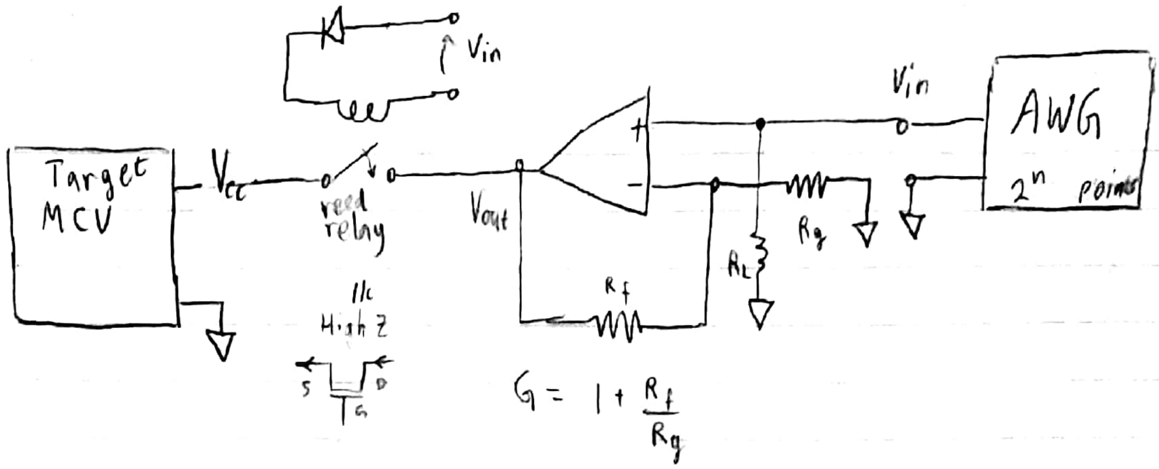
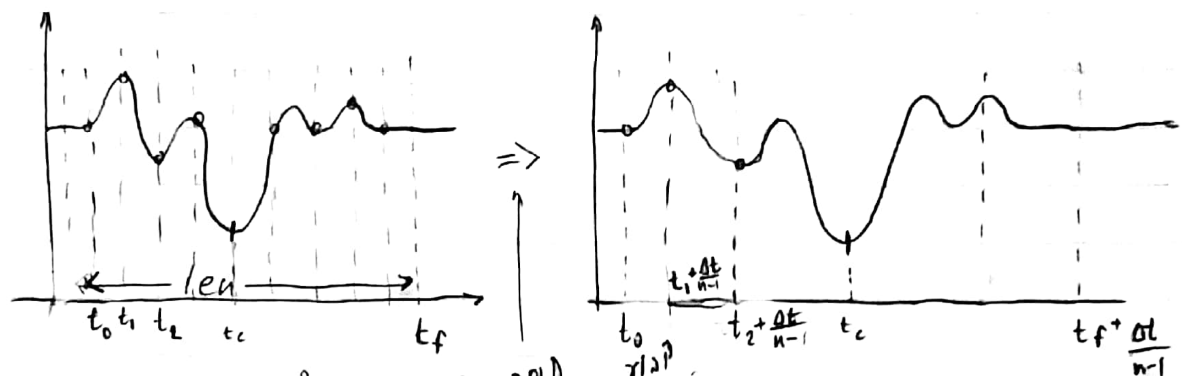


$$G = 1 + \frac{R_f}{R_g}$$



תוספת ל-len של Δt (אחרי crossover או מוטציה Δt יכול להיות חיובי או שלילי)

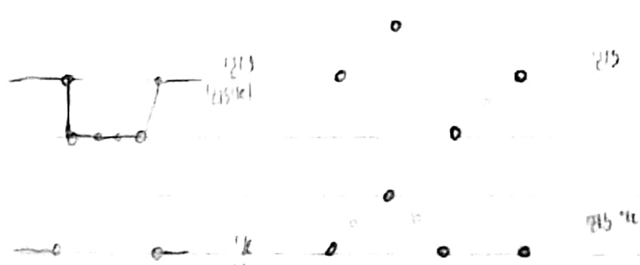
כל נקודה במרחק שווה מהסמוכה.
יש מספר שווה של נקודות מימין ומשמאל לנק' מרכז. סה"כ מספר אי זוגי של נקודות.

הרחבה ימינה או קיצור שמאלה

כל נקודה זזה לכיוונה המתאים
בשיעור: $\frac{\Delta t}{n-1}$ ← מס' נקודות פחות המרכז. (תמיד זוגי)

בתהליך crossover, נקודות (x,y) מתחלפות בצורה הבאה: עבור שתי נקודות (x_1, y_1), (x_2, y_2) של שני גרפים שונים, y_1 מתחלף עם y_2. לגבי ה x-ים: שתי אפשרויות: או שהם לא משתנים או שתהיה החלפה של רוחב (תלוי הסתברות).

אני מקבע רזולוציה בזמן: 10ns



$$\frac{\pi}{\frac{n-1}{2}} = \frac{2\pi}{n-1}$$

Search for good parameter values.

Waveform ⟹ Timing — doing this seperation reduces problem complexity.

(Waveform ↑) (Timing ↑ multiple sweeps on actual target)

(like in lpc1343 gl...)

When searching for a waveform, test it on your own simple loop program.

each solution/individual is tested N times:

another option for fitness: $F = \sum Results$. each Result $\in$ {reset, normal, sucess, undefined}

success > undefined > reset > normal
(10) (7) (2) (0)

- When mutating - use random? or gauss?

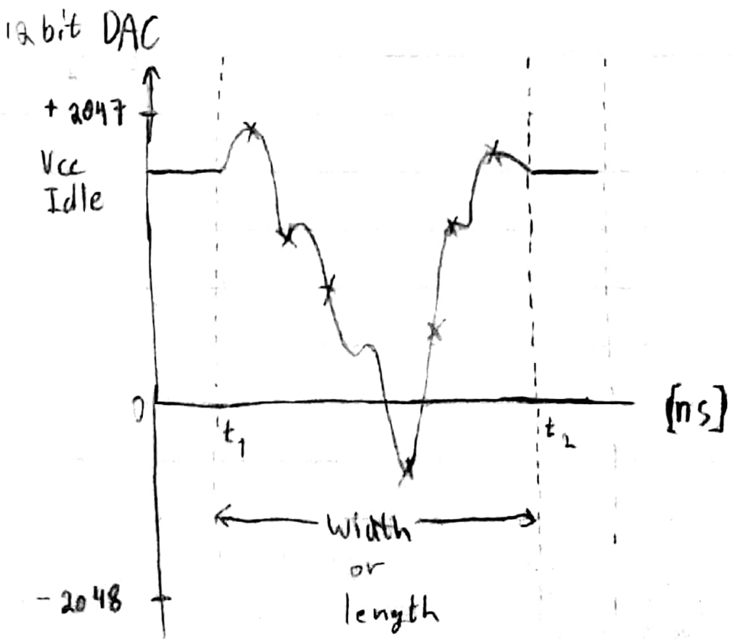
להכניס איזשהו local search כאשר הפתרונות מספיק "קרובים" או
לאחר מספר דורות כלשהו. כאשר GA נגמר. <Local search>
כדי למצוא shape טוב יותר.

מי שיש לו F גבוה יותר, יהיה בעל הסתברות גבוהה יותר לבחור
כהורה.

והסתברות מעניינת שאלה לבדוק לcrossover

- זה גורם ל GA שמחק את תכונות הפיזור.

```
for parameter/gene p do
    child.p = random in range[parent1.p, parent2.p]
end for
```



our team uses agilent 33250A arbitrary waveform generator.

each waveform can contain between 1 (DC voltage) to 65536 (64K) data points. or 16384 (16K)

According to manual.
Interpolation can be enabled: Linear or can be disabled: Step-like

max frequency 25MHz for AWG
max voltage: ±5V for 50Ω, ±10V for HiZ

Parameter Space - from shaping the glitch paper

Finite set (from 4 to 10) of (x, y) coordinates that are interpolated with cubic interpolation on a 2048 x 4096 grid, and fed into DAC. The glitch length (width) is encoded as frequency or period of the arbitrary waveform generated. That's along with other parameters: timing, and voltage (idle)

fitness function $= F = \frac{n_S}{n_T} - n_P$

$-\alpha \leq F \leq \alpha \quad \alpha \in \mathbb{R}$

n_P = penalty: target hangs or reset or false positive, i.e., incorrect byte extracted

n_S = number of successful glitches/tests

n_T = total number of tests.

Population is generated with random chromosomes.
each chromosome is an individual that contains the glitch parameters: - delay, timing, width, voltage idle, and (x, y) coordinates.

Selection: fitness proportionate and tournament are acceptable.
Crossover: uniform, in particular, every (x, y) point in the glitch waveform can be mixed between 2 parents with 0.5 probability.
Mutation: every parameter has a different mutation probability. The glitch width parameter has the highest probability, the waveform has a greater probability in the first generations, together with a higher likelihood of mutating by a small extent. elite individuals do not mutate, so we won't lose them.
Replacement: a replace-worst strategy is adopted, which replaces the worst individual of current population.
⊛ Mating of 2 parents create 2 offsprings
Start with 50 tests per individual and increment this value at each generation
The algorithm coverges when the success rate is good enough within the timeout limit.

Target responses: NORMAL - glitch did not affect, RESET - glitch resets the target,
Classification ~~MUTE - target stops all communication due to glitch,~~
~~CHANGING - target gives different responses for the same glitch~~
UNKNOWN - undefined response
SUCCESS - response is a specific predetermined that does not happen under normal operation. {map this to fitness?