Projet de fin d'études: Implémentation du modèle Izhikevich sur FPGA

Saulquin Aurélie

Université de Lille

24 Février 2022

Études du modèle

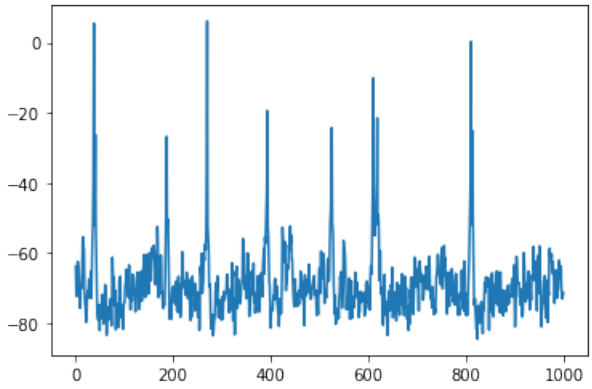
$v' = 0{,}04v^2 + 5v + 140 - u + I$
$u' = a(bv-u)$

if v=30mv
then v=c, u=u+d

↓

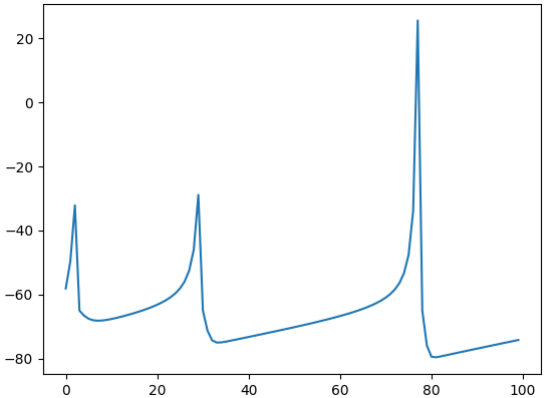
if v(n)=30mv
then v(n)=c, u(n)=u(n)+d

$v(n+1) = v(n) + 0{,}04v(n)^2 + 5v(n) + 140 - u + I$
$u(n+1) = u(n) + a(bv(n+1)-u(n))$



Études du modèle

1. Retranscription en Python du fichier matlab.
2. Réécriture du fichier python pour supprimer l'aléatoire.



Premières version des modèles

Écriture de plusieurs modèle VHDL.
float32 et le package math_real ne synthétise pas

```
entity model_4 is
    Port ( Vin : in float32;
           Uin : in float32;
           Iin : in float32;
           a : in float32;
           b : in float32;
           c : in float32;
           d : in float32;
           clk : in std_logic;
           Vout : out float32;
           Uout : out float32;
           Iout : out float32 );
end model_4;
```

Modèles synthétisable

- Format fixed point signé sur 32 bits : 16.16.
- Utilisation des process.
- Vin et Uin sont intériorisé dans le modèle.
- Utilisation des ressources élever (2 neurones implémentable).

```
entity model_sync_2 is
Port (      Iin : in std_logic_vector(31 downto 0);
            a : in std_logic_vector(31 downto 0);
            b : in std_logic_vector(31 downto 0);
            c : in std_logic_vector(31 downto 0);
            d : in std_logic_vector(31 downto 0);
            clk : in std_logic;
            Vout : out std_logic_vector(31 downto 0);
            Uout : out std_logic_vector(31 downto 0) );
end model_sync_2;
```

Modèles moins gourmand

- Réduction des DSP en utilisant le std_logic_vector plutôt que le *signed*.
- Développement de 3 modèles différents utilisable.

Modèle proposé : model_sync_4

Calcul de v et de u, détection du dépassement de seuil et re-calcul avec les valeurs resets.

$v(n+1) = v(n) + dt.(0{,}04.v(n)^2 + 5.v(n) + 140 - u(n) + I)$
$u(n+1) = u(n) + dt.a.(b.v(n+1) - u(n))$

Si $v(n+1) >= 30$:
$u\_reset = u(n+1) + d$
$v(n+1) = c + dt.(0{,}04.c^2 + 5.c + 140 - u\_reset + I)$
$u(n+1) = u\_reset + dt.a.(b.v(n+1) - u\_reset)$
fin si.

Modèle proposé : model_sync_5

Calcul de v et de u, détection du dépassement de seuil et reset simple des valeurs.

$v(n+1) = v(n) + dt.(0{,}04.v(n)^2 + 5.v(n) + 140 - u(n) + I)$
$u(n+1) = u(n) + dt.a.(b.v(n+1) - u(n))$

Si $v(n+1) >= 30$:
$v(n+1) = c$
$u(n+1) = u(n+1) + d$
fin si

Modèle proposé : model_sync_6

Détection du dépassement de seuil, reset si nécessaire puis calcul.

Si v(n) >= 30 :
v(n) = c
u(n) = u(n)+d
fin si

$v(n+1) = v(n) + dt(0{,}04.v(n)^2 + 5.v(n) + 140 - u(n) + I)$
$u(n+1) = u(n) + dt.a(b.v(n+1)-u(n))$

Comparaison des modèles : précision

précision des modèle 32 bits		|v moyen|	|u moyen|	|v median|	|u median|
model_sync_4	dt=1	2,4679	0,2472	1,2462	0,3512
	dt=0,5	0,2971	0,0217	0,3757	0,2083
	dt=0,25	0,0921	0,02474	0,7962	0,2372
model_sync_5	dt=1	2,7646	0,1739	1,5296	0,5088
	dt=0,5	0,9471	0,05849	0,1286	0,0385
	dt=0,25	1,5274	0,1271	0,6804	0,1972
model_sync_6	dt=1	11,6064	0,5092	1,5296	0,4628
	dt=0,5	3,7592	0,3941	0,2942	0,04515
	dt=0,25	1,5521	0,1269	0,6987	0,1974
min		0,0921	0,0217	0,1286	0,0385

précision des modèle 64 bits		|v moyen|	|u moyen|	|v median|	|u median|
model_sync_4	dt=1	2,7475	0,1497	1,2243	0,3613
	dt=0,5	0,7017	0,1585	0,7979	0,2047
	dt=0,25	0,1659	0,0648	0,9542	0,2316
model_sync_5	dt=1	9,9248	0,0939	1,578	0,5035
	dt=0,5	0,1035	0,0282	0,5184	0,1241
	dt=0,25	0,0423	0,7978	0,8084	0,1918
model_sync_6	dt=1	13,3796	0,4139	1,4755	0,4727
	dt=0,5	4,5931	0,3629	0,5335	0,1241
	dt=0,25	2,5404	0,1578	0,8312	0,1917
min		0,0423	0,0282	0,5184	0,1241

Comparaison des modèles ; précision

```
0
[array([-58.105])]

1
[array([-58.105]), array([-49.67024344])]

2
[array([-58.105]), array([-49.67024344]), array([-32.14843692])]

3
[array([-58.105]), array([-49.67024344]), array([-32.14843692]), array([46.97514719])]

4
[array([-58.105]), array([-49.67024344]), array([-32.14843692]), array([-65.]), array([-66.56464784])]
```

Comparaison des modèles : précision

précision des modèle 32 bits		|v moyen|	|u moyen|	|v median|	|u median|
model_sync_4	dt=1	2,4679	0,2472	1,2462	0,3512
	dt=0,5	0,2971	0,0217	0,3757	0,2083
	dt=0,25	0,0921	0,02474	0,7962	0,2372
model_sync_5	dt=1	2,7646	0,1739	1,5296	0,5088
	dt=0,5	0,9471	0,05849	0,1286	0,0385
	dt=0,25	1,5274	0,1271	0,6804	0,1972
model_sync_6	dt=1	6,8719	0,18892	1,7652	0,4629
	dt=0,5	0,9753	0,0741	0,2942	0,04515
	dt=0,25	3,1824	0,1931	0,6987	0,1837
min		0,0921	0,0217	0,1286	0,0385

précision des modèle 64 bits		|v moyen|	|u moyen|	|v median|	|u median|
model_sync_4	dt=1	2,7475	0,1497	1,2243	0,3613
	dt=0,5	0,7017	0,1585	0,7979	0,2047
	dt=0,25	0,1659	0,0648	0,9542	0,2316
model_sync_5	dt=1	9,9248	0,0939	1,578	0,5035
	dt=0,5	0,1035	0,0282	0,5184	0,1241
	dt=0,25	0,0423	0,7978	0,8084	0,1918
model_sync_6	dt=1	8,6449	0,939	1,7076	0,4727
	dt=0,5	0,1413	0,0429	0,5306	0,1194
	dt=0,25	2,194	0,1621	0,8312	0,1779
min		0,0423	0,0282	0,5184	0,1194

Comparaison des modèles : puissance

puissance des modèle 32 bits		basys 3	nexus 4 DDR
model_sync_4	dt=1	0,171	0,176
	dt=0,5	0,173	0,197
	dt=0,25	0,174	0,196
model_sync_5	dt=1	0,143	0,17
	dt=0,5	0,144	0,17
	dt=0,25	0,144	0,169
model_sync_6	dt=1	0,152	0,176
	dt=0,5	0,15	0,176
	dt=0,25	0,15	0,176
min		0,143	0,169

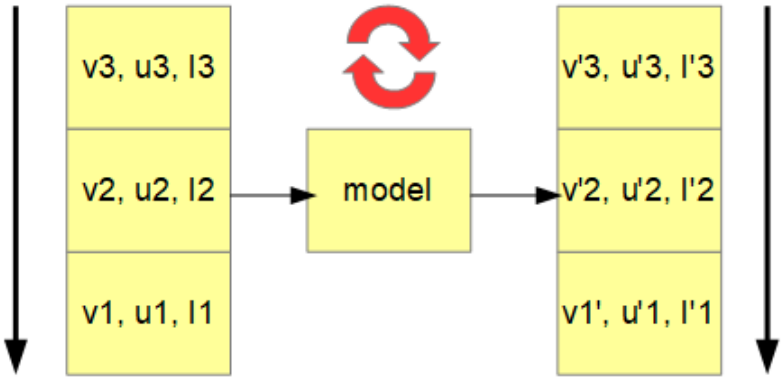
puissance des modèle 64 bits		basys 3	nexus 4 DDR
model_sync_4	dt=1	0,231	0,256
	dt=0,5	0,243	0,257
	dt=0,25	0,243	0,259
model_sync_5	dt=1	0,185	0,212
	dt=0,5	0,187	0,209
	dt=0,25	0,188	0,21
model_sync_6	dt=1	0,195	0,219
	dt=0,5	0,194	0,216
	dt=0,25	0,194	0,218
min		0,185	0,209

Comparaison des modèles : nombre de neurone

nombre de neurones 32 bits	basys 3	nexus 4 DDR
model_sync_4	11	30
model_sync_5	17	48
model_sync_6	22	59
max	22	59

nombre de neurones 64 bits	basys 3	nexus 4 DDR
model_sync_4	1	3
model_sync_5	2	6
model_sync_6	2	6
max	2	6

Comparaison des modèles : nombre de neurones



Comparaison des modèles : conclusion

- Précision 32 bits : model_sync_4 ou model_sync_5
- Précision 64 bits : model_sync_5
- Consommation : model_sync_5
- Nombre de neurones : model_sync_6
- Réaliste : model_sync_6

Démonstration sur carte