Wprowadzenie do impulsowych sieci neuronowych (SNN – Spiking Neural Network)

Metody symulacji i implementacji

Prowadzący: dr inż. Andrzej Skoczeń
CERN, TE-MPE-EP
AGH, KOiDC
e-mail: skoczen@fis.agh.edu.pl

Wykład 2
2022

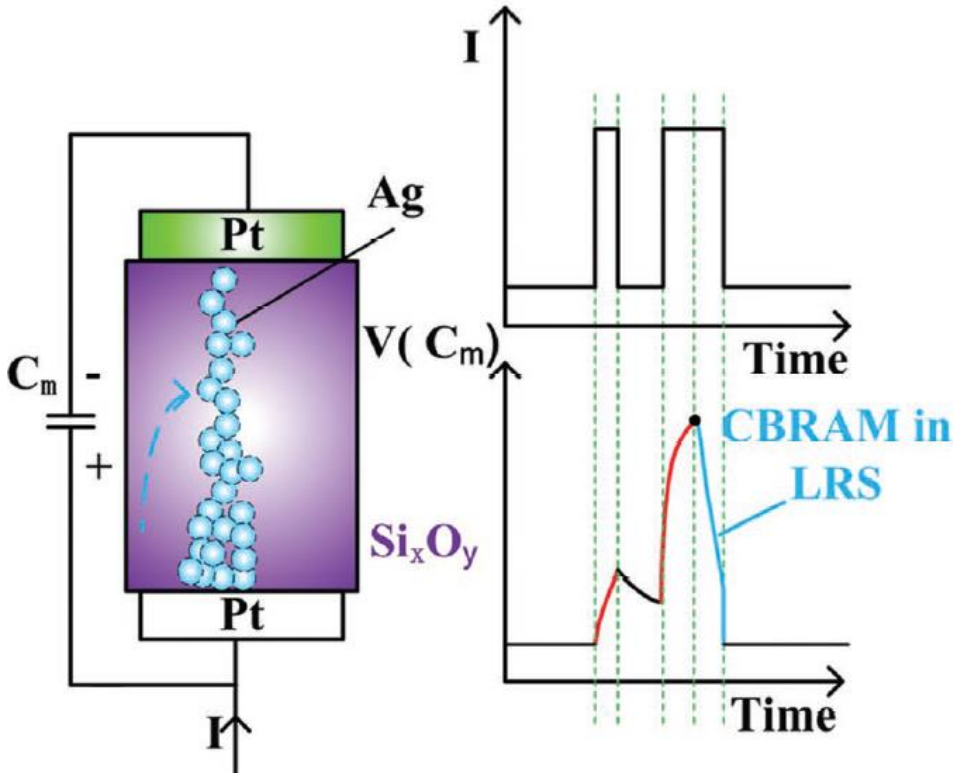
15 lipiec 2022

Fizjologia – technologia - informatyka:

- Memrystory c. d.
- Analogowe mnożenie wektor- macierz MVM
- Plastyczność neuronalna
- Jak to symulować?
- Przykłady projektów badawczych

Mechanizmy sztucznych neuronów i synaps

CBRAM – sztuczny neuron



CBRAM - oparte na metalowym włóknie (dyfuzyjne memrystory) naśladują zarówno funkcję “integrate-and-fire” (neuron) jak i funkcje synaptyczną.

Włókna Cu lub Ag (lub inny elektrochemicznie aktywny metal) lub nanoklastry są formowane lub rozpuszczane w celu dostrojenia przewodności imitującej zmianę wagi synaptycznej.

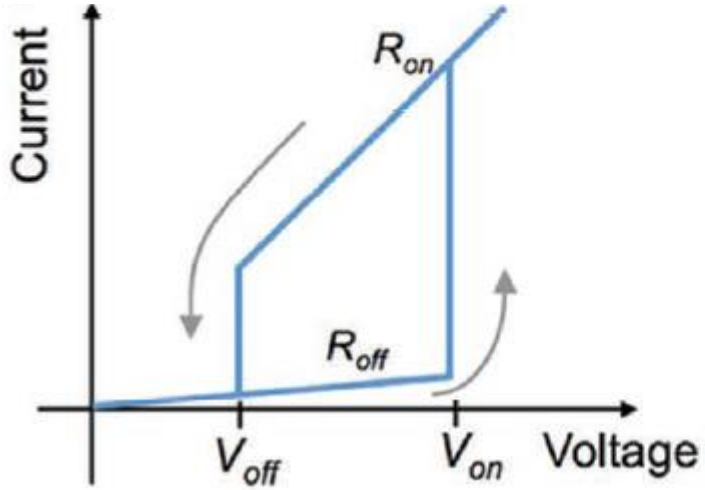
Urządzenia te spontanicznie po odpaleniu powracają do swojego wysoce rezystancyjnego stanu wyłączenia (bez potrzeby operacji RESET), co bardzo przypomina proces repolaryzacji neuronów biologicznych.

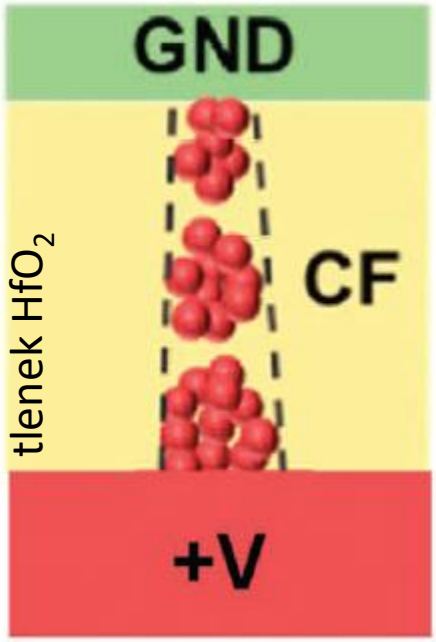
Przełączanie rezystancyjne

Stosuje się różne typy mechanizmów fizycznych, które najczęściej prowadzą do **przełączania rezystancyjnego**.

- Neuronopodobne zachowania integrujące i odpalające są zwykle realizowane przez progowe charakterystyki przełączania memrystorów - **przełączanie progowe**
- Zmiany wag synaptycznych w sztucznych synapsach są modulowane za pomocą analogowego (przyrostowego) nieulotnego przełączania na różne poziomy rezystancji (lub konduktancji) - **przełączanie pamięci**

Przełączanie progowe w sztucznych neuronach





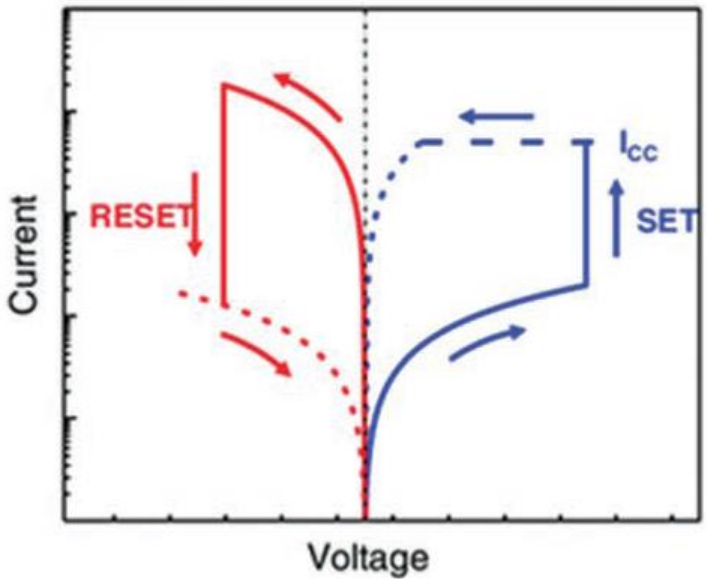
Przełączanie progowe w elementach RRAM (w trybie niskiego napięcia roboczego) może być wykorzystywane jako ulotne zachowanie „włączeniowe” (turn-on), przypominające funkcję neuronów.

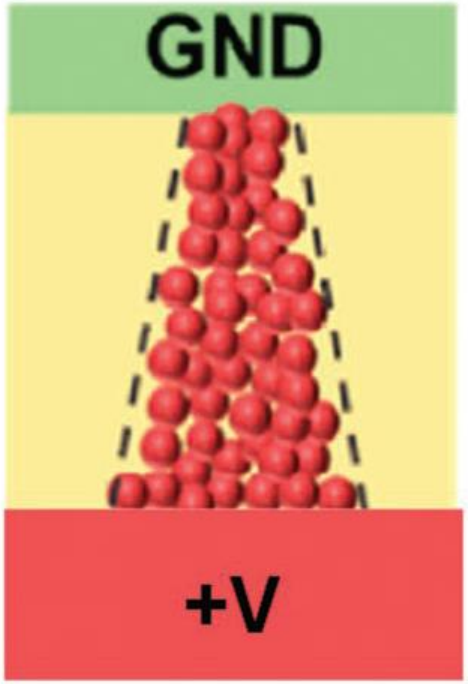
Wakansje tlenowe są generowane w warstwie tlenku pod wpływem zewnętrznego pola elektrycznego i dryfują w kierunku przeciwelektrody, tworząc przewodzące włókno.

Niska podatność na napięcie sprawia, że uformowane włókno pozostaje cienkie i niestabilne, tak że spontanicznie rozpada się po usunięciu polaryzacji.

Taki przejściowy skok przewodnictwa emuluje zachowannie obserwowane w neuronach biologicznych.

Przełączanie pamięci w sztucznych synapsach



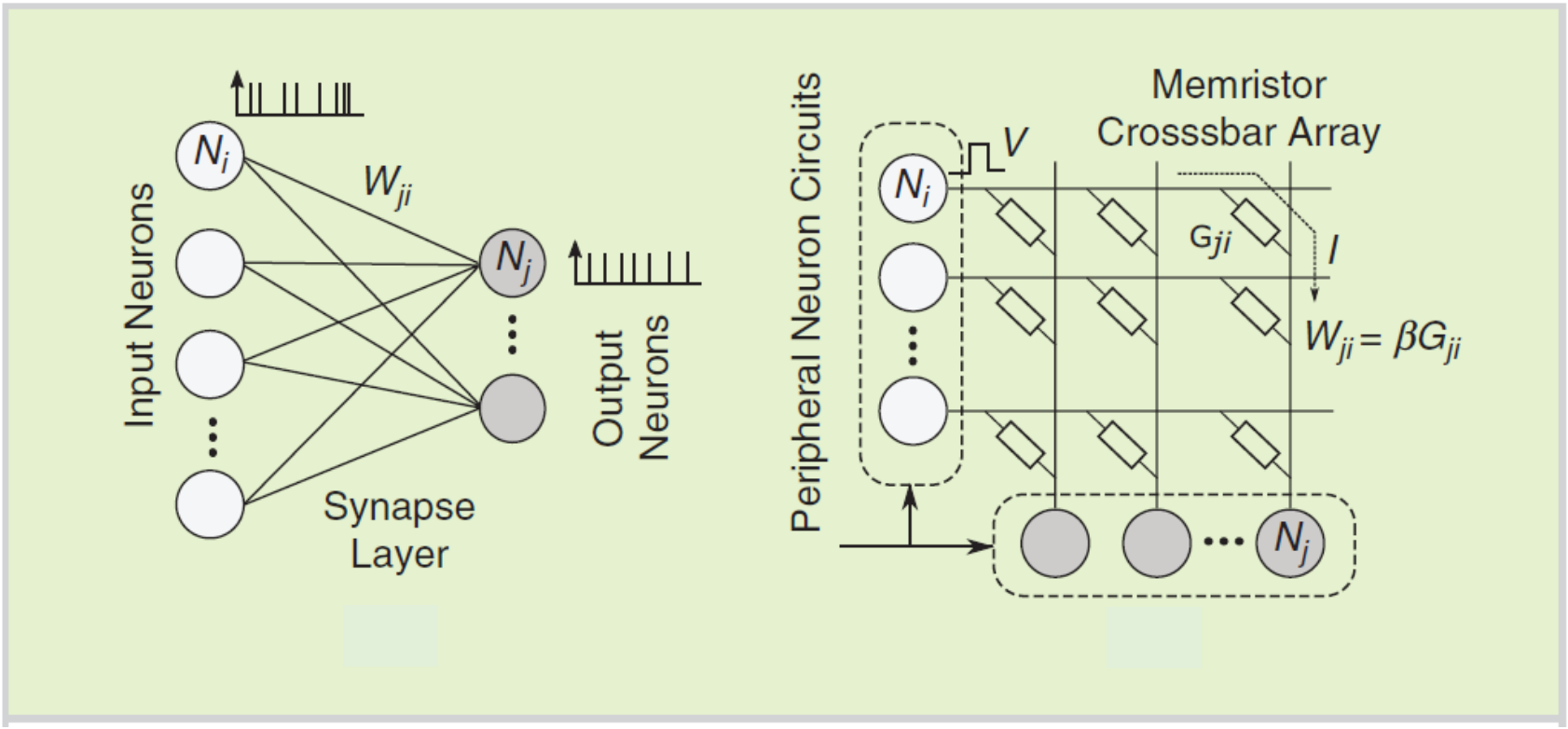


Element do implementacji analogowych synaps potrzebuje właściwości stopniowego przełączania (modulowania) konduktancji.

Różne poziomy stabilnej konduktancji można uzyskać poprzez staranne dostosowanie amplitudy lub szerokości impulsu programującego wielkości włókna przewodzącego i/lub ich liczby.

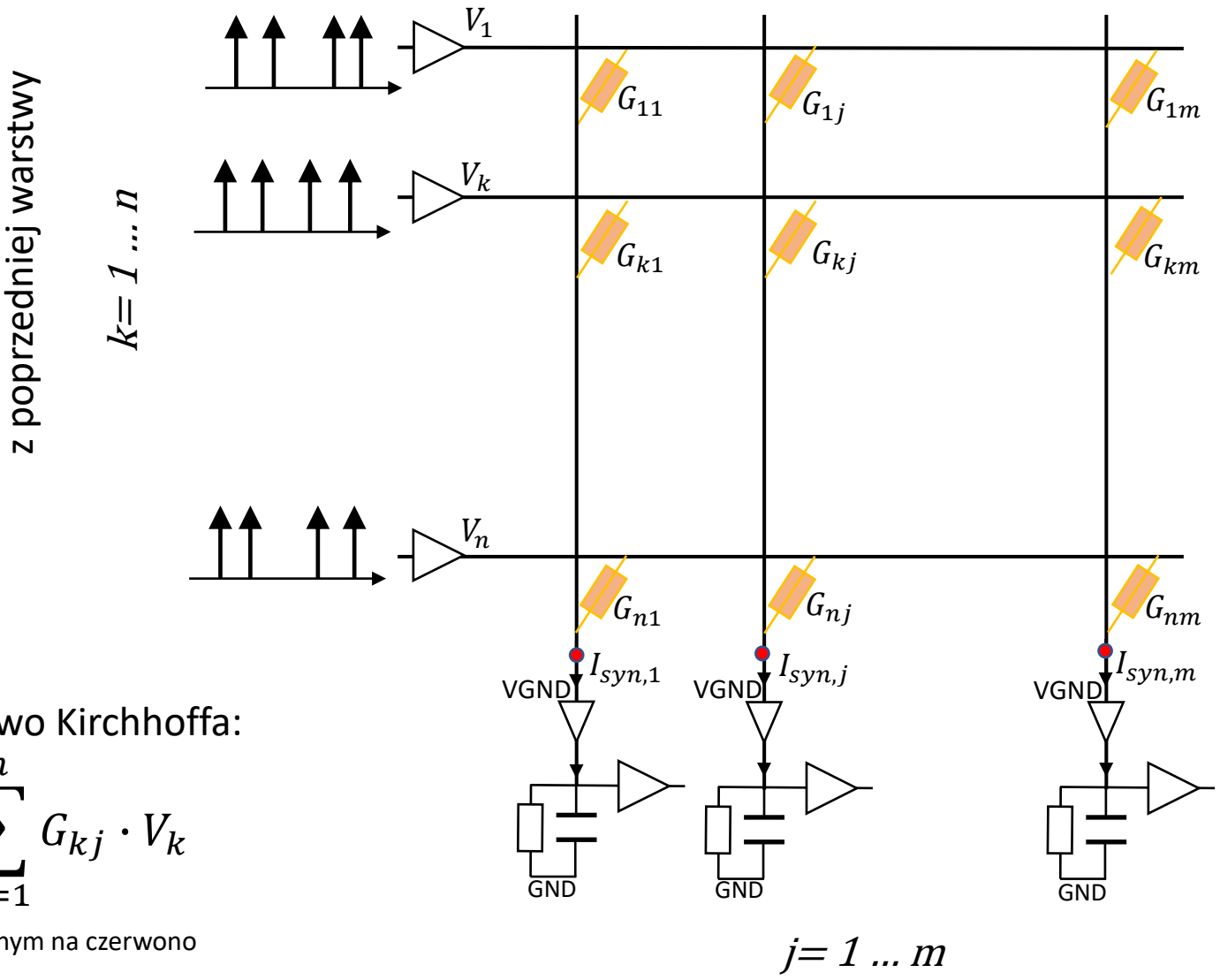
Macierze krzyżowe

Macierze krzyżowe (cross-bar) pamięci memrystycznej do implementacji jednostek mnożących macierz-wektor MVM (Matrix-Vector-Multiplication) w architekturach przyszłych analogowych komputerów dla aplikacji sieci neuronowych.



Macierze poprzeczne z urządzeniami memrystywnymi na skrzyżowaniach skutecznie implementują równoległą łączność, komunikację synaptyczną i plastyczność w sprzęcie.

Macierze krzyżowe



Pierwsze prawo Kirchhoffa:

$$I_{syn,j} = \sum_{k=1}^{n} G_{kj} \cdot V_k$$

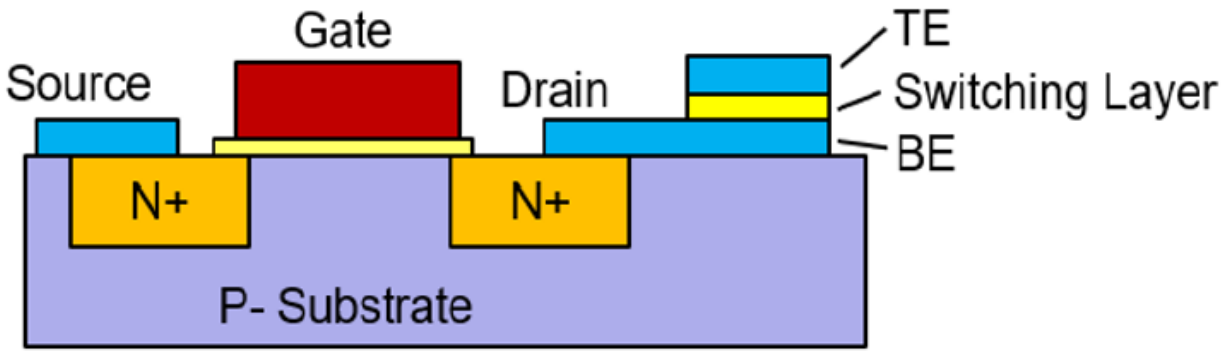
W węźle oznaczonym na czerwono

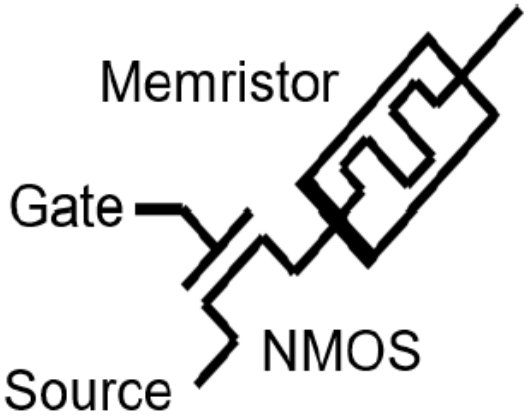
Struktura 1T1R

Konieczne jest zastosowanie tranzystora MOSFET szeregowo z każdym memrystorem :

- w celu ograniczenia prądów przepływających przez każdy memrystor, unikając uszkodzeń spowodowanych przejściowymi wysokimi prądami.

  Gdy tranzystor zostanie pominięty, ograniczenie prądu odbywa się w peryferyjnych obwodach CMOS, ograniczając rozmiar macierzy w celu zmniejszenia ryzyka lokalnych wysokich pasożytniczych prądów przejściowych.
- również jako urządzenie selekcyjne (adresowe), które aktualizuje indywidualnie każdy memrystor, unikając zmian w pobliskich urządzeniach.





Memrystory - konkluzja

Dotychczas eksperymentalne demonstracje klasyfikacji i szkolenia opartych na pamięci analogowej (analogue in-memory) systemów uczenia się z pamięcią analogową:

- odbywały się na ograniczonych układach i
- nie osiągały monolitycznej integracji CMOS z częścią memrystyczną i
- cierpiały na niedokładności klasyfikacji z powodu niedoskonałości urządzenia memrystcznego jak:
  - trudności z poprawną aktualizacją wag synaptycznych,
  - problemy z programowaniem wartości wielopoziomowych, czy
  - niestałość zmienności zakresu przewodnictwa urządzenia.

Memrystory - konkluzja

Chociaż memrystory są bardzo obiecującą technologią do implementacji pamięci analogowych o wysokiej gęstości w pobliżu systemu obliczeniowego, który potencjalnie mógłby wdrożyć system kognitywny szybkiego uczenia się o małej mocy, nadal istnieją znaczące ograniczenia technologiczne, które są obecnie badane, i które obecnie nie pozwalają na wdrożenie takich wielkoskalowych systemów.

25 sierpnia 2020 firma TSMC ogłosiła, że oferuje wbudowaną rezystancyjną pamięć nieulotną RAM w swoich procesach produkcyjnych 40 nm i 22 nm oraz planuje MRAM w 16 nm FinFET. >

13 stycznia 2022 firma Samsung ogłosiła innowację MRAM, twierdząc, że jest to pierwsze na świecie przetwarzanie w pamięci (in-memory) oparte na MRAM umożliwiające zarówno przechowywanie danych, jak i przetwarzanie danych w ramach jednej sieci pamięci. Firma twierdzi, że jej układ macierzy MRAR jest kolejnym krokiem do realizacji układów AI o niskim poborze mocy. >

Najbardziej krytyczna cecha sztucznych synaps jest wykazywanie stopniowych i wiarygodnych zmian przewodnictwa.

Dla praktycznych aktualizacji przewodnictwa zostały opracowane głównie trzy podejścia:

- Jednym z nich jest użycie dwóch precyzyjnie zaprojektowanych impulsów podawanych na dwa zaciski urządzenia, tak aby przewodność była modulowana tylko wtedy, gdy **nakładają się w czasie** (overlapping). Może to także umożliwić równoległe aktualizacje w macierzy.
- Drugie podejście polega na zastosowaniu **ciągu impulsów**, który w sposób ciągły programuje urządzenie docelowe, aż zmiana przewodności osiągnie żądaną wartość przed zaprogramowaniem następnego - aktualizacja sekwencyjna.
- Trzecie podejście polega na wbudowaniu **dynamiki relaksacji** w sztucznej synapsie, tak aby mogła “wyczuć” różnicę czasu między impulsami przed i po, i odpowiednio modulować wagę synaps, tak jak robi to synapsa biologiczna.

Plastyczność neuronalna

W systemie biologicznym odnosi się do zdolności neuronów lub synaps do zmiany ich właściwości, często zależnej od aktywności neuronalnej i uważa się, że stanowi podstawę funkcji uczenia się i pamięci mózgu.

W układzie sztucznym plastyczność zachodzi przez zmiany przewodności urządzenia.

Plastyczność synaptyczna to zdolność do zmiany właściwości synaptycznych, najczęściej wzmocnienie lub osłabienie siły synaptycznej. Plastyczność synaps można podzielić na:

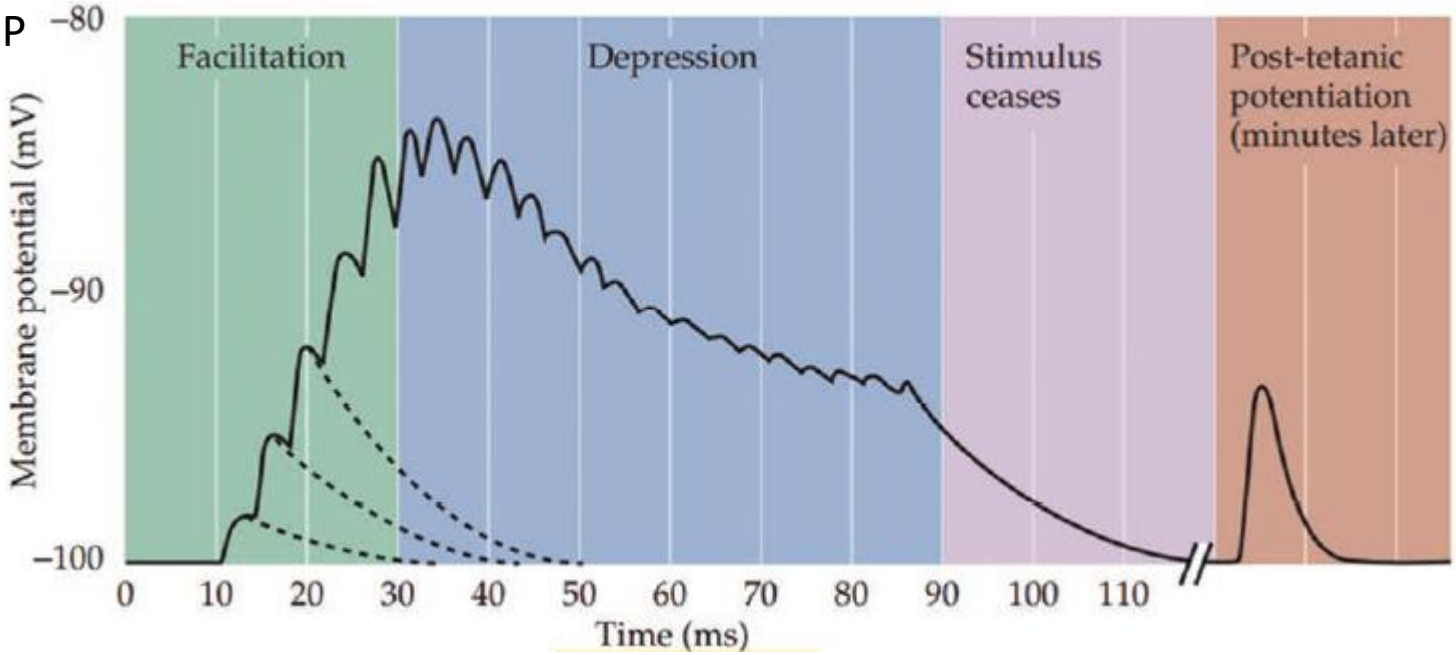
- krótkoterminową *STSP (Short-Term Synaptic Plasticity) i*
- długoterminową *LTSP (Long-Term Synaptic Plasticity)*

w zależności od czasu trwania jej oddziałania na siłę synaptyczną.

Skutecznością synaptyczną nazywamy zdolność wejścia presynaptycznego do wpływania na wyjście postsynaptyczne.

STSP (Short-Term Synaptic Plasticity)

Przebieg czasowy STSP



STSP trwa kilka minut lub krócej. Wpływ STSP na skuteczność synaptyczną może być albo wzmocnieniem, albo osłabieniem. Wzmocnienie synaptyczne może być:

- ułatwienie sparowanych impulsów (PPF) i
- wzmocnienie poskórczowe (PTP) zgodnie ze skalą czasową.

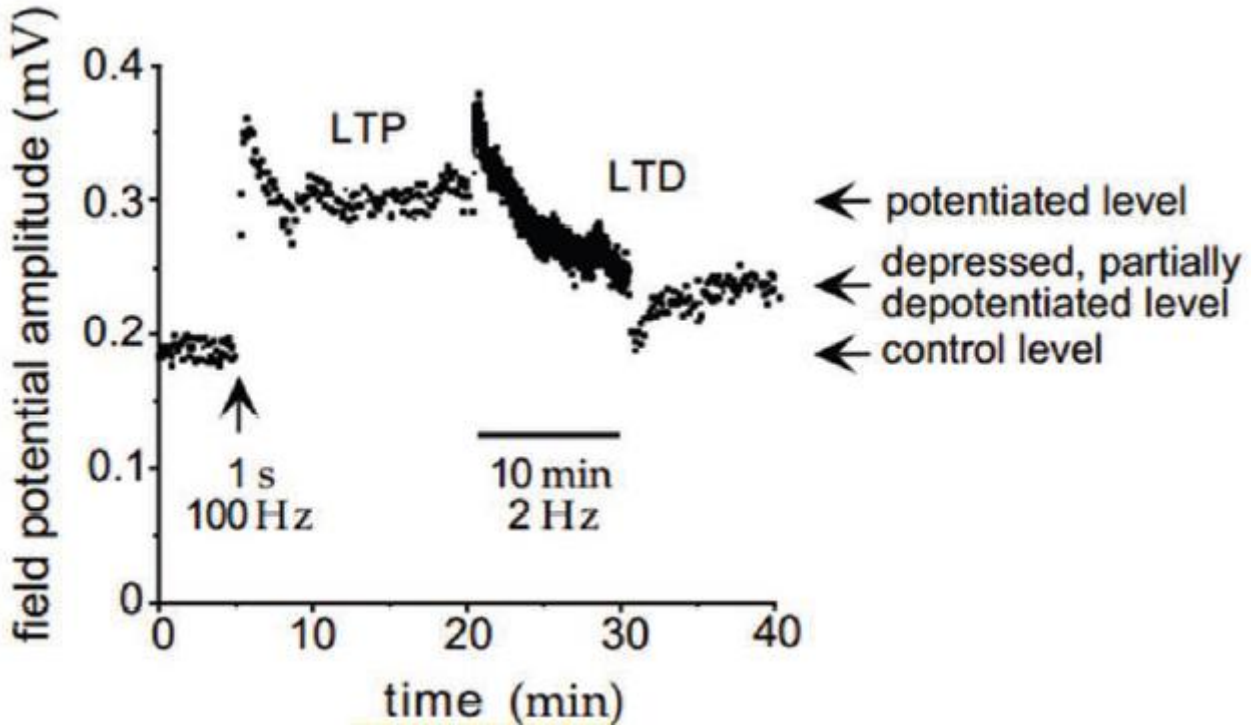
PPF występuje, gdy dwa AP docierają do synapsy w krótkim odstępie czasu w ciągu kilkudziesięciu milisekund. Jest to najprawdopodobniej spowodowane zwiększonym poziomem Ca^{2+} w komórce presynaptycznej po AP. Wysokie stężenie Ca^{2+} umożliwia uwalnianie większej liczby neuroprzekaźników i powoduje większy potencjał postsynaptyczny pobudzający. PTP działa w dłuższej skali czasu od kilkudziesięciu sekund do minut.

LTSP (Long-Term Synaptic Plasticity)

LTSP może trwać kilka godzin lub dłużej.

- Trwały wzrost siły synaptycznej nazywany jest długotrwałym wzmocnieniem **LTP** (Long-Term Potentiation).
- przeciwnie, uporczywy spadek siły synaptycznej jest znany jako długotrwała depresja **LTD** (Long-Term Depression).

Mechanizmy komórkowe i molekularne LTP lub LTD różnią się w różnych synapsach.



Schemat LTP i LTD. Stymulacja o wysokiej częstotliwości przez krótki okres indukuje LTP, podczas gdy stymulacja o niskiej częstotliwości przez długi okres indukuje LTD.

Dla SNN nie ma znanych skutecznych metod nadzorowanego szkolenia (supervised training), które prowadzą do wyższej wydajności niż sieci drugiej generacji.

Popularne strategie uczenia się z propagacją wsteczną BP (backpropagation) nie są bezpośrednio stosowalne do sieci SNN, gdyż:

- jeśli skoki (spikes) są reprezentowane obliczeniowo jako wystąpienie zdarzenia wyjściowego w określonym czasie, **nie są one różniczkowalne**;
- różniczkowanie błędu powracającego w warstwach przestrzennych (tak jak ma to miejsce w algorytmie BP wstecznej propagacji błędów) powoduje **utratę kluczowej informacji czasowej** zawartej w przebiegu czasowym pików (spikes).

Dla właściwego wykorzystania zalet SNN konieczne jest opracowanie skuteczych metodę nadzorowanego uczenia się, które jednocześnie uwzględnia przestrzeń i czas.

Przyjęto kilka podejść:

- Trenowanie ANN (II generacji) i konwersja do SNN
- Nadzorowane szkolenie w dziedzinie impulsowej
- Szkolenie nienadzorowane w dziedzinie skoków
- Łączenie nienadzorowanych metod wyodrębniania cech z nadzorowanym szkoleniem w zakresie kategoryzacji

Nadzorowane szkolenie w SNN

Metody uczenia SNN opierały się na adaptacji reguły Delta Learning i były odpowiednie do uczenia architektur jednowarstwowych.

Aby zastosować regułę wstecznej propagacji BP opadania gradientu do domeny czasowej:

- stosuje się takie kodowanie czasów impulsów, aby mieć różniczkowalny związek z podzbiorem poprzednich impulsów lub
- przybliżenie aktywności odpowiedzi na kształt impulsu, aby była zróżnicowana między warstwami neuronowymi.

Metoda wsteczną propagację w czasie BTT (back-propagation-through-time) przybliża kształt impulsu jako funkcję ciągłą różniczkowalną, która propaguje błąd wsteczny w przestrzeni i w wymiarze czasowym.

Metoda SLAYER uwzględnia propagację wsteczną w przestrzeni i czasie oraz trenuje zarówno wagi, jak i opóźnienia połączeń synaptycznych.

Plastyczność zależna od czasu impulsu

Spike-Timing-Dependent Plasticity (STDP)

1940 – Hebb - sformułował regułę plastyczności synaptycznej. Stwierdza ona, że synapsy zwiększają swoją wydajność, jeśli trwale biorą udział w odpalaniu neuronu postsynaptycznego.

1993 – Gerstner - opracował algorytmy uczenia się STDP jako udoskonalenie tej reguły, biorąc pod uwagę dokładny względny czas poszczególnych impulsów przed- i postsynaptycznych, a nie ich średnie wartości w czasie.

W porównaniu z tradycyjną plastycznością opartą na korelacji Hebba, STDP okazał się lepiej przystosowany do wyjaśniania zjawisk w korze mózgowej i okazał się skuteczny w uczeniu się ukrytych wzorców impulsów lub uczeniu się wzorców kompetycyjnych.

1998 – Makram, Bi, Poo - uczenie się STDP zostało eksperymentalnie zaobserwowane w neuronach biologicznych

Lokalne zasady uczenia się:

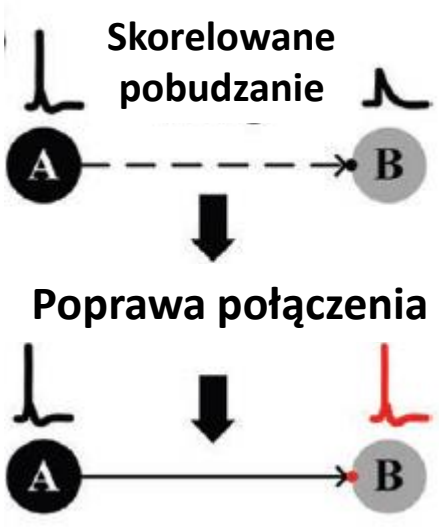
- plastyczność zależna od **czasu** impulsów (STDP) i
- plastyczność zależna od **szybkości** impulsów (SRDP).

W STDP, aktualizacja wagi synaptycznej zależy od względnego czasu między wyrzutem presynaptycznym i postsynaptycznym. Jeśli impuls neuronu presynaptycznego (PRE) poprzedza impuls neuronu postsynaptycznego (POST), a mianowicie względne opóźnienie impulsów, $\Delta t = t_{post} - t_{pre}$, jest dodatnie, to interakcja między dwoma impulsami powoduje zwiększenie wagę synapsy, co nosi nazwę wzmocnienia synaptycznego. Z drugiej strony, jeśli skok PRE następuje po skoku POST, tj. Δ t jest ujemny, wówczas waga synapsy ulega zmniejszeniu czyli depresji synaptycznej.

W SRDP tempo impulsów emitowanych przez neurony stymulowane zewnętrznie dyktuje wzmocnienie lub depresję synapsy, przy czym stymulacja o wysokiej i niskiej częstotliwości prowadzi odpowiednio do wzmocnienia i depresji synaptycznej. W przeciwieństwie do STDP opierającego się na parach impulsów, SRDP przypisano złożonej kombinacji trzech kolców (tryplet) lub więcej.

Plastyczność zależna od szybkości impulsów

Zasada uczenia się **Hebba** mówi, że wzrost wydajności synaptycznej wynika z powtarzającej się i trwałej aktywacji komórki postsynaptycznej B przez komórkę presynaptyczną A.



$$\Delta w_{ij} = \beta \cdot f_i(a_i) \cdot f_j(a_j)$$

gdzie
Δw_{ij} to zmiana siły połączenia między neuronami i oraz j,
β to parametr szybkości uczenia się i określa wielkość zmiany, a
$f_i(a_i)$ i $f_j(a_j)$ to funkcje zależne od aktywności presynaptycznej (a_i) i aktywnosci postsynaptycznej (a_j), odpowiednio

Zasada uczenia się **BCM** (Bienenstock, Coopera, Munro), w której przesuwający się próg indukcji LTP lub LTD jest określany przez ogólną aktywność postsynaptyczną:

- niska aktywność postsynaptyczna sprzyja LTD (osłabienie), natomiast
- wysoka aktywność postsynaptyczna sprzyja LTP (wzmocnienie).

Plastyczność zależna od szybkości impulsów

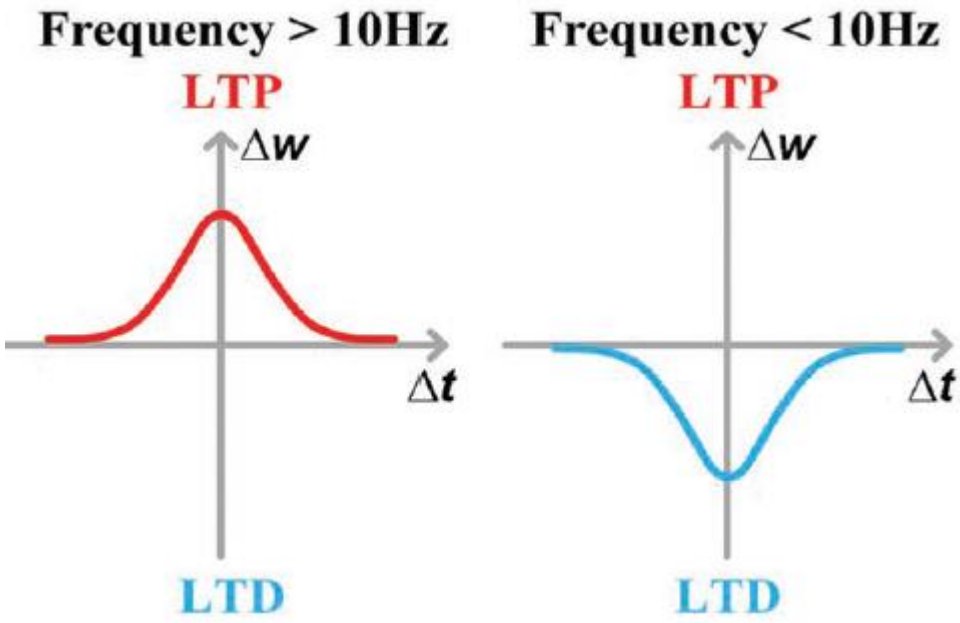
SRDP Spike Rate-Dependent Plasticity

SRDP to zmiana przewodnictwa w memrystorach zgodnie z stymulacją pewnym ciągiem impulsów (spike train) o określonej częstości (rate) impulsów.

Różne typy memrystorów wykorzystano do pokazania SRDP w odpowiedzi na przychodzące impulsy o różnych częstotliwościach:

- Ciąg impulsów presynaptycznych o częstotliwości (20-100 Hz) skutkuje LTP siły synaptycznej,
- podczas gdy ciąg impulsów o niskiej częstotliwości (1-5 Hz) skutkuje LTD.

Protokoły te są zgodne z aspektami reguły BCM.



LTP - Long-Term Potentiation
LTD - Long-Term Depression

Plastyczność zależna od czasu impulsów

STDP Spike Time-Dependent Plasticity

Zasada STDP stwierdza, że:

- połączenie synapsy jest wzmocnione, gdy aktywność presynaptyczna następuje po aktywnością postsynaptyczną przez kilkadziesiąt milisekund, i
- podobnie jest osłabiona, gdy aktywność presynaptyczna i postsynaptyczna przebiega w odwrotnej kolejności czasowej.

Ta piękna i precyzyjna zasada uczenia się w czasie została poparta wieloma badaniami dotyczącymi różnych obszarów mózgu i gatunków.

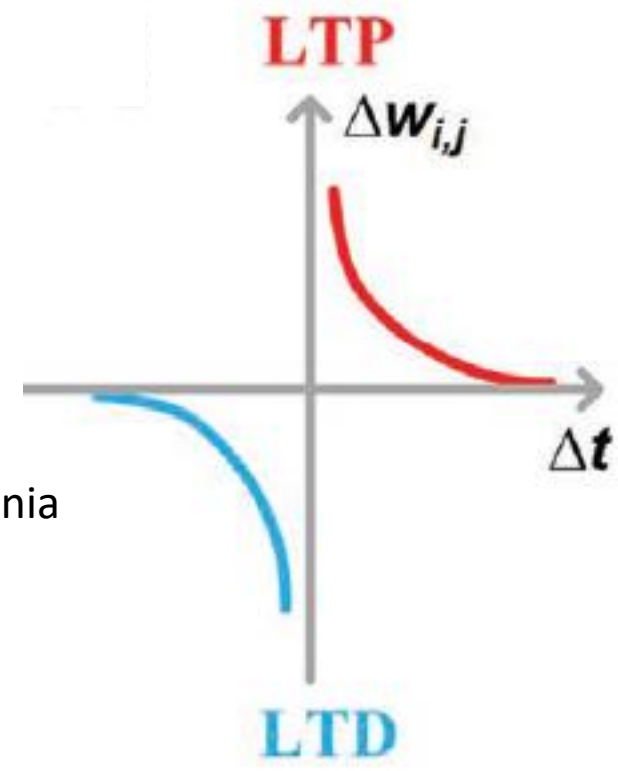
$$\Delta w = \begin{cases} \omega_0 \exp(^{-\Delta t}/_{\tau}) & \Delta t > 0 \\ -\omega_0 \exp(^{\Delta t}/_{\tau}) & \Delta t < 0 \end{cases}$$

gdzie

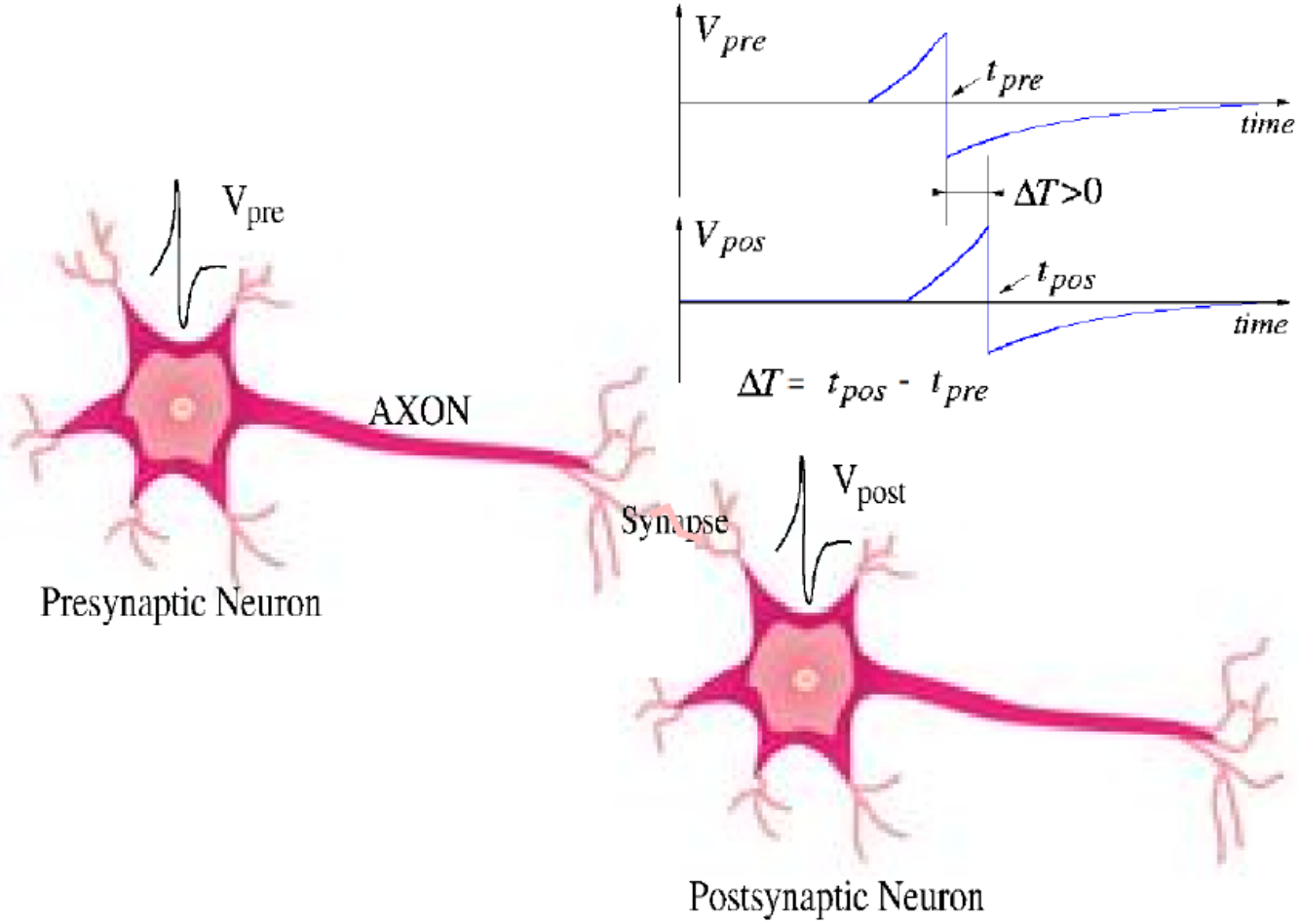
Δt jest różnicą czasu między post- i presynapsami,

w to siła połączenia , a

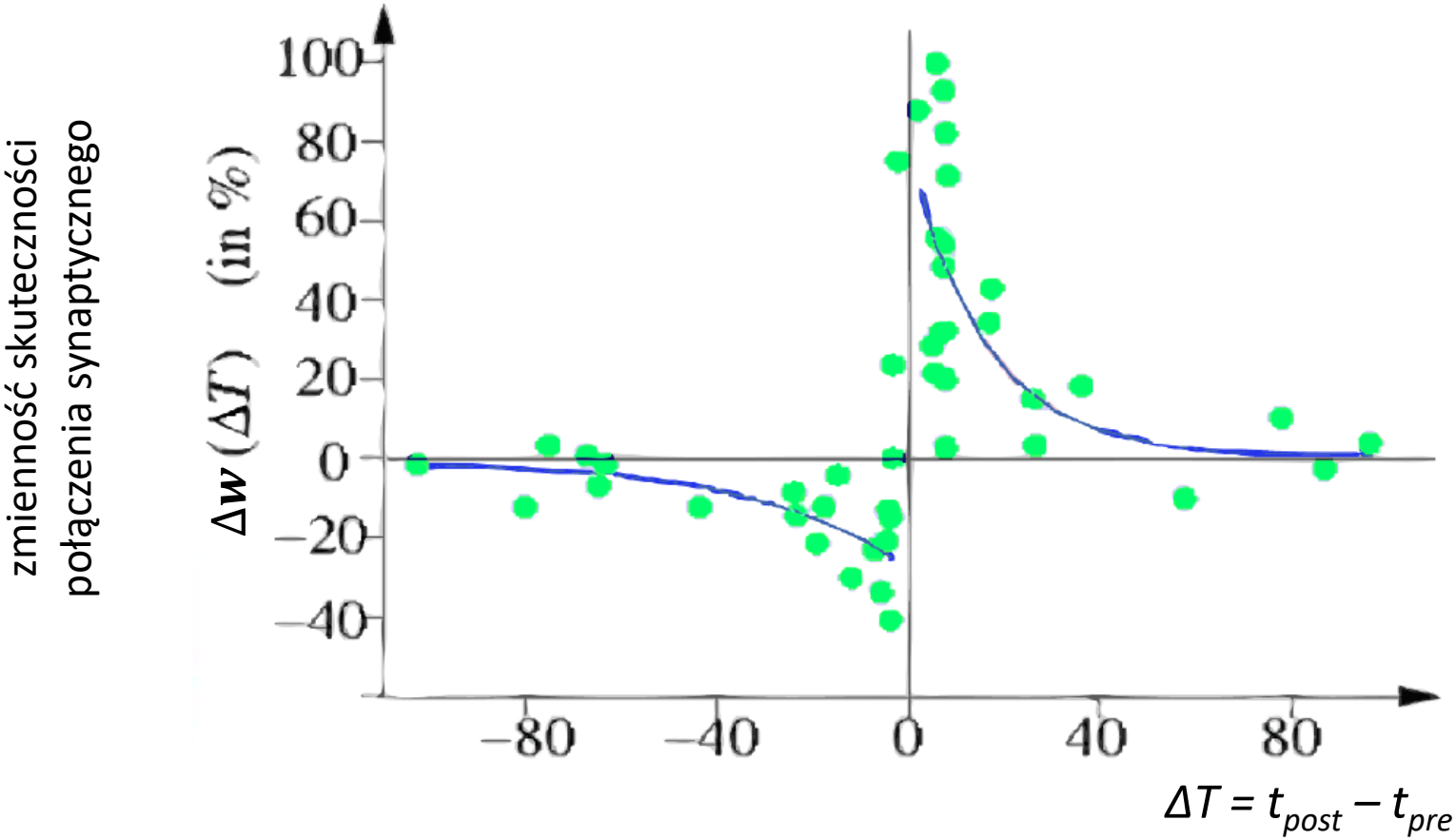
ω_0 i τ to dwie stałe, kontrolujące odpowiednio szybkość uczenia się i wrażliwość na czas.



Plastyczność zależna od czasu impulsu



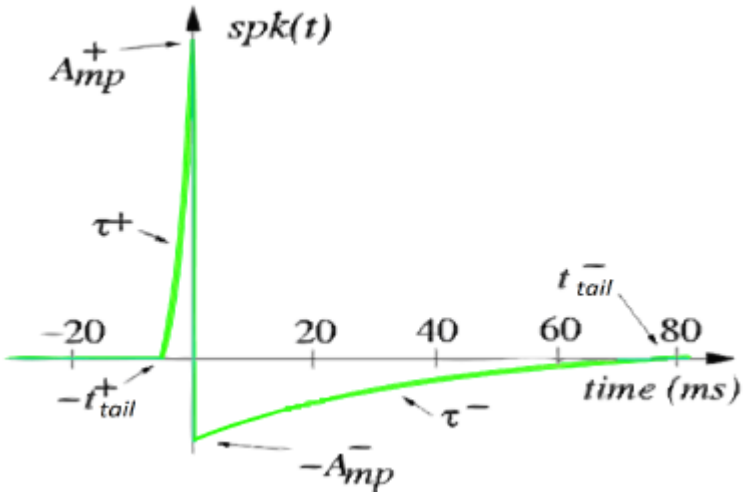
Neuron presynaptyczny o potencjale błonowym V_{pre}, który jest połączony przez synapsę o sile synaptycznej w z neuronem postsynaptycznym o potencjale błonowym V_{post}. Neuron presynaptyczny emituje impuls w czasie t_{pre}, który przyczynia się do powstania impulsu postsynaptycznego w czasie t_{post}.



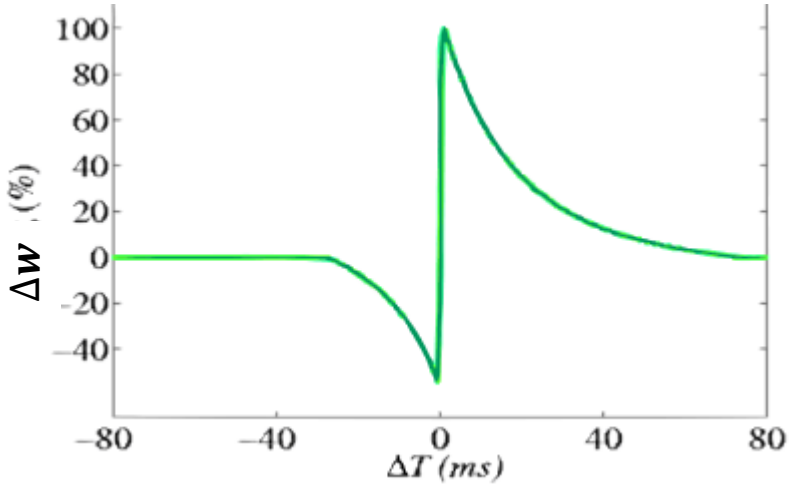
Dwa połączone neurony generują impulsy w krótkim czasie:

- jeśli $\Delta T > 0$, co oznacza, że impuls presynaptyczny przyczynił się przyczynowo do wygenerowania impulsu postsynaptycznego to istnieje pozytywna $\Delta w(\Delta T) > 0$;
- jeśli $\Delta T < 0$, to $\Delta w(\Delta T) < 0$ jest ujemna.

Modelowy przebieg potencjału czynnościowego membrany neuronu



Zmienność skuteczności synaptycznej



2009 - Linares – wykazano, że memrystancja może modelować STDP biologiczne.

Sygnały potencjałów czynnościowych (spikes), zastosowane do zacisków przed- i post-synaptycznych memrystywnego urządzenia pokazują, że zachowanie STDP pojawia się naturalnie.

Plastyczność zależna od czasu impulsów

STDP Spike Time-Dependent Plasticity

Eksperymentalnie, implementacja STDP przy użyciu memrystora polega na nałożeniu (overlaping) impulsów pre- i post- przyłożonych do zacisków elementu.

W ten sposób kluczowe jest przełożenie różnicy w timingu na różnicę w amplitudzie lub szerokości nałożonych impulsów, które następnie modulują wagę synaptyczną jako funkcję względnego timingu dwóch pików.

Symulacja sieci impulsowych

Symulatory SNN (Opne Source)

Symulator		Rodzaj	GPU	Język
Brain2	2019	clock-driven	Yes	C++ opakowany w Python
Bindsnet	2018	clock-driven	Yes	C++ opakowany w Python
PyNets	2009	clock-driven	Yes	C++ z interfejsem w Python
Nengo	2014	clock-driven	Yes	C++ opakowany w Python
NEURON	2009	clock-driven	No	C++ z interfejsem w Python
CARLsim	2015	clock-driven	Yes	C/C++ opakowany w Python
EDHA	2021	event-driven	No	Java

Obecnie metoda przyjęta przez symulatory SNN jest głównie oparta na zegarze (clock-driven), co oznacza, że każda aktualizacja obliczeń jest wykonywana w arbitralnych przedziałach czasu, a nie wtedy gdy naprawdę zachodzą zdarzeniach (events).

Sterowane zegarem symulatory dzielą czas na odcinki czasu poprzez próbkowanie czasu i realizowanie dyskretyzacji czasu w celu wykonania obliczeń.

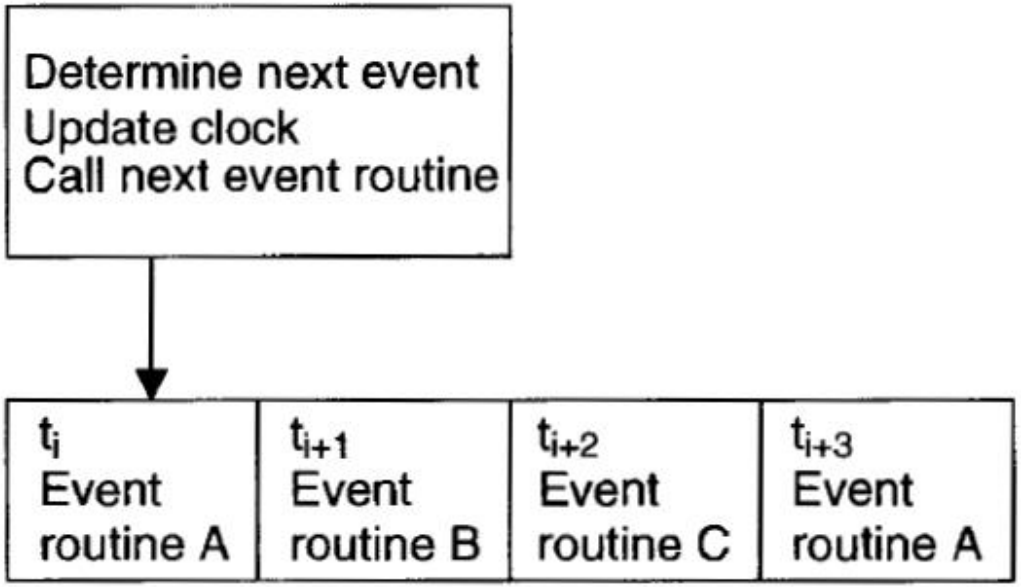
Symulacja zdarzeniowa

Symulacja zdarzeń dyskretnych to metoda reprezentacji zachowania systemu w czasie. System może być rzeczywisty lub hipotetyczny, ale zakłada się, że jego stan zmienia się w czasie z powodu kombinacji bodźca zewnętrznego i stanu wewnętrznego.

Podstawową ideą jest to, że zachowanie dowolnego systemu można rozłożyć na zbiór dyskretnych momentów czasu, w których coś się dzieje. Te chwile nazywane są zdarzeniami, a „rzeczy, które się dzieją” to zmiany stanu.

Jest to bardzo analogiczne do sposobu, w jaki digitalizujemy ciągłe zjawiska fizyczne, takie jak próbkowanie dźwięku. Zasadniczo digitalizujemy proces oparty na czasie, dzieląc go na odrębne zdarzenia. Łatwo zauważyć, że przy wystarczająco drobnej szczegółowości można uzyskać odpowiednio dokładną reprezentację dla niemal każdego celu.

Każde zdarzenie w systemie jest reprezentowane przez procedure obsługi. Procedura zdarzenia to kod, który ma być wykonany w celu reprezentowania akcji w tym zdarzeniu, co zwykle sprowadza się do zmiany stanu i określenia wystąpienia jednego lub więcej przyszłych zdarzeń. Zestaw możliwych zdarzeń jest zwykle (choć nie zawsze) stały, ale liczba wystąpień każdego typu zdarzenia może być zmienna.



lista przyszłych zdarzeń

Lista przyszłych zdarzeń jest listą tych zdarzeń, które zaplanowano na określone chwile w przyszłości (t_{i+1}, t_{i+2}, …). Pozycje na tej liście zawierają czas, w którym zdarzenie nastąpi i rodzaj zdarzenia; to znaczy, jaką rutynę zdarzeń należy wywołać, aby wykonać zachowanie tego zdarzenia. Często na tej liście zaplanowano więcej niż jedną instancję danego typu zdarzenia w różnych momentach.

Symulacja zdarzeniowa

Symulacja logiczna (Verilog, VHDL) jest zasadniczo procesem obliczania trajektorii stanu systemu w czasie. Prostą trajektorię stanu w funkcji czasu można zapisać w następujący sposób:

$$f(m, n, t+1) = m_t + n_t$$

gdzie m jest zmienną stanu, a n jest wejściem. Dodając druga funkcję:

$$g(x, n, t+1) = xt \wedge nt$$

Obliczanie tych dwóch funkcji razem wyglądałoby tak:

```
do (i= 1, time_of_end)
    t = t + 1
    m = m + n
    x = x ^ n
    n= new input
enddo
```

Ten zestaw funkcji stanu razem wziętych tworzy system ze zmienną stanu, która jest zdublowana (m, x). Uruchomienie tej symulacji oblicza trajektorię wektora stanu (m, x) w przedziale symulacji.

Symulacja zdarzeniowa

Zauważmy, że sposób, w jaki zdefiniowaliśmy funkcje f i g, stwierdza że wartość n użyta w obliczeniach jest wartością n dla czasu t, mimo że czas obliczeń wynosi już $t + 1$. Prowadzi to do dalszego udoskonalenia proces symulacji. W prawdziwym sprzęcie (sieci SNN) zmienne stanu (rejestry lub zatrzaski, synapsy) nie są aktualizowane natychmiast. Oznacza to, że obliczenie, nawet jeśli jest to proste przypisanie, zajmuje pewną niezerową ilość czasu.

```
do (i=l, endtime)
    t = t + 1
    // evaluate the functions
    t_m=m+n
    t_ n = new input
    // update the variable values
    m=t_m
    x= t_x
    n= t_n
enddo
```

Mamy teraz typową pętlę przetwarzania zdarzeń w cyfrowej symulacji logicznej:

- przesuń zegar,
- oblicz wszystkie funkcje logiczne i
- aktualizuj wartości zmiennych.

Chociaż nie uwzlędniliśmy rzeczywistych opóźnień czasowych związanych z obliczeniami, wyabstrahowaliśmy je w zachowaniu, które uwzględnia ich efekt.

Symulacja zdarzeniowa

Zauważmy, że zawiera to abstrakcję fizycznego zachowania w czasie, ponieważ przenosimy czas z jednej dyskretnej chwili do drugiej.
Zakładamy, że:

- nic ciekawego się nie dzieje między tymi dwiema chwilami.
- m, x i n nie zmieniają swoich wartości natychmiast, ale zmieniają się przed następną chwilą czasu ($t + 1$).

Zauważmy, że wszystkie aktualizują się równocześnie po zakończeniu wsyztskich obliczeń
Nazywa się to zwykle „cyklem delta” δ; co oznacza, że jest to zbiór zdarzeń, które mają miejsce w czasie t, ale po wszystkich operacjach obliczeniowych.
Przypisania nowych wartości mają miejsce w czasie $t + \delta t$.

Symulacja zdarzeniowa

Kolejna komplikacja pojawia się, gdy do obliczeń trzeba wprowadzić opóźnienia czasowe. Oznacza to, że obliczana funkcja może wyglądać tak:

$$f(m, n, t) = h(m_{t-k}, n_{t-k})$$

Oznacza to, że nowa wartość nie jest aktualizowana do zmiennej stanu aż do k jednostek czasu po obliczeniach.

```
do (i=l, endtime)
    t=t+l
    // evaluate the functions
    t_m(t+k) = m + n
    t_n = new input
    // update the variable values
    m= t_m(t)
    n=t_n
enddo
```

Symulacja zdarzeniowa

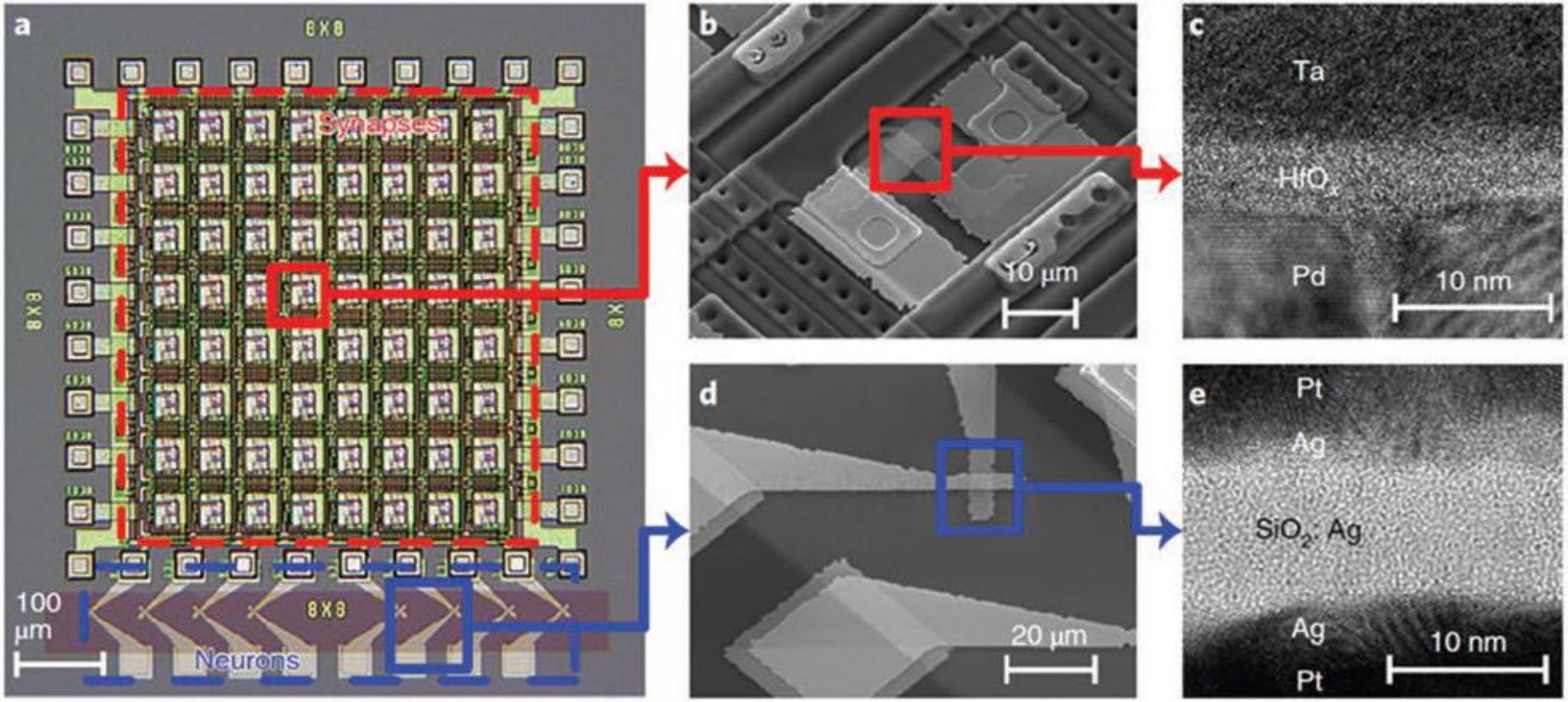
Następnie zastąpmy tablicę *t_m()* połączoną listą, która jest uporządkowana według indeksu *t*. Aby to zadziałało, opóźnienie jest zwykle włączane do obliczenia nowej wartości, a nowa wartość jest umieszczana na *liście przyszłych aktualizacji* w czasie $t + k$. *Lista przyszłych aktualizacji* jest analogiczna do *listy przyszłych zdarzeń* i w rzeczywistości nie musi być od niej oddzielona.

```
do (i=l, endtime)
    t = t + 1
    // evaluate the functions
    t_m=m+n
    put_on_future_update_list(t_m, t+k)
    t_n = new input
    // update the variable values
    m = take_off_future_update_list(t)
    n= t_n
enddo
```

Symulacja zdarzeniowa

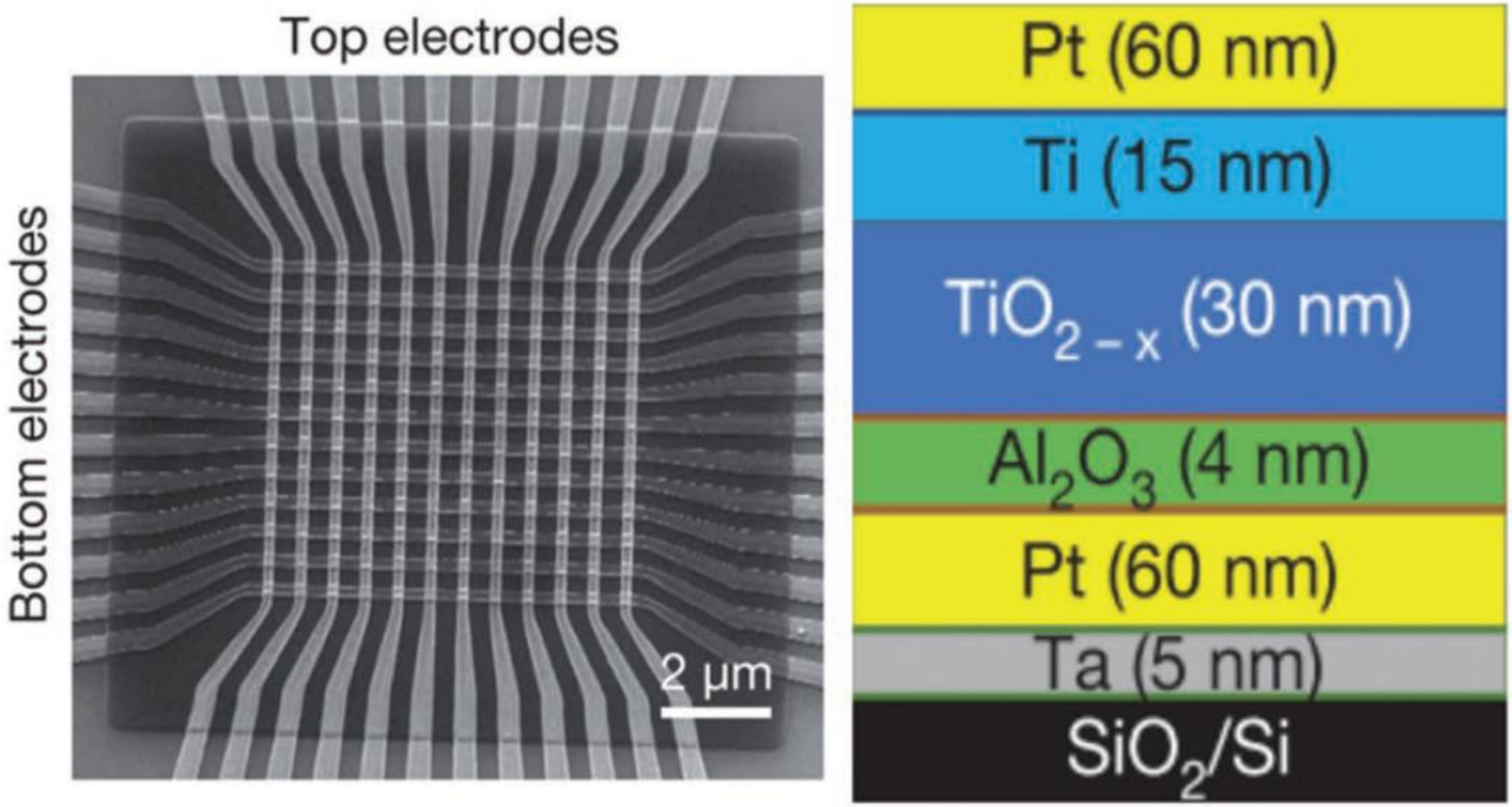
W powyższym prxzykładzie założyliśmy, że *lista przyszłych aktualizacji* zawiera wartości tylko dla zmiennej m. Ogólnie może istnieć wiele różnych zmiennych stanu, których nowe wartości mogą zostać zapisane na liście przyszłych aktualizacji, więc każdy wpis na liście musi identyfikować zmienną stanu, z którą powiązana jest wartość.

Zauważmy również, że nowa wartość jest zawsze na początku listy, jeśli lista jest uporządkowana według rosnącego czasu t. Oznacza to, że w danym momencie t na liście nie ma wartości z czasem $<t$ (byłyby to wartości z przeszłości), więc wszystkie aktualizacje zmiennych można znaleźć na początku listy. Oczywiście na liście może nie być żadnych wartości związanych z czasem t, to znaczy pierwszy wpis na liście może mieć czas $> t$, w którym to przypadku żadna zmienna nie byłaby aktualizowana w czasie t.

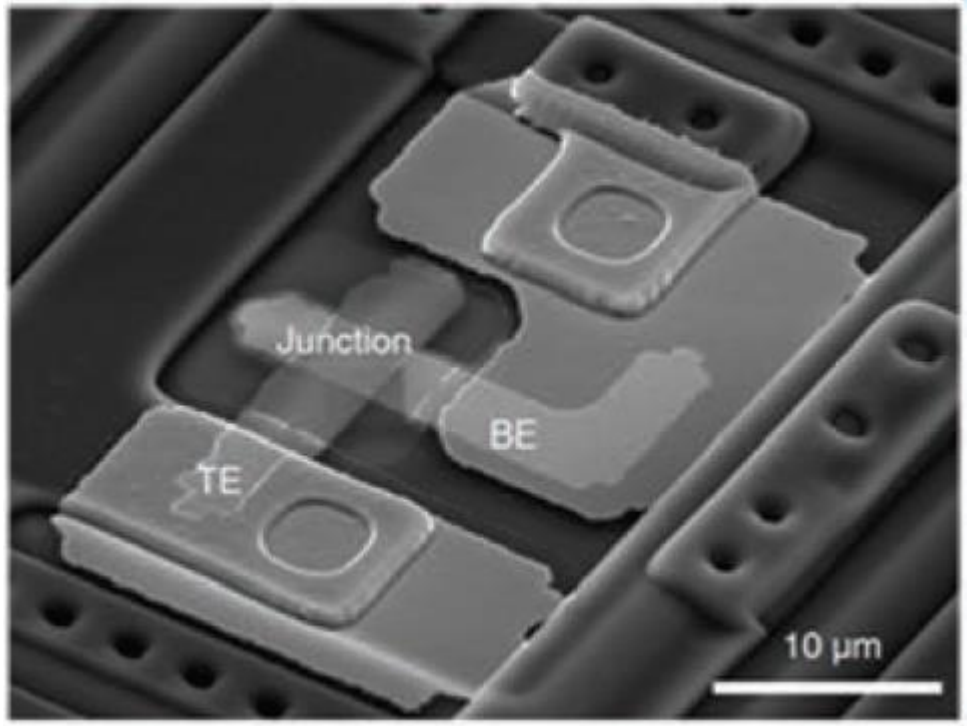


Tablica krzyżowa 8 × 8 synaps memrystorowych Pt/HfO_x/Ta połączonych z dyfuzyjnymi neuronami memrystorowymi Pt/Ag/SiO_x:Ag/Ag/Pt do uczenia nienadzorowanego.

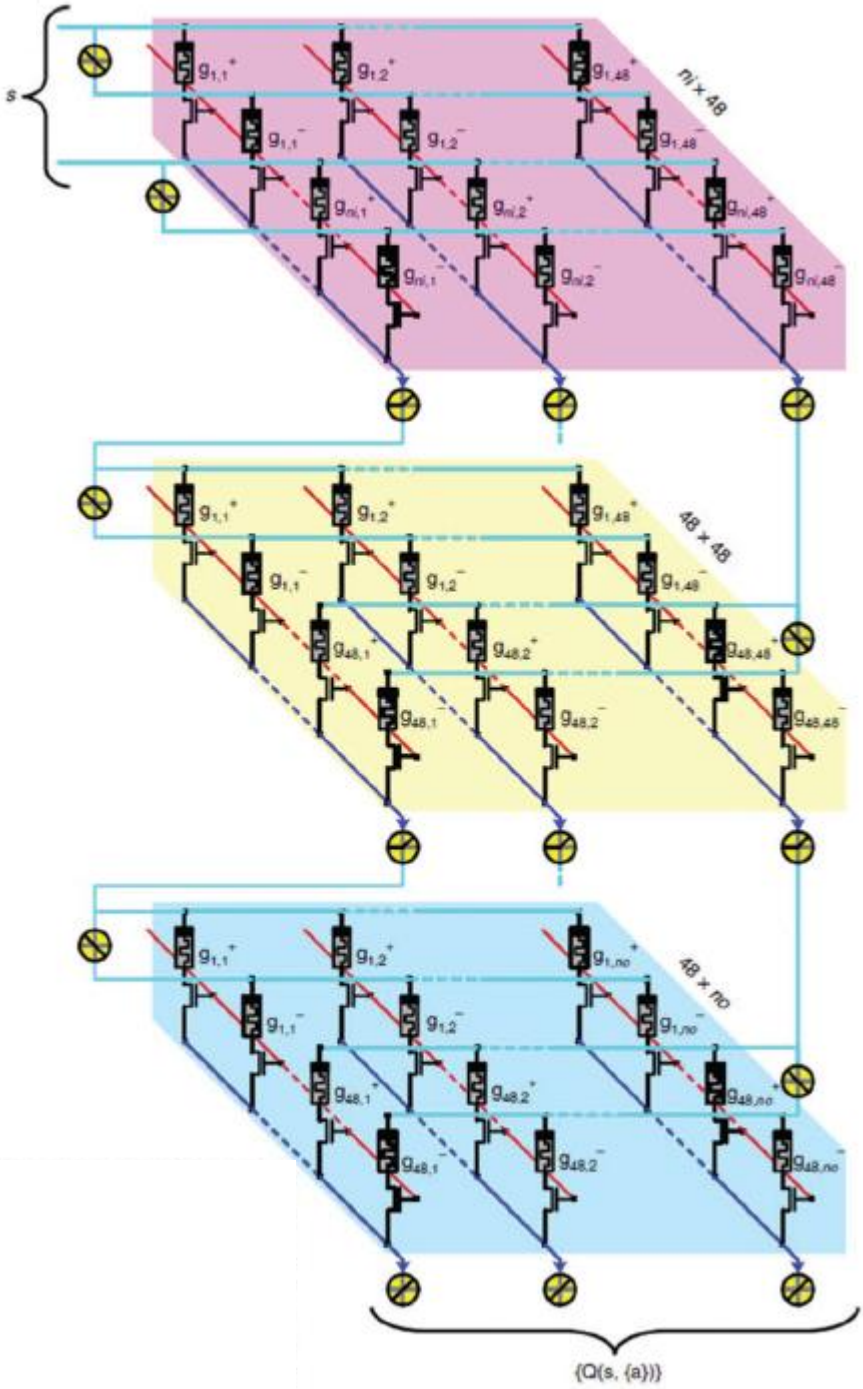
Układ krzyżowy 12 × 12 memrystorów $Pt/Al_2O_3/TiO_{2-x}/Ti/Pt$ do nadzorowanego uczenia się.



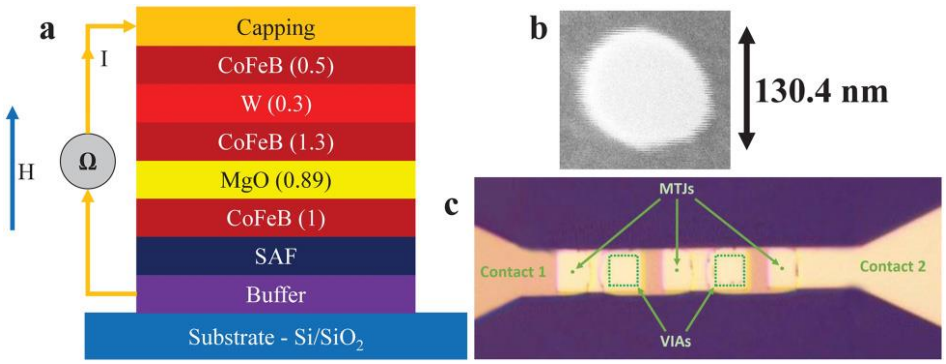
Trójwarstwowa, w pełni połączona sieć Q do uczenia się ze wzmocnieniem za pomocą memrystorów.



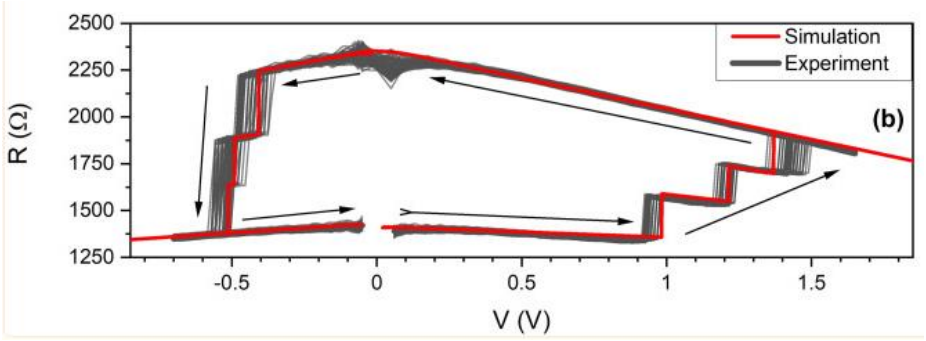
Skaningowa mikroskopia elektronowa memrystycznej synapsy $Pd/HfO_2/Ta$.



Badania na AGH

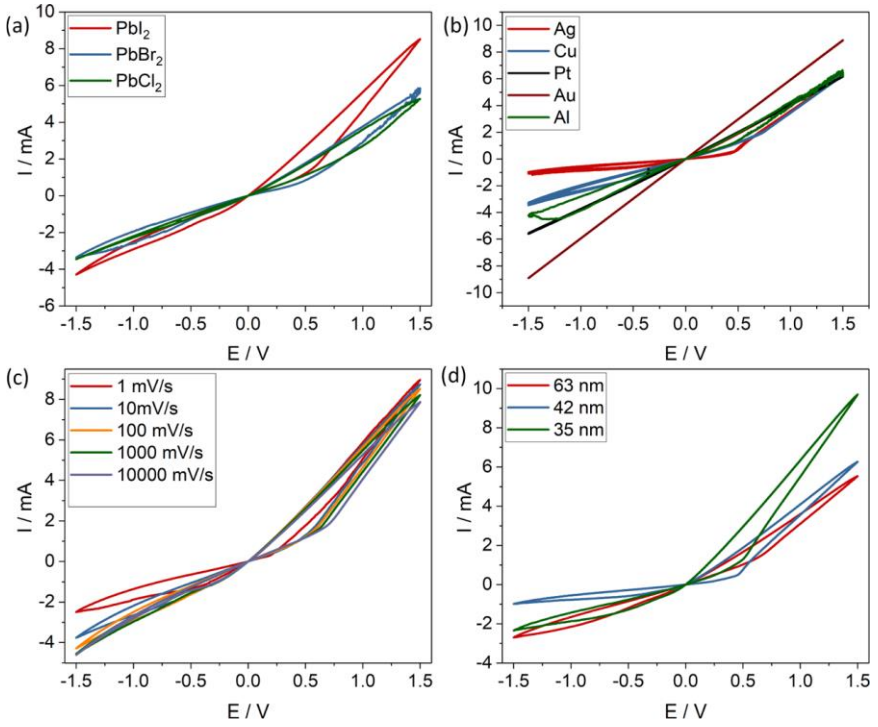


P. Rzeszut1,a), W. Skowroński1, S. Ziętek1, J. Wrona, T. Stobiecki Multi-bit MRAM storage cells utilizing serially connected perpendicular magnetic tunnel junctions featured Journal of Applied Physics 125, 223907 (2019); https://doi.org/10.1063/1.5097748,



Rzeszut P, Chęciński J, Brzozowski I, Ziętek S, Skowroński W, **Stobiecki** T. Multi-state MRAM cells for hardware neuromorphic computing. Sci Rep. 2022 May 3;12(1):7178. doi: 10.1038/s41598-022-11199-4. PMID: 35504980; PMCID: PMC9065142.

Charakterystyki prądowo-napięciowe różnych memrystorów halogenkowych ołowiowych(II) PbI_2 mierzonych drugą elektrodą Ag i szybkością skanowania 100 mV/s



Wlaźlak E, Marzec M, Zawal P, **Szaciłowski** K. Memristor in a Reservoir System-Experimental Evidence for High-Level Computing and Neuromorphic Behavior of PbI_2. ACS Appl Mater Interfaces. 2019 May 8;11(18):17009-17018. doi: 10.1021/acsami.9b01841. Epub 2019 Apr 24. PMID: 30986023.