiene mediante simulazioni tramite Metodo Monte Carlo. In particolare, sono state considerate 10000 iterazioni.



### 3.3. Pareto Energy-Delay Optimized by Python SciPy Algorithm

#### 3.3.1. Optimal Values Generation

Affinché vengano generate le coppie di fattori di dimensionamento ottime ( $S_1, S_2$ ) in riferimento al secondo e al terzo stadio del buffer in questione, è necessario eseguire lo script [optimization\\_analysis.py](#). Nello script appena citato vengono inizialmente caricate da file esterni i parametri caratteristici precedentemente calcolati e successivamente vengono inizializzate ulteriori variabili. Nello specifico, viene inizializzata la variabile *max\_iter* che indica il numero massimo di iterazioni che l'algoritmo deve eseguire per ogni passo di delay considerato, *d\_max* che rappresenta l'upper bound di delay da considerare durante l'ottimizzazione, *d\_min* che rappresenta il lower bound di delay da considerare durante l'ottimizzazione e *step\_opt* che rappresenta il passo di delay da considerare durante l'ottimizzazione. Pertanto, iterando su questi i-esimi valori di delay sopra citati, verrà effettuata una chiamata alla funzione *optimize\_considering\_delay*, cioè una funzione presente nello script `/optimization/optimize_function.py` all'interno della quale viene richiamato l'algoritmo della libreria `scipy`.

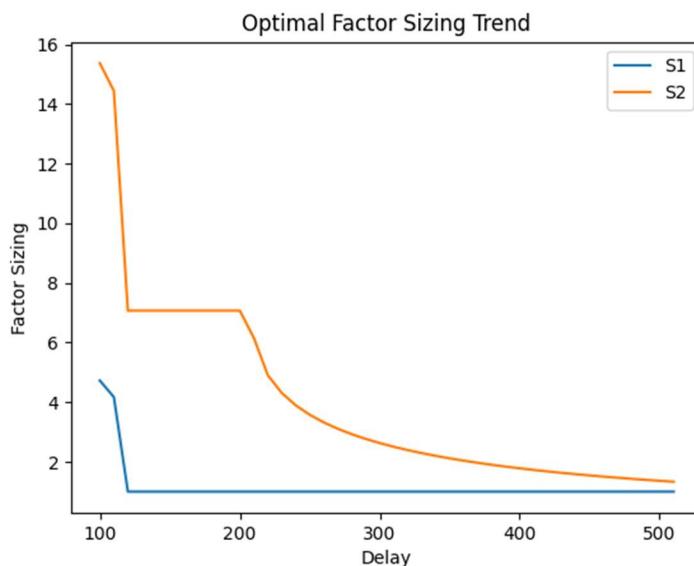
```
...  
for delay in range(d_max, d_min - 1, -step_opt):  
    variables, energy_value, iteration_count = optimize_considering_delay(  
        delay=delay,  
        tau_nom=tau_nom,  
        gamma_d=gamma_d,  
        s_load=s_load,  
        vdd=vdd,  
        c_min=c_min,  
        gamma_e=gamma_e,  
        s0=s0,  
        max_iter=max_iter  
    )  
...
```

In particolare, verrà considerato l'algoritmo di ottimizzazione `scipy.optimize.minimize(method='SLSQP')`, cioè un algoritmo con l'obiettivo di minimizzare una funzione scalare di una o più variabili utilizzando la programmazione sequenziale ai minimi quadrati (SLSQP).

```
optimize_function.py
```

```
def optimize_considering_delay(
    delay,
    tau_nom,
    gamma_d,
    s_load,
    vdd,
    c_min,
    gamma_e,
    s0,
    max_iter
):
    delay_constraints = (
        {
            'type': 'eq', 'fun': delay_constraint,
            'args': (delay, tau_nom, gamma_d, s_load)
        }
    )
    result = minimize(
        energy,
        s0,
        args=(gamma_e, vdd, c_min, s_load),
        constraints=delay_constraints,
        method='SLSQP',
        options={'maxiter': max_iter}
    )
    return result.x, result.fun, result.nit
```

Pertanto, eseguendo la procedura di ottimizzazione si otterranno i seguenti risultati:



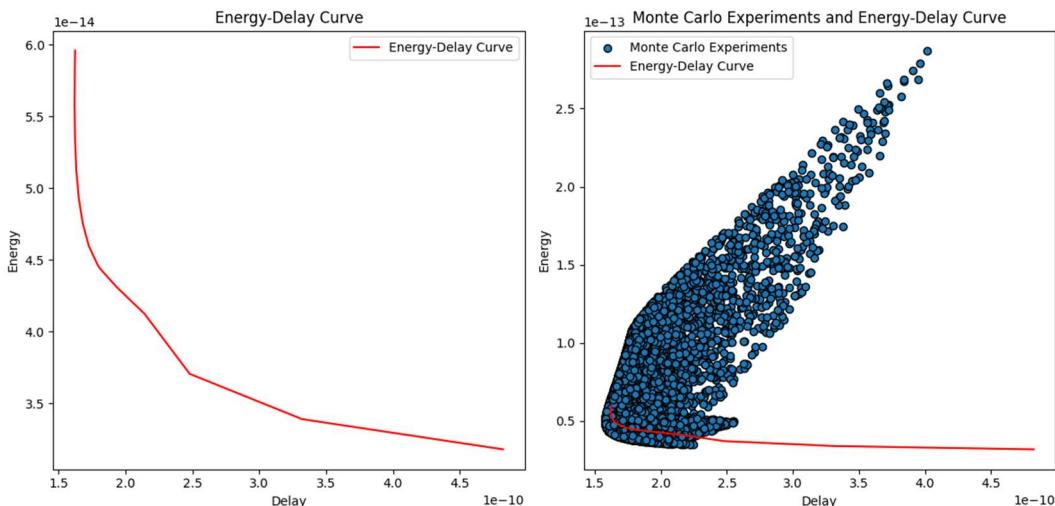
### Optimization Analysis

delay	optimal_point	objective_f_value	iterations
510	[1. 1.33264423]	33.23879370486311	13
500	[1. 1.3635375]	33.27208645350889	12
490	[1. 1.39595665]	33.30702349688241	13
480	[1. 1.43002206]	33.34373487151219	13
470	[1. 1.46586628]	33.38236526397441	12
460	[1. 1.5036451]	33.42307621309155	12
450	[1. 1.54352051]	33.46604874741513	12
440	[1. 1.58568349]	33.511486521734916	12
430	[1. 1.63034756]	33.559619652545564	13
420	[1. 1.6777551]	33.61070934226574	11
410	[1. 1.72818264]	33.66505358430954	10
400	[1. 1.78194738]	33.722994236470996	7
390	[1. 1.83941541]	33.784925801440345	6
380	[1. 1.90101201]	33.85130661610603	6
370	[1. 1.96723494]	33.92267307828939	6
360	[1. 2.03867152]	33.999658134492805	6
350	[1. 2.11602116]	34.08301552220824	7
340	[1. 2.20012549]	34.1736523231419	7
330	[1. 2.29200932]	34.27267267944303	7
320	[1. 2.39293768]	34.38144014734805	7
310	[1. 2.50449719]	34.50166448741946	8
300	[1. 2.62871555]	34.63553089108896	8
290	[1. 2.76824336]	34.7858958260805	8
280	[1. 2.92664238]	34.95659769553664	9
270	[1. 3.10886761]	35.15297635541817	9
260	[1. 3.32212816]	35.38280085877332	9
250	[1. 3.57756587]	35.658078406722026	9
240	[1. 3.89395242]	35.99903870578223	9
230	[1. 4.30747476]	36.44467932269003	10
220	[1. 4.90745886]	37.091264189600096	10
210	[1. 6.16007677]	38.44117293465406	10
200	[1. 7.0710674]	39.42292020323645	30
190	[1. 7.07106841]	39.422921295001785	40
180	[1. 7.07106807]	39.42292092256805	1614
170	[1. 7.07106828]	39.42292114913473	315
160	[1. 7.07106777]	39.42292060713031	67
150	[1. 7.07106824]	39.4229211416145	7
140	[1. 7.07103619]	39.422886571568725	10000
130	[1. 7.07106784]	39.42292068066554	10
120	[1. 7.07106796]	39.42292080748335	8
110	[4.1715742 14.44225169]	50.7845347435131	351
100	[4.72406271 15.36889897]	52.3785501924856	761

Si può notare come la variabile  $S_1$  rimane stabile al valore 1 nel range [510,120] per poi stabilirsi, negli ultimi 2 step, al valore di circa 4.5. Per quanto riguarda, invece, la variabile  $S_2$  assume un trend totalmente differente dalla variabile  $S_1$ . Inoltre, si può riscontrare come il valore ottimo delle variabili sia raggiunto in poche iterazioni per ogni passo di delay considerato tranne in corrispondenza di  $delay = 140$  in cui viene raggiunto il massimo numero di iterazioni.

#### 3.3.2. Optimized Buffer with Optimal Values

Pertanto, dopo aver generato i valori ottimi per le coppie di sizing factor, bisogna utilizzare tali valori ottimi all'interno dello schematic del buffer sopra citato così da ottenere la curva energy-delay ottima. Quindi, per fare ciò si esegue lo script [\*buffer\\_optimized\\_analysis.py\*](#) e successivamente lo script analogo di plotting.



Si può notare come la curva ottima generata risulta essere leggermente più in alto rispetto ai valori minimi (curva empirica) ottenuti tramite Metodo Monte Carlo. Questa approssimazione, anche se grossolana, può essere accettata considerando il fatto che ogni algoritmo di ottimizzazione presenta un'implementazione differente dagli altri che potrebbe essere migliore o peggiore. Inoltre, un ulteriore miglioramento a riguardo potrebbe essere quello di utilizzare un calcolatore con maggiore unità di calcolo ed eseguire il processo di ottimizzazione ponendo il numero massimo di iterazioni ad un valore notevolmente più grande ed impostando ulteriori constraint ad hoc per il problema in questione. Proprio per questo motivo, è stato implementato uno script in MATLAB che utilizza un algoritmo di ottimizzazione differente così da cercare di migliorare tale curva e cercare di approssimare al minimo il gap presente tra la curva ottima e i punti minimi sopra citati.

### 3.4. Pareto Energy-Delay Optimized by MATLAB fmincon Algorithm

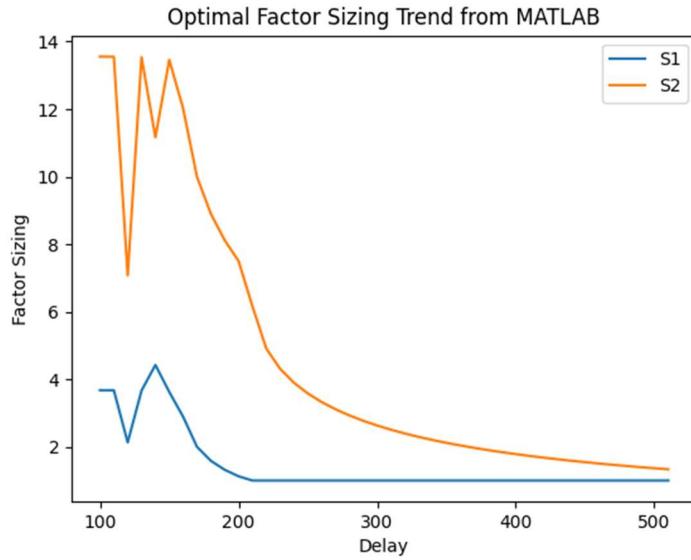
Affinché possa essere effettuata correttamente l'ottimizzazione tramite MATLAB, è necessario esportare su file i parametri calcolati tramite le analisi sul buffer ( $c_{\min}$ ,  $\gamma_e$ , ...). Pertanto, eseguendo lo script [optimization\\_analysis\\_matlab.py](#) verranno scritte automaticamente tali parametri nel file `buffer_data.txt` presente nella directory 'matlab/'.

#### 3.4.1. Optimal Values Generation

Quindi, ora sarà possibile eseguire lo script MATLAB [optimization.m](#). Tale script richiamerà l'algoritmo di ottimizzazione `fmincon`, nativo in MATLAB, utilizzando gli stessi parametri di ottimizzazione (`max_iter`, ...) descritti precedentemente.

```
...
%% optimization algorithm options
options = optimoptions('fmincon', 'Display', 'off');

%% optimization algorithm
for d = delay_range
    [optimal_s, optimal_energy] = fmincon( ...
        @(s) energy_model(s, vdd, c_min, s_load, gamma_e), ...
        s0, ...
        [], ...
        [], ...
        [], ...
        [], ...
        [0, 0], ...
        [], ...
        @(s) delay_constraints(s, d, tau_nom, s_load, gamma_d), ...
        options);
end
...
```

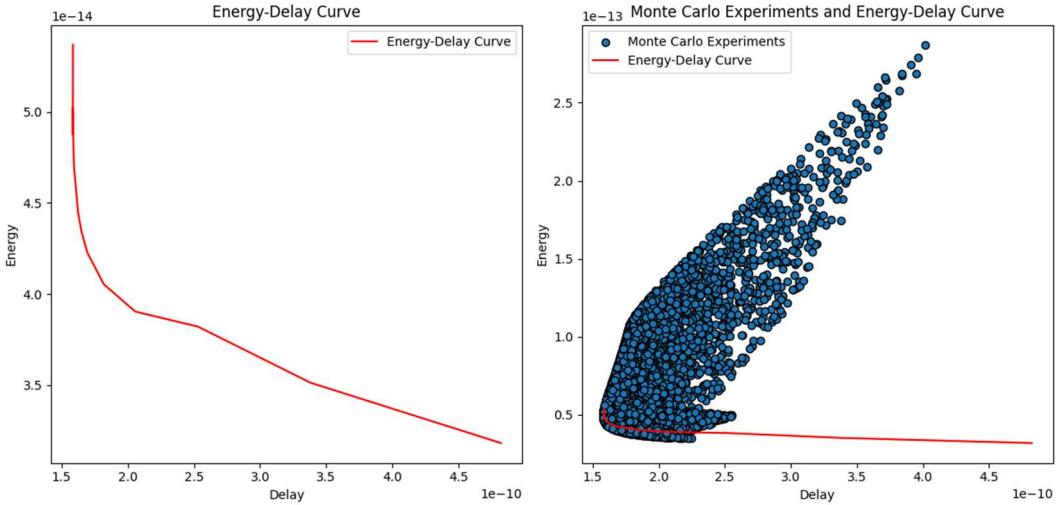


Optimization Analysis MATLAB

delay	s1_from_matlab	s2_from_matlab
510.0	1.0	1.332644
500.0	1.0	1.363538
490.0	1.0	1.395957
480.0	1.0	1.430022
470.0	1.0	1.465868
460.0	1.0	1.503645
450.0	1.0	1.543521
440.0	1.0	1.585684
430.0	1.0	1.630348
420.0	1.0	1.677755
410.0	1.0	1.728183
400.0	1.0	1.781947
390.0	1.0	1.839415
380.0	1.0	1.901012
370.0	1.0	1.967235
360.0	1.0	2.038672
350.0	1.0	2.116021
340.0	1.0	2.200126
330.0	1.0	2.292009
320.0	1.0	2.392938
310.0	1.0	2.504497
300.0	1.0	2.628716
290.0	1.0	2.768243
280.0	1.0	2.926642
270.0	1.0	3.108868
260.0	1.0	3.322128
250.0	1.0	3.577566
240.0	1.0	3.893952
230.0	1.0	4.307475
220.0	1.0	4.907459
210.0	1.0	6.160077
200.0	1.125097	7.500323
190.0	1.31729	8.115696
180.0	1.583916	8.899201
170.0	1.996932	9.992326
160.0	2.888084	12.01683
150.0	3.622778	13.45878
140.0	4.422265	11.16589
130.0	3.665288	13.53752
120.0	2.130154	7.077513
110.0	3.672968	13.55169
100.0	3.674849	13.55516

### 3.4.2. Optimized Buffer with Optimal Values

Pertanto, dopo aver generato i valori ottimi per le coppie di sizing factor tramite MATLAB, bisogna utilizzare tali valori ottimi all'interno dello schematic del buffer sopra citato così da ottenere la curva energy-delay ottima. Quindi, per fare ciò si esegue lo script [buffer\\_optimized\\_analysis\\_matlab.py](#) e successivamente lo script analogo di plotting.



Si può notare come la curva ottima ottenuta risulta essere più prossima ai punti minimi rispetto a quella ottenuta tramite l'algoritmo di ottimizzazione presente in Python. Questo si evince dall'ulteriore analisi effettuata dallo script in questione che mette a confronto i risultati ottenuti in Python con quelli ottenuti in MATLAB.

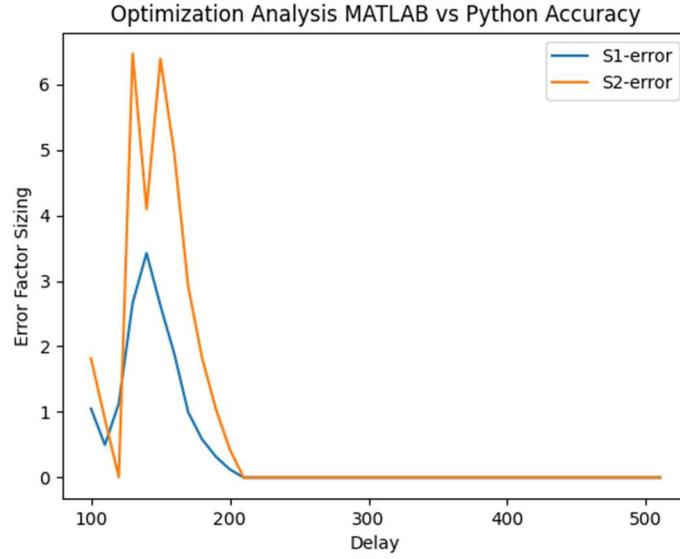
### 3.5. Comparison between Python SciPy Algorithm and MATLAB fmincon Algorithm

#### 3.5.1. Factor Sizing Generation

Eseguendo lo script [optimization\\_plotting\\_matlab.py](#) è possibile analizzare il confronto tra le coppie di factor sizing generati dall'algoritmo di ottimizzazione scipy presente in Python e quelli generati dall'algoritmo di ottimizzazione fmincon presente in MATLAB. Qui di seguito viene riportata una tabella contenente i valori appena citati e un plot relativo alla loro differenza in valore assoluto.

Optimization Analysis MATLAB vs Python

delay	s1_from_python	s1_from_matlab	s2_from_python	s2_from_matlab
510.0	1.0	1.0	1.332644227697826	1.32644
500.0	1.0	1.0	1.3635374961805513	1.363538
490.0	1.0	1.0	1.395956551525429	1.395957
480.0	1.0	1.0	1.430022056248067	1.430022
470.0	1.0	1.0	1.465863750511827	1.465868
460.0	1.0	1.0	1.5036450963061158	1.503645
450.0	1.0	1.0	1.5433205094464253	1.543521
440.0	1.0	1.0	1.585683485422175	1.585684
430.0	1.0	1.0	1.6303475577361939	1.630348
420.0	1.0	1.0	1.677755103385772	1.677755
410.0	1.0	1.0	1.728182638757264	1.728183
400.0	1.0	1.0	1.7819473832165638	1.781947
390.0	1.0	1.0	1.8394154067946091	1.839415
380.0	1.0	1.0	1.90101210352264	1.901012
370.0	1.0	1.0	1.9672349404635836	1.967235
360.0	1.0	1.0	2.038671517121078	2.038672
350.0	1.0	1.0	2.116021158039393	2.116021
340.0	1.0	1.0	2.2001254858297914	2.200126
330.0	1.0	1.0	2.2920093158787296	2.292009
320.0	1.0	1.0	2.392937677905153	2.392938
310.0	1.0	1.0	2.5044971906237192	2.504497
300.0	1.0	1.0	2.628715540090743	2.628716
290.0	1.0	1.0	2.7682433639987205	2.768243
280.0	1.0	1.0	2.926642381746395	2.926642
270.0	1.0	1.0	3.1088676082828397	3.108868
260.0	1.0	1.0	3.3221281642555898	3.322128
250.0	1.0	1.0	3.577565865916304	3.577566
240.0	1.0	1.0	3.893952421225638	3.893952
230.0	1.0	1.0	4.307474758219145	4.307475
220.0	1.0	1.0	4.907458589479405	4.907459
210.0	1.0	1.0	6.160076771781914	6.160077
200.0	1.0	1.0	7.07106739840253	7.500323
190.0	1000000000000002	1.31729	7.07106841148198	8.115696
180.0	1.0	1.0	7.071069065890352	8.899201
170.0	1000000000000004	1.096932	7.0710682761278765	9.992326
160.0	0.9999999999999991	2.888084	7.07106777316868	12.01683
150.0	1.0	3.622778	7.071068245675197	13.45678
140.0	0.999999999999984	4.422265	7.07103619064159	11.16589
130.0	0.999999999999966	3.665288	7.07106841422276	13.537572
120.0	1.000000000000027	2.130154	7.07106795100051	7.077513
110.0	4.1715741978037775	3.672968	14.442251689075418	13.55169
100.0	4.724062711129795	3.674849	15.36898974032227	13.55516



Quello che si può notare che il maggiore errore commesso sia per la generazione del factor sizing S1 che per S2 è in corrispondenza del range di ascisse [120,180]. Tale errore potrebbe spiegare la differenza relativa alle due curve energy-delay ottenute.

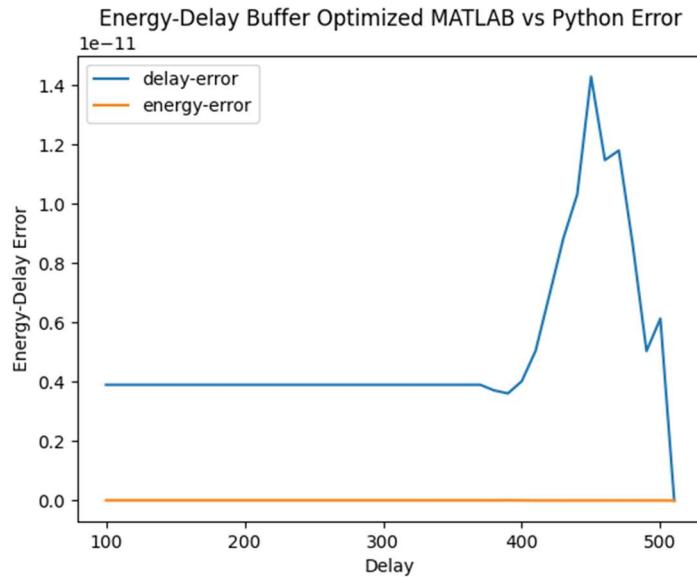
### 3.5.2. Pareto Energy-Delay Curve

Eseguendo lo script [buffer\\_optimized\\_plotting\\_matlab.py](#) è possibile analizzare il confronto tra le curve energy-delay ottenute dal buffer ottimizzato in seguito all'utilizzo delle coppie di sizing factor relative all'algoritmo di ottimizzazione scipy presente in Python e l'algoritmo di ottimizzazione fmincon presente in MATLAB.

Optimization Analysis MATLAB vs Python

delay	delay_from_python	delay_from_matlab	energy_from_python	energy_from_matlab
510.0	4.8225e-10	3.18069e-14	3.18069e-14	3.18069e-14
500.0	3.32125e-10	3.39098e-14	3.51213e-14	3.51213e-14
490.0	2.4813e-10	3.70632e-14	3.82046e-14	3.82046e-14
480.0	2.14257e-10	4.12587e-14	3.90366e-14	3.90366e-14
470.0	1.935765e-10	4.31029e-14	4.05432e-14	4.05432e-14
460.0	1.809275e-10	4.44167e-14	4.2277e-14	4.2277e-14
450.0	1.793485e-10	4.46407e-14	4.34152e-14	4.34152e-14
440.0	1.727340000000002e-10	4.59727e-14	4.44794e-14	4.44794e-14
430.0	1.68158e-10	4.75638e-14	4.70567e-14	4.70567e-14
420.0	1.651845e-10	4.93299e-14	5.02422e-14	5.02422e-14
410.0	1.633485e-10	5.12834e-14	4.97368e-14	4.97368e-14
400.0	1.631090000000002e-10	5.34508e-14	4.93416e-14	4.93416e-14
390.0	1.621160000000002e-10	5.57887e-14	4.87553e-14	4.87553e-14
380.0	1.622780000000002e-10	5.83692e-14	5.36995e-14	5.36995e-14
370.0	1.6244675e-10	5.95838e-14	5.36995e-14	5.36995e-14
360.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
350.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
340.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
330.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
320.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
310.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
300.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
290.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
280.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
270.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
260.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
250.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
240.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
230.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
220.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
210.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
200.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
190.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
180.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
170.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
160.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
150.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
140.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
130.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
120.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
110.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14
100.0	1.624675e-10	5.95838e-14	5.36995e-14	5.36995e-14

L'errore commesso in relazione alla precisione delle due curve generate è rappresentato dal seguente plot:



Quello che si può notare è che i valori di energia relativi a Python e MATLAB sono praticamente gli stessi dal momento che l'errore corrispondente è pari a zero. Per quanto riguarda, invece, i valori di delay, essi risultano essere leggermente differenti visto che il massimo errore corrispondente è dell'ordine di  $10^{-11}$ .

## 4. Conclusions

Dopo aver ottenuto la curva energy-delay ottima è possibile fare alcune considerazioni a riguardo. Tale curva rappresenta un insieme di punti ottimi di lavoro per il buffer in questione. Tanto è vero che essi sono delle coppie (*energy, delay*) , cioè punti che rappresentano un punto di lavoro in corrispondenza di una determinata coppia di factor sizing ( $S_1, S_2$ ) ottima. Pertanto, analizzando tale insieme di punti, è possibile individuare una coppia ( $S_1, S_2$ ) ottima che permette di adottare un approccio low-power per il buffer considerato.

Optimal Point for Buffer

delay	energy	s1	s2
4.82255e-10	3.18069e-14	1.0	1.332644227697826
3.32125e-10	3.39098e-14	1.0	1.363537496180513
2.4813e-10	3.70632e-14	1.0	1.395956551525429
2.14257e-10	4.12587e-14	1.0	1.430022058248067
1.935765e-10	4.31029e-14	1.0	1.4658682750511827
1.809275e-10	4.44167e-14	1.0	1.5036450983061158
1.793855e-10	4.46407e-14	1.0	1.5435205094464253
1.7273400000000002e-10	4.59727e-14	1.0	1.585683485422175
1.68158e-10	4.75638e-14	1.0	1.6303475577361939
1.651845e-10	4.93299e-14	1.0	1.677755103385772
1.633485e-10	5.12834e-14	1.0	1.728182638757264
1.6240500000000002e-10	5.34308e-14	1.0	1.7819473832165638
1.6211600000000002e-10	5.57887e-14	1.0	1.8394154067946091
1.6227800000000002e-10	5.83692e-14	1.0	1.901012013052264
1.624675e-10	5.95838e-14	1.0	1.9672349404635836
1.624675e-10	5.95838e-14	1.0	2.038671517121078
1.624675e-10	5.95838e-14	1.0	2.1160211588039393
1.624675e-10	5.95838e-14	1.0	2.2001254858297914
1.624675e-10	5.95838e-14	1.0	2.2920093158787296
1.624675e-10	5.95838e-14	1.0	2.392937677905153
1.624675e-10	5.95838e-14	1.0	2.5044971906237192
1.624675e-10	5.95838e-14	1.0	2.6287155540090743
1.624675e-10	5.95838e-14	1.0	2.7682433639987205
1.624675e-10	5.95838e-14	1.0	2.926642381746395
1.624675e-10	5.95838e-14	1.0	3.1088676082828397
1.624675e-10	5.95838e-14	1.0	3.3221281642555898
1.624675e-10	5.95838e-14	1.0	3.577565865916304
1.624675e-10	5.95838e-14	1.0	3.89395241225638
1.624675e-10	5.95838e-14	1.0	4.307474758219145
1.624675e-10	5.95838e-14	1.0	4.9074588599479405
1.624675e-10	5.95838e-14	1.0	6.160076771789194
1.624675e-10	5.95838e-14	1.0	7.07106739840253
1.624675e-10	5.95838e-14	1.0000000000000002	7.07106841148198
1.624675e-10	5.95838e-14	1.0	7.071068065890352
1.624675e-10	5.95838e-14	1.0000000000000004	7.0710682761278765
1.624675e-10	5.95838e-14	0.9999999999999991	7.07106777318688
1.624675e-10	5.95838e-14	1.0	7.071068243675197
1.624675e-10	5.95838e-14	0.9999999999999984	7.07103619064159
1.624675e-10	5.95838e-14	0.9999999999999966	7.071067841422276
1.624675e-10	5.95838e-14	1.0000000000000027	7.071067959100051
1.624675e-10	5.95838e-14	4.1715741978037775	14.442251689075418
1.624675e-10	5.95838e-14	4.724062711129795	15.368898974032227

Dalla tabella sopra allegata, generata tramite lo script *buffer\_optimized\_plotting.py*, è possibile notare l'insieme di valori corrispondenti alla curva ottima precedentemente ottenuta tramite l'algoritmo di ottimizzazione `scipy.optimize.minimize` presente in Python. Pertanto, analizzando tale tabella è possibile ricavare un punto di lavoro ottimo che risulti essere un buon compromesso tra energy e delay per il buffer considerato.