ÍNDEX MEMÒRIA

RESUM

Els generadors de patrons centrals (*Central Pattern Generators* en anglés) són un tipus de xarxa neuronal capaç de produir diferents patrons rítmics, modulables en funció d'un nombre reduït de paràmetres. Aquest projecte tracta la implementació d'un generador de patrons centrals fent servir una *FPGA*, per tal de controlar un robot hexàpode creat a la *EUETIB* com a resultat d'un treball col·laboratiu de diferents TFGs.

Un cop realitzat el sistema, s'aconsegueix produir 3 patrons de moviment hexàpode diferents, reproduint satisfactòriament els resultats de la bilbiolgrafia consultada.

RESUMEN

Los generadores de patrones centrales (*Central Pattern Generators* en inglés) son un tipo de red neuronal capaz de producir diferentes patrones rítmicos, modulables en función de un número reducido de parámetros. Este proyecto trata la implementación de un generador de patrones centrales usando una *FPGA*, con el objetivo de controlar un robot hexápodo creado en la *EUETIB* como resultado de un trabajo colaborativo de distintos TFGs.

Una vez realizado el sistema, se consigue producir 3 patrones de movimiento hexápodo diferentes, reproduciendo satisfactoriamente los resultados de la bibliografía consultada.

ABSTRACT

Central pattern generators are neuronal networks that are able to produce different rythmic patterns, which are modulable by a small number of parameters. This project is about the implementation of a central pattern generator using a *FPGA*, in order to control an hexapod robot made in the *EUETIB* by different collaborative *TFGs*.

Once the system is made, 3 different hexapod movement patterns are obtained, following the results obtained in the bibliography.

AGRAÏMENTS

Al Jordi Cosp, el meu tutor, per estar disponible sempre que he tingut dubtes o necessitava ajuda.

Al Javier Carpio, per implementar la part de comunicació dels *Arduinos* i mantenir-me informat de les modificacions que ha efectuat al robot.

CAPÍTOL 1: INTRODUCCIÓ

1.1. Introducció

Darrerament ha sorgit una tendència en el camp de la robòtica que intenta imitar els sistemes de locomoció biològics presents als éssers vius, degut a la gran flexibilitat que aquests presenten en superfícies irregulars i/o amb obstacles. El problema d'això és que requereix el disseny de sistemes de control relativament complexes. Així doncs, han sorgit diferents enfocaments per tal de dissenyar aquests sistemes, ja sigui mitjançant màquines d'estats finits, generadors de sinus, trajectòries de referència prèviament gravades o sistemes heurístics.

Un altre enfocament relativament nou en el camp és el de fer servir els sistemes basats en els que s'ha descobert que governen la locomoció animal: els CPGs o *Central Pattern Generators.* Aquests sistemes són xarxes neuronals bioinspirades que basen el seu funcionament en oscil·ladors interconnectats amb diferents paràmetres de control d'entrada, i permeten minimitzar el nombre de connexions entre el sistema central de decisió i el de locomoció, a més de no dependre de realimentacions per tal de poder efectuar una locomoció adequada.

En aquest treball es tracta el disseny d'un CPG per tal de controlar un robot hexàpode. Aquest disseny s'implementarà en una FPGA amb l'objectiu de poder realitzar diferents tipus de locomocions.

El robot hexàpode amb el que es treballarà és un robot dissenyat anteriorment per altres estudiants com a projecte de fi de grau, i amb el qual han treballat posteriorment més estudiants. Així doncs, l'enfocament cap al robot és el de construir una plataforma sobre la qual diferents estudiants puguin treballar d'una manera incremental, és a dir, on futurs estudiants puguin aprofitar el treball realitzat per altres estudiants com a base. Per tant, amb aquesta finalitat resulta d'especial importància la correcta realització dels documents pertinents (memòria, documents tècnics, etc.) per tal de facilitar el treball a futurs estudiants.

1.2. Abast del treball i objectius

L'objectiu principal d'aquest projecte és la implementació d'un generador de patrons centrals en una *FPGA,* fent servir el llenguatge de disseny *VHDL,* per tal de produir un o més sistemes de locomoció per controlar el robot hexàpode mencionat. Per tal d'aconseguir-ho es farà servir les equacions i paràmetres provinents de diferents articles mencionats a la bibliografia. La deducció dels paràmetres ideals per al sistema requereix l'ús de mètodes complexes, degut a que hi intervenen sistemes també complexes. Per tant, es comentarà només per sobre un dels possibles mètodes d'obtenció, però no s'utilitzaran per a obtenir els valors pertinents, ja que això podria ser l'abast d'un treball sencer.

Així doncs, els valors necessaris s'obtindran directament d'aquests articles, i aquest treball es centrarà en la implementació i adaptació per al cas concret del robot.

El pes del treball es divideix de la següent manera:

- Estudi de l'estat de l'art.
- Simulació de les equacions i sistemes d'equacions mencionats als articles, observant l'efecte dels diferents paràmetres implicats sobre aquests. Aquesta part es realitzarà fent servir el programari *Matlab*.
- Realització i simulació d'un disseny *VHDL* que implementi els sistemes estudiats i els adapti al robot.
- Connexió o adaptació de la *FPGA* al robot.
- Realització de proves amb el robot i la *FPGA* i ajustament dels paràmetres necessaris per tal de millorar el sistema.

Les parts més importants i que requeriran més temps són les simulacions del sistemes tant en *Matlab* com en *VHDL.*

CAPÍTOL 2: CENTRAL PATTERN GENERATORS

2.1. Introducció

Els generadors centrals de patrons són un tipus de xarxa neuronal capaç de produir patrons coordinats d'activitat rítmica, sense cap entrada provinent d'una realimentació sensorial o d'un centre de control.

La presència d'aquests sistemes al món animal s'ha descobert mitjançant experiments. Un exemple n'és el de la llamprea, un tipus de peix molt primitiu la medul·la espinal del qual, un cop extreta i aïllada, és capaç de produir patrons d'activitat molt semblants als patrons de la llamprea intacta.

Tot i que no és necessària la presència d'una realimentació sensorial per a la generació d'aquests patrons, sí que hi juga un rol molt important en el seu funcionament, ja que produeix una modulació en els patrons produïts per aquestes xarxes neuronals.

Així doncs, un *central pattern generator* es podria descriure com un sistema capaç de produir patrons per ell mateix i modulable per entrades externes relativament simples. Aquest sistema permet generar diferents comportaments locomotrius complexes.

2.2. Aplicació a la robòtica i característiques

La robòtica és un camp amb tendència imitar els sistemes presents als éssers vius, especialment en l'àmbit de la locomoció. Aquests sistemes presenten certs avantatges respecte els sistemes tradicionals. Per exemple, el desplaçament mitjançant braços robòtics presenta una flexibilitat i una capacitat d'adaptació molt més elevada que el desplaçament tradicional mitjançant rodes. També presenten desavantatges però: una elevada complexitat tant en el

desenvolupament mecànic com en l'electrònic, així com una complexitat de control molt més elevada.

En quant al problema del control locomotriu d'un robot, hi han diversos enfocaments diferents, cadascun amb els seus avantatges i inconvenients. Alguns exemples són:

- Màquines d'estat finit
- Generadors sinusoïdals
- Trajectòries gravades
- Sistemes de control heurístics
- Generadors de patrons centrals

Aquests últims, objecte del treball, són interessants ja que presenten certes propietats que poden ser útils per a la resolució del problema de la locomoció d'un robot:

1. L'objectiu dels models de CPG és proporcionar un comportament de cicle límit estable, és a dir, produir un comportament on les variables del sistema es moguin sempre dins d'un marge determinat un cop establert el règim permanent. Això permet al sistema tornar al règim estable ràpidament després d'una pertorbació a les variables d'estat involucrades.
2. La seva estructura és molt apropiada per una implementació distribuïda, característica molt interessant per a certs tipus de morfologies com els robots serp o robots reconfigurables.
3. Normalment els models de CPG tenen un nombre reduït de paràmetres de control. Aquest paràmetres són els que modulen el seu comportament, i permeten distribuir el sistema de manera que un sistema de decisió de més alt nivell s'encarregui de modificar aquests paràmetres quan sigui necessari, mentre el CPG s'encarrega de fer els càlculs pertinents, que són relativament costosos computacionalment. El fet de requerir pocs paràmetres de control també facilita la comunicació entre el CPG i el sistema de presa de decisions.
4. Les equacions diferencials típicament involucrades actuen com a filtres, de manera que una variació abrupta als paràmetres de control no implica una variació abrupta a l'estat del sistema. Això és especialment important ja que els moviments abruptes poden provocar un desgast més pronunciat als components mecànics (motors, reductors, etc.), i en última instància provocar problemes al robot.
5. La plasticitat de les connexions i senyals de realimentació pot permetre una gran flexibilitat per a la locomoció en entorns desconeguts.

No obstant, aquests sistemes presenten unes dificultats a causa de les quals actualment encara estan en fase de recerca.

En primer lloc, actualment no existeix cap metodologia establerta que permeti dissenyar un CPG de manera eficaç per tal de resoldre un problema de locomoció determinat. Això vol dir que diferents dissenyadors probablement faran servir diferents processos per tal d'arribar a una solució, que probablement no serà la mateixa.

En segon lloc, actualment no hi ha cap base teòrica per descriure CPGs que permeti analitzar i predir el seu comportament. Per exemple resulta molt difícil demostrar la estabilitat del sistema sencer CPG-robot.

En el procés de construcció d'un model CPG hi han certs punts a definir:

1. L'arquitectura general del model, és a dir, el tipus i nombre d'oscil·ladors presents.
2. El tipus i topologia de les connexions entre els diferents oscil·ladors. Aquestes condicions determinaran tant la sincronització dels oscil·ladors com el tipus de locomoció resultant, és a dir, les diferències de fase entre aquests.
3. Les formes d'ona que determinaran les trajectòries realitzades per cada articulació del robot.
4. L'efecte dels diferents paràmetres de control sobre la modulació resultant. És important determinar com aquests paràmetres afecten els valors com ara la freqüència, fase entre oscil·ladors o amplitud de l'ona.
5. L'efecte dels senyals de realimentació, en cas que n'hi hagi, sobre el CPG del robot.

La complexitat resideix en què tots aquest punts estan interconnectats entre ells, és a dir, tots tenen efecte, ja sigui en major o menor grau, sobre els altres.

2.3. Oscil·ladors acoblats com a CPGs

Els generadors de patrons centrals es poden modelar de diferents maneres en funció del nivell d'abstracció. Aquest treball es centrarà en un model abstracte que treballa amb oscil·ladors acoblats.

En aquest model, cada oscil·lador representa un element capaç de treballar independentment, al qual se li afegeixen connexions o contribucions externes per tal de provocar una certa sincronització entre els diferents elements que conformen la xarxa.

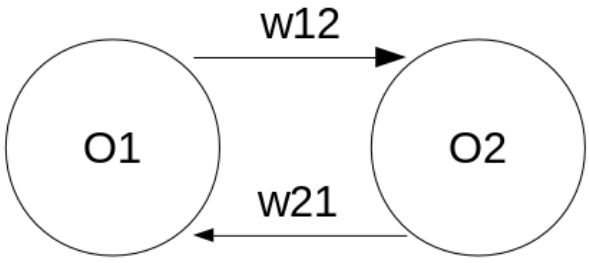

Figura 1: *Xarxa de 2 oscil·ladors.*

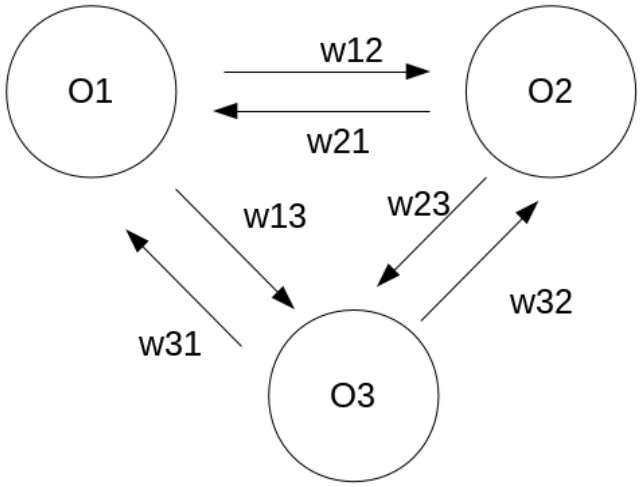

Figura 2: *Xarxa de 3 oscil·ladors.*

A les anteriors figures es poden apreciar els diferents elements que conformen una xarxa d'oscil·ladors. Per una banda, hi ha els oscil·ladors O_i , cadascun amb el seu estat intern. D'altra banda, es representen les connexions mitjançant fletxes. En aquest exemple la nomenclatura per a cada connexió és w_{ij}, sent w un valor numèric que representa la contribució de l'oscil·lador i sobre l'oscil·lador j. El valor d'aquesta connexió pot ser:

- **Positiu**, si es tracta d'una connexió que estimula la oscil·lació
- **Negatiu**, si la connexió inhibeix la oscil·lació.
- **Zero**, quan no hi ha connexió

Per tant, donada una xarxa de n oscil·ladors, els elements presents seran:

- **O_n oscil·ladors**, a cadascun del qual li corresponen un nombre de valors que representen el seu estat intern.
- **n x (n-1) connexions**, assumint que tots els oscil·ladors estan connectats entre ells (incloses les connexions nul·les). Aquestes connexions es poden representar mitjançant una matriu quadrada de $n \times n$ elements, i llavors s'inclouen tant les connexions externes com connexions autoexcitatrius. Aquests últimes tindran un valor nul en el model que es treballarà.

$$\begin{pmatrix} w_{1,1} & w_{1,1} & .. & w_{1,n} \\ w_{2,1} & w_{2,2} & .. & w_{2,n} \\ .. & .. & .. & .. \\ w_{n,1} & .. & .. & w_{n,n} \end{pmatrix}$$

Figura 3: *Format de la matriu de connexions entre oscil·ladors.*

2.4. Model dels oscil·ladors

Hi ha diferent tipus d'oscil·ladors acoblats que es poden fer servir per crear CPGs. La gran majoria d'aquests no són més que sistemes d'equacions diferencials dels quals se n'extreu una solució mitjançant integració numèrica, ja que és el més senzill computacionalment, i en molts casos les equacions ni tan sols tenen solució analítica.

En aquest treball es farà servir l'oscil·lador de *van der Pol* (abreviat *VDP*). La principal raó d'aquesta decisió és que hi ha bibliografia disponible en la qual es fa servir aquest oscil·lador amb el mateix objectiu, dissenyar un CPG. Això és d'especial importància ja que permetrà aprofitar valors de paràmetres i connexions que d'altra manera haurien sigut molt difícils de deduir, quedant fora de l'àmbit d'aquest treball.

2.4.1. L'oscil·lador de van der Pol

L'oscil·lador de *van der Pol* és un oscil·lador amb amortiment no lineal bastant utilitzat en l'àmbit de la biologia, entre d'altres. Va ser proposat per l'enginyer *Balthasar van der Pol,* fent servir vàlvules de buit, i consisteix en la següent equació diferencial:

$$\ddot{x}-\alpha\left(p^2-x^2\right)\dot{x}+\omega^2 x=0 \quad (1)$$

Es tracta, per tant, d'una equació diferencial de segon ordre. Els paràmetres que la governen són els següents:

- **α:** Afecta al grau de no-linealitat del sistema. Com més elevat sigui aquest valor, menys s'assemblarà l'ona resultant a una ona sinusoïdal.
- **p:** Afecta la amplitud de les oscil·lacions.
- **ω:** Afecta la freqüència de les oscil·lacions

Cal notar que tot i que cada paràmetre influencia principalment una característica determinada de l'oscil·lació, també té certa influencia a les altres característiques. Per exemple, tot i que el paràmetre *p* és el que té una influencia més directa a l'amplitud de l'ona, els paràmetres α i ω també en tenen certa influencia, però molt més petita.

A continuació es realitzen una sèrie de simulacions de l'equació amb el programari *Matlab*, per tal d'estudiar el seu comportament amb diferents paràmetres. Aquest programari proporciona una sèrie de funcions dedicades a la integració numèrica basant-se en mètodes com ara els de *Runge-Kutta*. Aquestes funcions tenen com a paràmetres d'entrada la funció a integrar, les condicions inicials i el marge de temps a integrar. Un aspecte a tenir en compte és que les funcions només accepten com a entrada sistemes d'equacions diferencials de primer ordre. Així doncs, per tal d'introduir-hi l'equació de segon ordre de l'oscil·lador, cal convertir-la en un sistema d'equacions de primer ordre:

$$\begin{cases} y = \dot{x} \\ \dot{y} = \alpha(p^2 - x^2)y - \omega^2 x \end{cases} \quad (2)$$

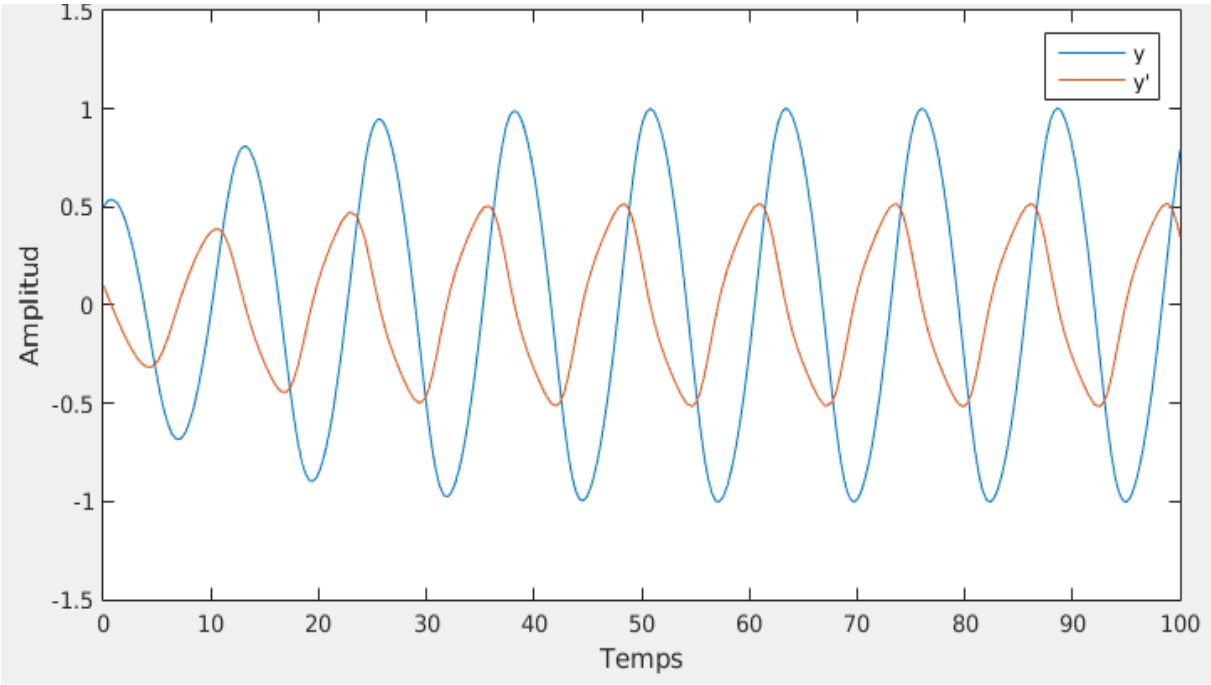

Figura 4: *Solució de l'oscil·lador de van der Pol per a α = p = ω = 0.5. Magnituds adimensionals.*
y(0) = 0,1
y'(0) = 0,5

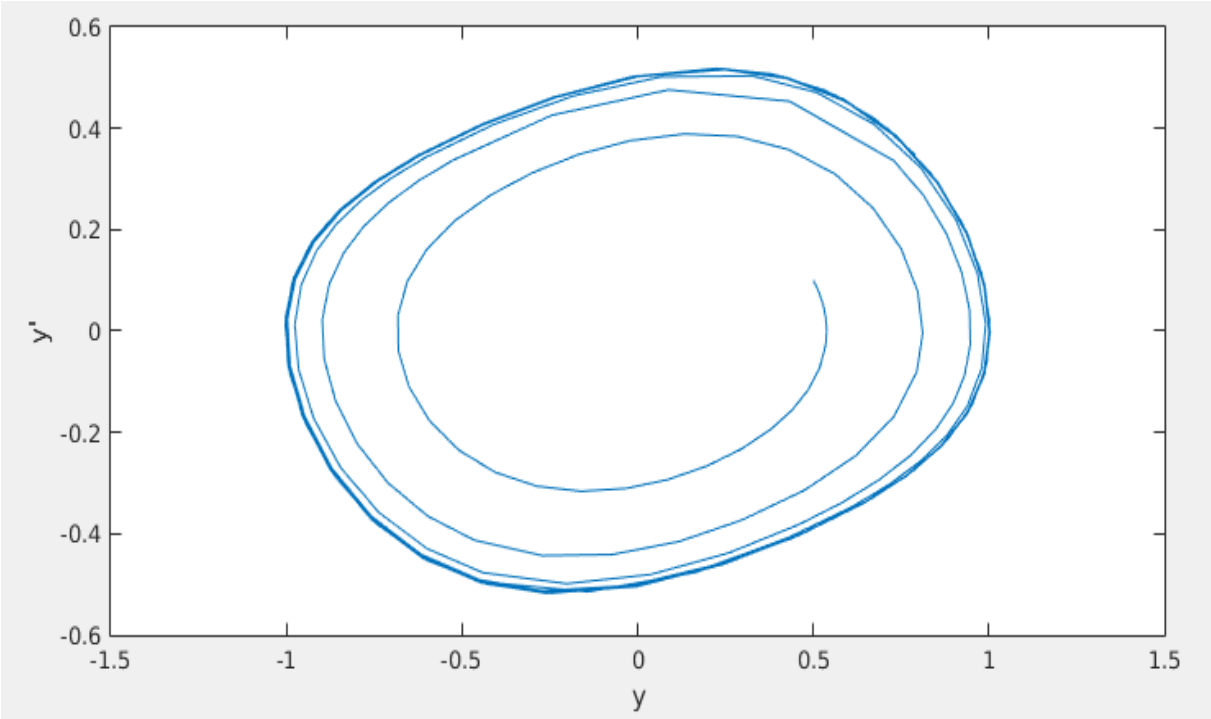

Figura 5: *Diagrama de fase de la solució obtenida a la figura anterior. A l'eix horitzontal hi ha el valor de y, mentre que a l'eix vertical s'hi troben els valors de y'. Magnituds adimensionals.*

Les dues simulacions han estat realitzades mitjançant la funció *ode45()* de *Matlab. Es* pot apreciar que s'arriba a un règim permanent estable.

Realitzades diverses simulacions amb diferents condicions inicials es comprova que el sistema sempre sempre oscil·la, un cop establert el règim permanent, de la mateixa manera.

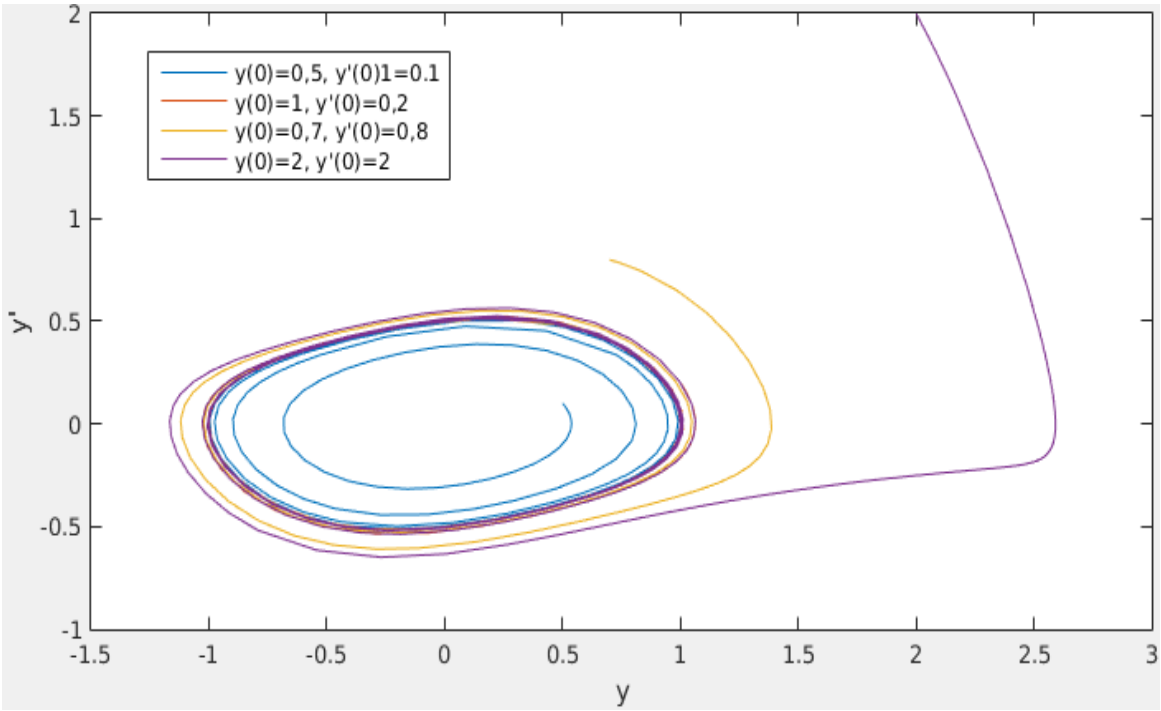

Figura 6: *Diagrama de fase de l'oscil·lador amb els paràmetres anteriors i diferents condicions inicials. Magnituds adimensionals. Les condicions inicials estan indicades a la llegenda.*

Seguidament es simula el comportament del sistema davant d'un canvi brusc dels paràmetres. S'observa la característica comentada anteriorment: l'equació de segon ordre actua com a filtre suavitzant la transició entre les dues etapes.

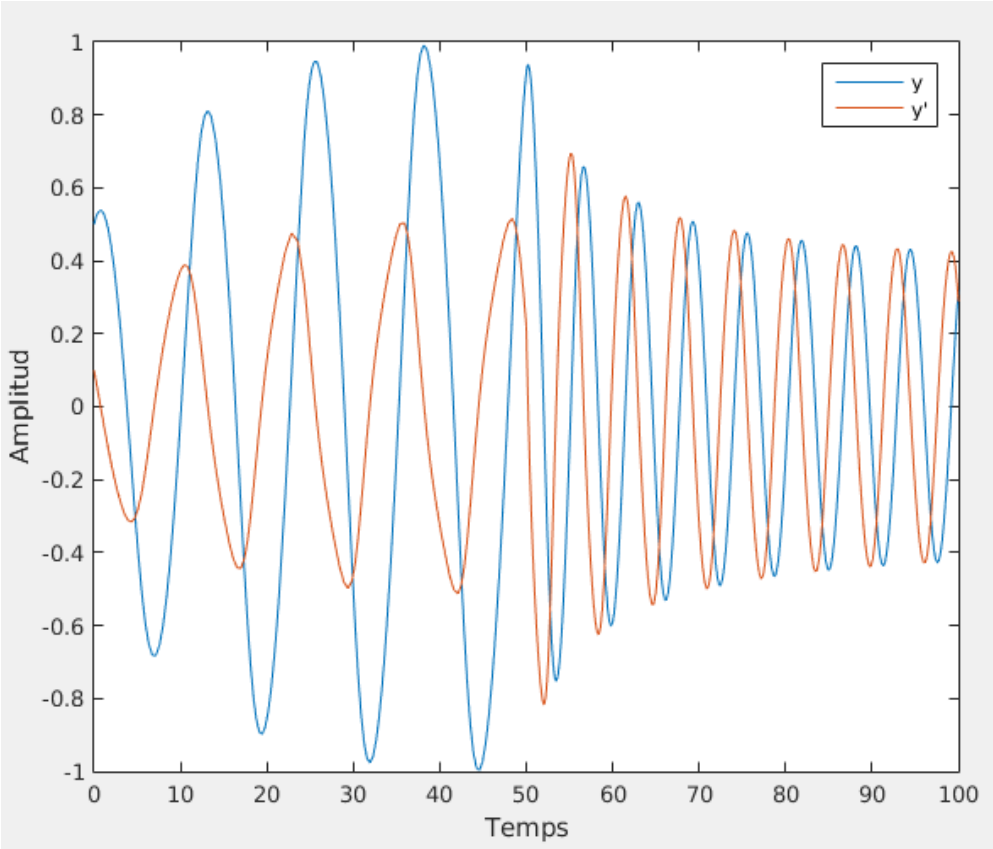

Figura 7: *Evolució temporal de l'oscil·lador amb un canvi brusc de paràmetres. Magnituds adimensionals.*
y(0) = 0,5
y'(0) = 0,1
Interval [0,50] → α = p = ω = 0.5
Interval [50,100] → α = ω = 1, p = 0,2

Els paràmetres de la primera etapa són α = p = ω = 0.5, mentre que a la segona etapa els valors són α = ω = 1, i p = 0.2. Això provoca una disminució de l'amplitud i un augment de la freqüència.

2.4.2. L'oscil·lador de van der Pol connectat en xarxa

El model exposat a l'apartat anterior representa un únic oscil·lador de van der Pol independent. Per tal de permetre-hi contribucions externes, per realitzar la xarxa neuronal, cal realitzar una lleugera modificació:

$$\ddot{x}-\alpha\left(p^2-x^2\right)\dot{x}+\omega^2 x=0 \qquad (1)$$

$$\ddot{x}_i-\alpha\left(p^2-x_{ai}^2\right)\dot{x}_i+\omega^2 x_{ai}=0 \qquad (2)$$

L'equació (1) és l'equació original mentre que la (2) és la versió que es farà servir per realitzar el *CPG*. El subíndex *i* representa el nombre de l'oscil·lador en qüestió, entre 1 i *n* (on *n* és el nombre d'oscil·ladors a la xarxa). El valor x_{ai} representa la suma de contribucions cap a un oscil·lador determinat:

$$x_{ai}=x_i+\sum w_{ii}\,x_i \qquad (3)$$

A l'equació (3) s'hi aprecien dues contribucions diferenciades: per una banda el valor intern del propi l'oscil·lador, x_i, i per altra banda la suma de totes les contribucions externes pels altres oscil·ladors. Cada contribució externa és simplement el valor de l'estat de l'oscil·lador, x_j, multiplicat per un factor de connexió. El valor de w_{ij} prové de la matriu definida anteriorment.

Cal notar que, degut a que la contribució interna del propi l'oscil·lador *i* ja queda representada en el terme x_i a l'equació (3), a la matriu de connexions definida anteriorment s'haurà de tenir això en compte i considerar els valor de la diagonal principal com a nuls:

$$\begin{pmatrix} 0 & w_{1,1} & .. & w_{1,n} \\ w_{2,1} & 0 & .. & w_{2,n} \\ .. & .. & .. & .. \\ w_{n,1} & .. & .. & 0 \end{pmatrix}$$

Figura 8: *Format d'una matriu de connexions que no considera els pesos de connexió d'un oscil·lador sobre ell mateix.*

En cas contrari s'estaria considerant el pes intern de l'oscil·lador més d'una vegada, ja que s'inclouria el valor x_i i el valor $x_i * w_{ii}$ *alhora*. Això és només una qüestió de preferència a l'hora de representar els diferents elements al sistema. Una altra possibilitat seria fer servir l'equació (4) en comptes de l'equació (3) i incloure la contribució interna dels oscil·ladors a la diagonal principal de la matriu:

$$x_{ai}=\sum_j w_{ij}\,x_j \qquad (4)$$

$$\begin{pmatrix} 1 & w_{1,1} & .. & w_{1,n} \\ w_{2,1} & 1 & .. & w_{2,n} \\ .. & .. & .. & .. \\ w_{n,1} & .. & .. & 1 \end{pmatrix}$$

Figura 9: *Format d'una matriu de connexions on es té en compte la influència del propi oscil·lador sobre ell mateix.*

Així doncs, es tenen els 3 paràmetres explicats anteriorment, *α, p* i *ω,* que defineixen un comportament que segons les equacions exposades serà comú a tots els oscil·ladors. D'altra banda la matriu de pesos o connexions definirà les relacions entre els elements que conformen la xarxa, i provocarà en la majoria de casos que les oscil·lacions de cada element (oscil·lador) siguin diferents.

Per tant, en última instància el resultat d'aquest model és un sistema de tantes d'equacions diferencials de segon ordre com oscil·ladors hi hagin:

$$\begin{aligned} \ddot{x}_1 - \alpha\left(p^2 - \left(\sum_j w_{1j}x_j\right)^2\right)\dot{x}_1 + \omega^2\left(\sum_j w_{1j}x_j\right) = 0 \\ \ddot{x}_2 - \alpha\left(p^2 - \left(\sum_j w_{2j}x_j\right)^2\right)\dot{x}_2 + \omega^2\left(\sum_j w_{2j}x_j\right) = 0 \\ \vdots \\ \ddot{x}_n - \alpha\left(p^2 - \left(\sum_j w_{nj}x_j\right)^2\right)\dot{x}_n + \omega^2\left(\sum_j w_{nj}x_j\right) = 0 \end{aligned} \qquad (5)$$

2.5. Model de la xarxa neuronal en funció del robot

El nombre d'oscil·ladors i les connexions del sistema depenen de la morfologia del robot. Típicament, cada oscil·lador controla una extremitat, mentre que les connexions entre els diferents oscil·ladors defineixen el tipus de coordinació entre les diferents extremitats.

En aquest treball es farà servir, per tant, un oscil·lador per cada pota del robot hexàpode. Llavors el sistema consistirà en $n = 6$ equacions diferencials com les de l'eq. (5).

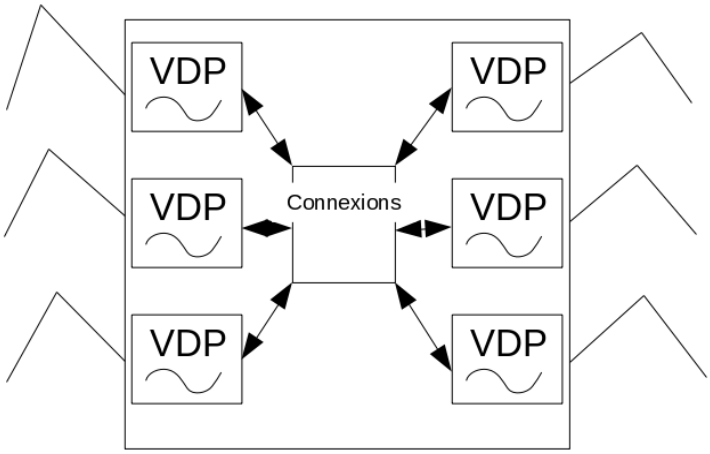

Figura 10: *Diagrama de l'estructura de la xarxa neuronal.*

2.5.1. Diferents patrons de locomoció

L'objectiu d'un generador de patrons central és, en última instància, generar un patró de moviment adequat per a la locomoció d'un sistema. Per tant cal estudiar quins tipus de locomoció són adequats en funció de la morfologia del robot. A continuació es descriuen diferents tipus de locomoció adequats per a un sistema hexàpode, en funció del desfasament d'ona entre els diferents oscil·ladors. Per tal de representar les potes del sistema es fa servir el següent tipus de diagrama, on L*n* representa la pota a la posició *n* de l'esquerra (*Left*) i R*n* la pota *n de* la dreta (*Right*).

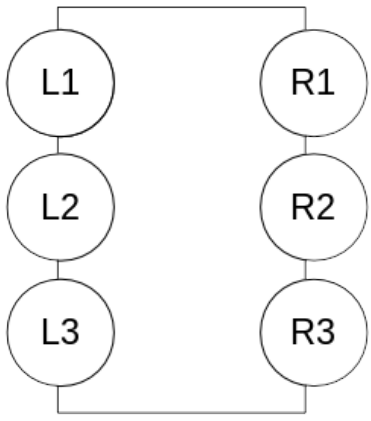

Figura 11: *Format del diagrama.*

Es consideren tres tipus de moviment:

- Moviment lent. En aquest cas, primer es mouen totes les potes d'un lateral successivament, i després totes les potes de l'altre lateral. És el sistema més estable.

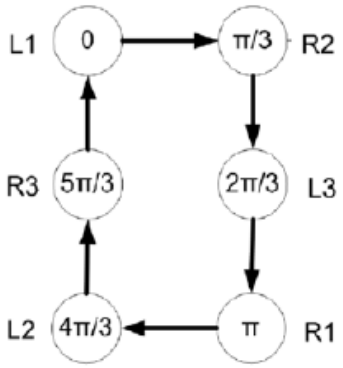

Figura 12: *Fases del moviment lent. Les potes no es mostren en ordre. Font: [2]*

- Moviment mitjà. En aquest cas hi han 3 potes tocant a terra i una o dues potes aixecant-se.

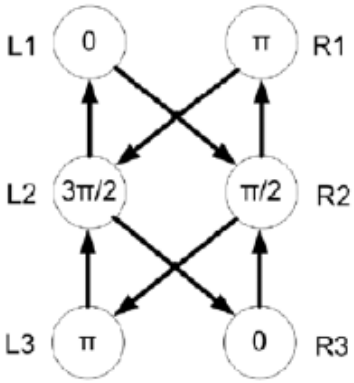

Figura 13: *Fases del moviment mitjà. Font: [2]*

- Moviment ràpid o trípode. És el moviment més típic d'un hexàpode. En aquest patró es mouen 3 potes cada vegada, mentre les 3 potes restants subjecten l'estructura formant un trípode estable. Seguidament es mouen aquestes 3 potes mentre les altres subjecten l'estructura, i així consecutivament. Es tracta per tant de dos trípodes que es van alternant per tal de subjectar l'estructura mentre es mouen les altres potes.

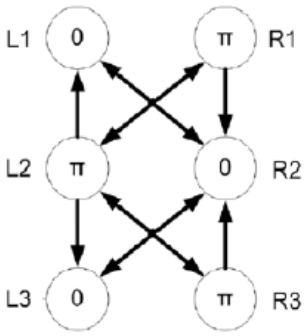

Figura 14: *Fases del moviment ràpid o trípode. Font: [2]*

2.5.2. Obtenció dels paràmetres adients per als patrons locomotrius

Un cop descrit el funcionament dels oscil·ladors i els diferents patrons que es volen aconseguir, només queda definir les connexions entre aquests. Aquesta és probablement la part més complexa del disseny d'un *CPG*, ja que com s'ha mencionat abans no existeix encara cap fundació teòrica sòlida per descriure el comportament d'una xarxa amb unes connexions determinades, ni cap metodologia per tal d'aconseguir un patró determinat.

Per aquesta raó, són molt útils els algorismes d'aprenentatge i d'optimització, els quals calculen iterativament els valors òptims per les connexions, avaluant-ne a cada iteració el comportament de la xarxa sencera i establint un set de paràmetres que estableixen com d'adequat és el resultat obtingut a cada iteració.

La realització d'aquests algorismes per obtenir els valors de les connexions queda fora de l'àmbit d'aquest projecte, i podria ser el tema per a un projecte separat sencer. Per tant, es faran servir els valors obtinguts a articles científics de la bibliografia. Seguidament es fa un resum del procediment d'obtenció.

2.5.3. Algorisme genètic per a la obtenció dels paràmetres

Els valors que es fan servir en aquest projecte provenen d'un algorisme genètic presentat a un article realitzat per Barron-Zambrano i Torres-Huitzil (2011).

En aquest article es fa servir un algorisme genètic on es calculen els paràmetres rellevants:

- Primerament es calculen els paràmetres α, p i ω per tal d'obtenir una amplitud i una freqüència desitjada. Això s'aconsegueix mitjançant un algorisme iteratiu en el qual s'avalua com és d'adequat el resultat obtingut, fent servir una expressió en funció dels paràmetres resultants de l'ona com ara l'amplitud i la freqüència. A cada iteració s'hi afegeix un cert element d'atzar, com ara la possibilitat de combinar resultats anteriors o fer servir valors aleatoris.

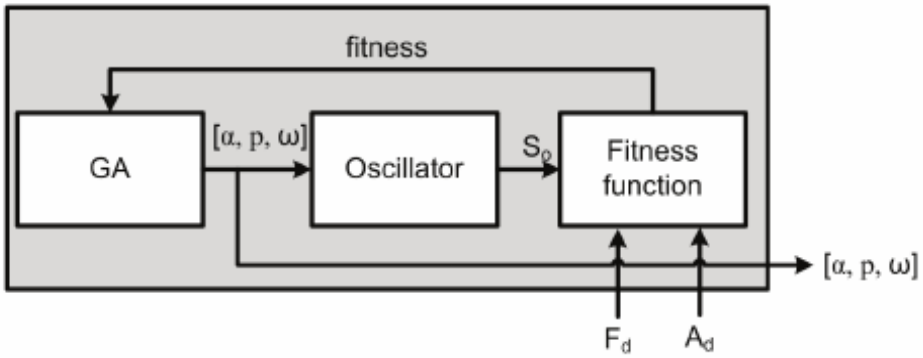

Figura 15: *Diagrama de blocs de l'algorisme fet servir a l'article per calcular α, p i ω.*
Font: [3]

- En segon lloc, es fa servir un algorisme semblant a l'anterior, però en aquest cas la variable a optimitzar és el desfasament entre els diferents oscil·ladors per tal d'aconseguir els patrons esmentats. D'aquí s'aconsegueix la matriu dels valors de les connexions pertinent.

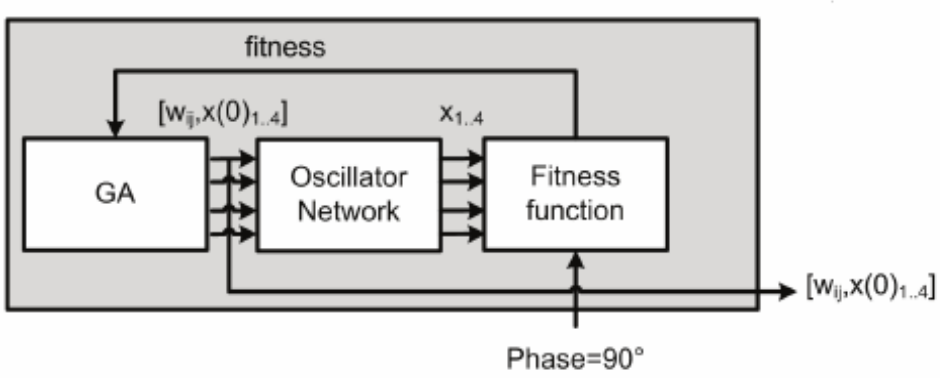

Figura 16: *Diagrama de blocs de l'algorisme fet servir per obtenir els valors de les connexions entre els oscil·ladors i els valors inicials d'aquests.*
Font: [3]

A les figures 15 i 16 es mostren diagrames de blocs de l'algorisme genètic que es fa servir a l'article. L'algorisme intenta optimitzar els paràmetres per tal d'obtenir una amplitud i una freqüència determinades, a més de les condicions inicials i pesos de les connexions adients per tal d'obtenir un desfasament de 90º amb 4 oscil·ladors.

2.5.4. Matriu de connexions obtingudes i simulacions

Finalment, les matrius de pesos o connexions que es fan servir són les següents:

- Moviment lent

$$\text{Slow} = \begin{array}{c} \\ L1 \\ L2 \\ L3 \\ R1 \\ R2 \\ R3 \end{array} \begin{array}{c} \begin{array}{cccccc} L1 & L2 & L3 & R1 & R2 & R3 \end{array} \\ \begin{bmatrix} 0 & -0.1 & 0 & -0.1 & 0 & 0 \\ 0 & 0 & -0.1 & 0 & -0.1 & 0 \\ -0.1 & 0 & 0 & 0 & 0 & -0.1 \\ -0.1 & 0 & 0 & 0 & -0.1 & 0 \\ 0 & -0.1 & 0 & 0 & 0 & -0.1 \\ 0 & 0 & -0.1 & -0.1 & 0 & 0 \end{bmatrix} \end{array}$$

Figura 17: *Matriu de moviment lent.*
Font: [2]

- Moviment mitjà

$$\text{Medium} = \begin{array}{c} \\ L1 \\ L2 \\ L3 \\ R1 \\ R2 \\ R3 \end{array} \begin{array}{c} \begin{array}{cccccc} L1 & L2 & L3 & R1 & R2 & R3 \end{array} \\ \begin{bmatrix} 0 & -0.1 & 0 & -0.1 & 0 & 0 \\ 0 & 0 & 0 & -0.1 & -0.1 & 0 \\ -0.1 & 0 & 0 & 0 & -0.1 & 0 \\ -0.1 & 0 & 0 & 0 & -0.1 & 0 \\ -0.1 & -0.1 & 0 & 0 & 0 & 0 \\ 0 & -0.1 & 0 & -0.1 & 0 & 0 \end{bmatrix} \end{array}$$

Figura 18: *Matriu de moviment mitjà.*
Font: [2]

- Moviment ràpid

$$\text{Fast} = \begin{array}{c} \\ L1 \\ L2 \\ L3 \\ R1 \\ R2 \\ R3 \end{array} \begin{array}{c} \begin{array}{cccccc} L1 & L2 & L3 & R1 & R2 & R3 \end{array} \\ \begin{bmatrix} 0 & -0.1 & 0 & -0.1 & 0 & 0 \\ -0.1 & 0 & -0.1 & 0 & 0 & 0 \\ 0 & -0.1 & 0 & 0 & 0 & -0.1 \\ -0.1 & 0 & 0 & 0 & -0.1 & 0 \\ 0 & 0 & 0 & -0.1 & 0 & -0.1 \\ 0 & 0 & -0.1 & 0 & -0.1 & 0 \end{bmatrix} \end{array}$$

Figura 19: *Matriu de moviment ràpid.*
Font: [2]

A continuació es simula la xarxa neuronal amb aquestes matrius i els següents paràmetres:

- **α** = 1
- **p** = 2
- **ω** = 10

Com s'ha mencionat anteriorment, per introduir aquest sistema a *Matlab*, cal expressar-ho tot com un sistema d'equacions diferencials de primer ordre. Per tant, un sistema de 6 equacions de segon ordre es representarà com a un sistema de 12 equacions de primer odre.

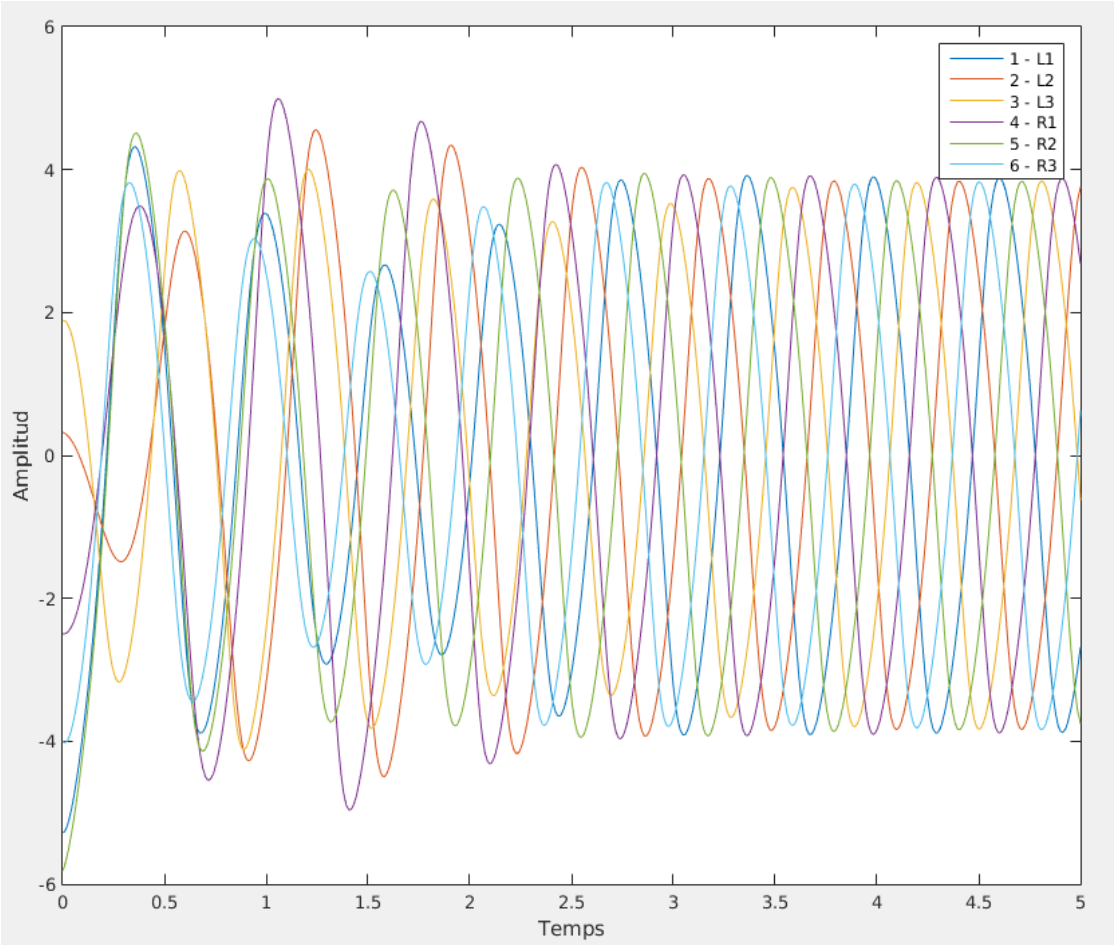

Figura 20: *Evolució temporal d'una xarxa fent servir la matriu de moviment lent. Magnituds adimensionals. Condicions inicials aleatòries entre [6,-6]:*
y1(0)=-5.274346, y1'(0)=-1.208907
y2(0)=0.322510, y2'(0)=-0.998406
y3(0)=1.882319, y3'(0)=1.535680
y4(0)=-2.496191, y4'(0)=-0.820186
y5(0)=-5.814154, y5'(0)=5.808765
y6(0)=-3.993979, y6'(0)=-4.725404

S'hi pot apreciar una etapa inestable transitòria evolucionant cap a una etapa estable, on les diferents ones tenen un desfasament de 2π/6 tal i com s'ha mostrat al diagrama anteriorment.

Un aspecte a tenir present són les diferents condicions inicials dels oscil·ladors. Avaluant diferents condicions inicials al sistema anterior es poden apreciar certs aspectes d'especial importància:

1. Tant la duració del transitori com l'amplitud pics de l'ona durant aquest es veuen afectats per el conjunt de condicions inicials. Això pot implicar, si

no es prenen les mesures adequades, que el robot trigui més o menys en arribar al moviment estable, i que l'estructura pugui no ser estable durant aquest període. A la següent figura es pot comprovar aquest fet, ja que el sistema triga més a estabilitzar-se que en el cas anterior.

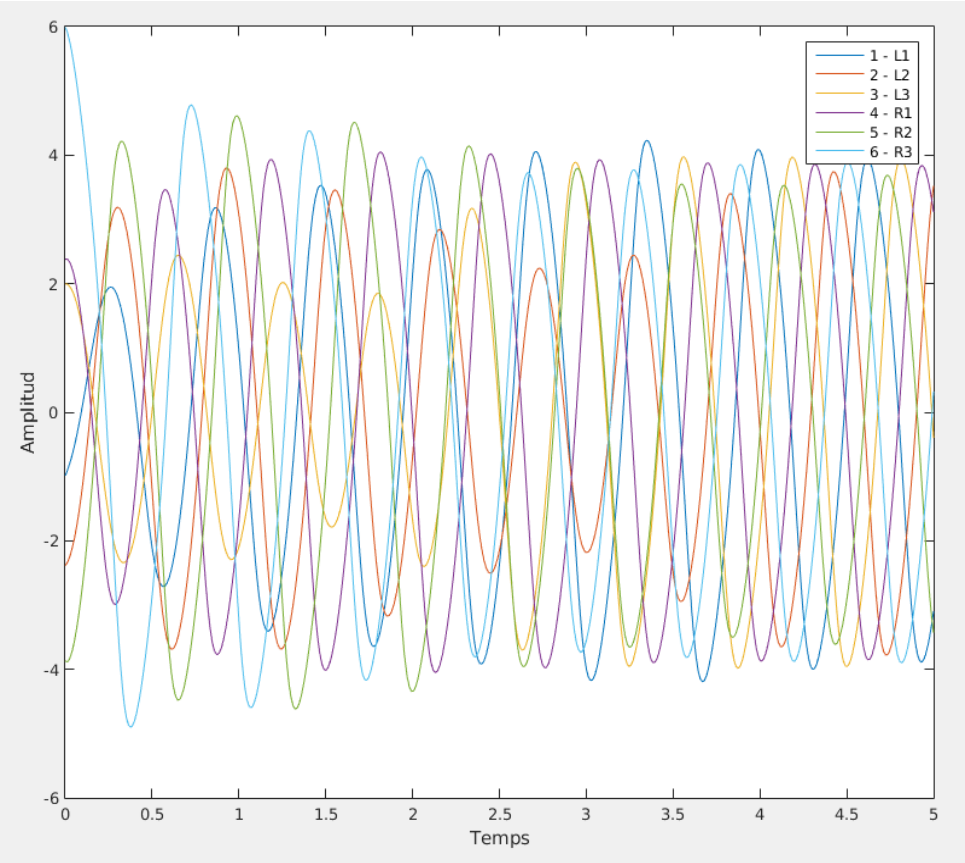

Figura 21: *Evolució temporal del sistema anterior amb condicions inicials diferents. Magnituds adimensionals.*
y1(0)=-0.987071, y1'(0)=5.796630
y2(0)=-2.382541, y2'(0)=2.413185
y3(0)=1.996066, y3'(0)=0.469518
y4(0)=2.377266, y4'(0)=1.998335
y5(0)=-3.862411, y5'(0)=-4.463827
y6(0)=5.988965, y6'(0)=-3.946547

2. Si els valors de les condicions inicials dels diferents oscil·ladors són massa propers entre ells, és probable que les diferents oscil·lacions s'acoblin en una única i no s'aconsegueixin les diferències de fase desitjades.

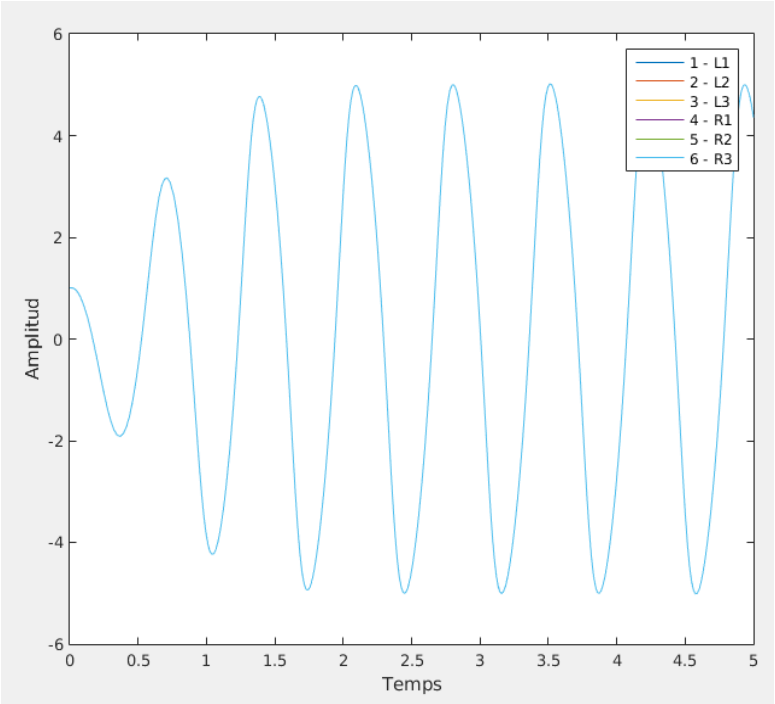

Figura 22: *Evolució temporal del sistema anterior fent servir com a condicions inicials el mateix valor per a tots els oscil·ladors. Magnituds adimensionals.*
y1(0)=1.000000, y1'(0)=1.000000
y2(0)=1.000000, y2'(0)=1.000000
y3(0)=1.000000, y3'(0)=1.000000
y4(0)=1.000000, y4'(0)=1.000000
y5(0)=1.000000, y5'(0)=1.000000
y6(0)=1.000000, y6'(0)=1.000000

3. Si els valors de les condicions inicials són 0, el sistema no arriba mai a oscil·lar.

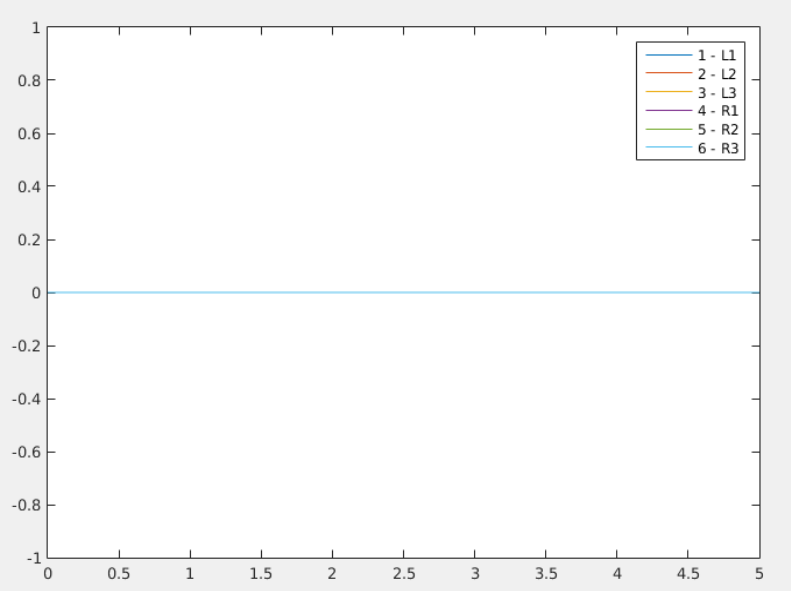

Figura 23: *Evolució temporal del sistema anterior fent servir condicions inicials nul·les. Magnituds adimensionals.*

4. La matriu de connexions que s'hagi fet servir té influencia en aquestes possibilitats. Per exemple, quan es fa servir la matriu de moviment ràpid o trípode és molt més probable que les oscil·lacions s'acoblin entre elles que quan es fa servir la matriu de moviment lent.

A la següent figura es simula el comportament fent servir la matriu de moviment mitjà. Es comprova que les relacions de fase són les indicades anteriorment i que per tant funciona correctament.

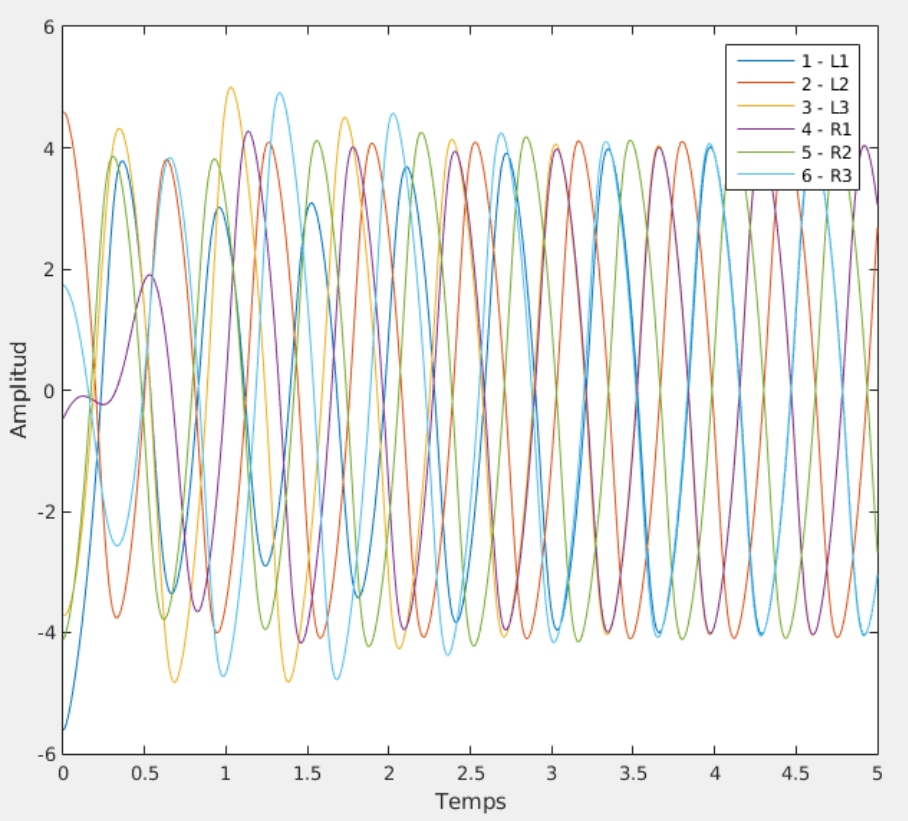

Figura 24: *Evolució temporal del sistema amb la matriu de moviment mitjà i condicions inicials aleatòries entre [-6,6]. Magnituds adimensionals.*
y1(0)=-5.608790, y1'(0)=0.734398
y2(0)=4.582398, y2'(0)=2.030104
y3(0)=-3.714801, y3'(0)=-1.573001
y4(0)=-0.471289, y4'(0)=5.779655
y5(0)=-4.123141, y5'(0)=4.266274
y6(0)=1.737174, y6'(0)=-1.484733

Finalment, el comportament de la matriu de moviment ràpid:

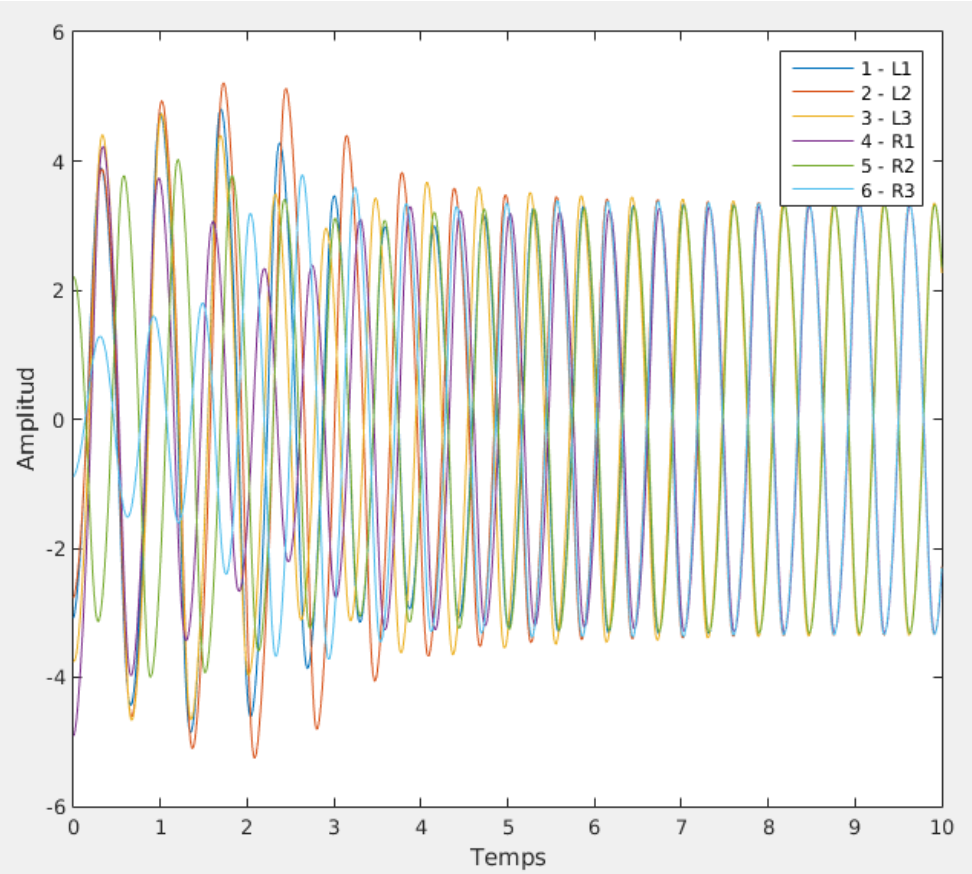

Figura 25: *Evolució temporal del sistema fent servir la matriu de moviment ràpid o trípode i condicions inicials aleatòries entre [-6,6]. Magnituds adimensionals.*

y1(0)=-3.085805, y1'(0)=5.009092
y2(0)=-2.771261, y2'(0)=3.186000
y3(0)=-3.736056, y3'(0)=-2.550022
y4(0)=-4.906638, y4'(0)=0.914513
y5(0)=2.200359, y5'(0)=0.559117
y6(0)=-0.891254, y6'(0)=1.733313

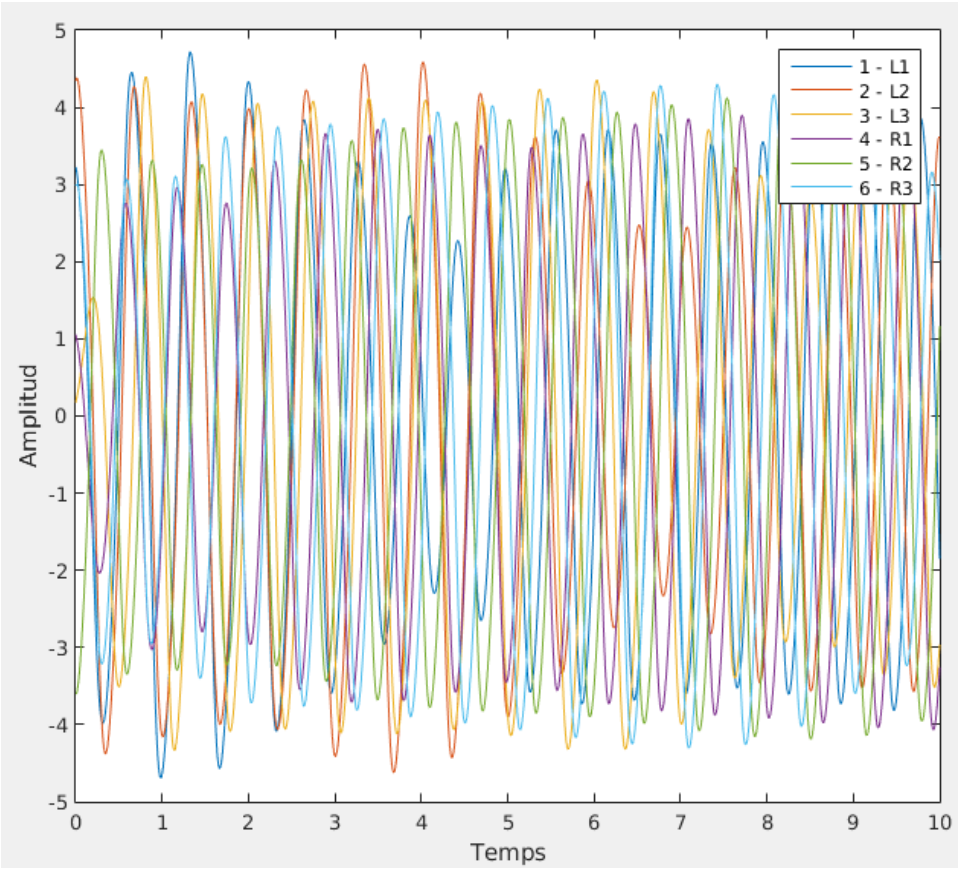

Figura 26: *Evolució temporal del sistema anterior, fent servir condicions inicials diferents. Magnituds adimensionals.*

y1(0)=3.226251, y1'(0)=-3.992957
y2(0)=4.343766, y2'(0)=5.878466
y3(0)=0.173081, y3'(0)=4.611372
y4(0)=1.056313, y4'(0)=-4.142972
y5(0)=-3.601646, y5'(0)=-1.116542
y6(0)=2.984469, y6'(0)=3.907006

A la figura 25 s'observa que 3 oscil·ladors s'acoblen en una oscil·lació única i els altres 3 en una altra oscil·lació, resultant en dues oscil·lacions diferents tal i com s'esperava.

Fent servir la matriu de moviment ràpid i condicions inicials aleatòries, en molts casos casos el sistema no s'estabilitza de la manera desitjada. A la figura 26 s'observa un comportament inadequat en la xarxa. Depenent de les condicions inicials pot passar que el sistema trigui molt més del necessari a estabilitzar-se, o resulti en un acoblament entre oscil·ladors incorrecte (per exemple, tres desfasaments diferents en comptes de dos) Per tant, caldrà anar amb compte quan es faci servir aquesta matriu.

L'efecte de les condicions inicials al tipus de comportament del sistema resulta important ja que, tot i que les condicions inicials a $t = 0$ es poden fixar per tal d'assegurar un comportament adequat, si es vol canviar de tipus de moviment un cop el sistema ja està funcionant llavors les condicions inicials de la següent etapa són les condicions finals de l'etapa anterior, i aquestes no es poden fixar.

Afortunadament, el desfasament dels oscil·ladors quan es fa servir la matriu de moviment lent assegura que la distància entre els valors dels oscil·ladors en tot

moment, un cop assolit el règim estable, sigui tal que garanteixi el correcte funcionament si en qualsevol moment es vol canviar a la matriu de moviment ràpid. Això es pot apreciar a la figura 27.

A la següent figura es mostra l'evolució temporal del sistema i el seu comportament quan es canvia sobtadament la matriu de connexions. Els paràmetres són:

- $\alpha = 1$
- $p = 2$
- $\omega = 5$

El gràfic es divideix en etapes de 10 segons, entre les quals s'efectuen els canvis de matriu. Es fan servir les matrius exposades anteriorment, i l'ordre d'aplicació és el següent:

Moviment lent → moviment ràpid → moviment mitjà → moviment ràpid

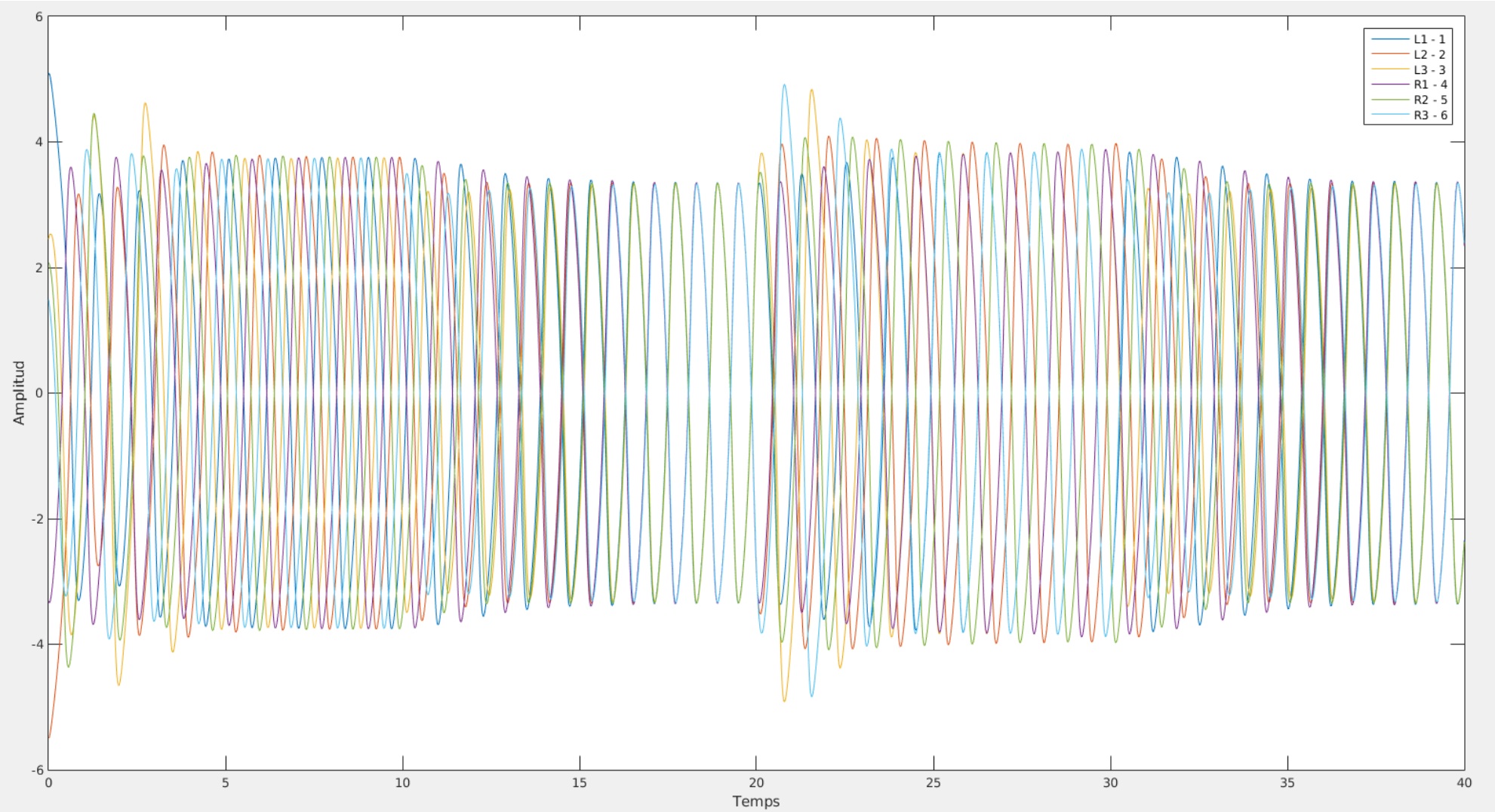

Figura 27: *Evolució temporal del sistema amb diferents canvis de matriu. Magnituds adimensionals. Els canvis de matriu es realitzen en els instants 10, 20 i 30.*
y1(0)=5.067984, y1'(0)=3.251451
y2(0)=-5.488082, y2'(0)=-1.461766
y3(0)=2.452075, y3'(0)=2.754157
y4(0)=-3.308675, y4'(0)=-2.771343
y5(0)=2.076374, y5'(0)=-0.270094
y6(0)=1.484597, y6'(0)=-3.162661

CAPÍTOL 3: DISSENY I IMPLEMENTACIÓ DE LA XARXA

3.1. Introducció

Un cop simulat el generador de patrons fent servir diferents paràmetres i comprovant que el seu comportament és l'adequat, el següent pas és la implementació al robot hexàpode.

En primer lloc es farà un resum dels aspectes importants del robot que es farà servir. Seguidament es compararan diferents suports per a la implementació hardware del sistema i es justificarà el suport escollit, i finalment es tractaran temes més específics del disseny.

3.2. Característiques del robot

3.2.1. Estructura

El robot que es farà servir és un robot hexàpode modular desenvolupat a l'EUETIB gràcies al treball col·laboratiu en diversos TFG. Es tracta d'un sistema modular en el qual cada mòdul inclou dues potes, cadascuna de les quals consta de dos servos (un horitzontal i un vertical), i una placa *Arduino Mini* que s'encarrega del control d'aquests.

El sistema consta de 3 mòduls com el mencionat, a més d'un mòdul de control on en un principi hi havia una placa *Arduino Uno* que s'encarregava d'enviar els senyals de control als altres mòduls. Aquest últim no es farà servir ja que s'hi

col·locarà una FPGA amb més capacitat de càlcul al seu lloc per tal d'efectuar el control. La jerarquia de control és la següent:

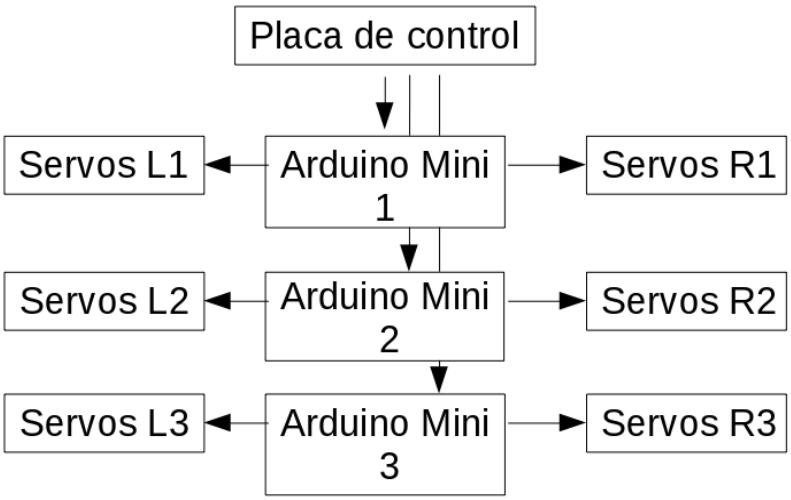

Figura 28: *Jerarquia de control al robot.*

S'hi pot apreciar que el control està centralitzat en una placa independent, la qual s'encarrega de controlar els altres elements.

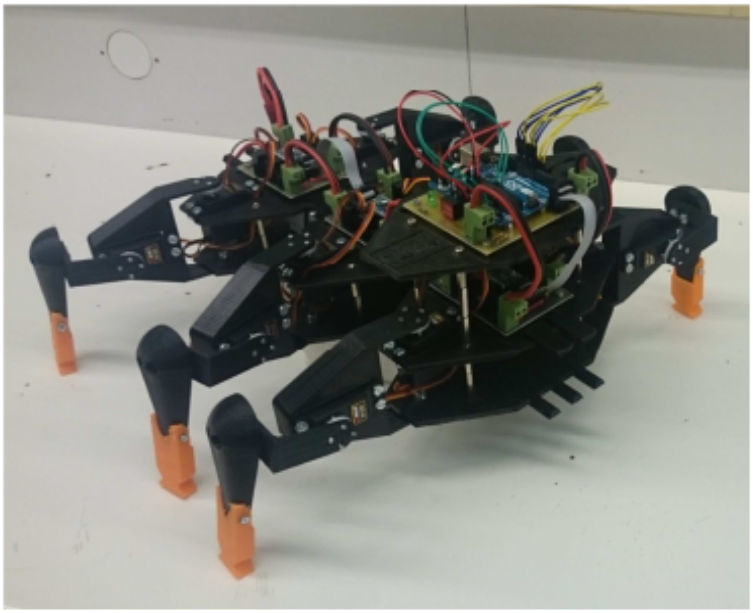

Figura 29: *Robot hexàpode original.*
Font: [10]

Un aspecte important a tenir en compte és que la qualitat dels servos actuals no és gaire alta, i per tant quan el robot està en funcionament poden tremolar bastant. A més l'acceleració no és massa elevada, cosa que limita la quantitat de canvis de posició que es poden fer en un determinat temps. Això últim s'ha de tenir en compte a l'hora d'escollir un interval d'actualització dels servos.

3.3. Comunicació

3.3.1. Procediment

Seguint aquesta jerarquia, el procediment per assignar una nova posició a un servo determinat és el següent:

1. La placa de control estableix una consigna i l'envia per el bus de comunicació, juntament amb la selecció del servo desitjat.
2. L'Arduino del mòdul al qual la consigna va dirigida envia la consigna al servo pertinent, que actualitza seguidament la seva posició.

Per tant, per cada servo que es vulgui moure, a la placa de control s'han d'establir 3 punts diferents:

1. Posició del servo desitjada.
2. Mòdul al qual el servo en qüestió pertany.
3. Selecció d'un dels 4 servos controlats pel mòdul en qüestió (2 horitzontals i 2 verticals).

Per tal d'efectuar aquesta comunicació s'aprofitarà el bus SPI present al robot. Així doncs, serà necessari estudiar el funcionament de l'estàndard SPI, a més d'establir un protocol que permeti encapsular els 3 punts mencionats.

3.3.2. L'estàndard SPI

L'estàndard SPI (de l'anglés *Serial Peripheral Interface*) és un estàndard síncron i sèrie de comunicacions per a la transferència d'informació entre circuits. El sistema fa servir una arquitectura mestre-esclau, en el qual el mestre decideix quan s'efectuen les transmissions. A més es tracta d'un sistema *full dúplex*, és a dir, permet una comunicació bidireccional entre el mestre i l'esclau. En aquest projecte aquesta característica no serà rellevant ja que només serà necessària la comunicació en sentit Mestre → Esclau.

El sistema consta dels següents senyals:

- SCLK (*Serial Clock)*. Senyal de rellotge generat pel mestre.
- MOSI (*Master Output, Slave Input*). Sortida de dades del mestre.
- MISO (*Master Input, Slave Output).* Entrada de dades del mestre.
- SS_n (*Slave Select n*). Senyal d'activació de l'esclau. Hi ha tants com esclaus amb els quals el mestre s'hagi de comunicar. És un senyal actiu a nivell baix.

El procediment de comunicació és el següent:

1. El mestre configura el rellotge a la freqüència desitjada.
2. El mestre produeix un '0' (nivell baix) al *Slave* Select adient.
3. Mentre hi hagi el '0' al SS, a cada cicle del rellotge (*SCLK*) s'intercanvia un bit entre el mestre i l'esclau, fent servir les línies *MOSI* i *MISO*. Típicament els bits que es van transferint s'emmagatzemen a un registre.
4. La transmissió finalitza quan el mestre torna a deixar el senyal SS a nivell alt.
5. En funció del hardware es poden realitzar operacions addicionals com ara generar interrupcions per avisar de que s'ha completat la transferència en el cas dels microcontroladors.

A més, en funció de diferents aspectes, hi han quatre modes de funcionament diferents. Aquests modes influencien la polaritat i la fase dels senyals i per tant és indispensable que tots els punts connectats al bus estiguin configurats per fer servir el mateix mode.

Aquests modes són el resultat de les possibles combinacions de dues opcions:

- CPOL: Estableix la polaritat del senyal de rellotge SCLK:
  - CPOL = 0: L'estat actiu del rellotge és 1 (nivell alt). Aquest és el funcionament típic dels senyals de rellotge.
  - CPOL = 1: L'estat actiu del rellotge és 0 (nivell baix), i l'actiu 1 (nivell alt). Aquest mode és menys comú.
- CPHA: Estableix en quin flanc s'efectua la transmissió de dades.
  - CPHA = 0: La transmissió s'efectua en el flanc *Nivell actiu → Nivell inactiu* (en funció de CPOL).
  - CPHA = 1: La transmissió s'efectua en el flanc *Nivell inactiu → Nivell actiu* (en funció de CPOL).

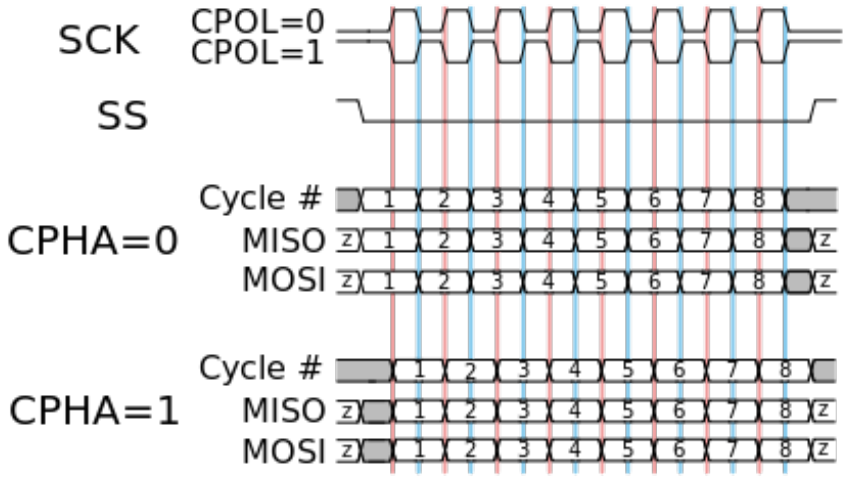

Figura 30: *Diagrames dels diferents modes SPI.*
Font: [12]

Molts microcontroladors fan servir la següent nomenclatura:

Taula 1: *Mode SPI en funció de CPOL i CPHA.*

Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

D'aquesta manera només cal especificar el número del mode al codi pertinent.

3.3.3. Protocol per a l'enviament de les dades dels servos

Per tal de permetre el control dels diferents servos del robot a través del bus SPI, s'ha acordat el següent protocol amb un altre estudiant que també està treballant amb el robot, qui a més s'ha encarregat de programar els 3 *Arduino Mini* per tal d'adaptar el sistema al següent protocol:

- Totes les dades s'encapsulen en un byte
- La posició del servo desitjada és representada pels 6 bits menys significatius (en blau), mentre que els 2 bits més significatius (en vermell) seleccionen el servo:

Bit 7 (MSB) 00000000 Bit 0 (LSB)

Taula 2: *Bits de selecció de servo.*

Bit 7	Bit 6	Servo
0	0	Servo dret vertical
0	1	Servo dret horitzontal
1	0	Servo esquerre vertical
1	1	Servo esquerre horitzontal

La selecció del mòdul (*Arduino)* que es vol fer servir es realitza mitjançant els senyals Ss_n del bus SPI. Per tant no és necessari considerar-ho aquí.

3.4. Tecnologies disponibles

3.4.1. Tipus d'implementacions

Com s'ha explicat anteriorment, el sistema consisteix en un sistema d'equacions diferencials que s'han de resoldre per tal de generar les formes d'ona necessàries, que en última instància mouran són les que definiran el moviment de les diferents potes.

El suport en el qual es realitza aquest projecte ha estat decidit a priori, abans de començar-lo. No obstant, és interessant comparar-lo amb altres opcions i justificar la seva utilitat.

A la literatura rellevant s'aprecien tres tipus d'implementacions diferents:

- **Implementació analògica:** Es fan servir circuits analògics per tal d'aconseguir el comportament desitjat. Aquesta implementació té l'avantatge que ofereix un comportament similar al de les neurones biològiques i una eficiència energètica elevada. L'inconvenient és que requereix cicles de disseny bastant llargs per a cada modificació, i a vegades una modificació relativament petita pot implicar la necessitat de tornar a dissenyar el circuit sencer.
- **Implementació amb microprocessadors:** Aquests sistemes ofereixen, a diferència de les implementacions analògiques, una gran precisió i flexibilitat per ser re-configurats, però tenen l'inconvenient d'ocupar més àrea i requerir més energia, factors que redueixen la seva utilitat en l'ús en robots.
- **Implementació amb FPGA:** Ofereixen una flexibilitat més elevada i uns temps de reacció molt més reduïts que les implementacions amb microprocessadors, a més d'una millor relació rendiment/àrea*consum d'energia. Els temps de reacció reduïts són deguts a la possibilitat d'establir elements de computació en paral·lel o en sèrie segons sigui convenient, fet que no és possible als microprocessadors, que tenen una arquitectura fixa. Un altre aspecte a tenir en compte és que la realització de disseny en FPGA es fa típicament de manera modular, fet que juntament amb la modularitat intrínseca dels generadors de patrons centrals fa la seva implementació en aquests sistemes més intuïtiva. A més a més, serveixen com a plataforma de proves pel cas que es vulgui realitzar posteriorment el sistema amb un *ASIC* (circuit integrat d'aplicació específica).

3.4.2. Funcionament d'una FPGA

Una FPGA (de l'anglés, *Field-Programable Gate* Array) és un tipus de circuit integrat, dissenyat de manera que pugui ser reconfigurable per qui el faci servir. Aquest circuit disposa d'una sèrie de blocs lògics programables i una jerarquia de connexions internes reconfigurables. La combinació d'aquests elements permet la realització tant de circuits combinacionals com seqüencials, mitjançant la utilització de funcions combinacionals i elements de memòria com ara els *flip-flops.*

Mentre que als microprocessadors típicament l'element característic i condicionant del seu rendiment és la velocitat a la qual aquests poden realitzar operacions, a les FPGAs la característica típicament determinant és la quantitat de recursos disponibles, és a dir, la quantitat de blocs i connexions que es poden fer servir.

El disseny es realitza típicament mitjançant llenguatges *HDL* (de l'anglés, *Hardware Description Language).* També és possible un disseny esquemàtic, que resulta més intuïtiu però és poc adequat quan es treballa amb estructures complexes. Actualment hi han dos llenguatges *HDL* disponibles: *VHDL* i *Verilog.*

El disseny per a la FPGA en aquest projecte es realitzarà únicament en *VHDL*.

3.4.3. FPGAs disponibles

- *Digilent Nexys 2 Spartan 3E*

  Es tracta d'una placa de desenvolupament, és a dir, dedicada especialment a l'aprenentatge i pràctica del desenvolupament de dissenys de *FPGA*. Incorpora multitud d'entrades i sortides disposades en diferents tipus de busos i interruptors, a més dels circuits d'alimentació i adaptació necessaris per al seu funcionament.

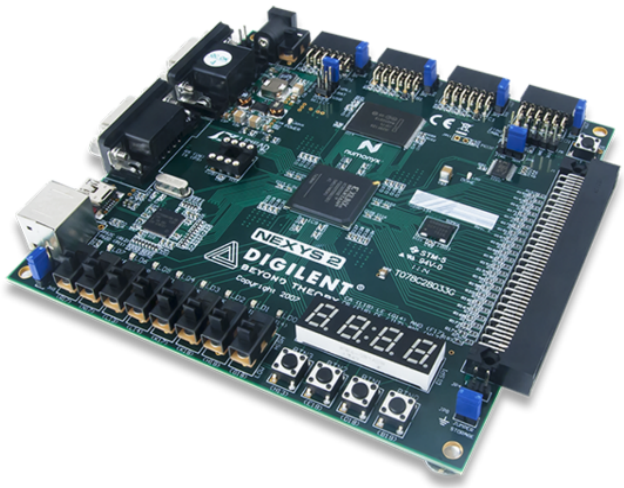

Figura 31: *Nexys 2 Spartan-3E.*
Font: [4]

  El rellotge integrat és de 50 MHz, de sobres per l'aplicació d'aquest projecte. Com a suport per realitzar proves és molt útil, però té l'inconvenient que es massa gran per incorporar-ho a cap aplicació pràctica.

- *Digilent Cmod S6 (Spartan 6)*

  És una placa més petita que la anterior i per tant més practica a l'hora d'incorporar-la al robot. Disposa una *FPGA* més moderna que la *Spartan-6*, però no obstant té menys recursos disponibles.

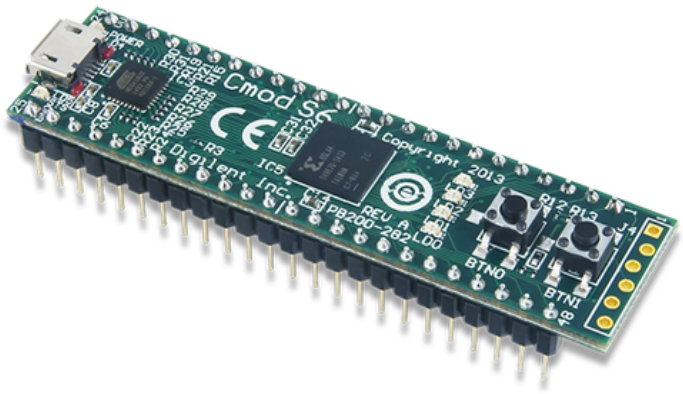

Figura 32: CMod S6 de Digilent.
Font: [5]

3.5. Implementació de la xarxa en una FPGA

3.5.1.Aspectes importants

Les característiques importants de la FPGA a l'hora de fer un disseny són

- La freqüència del rellotge
- Els recursos (àrea) disponibles

El primer punt no serà rellevant en aquest disseny, ja que típicament les FPGAs tenen un rellotge de l'ordre de MHz, i la freqüència de les oscil·lacions dels oscil·ladors serà de l'ordre d'1 Hz. La freqüència a la qual s'actualitzaran els valors dels oscil·ladors serà com a molt de l'ordre d'una desena d'Hz, i per tant es disposa d'un marge molt elevat.

La quantitat de recursos disponibles, però, pot arribar a ser un element condicionant en aquest disseny, degut a la necessitat de realitzar bastants càlculs per a cada oscil·lador de la xarxa. Aquest tema es tractarà més endavant.

3.5.2. Representació numèrica i realització de càlculs a la FPGA

Dins del llenguatge VHDL, l'estàndard IEEE (*de l'acrònim anglés Institute of Electrical and Electronics Engineers*) proporciona un conjunt llibreries que faciliten en gran mesura la realització d'operacions aritmètiques. Sense aquestes llibreries seria necessari definir cadascuna de les operacions necessàries, fent servir únicament portes lògiques.

Aquestes llibreries s'encarreguen de definir tant els tipus de senyals possibles com el funcionament del conjunt d'operacions disponibles entre aquests senyals.

Per exemple, un tipus molt útil que proporciona la llibreria és el tipus *integer*, que permet representar i operar amb nombres enters entre -2147483648 i +2147483647 quan es configura per fer servir 32 bits (sent el bit més significatiu el que indica el signe). A més, permet reduir el rang possible per tal de reduir el nombre de bits utilitzats i estalviar recursos a la FPGA.

En aquest projecte es faran servir principalment els següents tipus de senyals:

- *integer*, per la realització de comptadors.
- *STD_LOGIC* i *STD_LOGIC_VECTOR*, per a la representació d'estats lògics i vectors de bits quan sigui necessari treballar amb aquests directament.
- *sfixed*, per a la representació de nombres amb signe fent servir punt fixe.

El tipus *sfixed* serà bastant crític en aquest disseny, ja que s'ha escollit per representar els valors dels oscil·ladors i realitzar les operacions pertinents. La raó principal per la qual es fa servir punt fixe i no punt flotant, és que la aritmètica en punt fixe requereix menys recursos, factor que com s'ha esmentat abans, és critic en un disseny per a *FPGA*. A més, degut a que el rang de valors possibles a les oscil·lacions ve definit per la seva amplitud, no és necessari el gran rang dinàmic que ofereix el punt flotant.

També seria possible fer servir senyals d'un altre tipus com ara enters, però llavors es complicaria bastant el disseny ja que per modificar qualsevol magnitud possiblement caldria modificar totes les altres magnituds presents al codi. El tipus *sfixed* proporciona un conjunt de funcions que fan el disseny molt més intuïtiu, permetent per exemple especificar fàcilment el rang i precisió dels senyals i realitzar assignacions entre senyals de diferent nombre de bits, amb la possibilitat de controlar aspectes com ara l'arrodoniment/truncament o la saturació dels nombres. Això permet mantenir un ús de recursos no massa elevat mitjançant la utilització de senyals de nombre amb un nombre bits limitat, a la vegada que es controlen els errors per arrodoniment o saturació.

3.5.3. Resolució dels sistemes d'equacions

Per tal de resoldre els sistemes d'equacions diferencials del sistema es farà servir un mètode d'integració numèrica. Concretament, es farà servir el mètode d'integració d'Euler.

Aquest mètode es caracteritza per ser el mètode d'integració d'ODEs més bàsic. És un mètode de primer ordre i té un error més elevat que altres mètodes més complexes, però resulta útil per aquest projecte ja que és molt fàcil d'implementar i requereix pocs recursos.

Per resoldre una equació de segon ordre com la que s'ha exposat anteriorment, el procediment és el següent:

1. En primer lloc cal representar l'equació de segon ordre com a sistema d'equacions de primer odre:

$$\ddot{x}(t)=f(t,\dot{x}(t),x(t)) \Rightarrow \begin{matrix} y(t)=\dot{x}(t) \\ \dot{y}(t)=f(t,x(t),y(t)) \end{matrix} \qquad (1)$$

2. Es parteix d'unes condicions inicials prèviament especificades i es divideix el temps a integrar en una sèrie de subintervals t_n.

$$\begin{matrix} x(t_o)=x_0 \\ \dot{x}(t_0)=\dot{x_0} \end{matrix} \qquad (2)$$

$$t_{n+1}=t_n+h \qquad (3)$$

3. Per a cada iteració es calculen els valors de les variables en funció dels valors obtinguts a la iteració anterior:

$$\begin{aligned} y(t_n) &= y(t_{n-1}) + h\,\dot{y}(t_{n-1}) \\ \dot{y}(t_n) &= \dot{y}(t_{n-1}) + hf(t_{n-1}, x(t_{n-1}), y(t_{n-1})) \end{aligned} \qquad (4)$$

A continuació es comparen diferents simulacions d'un oscil·lador únic, fent servir el mètode de *Matlab* ode45() i el mètode d'Euler implementat. Els paràmetres de l'oscil·lador utilitzats són α = p = ω = 0,5.

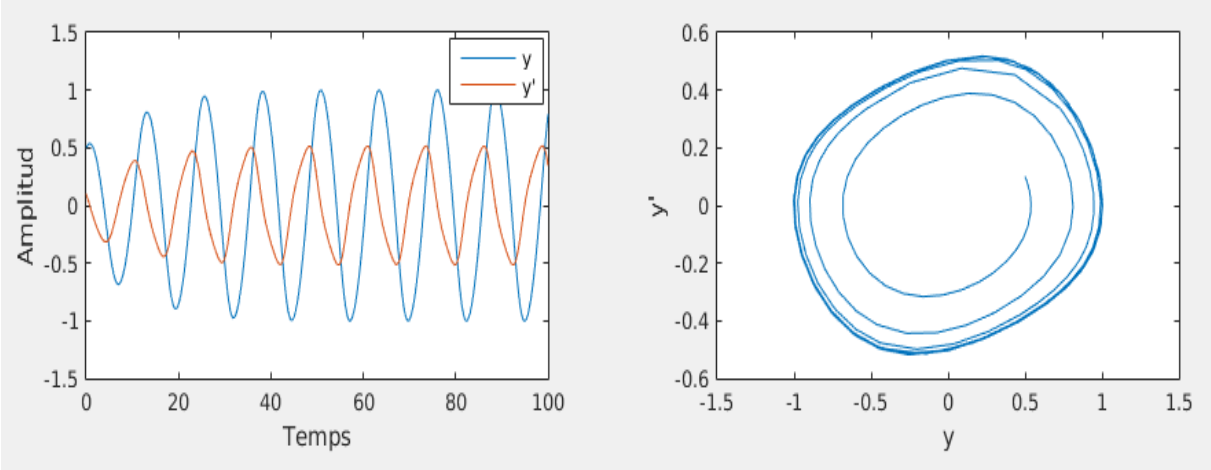

Figura 33: *Evolució temporal i diagrama d'estat de l'oscil·lador de van der Pol obtinguda amb la funció ode45() de Matlab. Magnituds adimensionals.*
y(0) = 0,5, y'(0) = 0,1

A la figura anterior, el mètode de *Matlab* ha fet servir 261 punts d'integració, no necessàriament equidistants.

A la següent figura es fa servir el mètode d'Euler amb 261 punts d'integració equidistants. Es pot apreciar una distorsió apreciable respecte a la figura anterior.

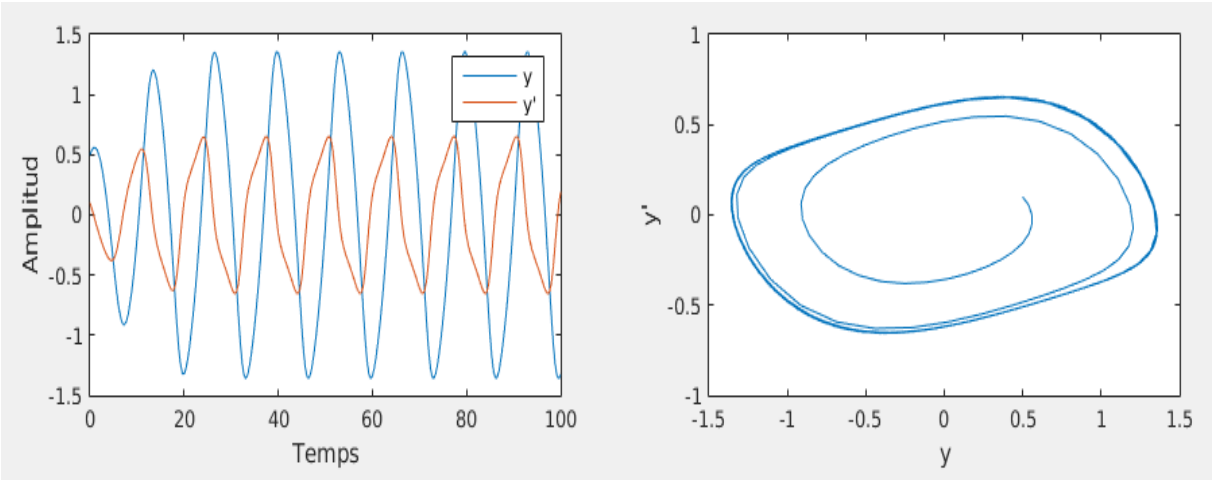

Figura 34: *Evolució temporal i diagrama d'estat de l'oscil·lador fent servir el mètode d'Euler amb 261 punts d'integració equidistants. Magnituds adimensionals.*
y(0) = 0,5, y'(0) = 0,1

No obstant, si es fa servir un pas d'integració suficientment petit, el resultat s'apropa més al resultat esperat, tal i com es pot apreciar a la següent figura.

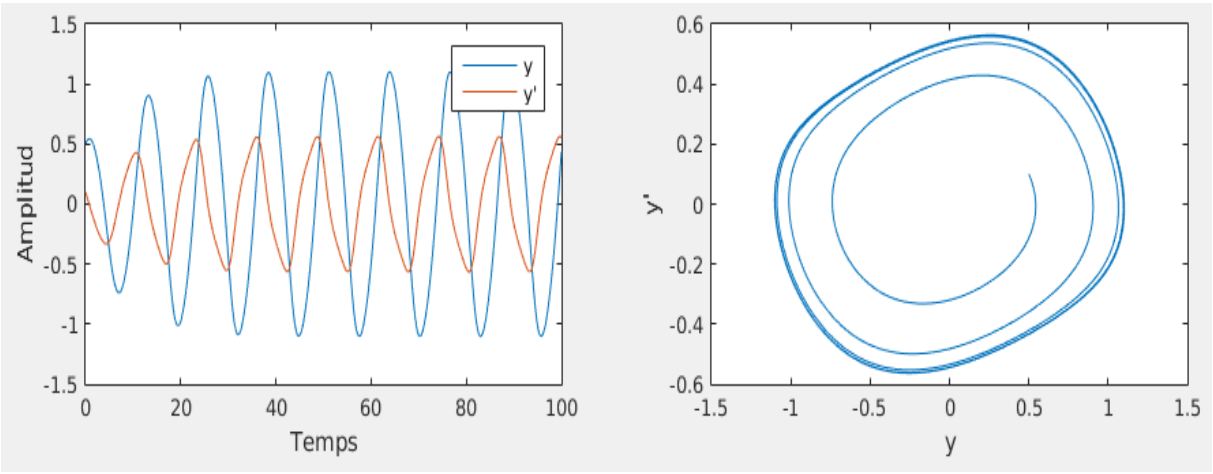

Figura 35: *Figura 18: Evolució temporal i diagrama d'estat de l'oscil·lador fent servir el mètode d'Euler amb 1000 punts d'integració equidistants. Magnituds adimensionals.*
y(0) = 0,5, y'(0) = 0,1

3.5.4. Implementació d'un oscil·lador únic a la FPGA

Un cop estudiat i comprovat el mètode d'integració, es procedeix a la implementació de l'oscil·lador de van der Pol en el disseny de la FPGA.

En primer lloc és important tenir en compte les operacions que s'han d'efectuar:

- L'equació de l'oscil·lador fa servir sumes, restes i multiplicacions.
- A cada iteració s'efectuen càlculs amb els valors de la iteració anterior. Per tant, són necessaris elements de memòria per tal d'emmagatzemar els

valors anteriors. Es consideraran registres síncrons amb senyal d'actualització (*enable*).

Per a un oscil·lador únic, un possible disseny seria el següent:

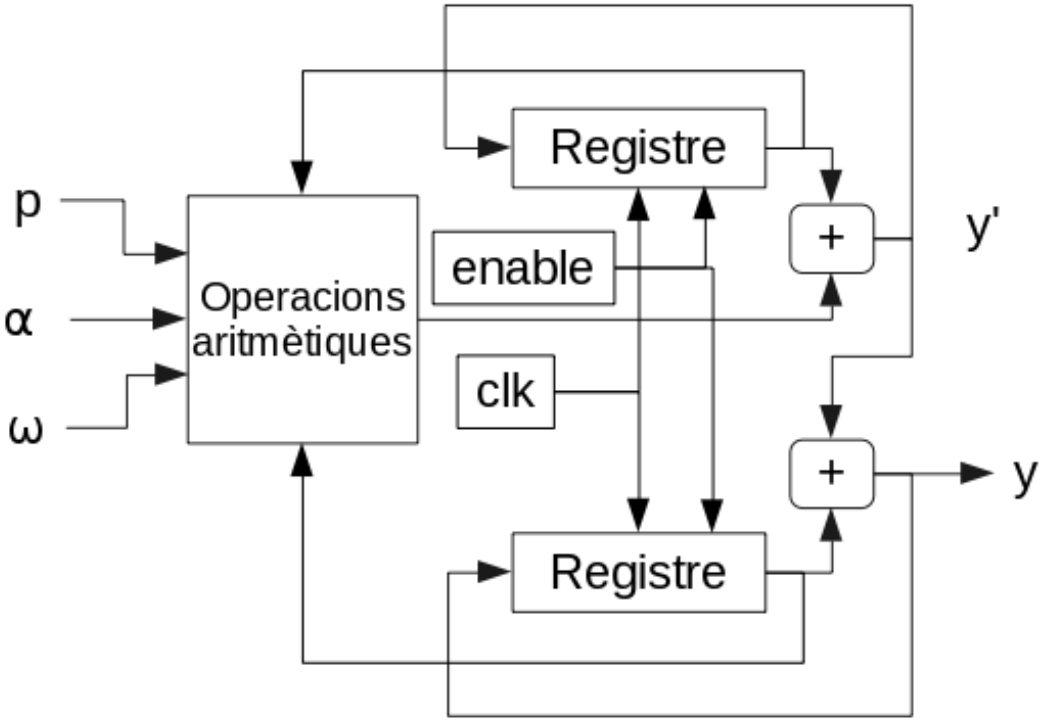

Figura 36: *Diagrama del disseny d'un oscil·lador de van der Pol únic.*

En aquest disseny, per simplicitat s'assumeix un pas d'integració d'una unitat. Tots els blocs, excloent-hi els registres, són combinacionals. A cada cicle de rellotge, si el senyal d'*enable* està a nivell alt, s'assignen els nous valors de les variables *y* i *y'* als dos registres. El senyal d'*enable* permet actualitzar el sistema a la freqüència desitjada, mecanisme que permet modificar la velocitat d'oscil·lació sense haver de modificar els paràmetres numèrics de l'equació.

Aquest sistema no és més que una implementació de l'equació esmentada anteriorment:

$$\begin{aligned} y(t_n) &= y(t_{n-1}) + h\,\dot{y}(t_{n-1}) \\ \dot{y}(t_n) &= \dot{y}(t_{n-1}) + hf(t_{n-1}, x(t_{n-1}), y(t_{n-1})) \end{aligned} \qquad (4)$$

Si es vol fer servir un pas d'integració diferent, caldria incloure el pas d'integració *h* abans dels registres. Un possible disseny seria el següent:

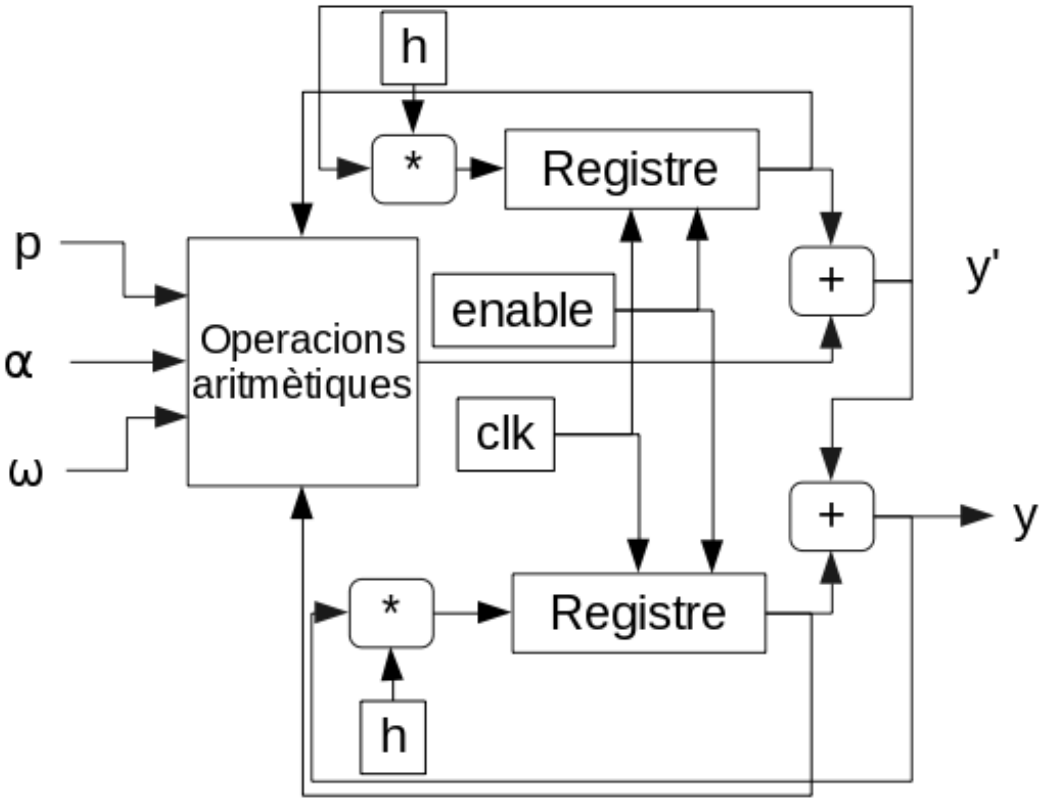

Figura 37: *Diagrama del disseny de l'oscil·lador de VDP incloent-hi un pas d'integració variable h.*

Degut a l'elevada freqüència a la que les FPGAs treballen, en molts casos pot ser interessant reduir el pas d'integració per tal de:

1. Reduir la freqüència d'oscil·lació del sistema.
2. Reduir la variació dels valors entre cada cicle, augmentant per tant la precisió de la integració.

Reduir el pas d'integració implica fer servir un pas $h < 1$. Per tant, realment el que s'estaria efectuant és una divisió.

Tenint en compte que tant les multiplicacions com les divisions requereixen bastants recursos, és possible aprofitar una propietat de la representació binària dels nombres: l'operació de moure tots els dígits d'un nombre binari cap a l'esquerra *n* posicions és equivalent a multiplicar aquest nombre per 2^n, mentre que si el moviment es produeix cap a la dreta l'operació equivalent és la de dividir per 2^n. Aquesta operació es coneix en anglés com a «*shifting*», i és interessant ja que requereix pocs recursos.

Per tant, si es vol fer servir un pas d'integració de 2^{-n}, un disseny que estalviaria recursos respecte el disseny anterior seria el següent:

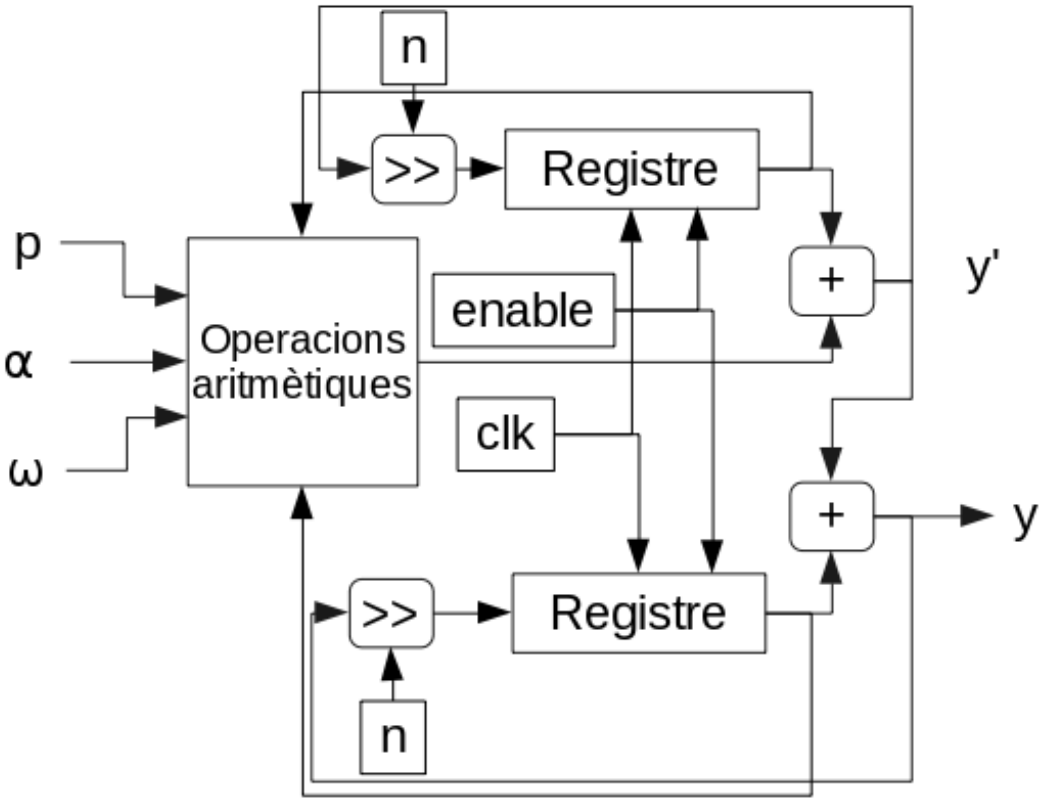

Figura 38: *Variació del disseny anterior.*

El símbol «>>» representa l'operació de moure els bits cap a la dreta. Una limitació d'això és que si el nombre de bits dels senyals és massa petit, l'operació possiblement resultarà en 0 i per tant l'oscil·lador s'aturarà.

El bloc d'operacions aritmètiques representa la part $f(t_{n-1}, x(t_{n-1}), y(t_{n-1}))$ de l'equació (4) anterior, i a més dels estats anteriors (y i y') s'inclouen els paràmetres α, p i ω com a entrades. És un bloc totalment combinacional, i la representació d'aquest bloc és la següent:

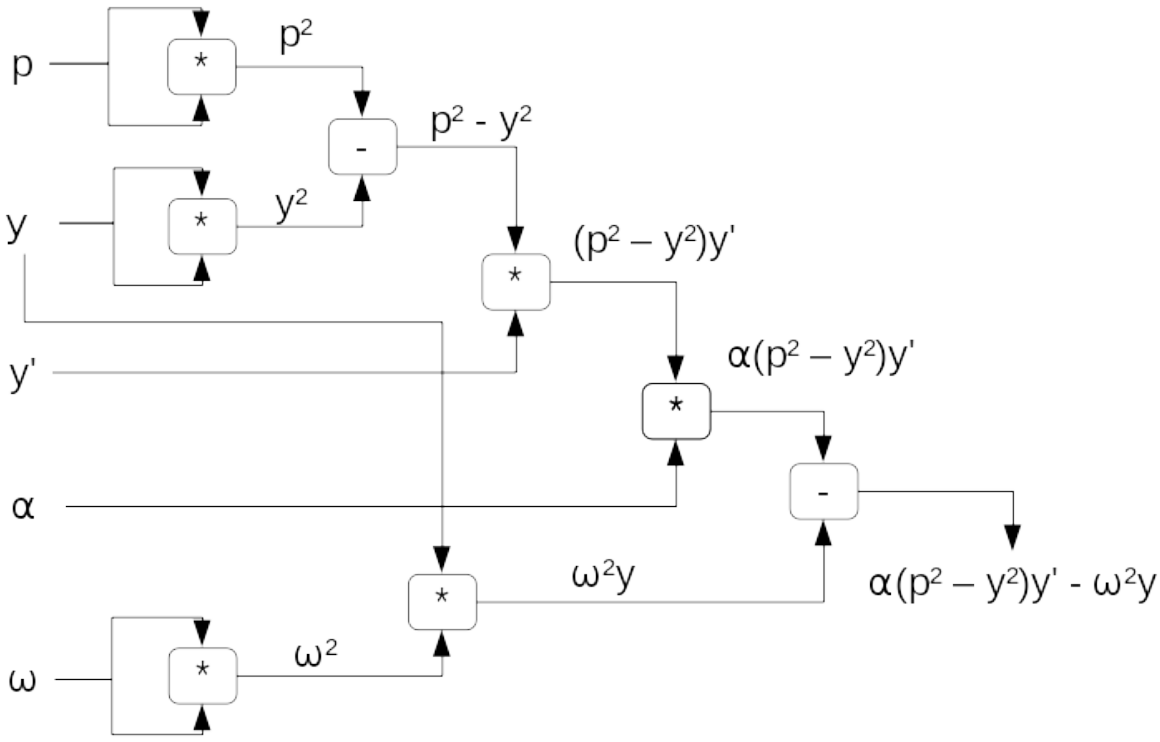

Figura 39: *Representació del bloc aritmètic.*

Un cop definit el sistema es procedeix a simular el seu comportament al programari *Vivado* de *Xilinx*.

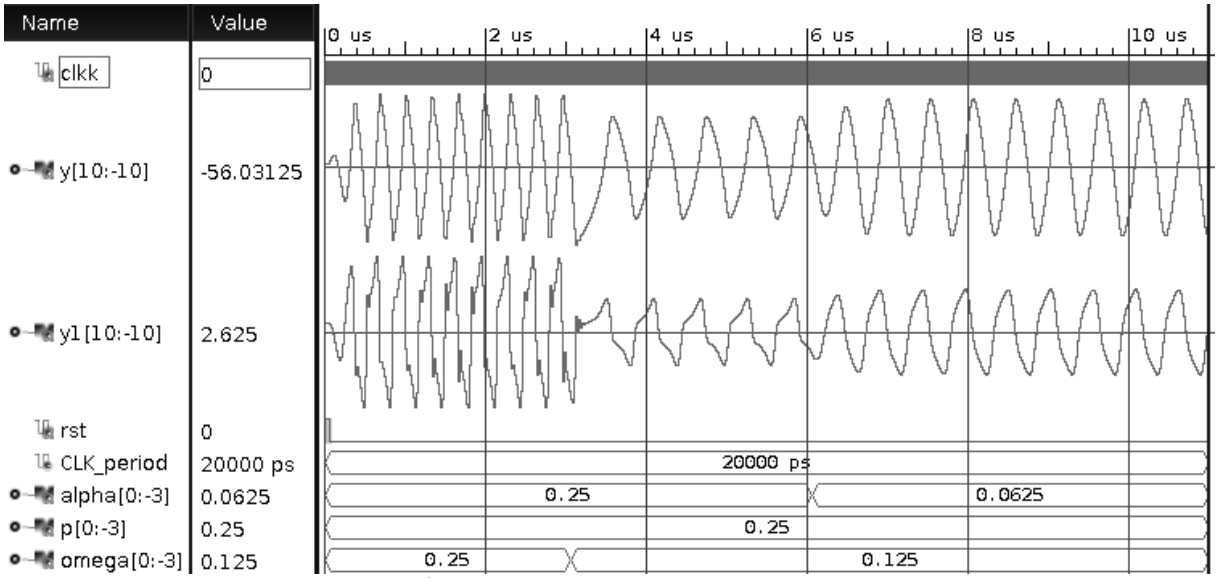

Figura 40: *Simulació del disseny en VHDL d'un oscil·lador de van der Pol únic.*

S'observa l'evolució temporal dels paràmetres *y* i *y'* (*y1* a la simulació). Els paràmetres α, p i ω es modifiquen durant la simulació, provocant canvis en l'ona resultant.

El disseny simulat és semblant al de la figura 20, ja que el pas d'integració és unitari. En aquest cas, però, no es fa servir senyal d'activació o *enable*, i l'oscil·lador actualitza el seu estat a cada cicle de rellotge. Més endavant es farà servir un disseny que amb senyal d'activació que sí que permetrà ralentitzar el funcionament del sistema.

En aquest cas s'han fet servir senyals de punt fixe amb signe de 21 bits per emmagatzemar els estats de l'oscil·lador, distribuïts de manera que els 11 bits més significatius representen la part entera i els 10 bits menys significatius la part decimal. El bit més significatiu és el bit de signe, i per tant queden 10 bits per representar la part entera. Per tant, el rang de valors enters disponible en aquest cas és $[(2^{10}-1),-2^{10}]=[1023,-1024]$. A la simulació el valor de y es manté dins del rang [57, -57], i per tant es podrien estalviar recursos assignant menys bits a la part entera. Concretament, amb 6 bits a la part entera més el bit de signe, el rang seria de $[(2^{6}-1),-2^{6}]=[63,-64]$.

En quant a la precisió decimal, el valor més petit que es pot representar amb 10 bits a la part decimal és de $2^{-10}=976{,}5625*10^{-6}$. La precisió necessària en un sistema com aquest és difícil de determinar, i serà més difícil encara quan es tracti la xarxa amb els 6 oscil·ladors acoblats. No obstant, s'ha comprovat empíricament, mitjançant repetides simulacions, que si la precisió decimal no és suficientment elevada, la xarxa d'oscil·ladors no funciona correctament.

En quant a les operacions aritmètiques internes, es fan servir senyals temporals els amb una longitud de bits tal que permeti que:

a) el resultat d'una operació mai pugui excedir del rang de nombres enters disponible,

b) el resultat d'una operació mai pugui implicar un error d'arrodoniment per pèrdua de decimals.

Això implica que a la majoria d'operacions serà necessari emmagatzemar el resultat en un senyal amb més bits que els senyals dels operands. La llibreria fa servir el següent protocol:

Table G.2—Index range of result of an operation

Operation	Result range
A + B	Max(A'left, B'left) + 1 **downto** Min(A'right, B'right)
A – B	Max(A'left, B'left) + 1 **downto** Min(A'right, B'right)
A * B	A'left + B'left + 1 **downto** A'right + B'right
A **rem** B	Min(A'left, B'left) **downto** Min(A'right, B'right)
Signed A/B	A'left – B'right + 1 **downto** A'right – B'left
Signed A **mod** B	Min(A'left, B'left) **downto** Min(A'right, B'right)
Signed reciprocal(A)	–A'right **downto** –A'left – 1
abs A	A'left + 1 **downto** A'right
– A	A'left + 1 **downto** A'right
Unsigned A/B	A'left – B'right **downto** A'right – B'left – 1
Unsigned A **mod** B	B'left **downto** Min(A'right, B'right)
Unsigned reciprocal(A)	–A'right + 1 **downto** –A'left

Figura 41: *Taula de mides de senyals en funció dels operadors. El valor a l'esquerra de la paraula «downto» es correspon amb els bits de la part entera, mentre que els de la dreta són els de la part decimal.*
Font: [7]

Per tant, hi haurà una utilització de recursos considerable, però gràcies a això no hi haurà pèrdua d'informació. No obstant, com el resultat final depèn en primer lloc del senyal dins del qual s'emmagatzema el valor de l'oscil·lador, òbviament aquest resultat vindrà en un senyal de més bits que el senyal (registre) al qual es vol emmagatzemar finalment. Per tant, una assignació directa no és possible, però la llibreria que proporciona els senyals i operadors de punt fixe també proporciona una funció que permet canviar la mida en bits d'un valor, amb la conseqüent pèrdua d'informació. Aquesta funció permet especificar factors com el comportament si el valor satura un cop adaptat al senyal desitjat, o si es vol fer servir arrodoniment o truncament de decimals. Això és molt útil ja que és una eina més per controlar el balanç precisió/utilització de recursos.

El fet de realitzar totes les operacions aritmètiques sense canviar la mida dels valors resultants, i realitzar finalment un únic canvi de mida és una decisió de disseny. La raó d'això és que si s'acumulen errors a les diferents operacions, ja sigui per arrodoniment o per saturació, és molt probable que el sistema no oscil·li correctament, especialment un cop implementada la xarxa sencera. El procediment utilitzat facilita trobar possibles problemes ja que només es produeix pèrdua d'informació en una assignació concreta, i per tant es pot controlar fàcilment.

A les simulacions es treballarà amb un nombre de bits elevat per tal d'assegurar que no hi hagin problemes per manca de precisió o un rang massa petit. A la

simulació, per exemple, el protocol descrit resulta en un valor final de 69 bits: 35 per la part entera, 1 bit de signe i 33 bits de la part decimal. A l'assignació final s'adapta el valor al senyal de 21 bits prèviament explicat.

Més endavant, abans de la implementació a la FPGA ja s'escurçaran els senyals si és convenient, per tal d'estalviar recursos. El motiu d'això és evitar problemes a les primeres simulacions per manca de precisió. Un cop el sistema simuli bé, es tornarà a simular, però fent servir senyals de menys bits. Això permetrà comprovar a partir de quines longituds de bits el sistema deixa de funcionar correctament.

3.5.5. Implementació de la xarxa neuronal a la FPGA

Un cop definit el disseny d'un oscil·lador únic, la xarxa es pot realitzar combinant-ne diferents blocs d'oscil·ladors idèntics i establint-ne les relacions necessàries. Per tal d'incloure-hi la contribució externa dels altres oscil·ladors, caldrà efectuar una petita modificació al disseny:

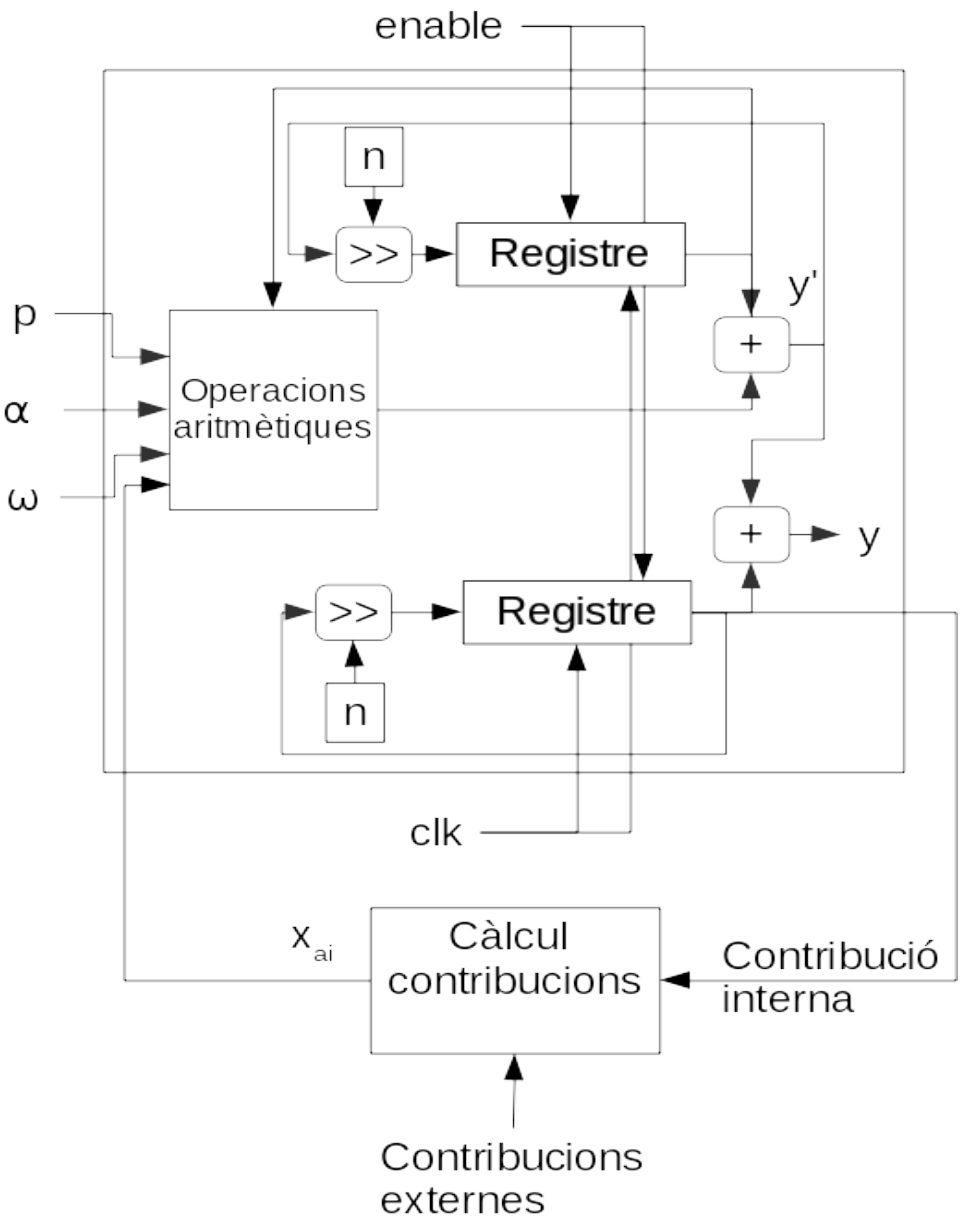

Figura 42: *Disseny de l'oscil·lador per a la xarxa neuronal.*

En el disseny de la figura anterior, el valor de *y*' està realimentat al bloc d'operacions, tal i com succeeix als dissenys anteriors. La diferència recau en què no hi ha realimentació interna del valor de *y* cap al bloc d'operacions. En comptes d'això, hi ha el valor x_{ai} esmentat anteriorment, on s'inclou la suma de les contribucions externes. Així doncs, serà necessari dissenyar un bloc extern encarregat del càlcul d'aquest valor. Abans d'això, però, es mostra el bloc d'operacions adaptat per fer servir aquest valor, seguint l'equació prèviament esmentada. L'únic canvi necessari és substituir el senyal *y* per x_{ai}.

$$\ddot{x}_i - \alpha\left(p^2 - x_{ai}^2\right)\dot{x}_i + \omega^2 x_{ai} = 0$$

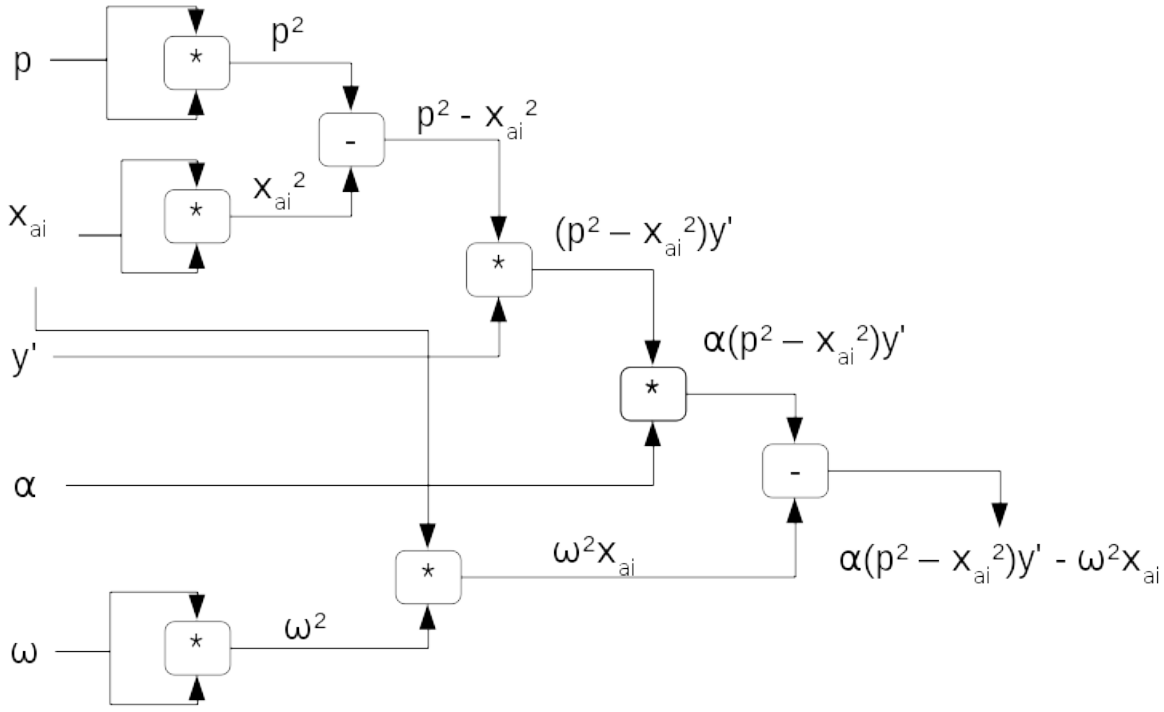

Figura 43: Diagrama del bloc d'operacions per a un oscil·lador integrat en una xarxa neuronal.

Finalment, la part que s'encarrega de calcular les contribucions dels oscil·ladors externs és la següent:

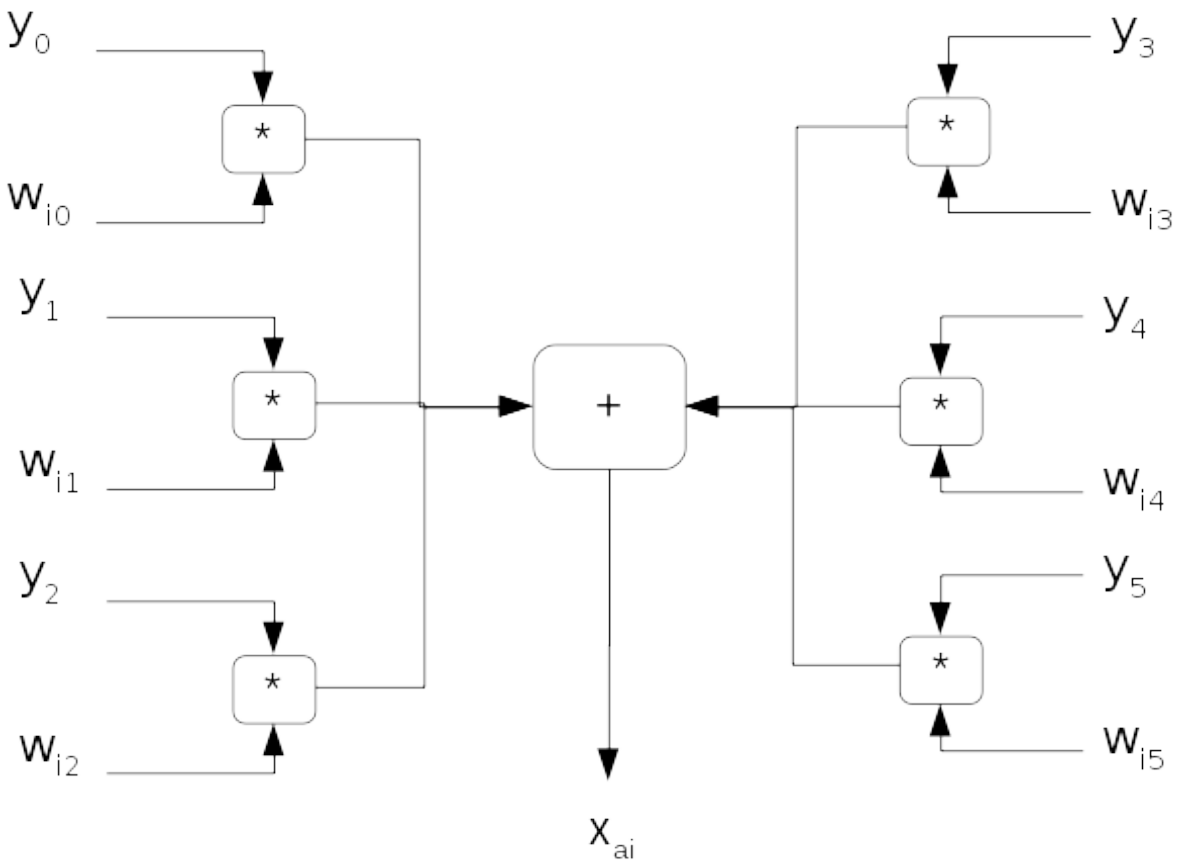

Figura 44: *Bloc de càlcul de contribucions.*

Es farà servir un bloc com aquest per a cada oscil·lador *i.* Els valors *w* són els diferents valors de la matriu de pesos o connexions esmentada anteriorment, sent w_{in} el pes de la influència de l'oscil·lador *n* a l'oscil·lador *i*.

Seguidament es simula el sistema dissenyat:

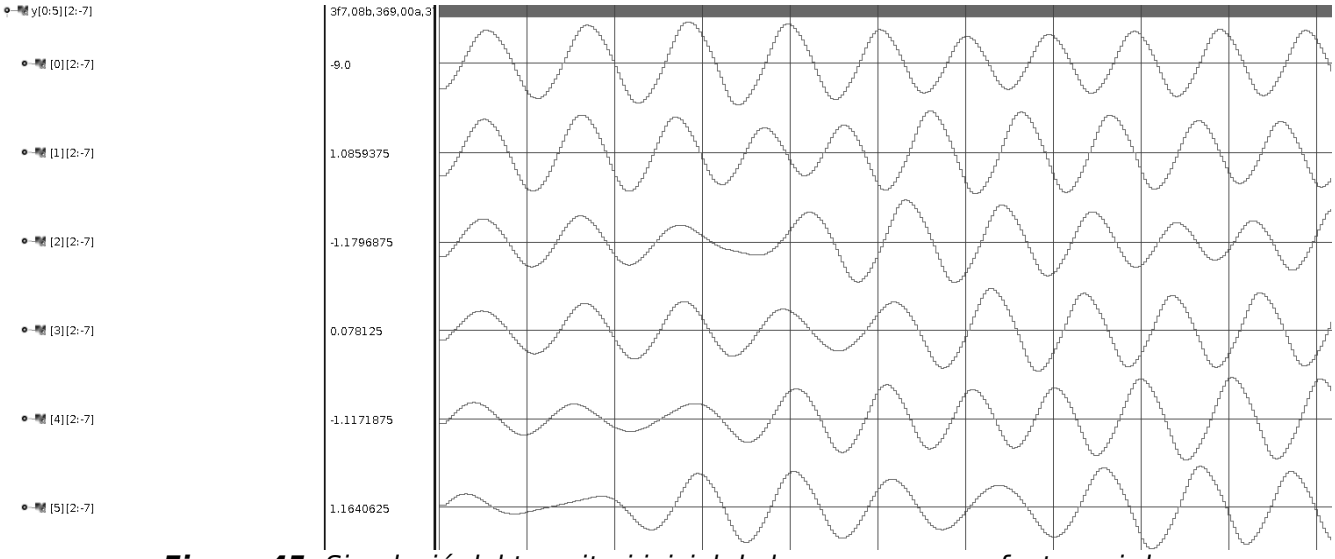

Figura 45: *Simulació del transitori inicial de la xarxa sencera fent servir la matriu de moviment lent.*

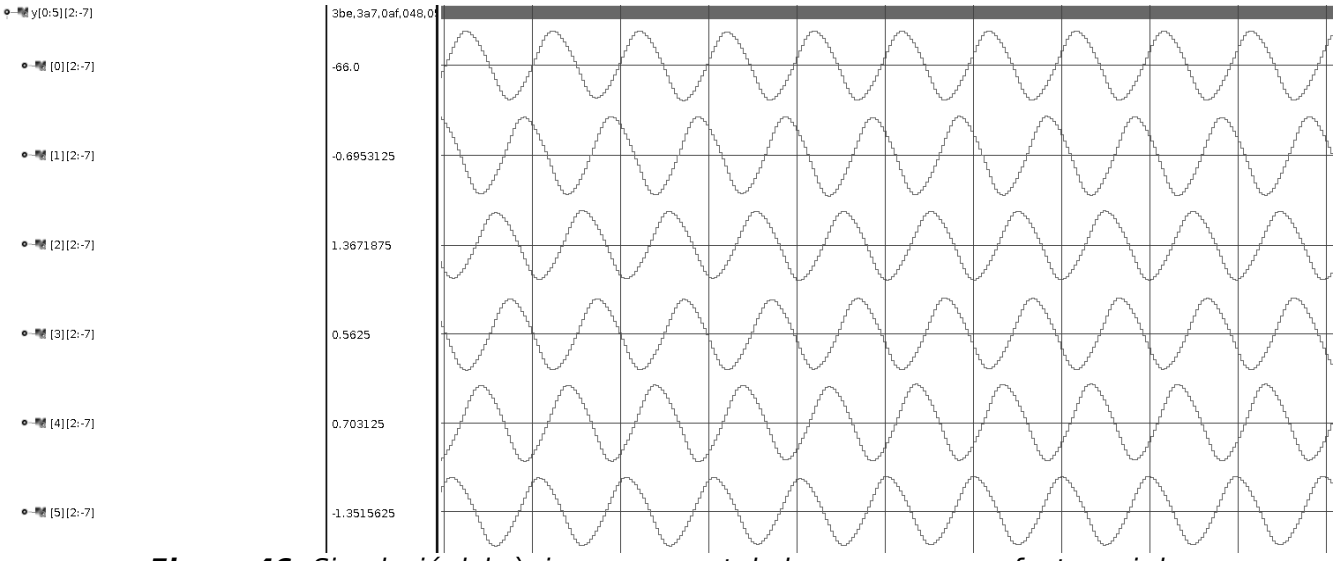

Figura 46: *Simulació del règim permanent de la xarxa sencera fent servir la matriu de moviment lent.*

A la simulació s'ha fet servir una longitud de 10 bits per l'estat de cada oscil·lador. S'observa que un cop establert el règim permanent, els desfasaments de les ones són els adequats.

En quant a la matriu de moviment mitjà, les simulacions són les següents.

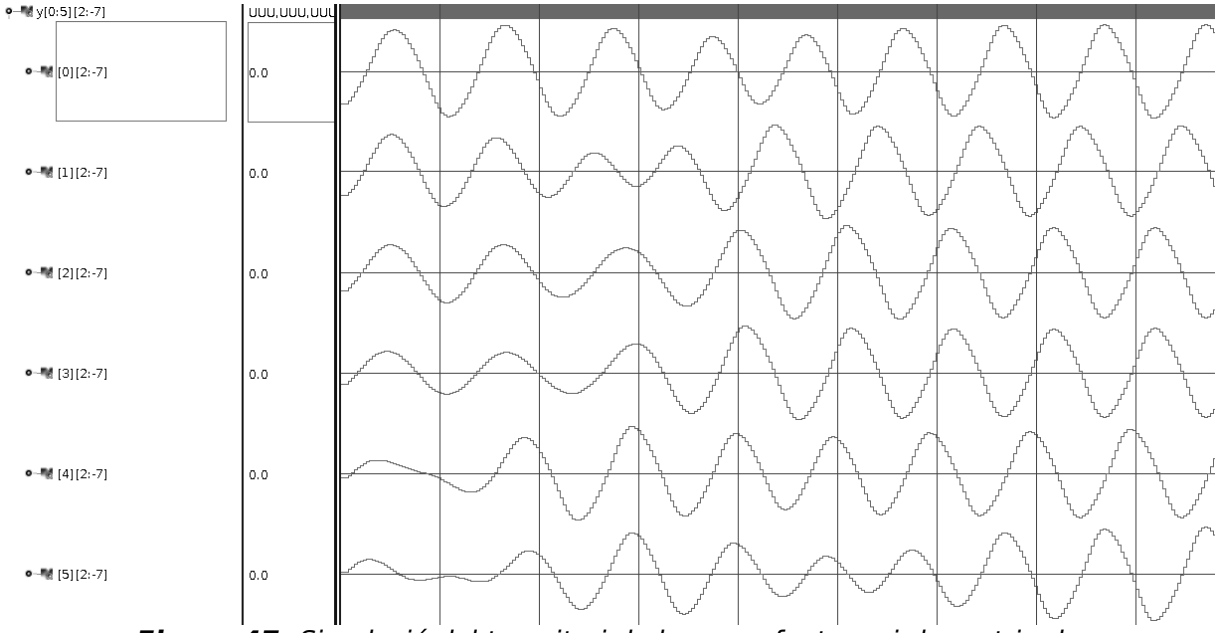

Figura 47: *Simulació del transitori de la xarxa fent servir la matriu de moviment mitjà.*

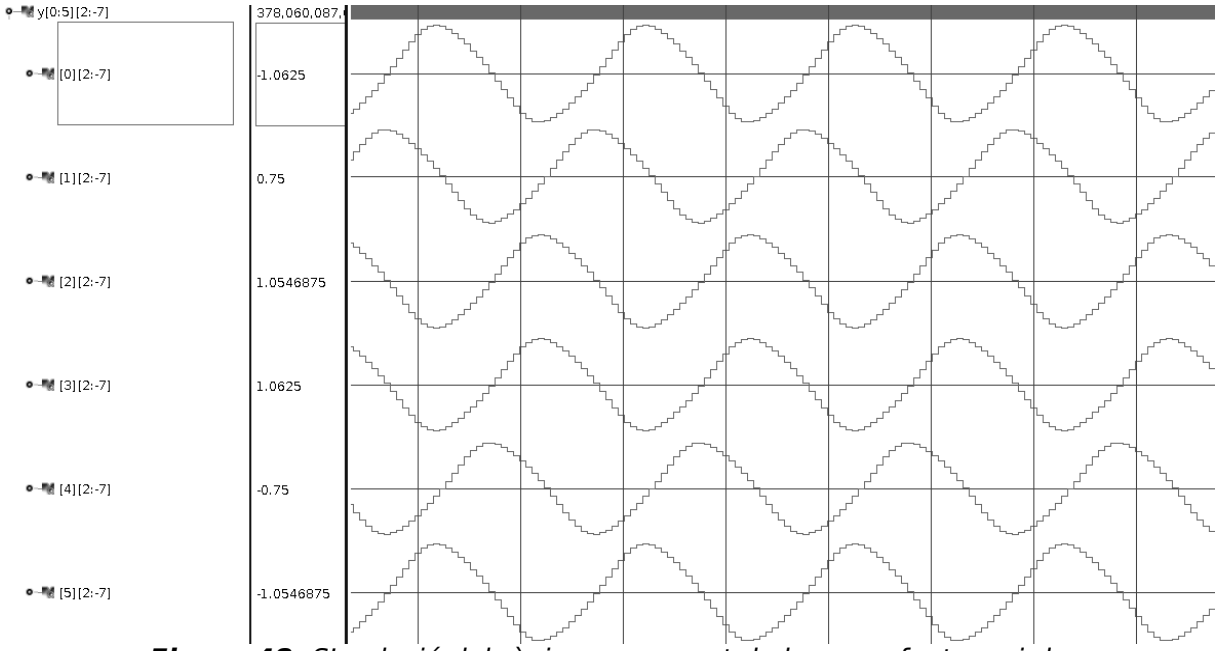

Figura 48: *SImulació del règim permanent de la xarxa fent servir la matriu de moviment mitjà.*

A la figura anterior s'hi aprecien 4 desfasaments diferents. Els oscil·ladors L1 i R3 ([0] i [5] a la simulació) tenen la mateixa fase. El L3 i R1 ([2] i [3]) comparteixen una altra fase, i el L2 i R2 ([1] i [4]) tenen desfasaments diferents. Per tant el comportament és l'esperat.

Seguidament es simula la xarxa fent servir la matriu de moviment ràpid.

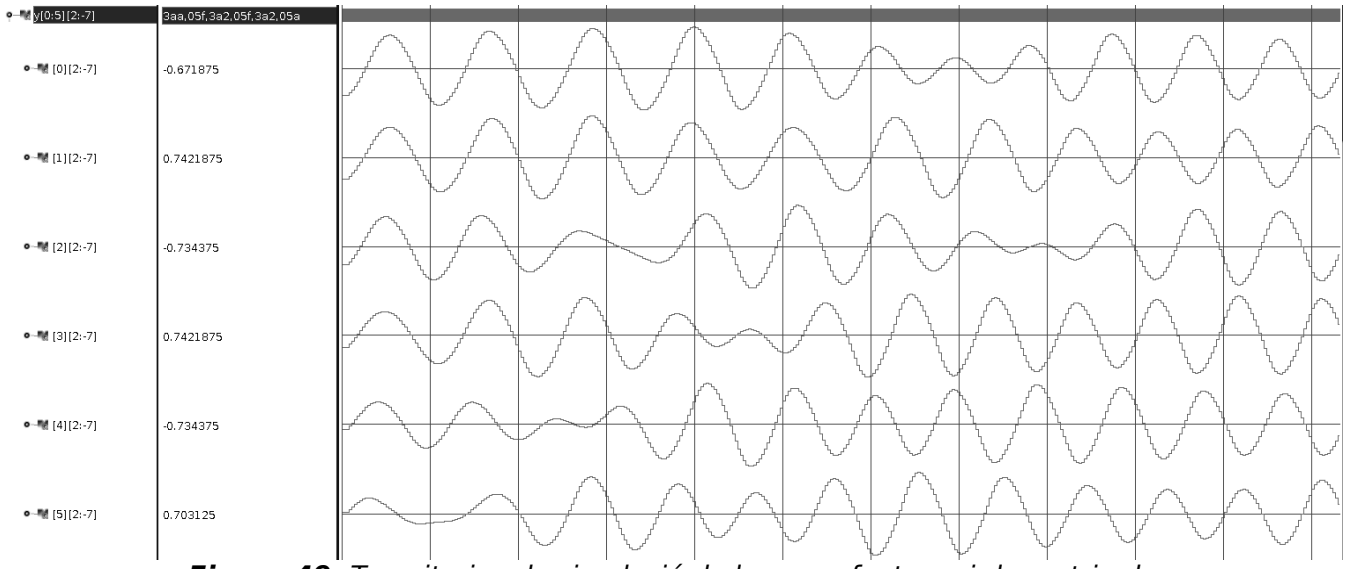

Figura 49: *Transitori en la simulació de la xarxa fent servir la matriu de moviment ràpid o trípode.*

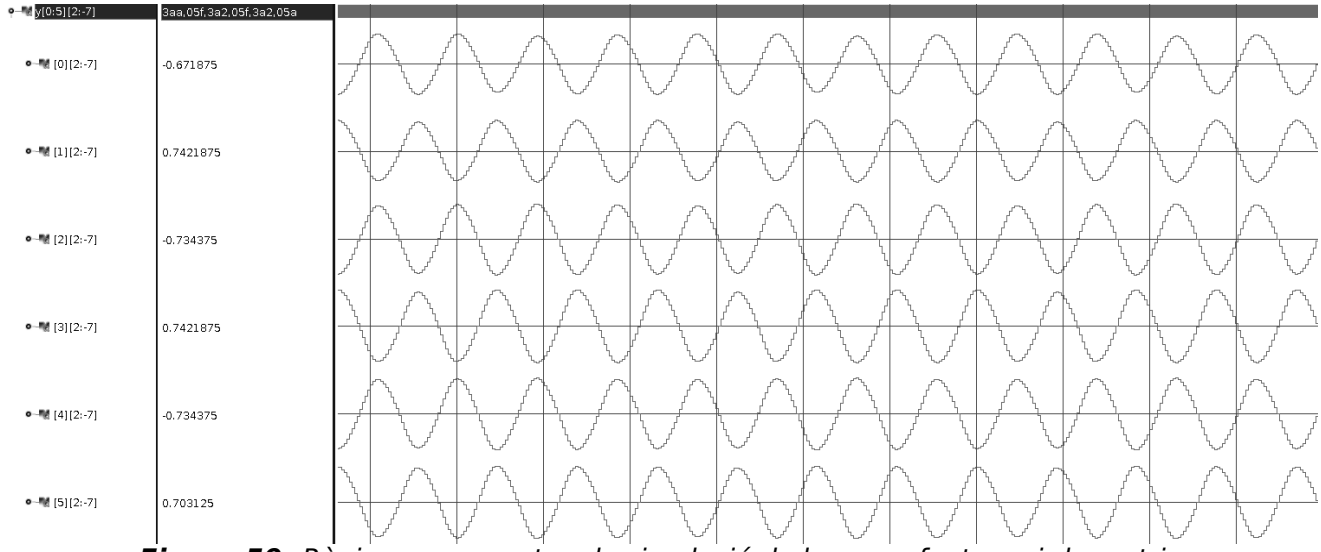

Figura 50: *Règim permanent en la simulació de la xarxa fent servir la matriu de moviment ràpid o trípode.*

Tal i com s'esperava amb la matriu de moviment trípode, totes les ones tenen un desfasament de 0º o 180º

El robot disposa de 12 servos: 6 horitzontals i 6 verticals. La xarxa neuronal controla 6 senyals, que s'enviaran a cadascun dels servos horitzontals. En quant al moviment vertical, per produir un moviment d'avenç en el robot és necessari que la pota estigui tocant a terra durant els pendents de pujada de l'ona sinusoïdal, mentre que durant les pendents de baixada la pota haurà d'estar aixecada per tal que el robot no torni sempre a la mateixa posició en cada oscil·lació.

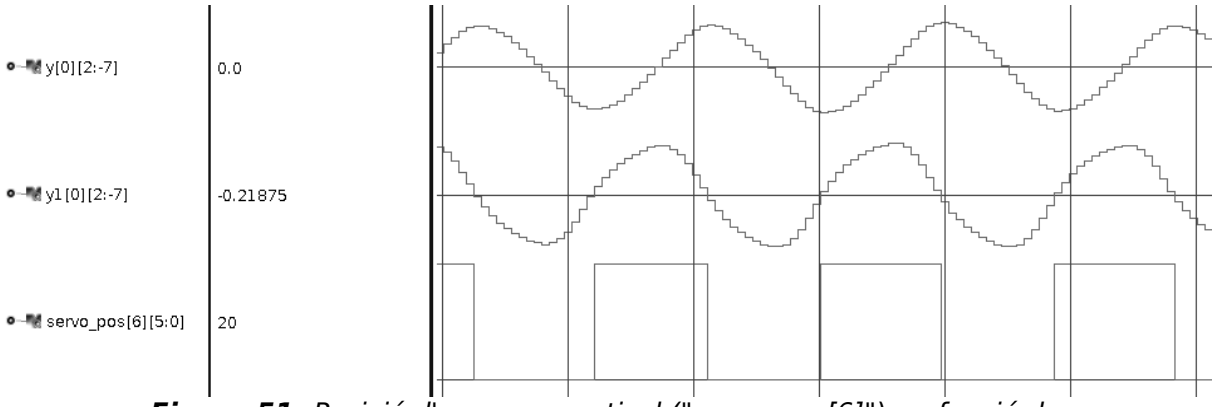

Figura 51: *Posició d'un servo vertical ("servo_pos[6]") en funció de l'ona del servo horitzontal.*

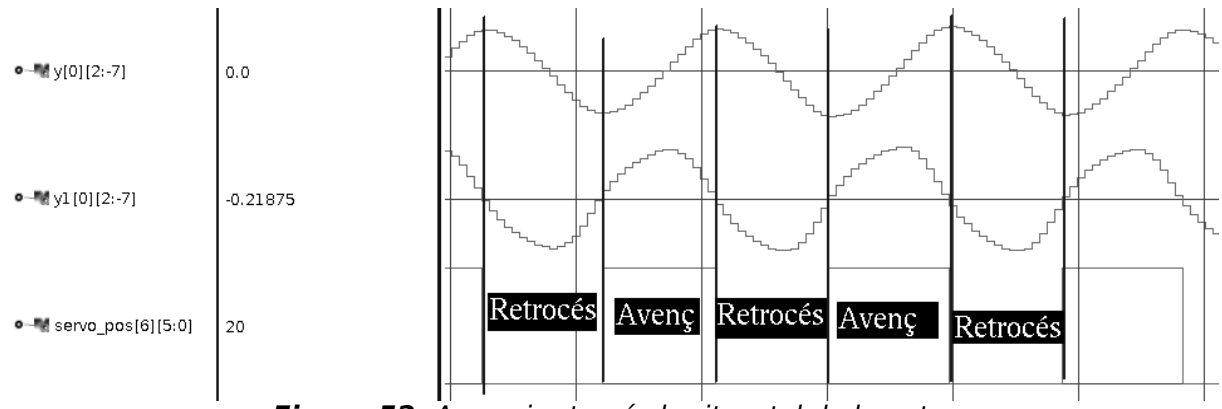

Figura 52: *Avenç i retrocés horitzontal de la pota.*

A les figures anteriors s'hi aprecien els valors de y, y' (y1) i el servo vertical de la pota (*servo_pos*). Es comprova que, tal i com s'ha explicat, el servo vertical puja i baixa en funció de si el valor del servo horitzontal (*y)* està pujant o baixant. Això s'aconsegueix observant l'ona de la derivada (*y1*), ja que aquesta té un valor positiu quan *y* puja i un valor negatiu quan *y* baixa. Per tal de detectar si el valor de *y1* és positiu o negatiu, s'observa el bit més significatiu (*MSB*) de *y1,* ja que aquest és el que emmagatzema el signe. Si el seu valor és 1 en un instant donat vol dir que *y1* és negatiu, mentre que si val 0 vol dir que y1 és positiu. El fet de comprovar el bit directament en comptes de comprovar si el valor de *y1* és més gran o més petit que 0 és una manera d'estalviar recursos, ja que les operacions i comprovacions directes de bits requereixen pocs recursos.

3.5.6. Optimització del disseny

El disseny explicat té l'inconvenient que requereix molts recursos, degut a la gran quantitat d'operacions que es realitzen concurrentment. Concretament, les operacions més costoses que es fan servir són les multiplicacions, i a cada oscil·lador n'hi ha varies amb longituds de bits relativament grans. A més, per cadascun dels 6 oscil·ladors hi ha un mòdul de càlcul de les contribucions com el de la figura 44 , resultant en un total de 6 * 6 = 36 multiplicacions més. Tot plegat resulta en un disseny que probablement no hi cabrà en FPGAs més petites.

Així doncs, és interessant optimitzar el disseny per tal de reduir l'ús de recursos. Òbviament la opció més interessant és reduir el nombre d'operacions del disseny. Això es pot aconseguir si en comptes de realitzar totes les operacions de manera paral·lela, s'efectuen les que consumeixin més recursos en sèrie, utilitzant un únic bloc operador per a totes les operacions del mateix tipus. Això és factible ja que la velocitat d'actualització dels valors requerida en el robot (pocs Hz) és diversos ordres més petita que la freqüència de la FPGA (de l'odre de desenes de MHz), i per tant hi ha marge de sobra per realitzar aquesta serialització.

A la figura 42 s'observa que hi ha un bloc sencer d'operacions aritmètiques per a cadascun dels 6 oscil·ladors que conformen la xarxa. Una manera d'aconseguir un estalvi significatiu de recursos és treure aquest bloc a fora del bloc de l'oscil·lador, i fer servir el mateix bloc per als 6 oscil·ladors mitjançant un multiplexor.

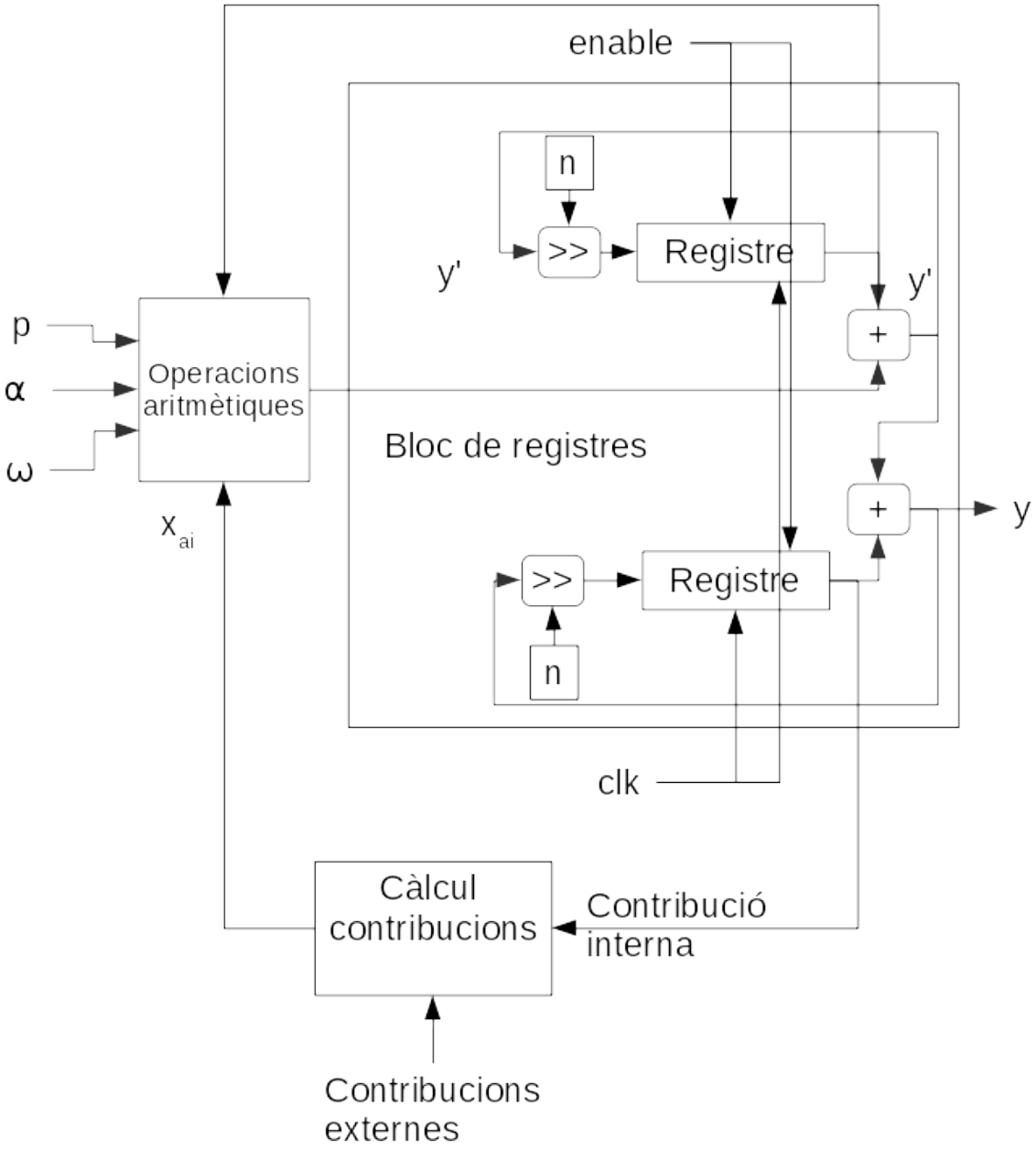

Figura 53: *Externalització del bloc de les operacions aritmètiques.*

A la figura 53 hi ha 3 blocs diferenciats: el bloc que abans era l'oscil·lador sencer ha passat a dividir-se en un bloc de registres i en el bloc d'operacions. El bloc de càlcul de contribucions és el que es correspon amb el descrit a la figura 44. Llavors, per fer servir el mateix bloc d'operacions per als 6 oscil·ladors s'introdueixen multiplexors.

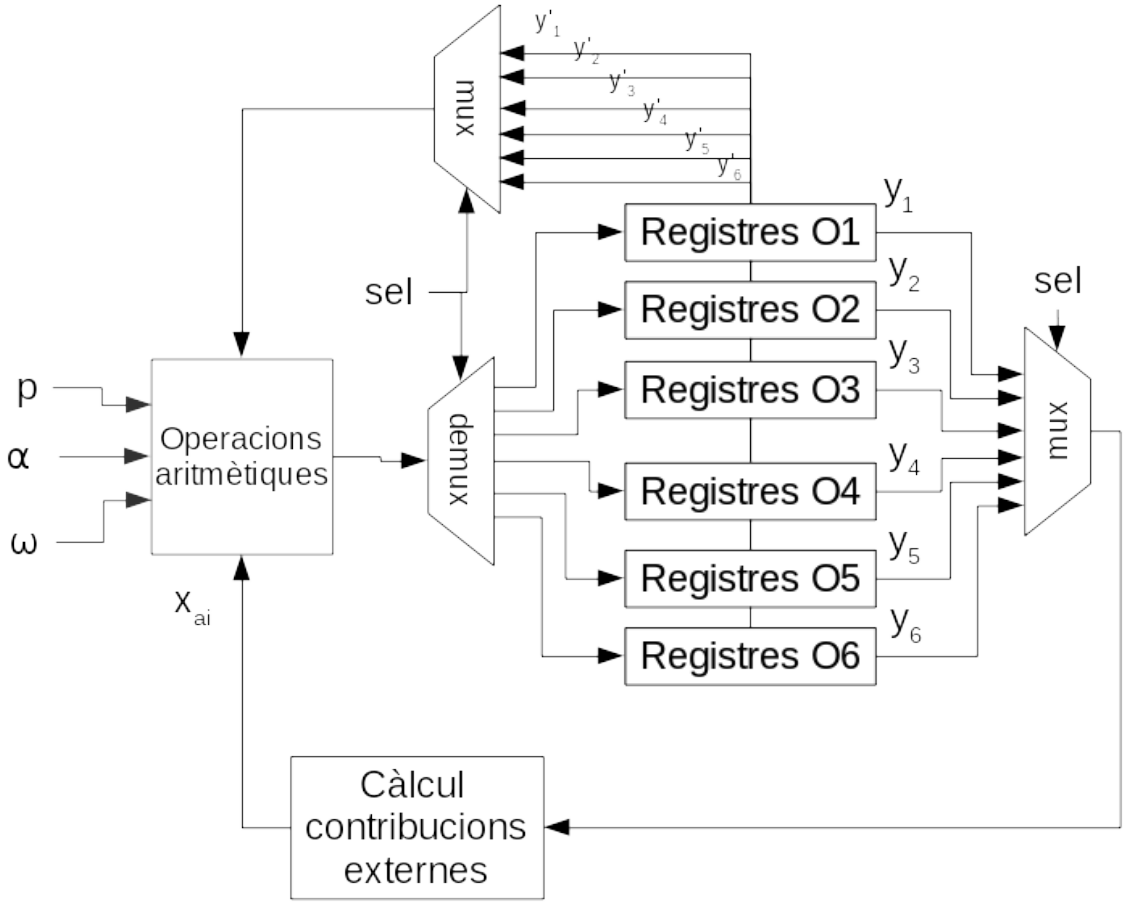

Figura 54: *Disseny amb les operacions serialitzades.*

D'aquesta manera, es fa servir un bloc d'operacions i un bloc de càlcul de contribucions externes en comptes de 6, amb un estalvi de recursos considerable. Per actualitzar els registres cal seleccionar el registre corresponent als mutiplexors/demultiplexors i esperar un cicle de rellotge per tal que s'actualitzin.

En general, la serialització de les operacions que requereixen molts recursos suposa un augment de la complexitat del disseny, però també suposa un estalvi de recursos considerable si es realitza correctament. El disseny encara es podria optimitzar molt més, de manera que, per exemple, un únic bloc s'encarregui de realitzar totes les multiplicacions del disseny. No obstant això complicaria el disseny molt més. Cal notar que si es serialitzen massa operacions es pot arribar a un punt en el qual el disseny sigui massa lent per a l'aplicació desitjada. Això depèn de la freqüència de la FPGA i de la quantitat d'operacions que s'hagin de realitzar.

Degut a problemes amb el disseny optimitzat però, es farà servir el disseny paral·lel descrit anteriorment, ja que aquest sí que funciona correctament. A causa d'això s'haurà de fer servir la placa *Nexys 2 Spartan-3E* en comptes de la *Cmod S6*, que hauria resultat més convenient per facilitat d'incorporació al robot.

3.6. Comunicació amb el robot

3.6.1. Component SPI per a la FPGA

Per tal d'efectuar la comunicació amb el robot, tal i com s'ha explicat anteriorment, és necessari un disseny que implementi el funcionament d'un mestre SPI a la FPGA.

S'ha decidit aprofitar un disseny *VHDL* disponible a internet, ja que dissenyar i comprovar aquest component des de 0 requeriria bastant de temps, i a més aquest no és el tema d'aquest projecte.

Per tal de fer-lo servir cal estudiar la documentació proporcionada, on s'esmenta el funcionament dels diferents senyals de entrada i sortida implicats. A continuació es descriuen els punts més importants.

Es tracta d'un disseny síncron, ja que requereix un senyal de rellotge. Com a entrades té els senyals típics de rellotge, reset i enable, a més de senyals com ara *cpha i cpol* per tal de seleccionar el mode SPI que es vol fer servir. La selecció de l'esclau es realitza amb el senyal *addr*. Amb el senyal *clk_div*es pot controlar el prescaler intern, per tal de seleccionar una freqüència de rellotge SPI més petita que el rellotge de la FPGA. La freqüència resultant ve donada per la següent fòrmula:

$$f_{SCLK} = \frac{f_{clock}}{2 \cdot clk\_div} \quad (1)$$

Les dades a enviar s'emmagatzemen al senyal *tx_data*, mentre que les dades obtingudes van a parar al senyal *rx_data.* A més disposa d'un senyal *busy* que indica quan s'està efectuant la transmissió de dades, molt útil per evitar sobreescriure el registre *tx_data* durant la transmissió.

També incorpora com a sortida tots els senyals necessaris típics del SPI: *sclk* (rellotge spi), els senyals *mosi i miso* (entrada i sortida sèrie) i els selectors d'esclau individuals (*ss_n(l)*). Tot això es pot apreciar a la següent figura.

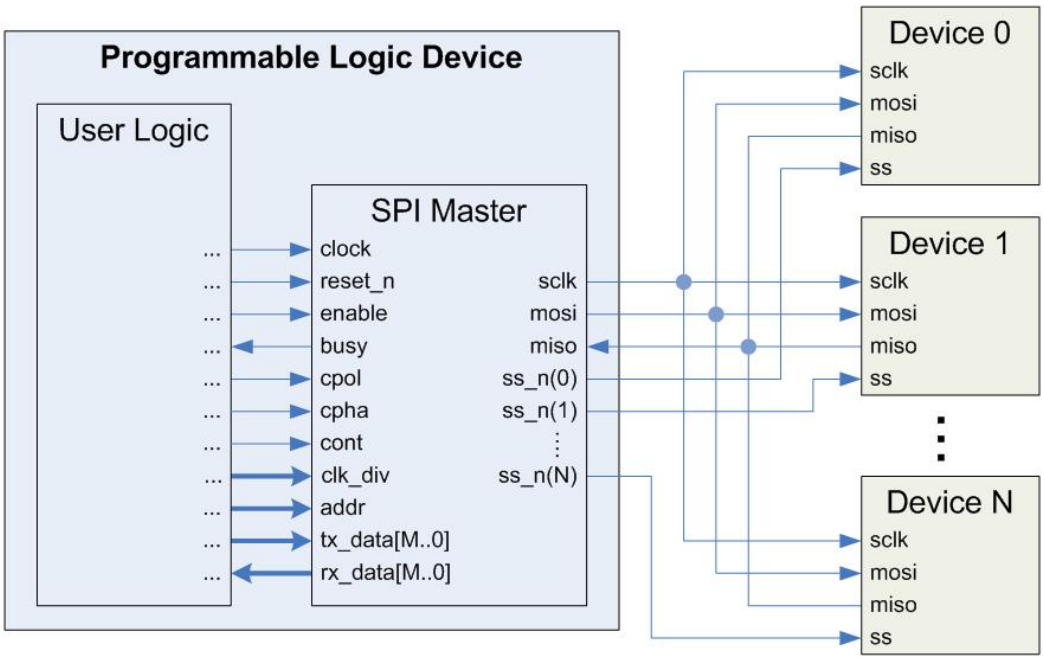

Figura 55: *Exemple d'aplicació del mòdul SPI.*
Font: [11]

Taula 3: Senyals del component SPI (en anglés).
Font: [11]

Port	Width	Mode	Data Type	Interface	Description
clock	1	in	standard logic	user logic	System clock.
reset_n	1	in	standard logic	user logic	Asynchronous active low reset.
enable	1	in	standard logic	user logic	H: latches in settings, address, and data to initiate a transaction, L: no transaction is initiated.
cpol	1	in	standard logic	user logic	SPI clock polarity setting.
cpha	1	in	standard logic	user logic	SPI clock phase setting.
cont	1	in	standard logic	user logic	Continuous mode flag.
clk_div	32	in	integer	user logic	Speed setting. The integer input is the number of system clocks per 1/2 period of sclk.
addr	32	in	integer	user logic	Address of target slave. The slaves are assigned addresses starting with 0.
tx_data	M*	in	standard logic vector	user logic	Data to transmit.
miso	1	in	standard logic	slave devices	Master in, slave out data line.
sclk	1	buffer	standard logic	slave devices	SPI clock.
ss_n	N^	buffer	standard logic vector	slave devices	Slave select signals.
mosi	1	out	standard logic	slave devices	Master out, slave in data line.
busy	1	out	standard logic	user logic	Busy / data ready signal.
rx_data	M*	out	standard logic vector	user logic	Data received from target slave.

Un cop implementat al disseny es realitza una simulació per comprovar que el seu comportament és tal i com el descrit a la documentació.

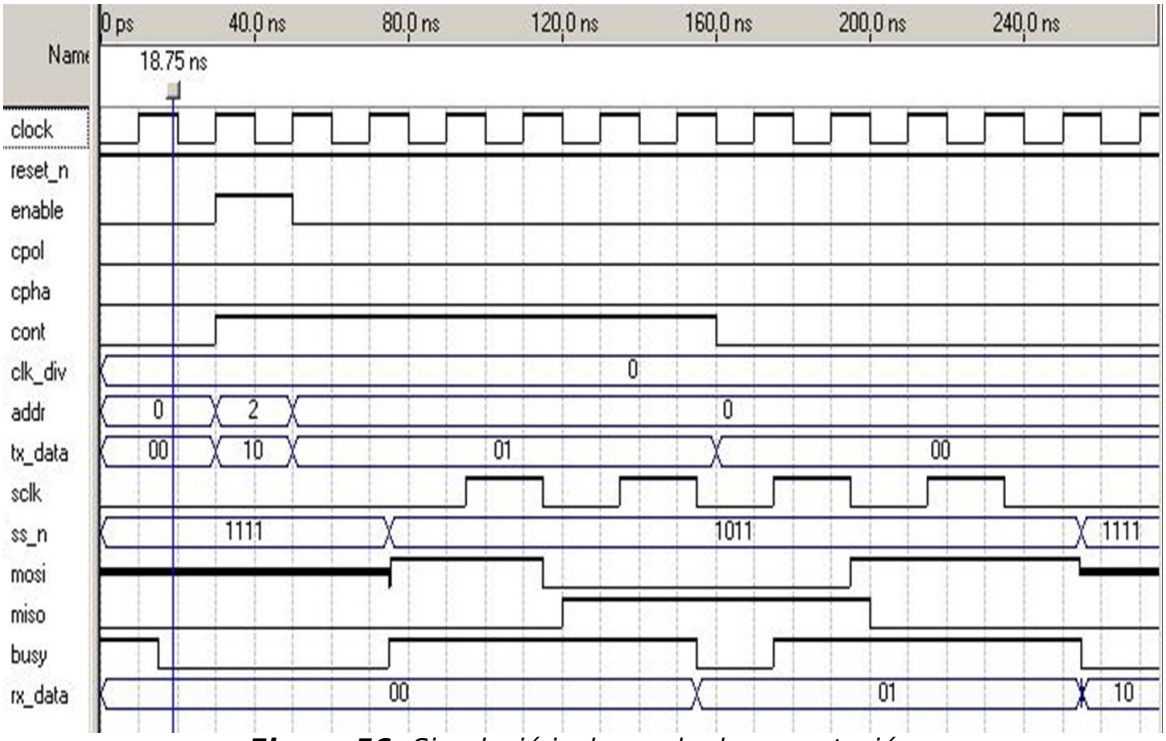

Figura 56: *Simulació inclosa a la documentació.*
Font: [11]

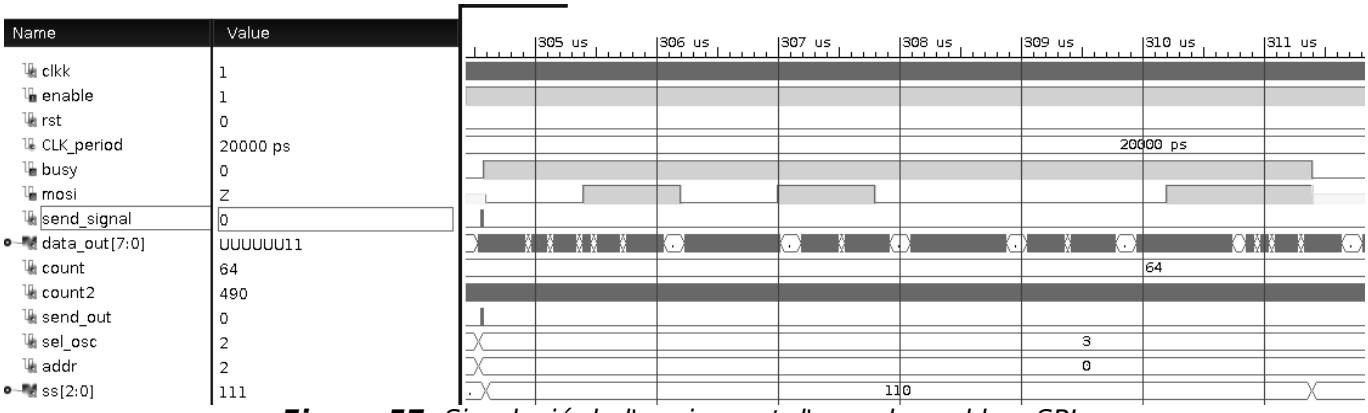

Figura 57: *Simulació de l'enviament d'un valor pel bus SPI.*

Es pot observar que el comportament de la figura 57 es correspon amb el comportament de la figura 56: quan s'activa el senyal d'enable (*send_signal* en aquest cas) el senyal *mosi* comença a transmetre les dades emmagatzemades prèviament al registre *tx_data* (connectat a *data_out* en aquest cas). Noti's que el registre data_out segueix canviant un cop començat el procès, degut a que a la simulació es fa servir una freqüència molt alta d'oscil·lació. No obstant, això no afecta a la transmissió que s'està efectuant ja que la càrrega de dades al mòdul SPI només s'efectua en el flanc de pujada del senyal d'activació (*send_signal).* També s'aprecia que el senyal *busy* es queda a nivell alt mentre s'efectua la transmissió, i un cop acabada torna a nivell baix i el senyal *mosi* torna a un estat d'alta impedància.

3.6.2. Ordre d'enviament dels senyals

Degut a que el robot hexàpode que es vol fer servir té 12 servos (2 per cada pota), és necessari l'enviament de 12 posicions pel bus SPI, fent servir el protocol explicat anteriorment. Per tal de fer servir només un component SPI, es multiplexen els valors a enviar i es canvien els paràmetres (adreça de l'esclau, adreça de la pota) per a cada cas. A més, cal tenir en compte que les plaques *Arduino* del robot no realitzen l'actualització de la posició d'un servo instantàniament, i que el propi servo també requereix un cert temps per fer-ho. Per tant, caldrà introduir una espera al sistema entre cada enviament de dades. El sistema d'enviament de dades a utilitzar és el següent:

- Quan s'activa el primer senyal d'enviament es comença a enviar la posició del primer servo, i mentrestant un comptador produeix una espera determinada.
- Un cop el comptador arriba al seu límit, es comença a enviar la posició del següent oscil·lador, i es torna a esperar fins que el comptador arribi al límit. Això succeeix per a cadascun dels 12 servos.
- Un cop enviades les posicions dels 12 servos, no es tornarà a començar el cicle d'enviament fins que un segon comptador, que ha començat a comptar amb el primer senyal d'enviament, hagi arribat al seu límit, que serà molt més elevat que el limit entre senyals.

Per tant, hi entren en joc dos comptadors: un s'encarrega de l'espera entre l'enviament dels senyals dels diferents servos i l'altre s'encarrega de l'espera entre tandes d'enviament de tots els senyals.

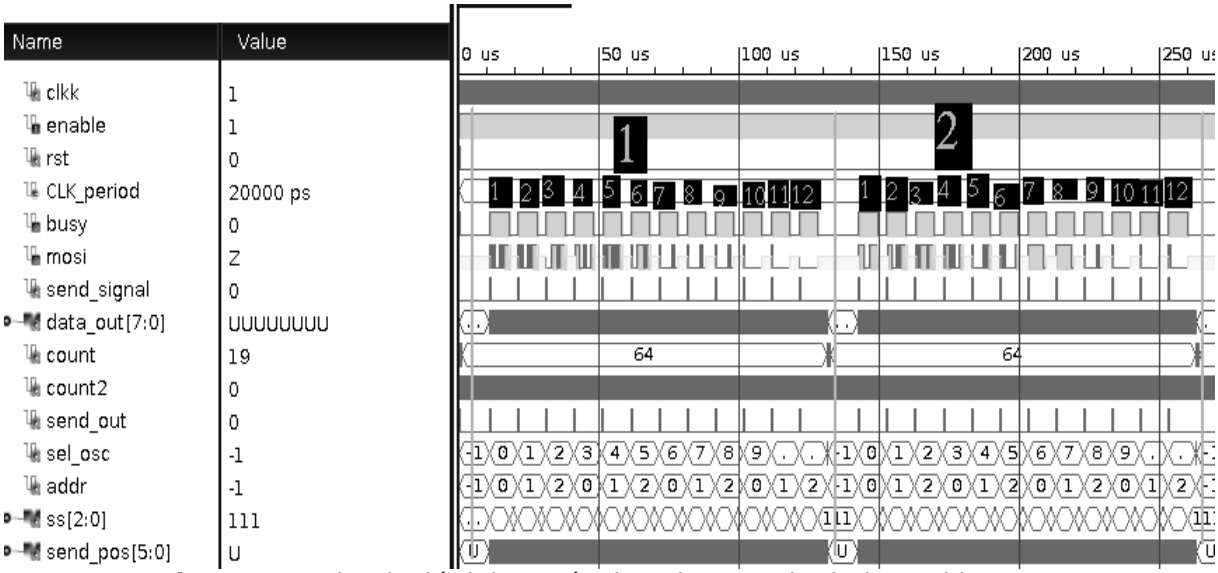

Figura 58: *Simulació del procés d'enviament de dades pel bus SPI.*

A la figura anterior s'hi aprecien dues tandes d'enviament de dades, i en cadascuna d'elles l'enviament, d'una manera seguida, de les posicions dels 12 servos. Cal notar que els temps que es poden apreciar a aquesta simulació no són els temps reals que es faran servir al robot. A les simulacions s'han fet servir unes esperes molt més petites per tal que:

a) la simulació no trigui massa temps

b) es pugin apreciar les diferents esperes amb un mateix nivell d'ampliació de la figura.

Al robot es faran servir uns temps d'espera tals que l'ordre de magnitud de l'espera entre tandes d'enviament sigui major que l'ordre de magnitud de l'espera entre els diferents servos. D'aquesta manera, en el resultat final donarà la impressió que tots els servos s'actualitzen a la vegada a intervals regulars.

A la simulació de la figura anterior aquests comptadors tenen el nom de *count* i *count2*. També s'hi aprecien els senyals que seleccionen quin senyal toca enviar a cada moment (*sel_osc)* i a quin dels 3 mòduls del robot pertany el servo corresponent (*addr)*. Quan no s'està enviant cap dada aquests dos senyals es queden a -1.

3.6.3. Connexió al robot

Finalment, l'ultim pas per comunicar la *FPGA* amb el robot és la connexió del bus SPI entre la *FPGA* i el robot.

Per efectuar la connexió caldrà tenir presents els següents punts:

- Col·locació dels diferents senyals al bus del robot.
- Nivells de voltatge del robot.
- Nivells de voltatge de la FPGA.
- En cas que sigui necessari un circuit d'adaptació, alimentació i característiques del circuit.

La placa que es farà servir per efectuar les proves és la *Nexys 2 Spartan-3E*, a la qual s'hi troben la pròpia *FPGA* i diversos circuits d'adaptació i regulació. La FPGA treballa amb els nivells de voltatges *CMOS*, és a dir, amb un voltatge de nivell alt de 3.3 V. Els tres *Arduino Mini* que hi ha connectats al bus SPI, però, funcionen amb nivells de voltatge TTL, amb un voltatge de nivell alt de 5 V. El microcontrolador que fa servir és l'*ATmega328* d'*Atmel*, i per tant és necessari tenir en compte les especificacions d'aquest component.

Symbol	Parameter	Condition		Min.	Typ.	Max.	Units
V_{IL}	Input Low Voltage, except XTAL1 and $\overline{\text{RESET}}$ pin	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V		-0.5 -0.5		$0.2V_{CC}$ (1) $0.3V_{CC}$ (1)	V
V_{IH}	Input High Voltage, except XTAL1 and $\overline{\text{RESET}}$ pins	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V		$0.7V_{CC}$ (2) $0.6V_{CC}$ (2)		V_{CC} + 0.5 V_{CC} + 0.5	V
V_{IL1}	Input Low Voltage, XTAL1 pin	V_{CC} = 1.8V - 5.5V		-0.5		$0.1V_{CC}$ (1)	V
V_{IH1}	Input High Voltage, XTAL1 pin	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V		$0.8V_{CC}$ (2) $0.7V_{CC}$ (2)		V_{CC} + 0.5 V_{CC} + 0.5	V
V_{IL2}	Input Low Voltage, $\overline{\text{RESET}}$ pin	V_{CC} = 1.8V - 5.5V		-0.5		$0.1V_{CC}$ (1)	V
V_{IH2}	Input High Voltage, $\overline{\text{RESET}}$ pin	V_{CC} = 1.8V - 5.5V		$0.9V_{CC}$ (2)		V_{CC} + 0.5	V
V_{IL3}	Input Low Voltage, $\overline{\text{RESET}}$ pin as I/O	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V		-0.5 -0.5		$0.2V_{CC}$ (1) $0.3V_{CC}$ (1)	V
V_{IH3}	Input High Voltage, $\overline{\text{RESET}}$ pin as I/O	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V		$0.7V_{CC}$ (2) $0.6V_{CC}$ (2)		V_{CC} + 0.5 V_{CC} + 0.5	V
		I_{OL} = 20mA, V_{CC} = 5V	T_A=85°C			0.9	
			T_A=105°C			1.0	

Figura 59: *Extracte del full d'especificacions de l'Atmel ATmega328 dels Arduinos.*
Font: [1]

Observant la figura anterior, el voltatge mínim que es considera com a nivell alt és el corresponent a $0{,}6V_{cc,}$ que són 3 V amb V_{cc} = 5 V. Tenint en compte que la comunicació s'efectua només en sentit *FPGA → Arduino*, teòricament es podria connectar les sortides de la FPGA directament al bus SPI, ja que en teoria hi hauria un marge de 300 mV. No obstant, aquest marge es considera insuficient, especialment a freqüències altes, i per tant s'opta per introduir-hi un circuit buffer entre els dos elements, per tal de proporcionar un marge de soroll més elevat i assegurar que el soroll no pugui provocar errors de comunicació. Això és d'especial importància ja que la transmissió incorrecta d'un valor de posició per als servos podria implicar que el robot perdés l'estabilitat i caigués, provocant danys estructurals en aquest.

El circuit integrat a utilitzar escollit és el *74HCT241N*, que pertany a una família lògica anomenada *IC06 74HC/HCT/HCU/HCMOS.* Tot i que hi han molts circuits integrats que servirien per a aquesta aplicació, s'ha escollit aquest per motius de disponibilitat a les tendes electròniques de la zona. Les especificacions elèctriques rellevants de la família són les següents:

DC CHARACTERISTICS FOR 74HC

Voltages are referenced to GND (ground = 0 V)

SYMBOL	PARAMETER	T_{amb} (°C)							UNIT	TEST CONDITIONS		
		74HC										
		+25			−40 to +85		−40 to +125			V_{CC} (V)	V_I	OTHER
		min.	typ.	max.	min.	max.	min.	max.				
V_{IH}	HIGH level input voltage	1.5 3.15 4.2	1.2 2.4 3.2		1.5 3.15 4.2		1.5 3.15 4.2		V	2.0 4.5 6.0		
V_{IL}	LOW level input voltage		0.8 2.1 2.8	0.5 1.35 1.8		0.5 1.35 1.8		0.5 1.35 1.8	V	2.0 4.5 6.0		
V_{OH}	HIGH level output voltage all outputs	1.9 4.4 5.9	2.0 4.5 6.0		1.9 4.4 5.9		1.9 4.4 5.9		V	2.0 4.5 6.0	V_{IH} or V_{IL}	$-I_O$ = 20 µA $-I_O$ = 20 µA $-I_O$ = 20 µA
V_{OH}	HIGH level output voltage standard outputs	3.98 5.48	4.32 5.81		3.84 5.34		3.7 5.2		V	4.5 6.0	V_{IH} or V_{IL}	$-I_O$ = 4.0 mA $-I_O$ = 5.2 mA
V_{OH}	HIGH level output voltage bus driver outputs	3.98 5.48	4.32 5.81		3.84 5.34		3.7 5.2		V	4.5 6.0	V_{IH} or V_{IL}	$-I_O$ = 6.0 mA $-I_O$ = 7.8 mA
V_{OL}	LOW level output voltage all outputs		0 0 0	0.1 0.1 0.1		0.1 0.1 0.1		0.1 0.1 0.1	V	2.0 4.5 6.0	V_{IH} or V_{IL}	I_O = 20 µA I_O = 20 µA I_O = 20 µA

Figura 60: *Extracte del full d'especificacions de la família 74HC.*
Font: [9]

S'observa que el voltatge d'entrada mínim que es considera com a nivell alt té un valor típic de 2.4 V amb V_{cc} = 4.5 V. Per tant, es disposa d'un marge de soroll de 3.3 V – 2.4 V = 0.9 V, el triple que en el cas anterior. En quant a la sortida, a 4.5 V el circuit pot proporcionar un voltatge a les sortides de 4.4 V. Per tant s'estima que amb V_{cc} = 5 V el voltatge a la sortida serà com a mínim d'uns 4.9 V.

Per realitzar les proves s'alimenta el buffer aprofitant la sortida de +5 V de la *Nexys 2 Spartan-3E*, que està alimentada pel USB de l'ordinador.

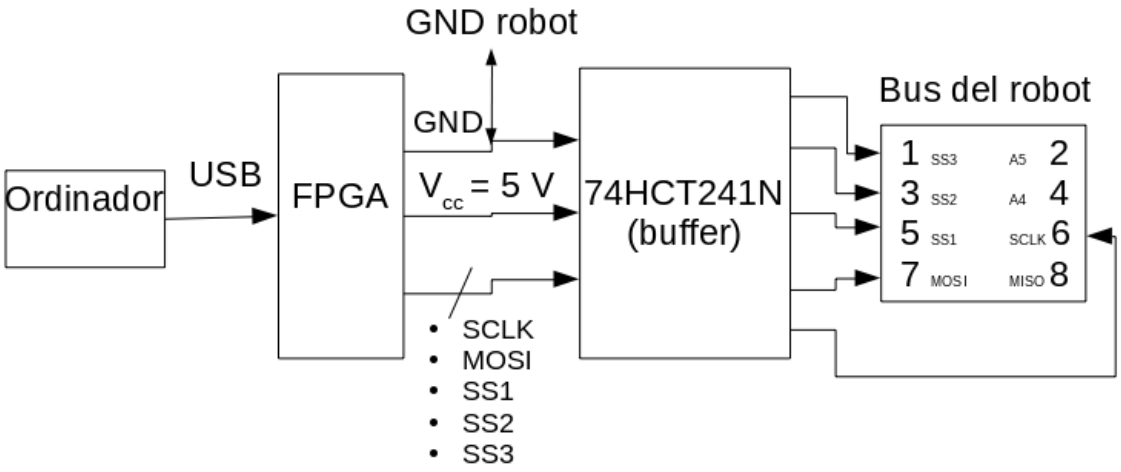

Figura 61: *Diagrama de connexions per efectuar les proves.*

CAPÍTOL 4: PROVES EXPERIMENTALS I RESULTATS AL ROBOT

Paral·lelament amb el disseny en VHDL de la xarxa es realitzen proves al laboratori de l'escola per tal de comprovar si les diferents parts del sistema funcionen correctament.

El primer element que es prova és el disseny SPI. En comptes de fer servir el robot per realitzar les primeres comprovacions, s'utilitza un circuit integrat *MCP4251*, un potenciòmetre digital de la marca *Microchip* amb una interfície SPI.

El procediment utilitzat és l'enviament de diferents valors pel bus SPI per tal de modificar la resistència d'un dels 2 potenciòmetres disponibles. Les dades enviades han de seguir el format indicat al full d'especificacions del circuit integrat.

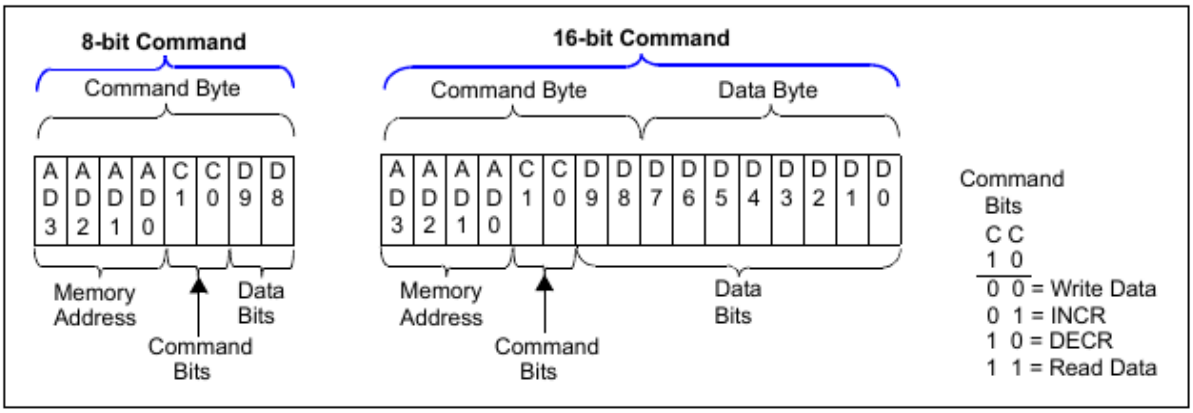

Figura 62: Format d'enviament d'ordres al potenciòmetre.
Font: [8]

C1:C0 Bit States	Command	# of Bits
11	Read Data	16-Bits
00	Write Data	16-Bits
01	Increment	8-Bits
10	Decrement	8-Bits

Figura 63: *Ordres de configuració del potenciòmetre disponibles.*
Font: [8]

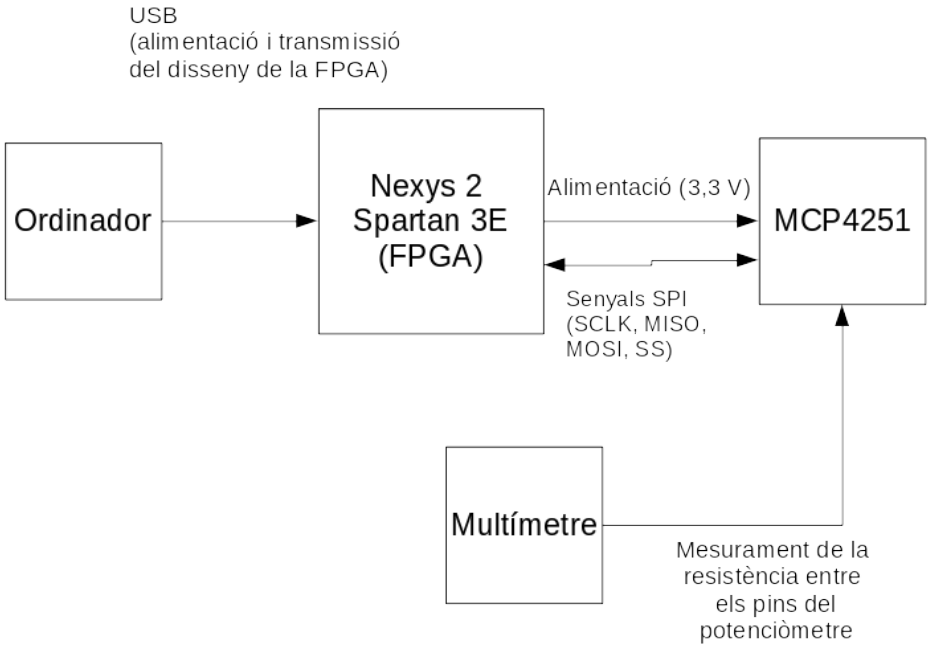

Figura 64: *Estructura per comprovar el correcte funcionament del SPI.*

Comprovat el correcte funcionament del SPI, es comencen les proves amb el robot. En primer lloc s'ha intentat moure un únic servo d'una pota, i després s'ha introduït un oscil·lador únic per tal de controlar les oscil·lacions d'una única pota. Seguidament s'ha comprovat el funcionament del sistema dissenyat per enviar les posicions dels 12 servos simultàniament, i finalment, s'ha introduït la xarxa neuronal. S'observa que, amb els paràmetres inicials, la coordinació entre les potes (desfasament de les ones) és l'adequada, però el moviment general resulta massa lent. Llavors s'ha intentat en primer lloc augmentar la velocitat d'actualització dels oscil·ladors sense canviar els paràmetres d'oscil·lació, modificant a quina freqüència s'envien els senyals d'activació d'aquests. No obstant, degut a les característiques dels servos i els microcontroladors, s'observa que si s'actualitzen massa ràpid el comportament no és l'adequat,

degut a que no tenen suficient marge de temps per efectuar tots els moviments. Llavors es decideix tornar a la velocitat d'actualització anterior, però modificant el paràmetre ω de les oscil·lacions. D'aquesta manera, s'incrementa la variació de l'angle entre cada actualització de la posició. Això resulta en uns moviments una mica més bruscos, però també més ràpids, i a més els servos treballen a una velocitat d'actualització adequada.

Per tal de comprovar les diferents matrius de moviment s'estableix un sistema de selecció bastant senzill amb els interruptors disponibles de la *Spartan 3E*.

Un aspecte important que s'ha notat és que, degut a la velocitat d'oscil·lació que es fa servir, els períodes d'establiment del règim permanent un cop es reseteja el robot o es canvia la matriu de moviment poden ser relativament llargs. Per tant s'implementa també un sistema mitjançant el qual s'augmenta dràsticament la velocitat d'actualització dels oscil·ladors, per tal d'arribar al règim permanent sense haver de fer que el robot realitzi tots els moviments de les fases transitòries.

Tal i com s'ha comentat anteriorment a l'apartat *FPGAs disponibles,* problemes amb el disseny optimitzat fan que no sigui possible fer servir la placa petita, i s'hagi de fer servir la *Nexys 2*, que és massa gran per acoblar-la al robot. Per tant, les proves de locomoció estaran limitades per la longitud dels cables de connexió. A les proves realitzades la longitud dels cables permetia fer uns 3-5 passos, suficient per per tenir una idea de l'estabilitat i velocitat del robot.

Per realitzar les proves es fa servir la memòria RAM volàtil de la FPGA, ja que és més ràpid pujar el disseny a la memòria RAM que a la ROM no volàtil. No obstant, si es vol alimentar la FPGA directament de la bateria del robot, és necessari fer servir la memòria no volàtil per tal de mantenir el disseny guardat mentre l'ordinador no hi estigui connectat. Per fer proves que impliquin canvis freqüents al codi, l'ús de la memòria ROM és poc convenient ja que els dissenys triguen més a pujar-se.

CAPÍTOL 5: CONCLUSIONS I FUTURES MILLORES

A continuació s'analitzen els punts definits al començament del treball i fins a quin punt s'han aconseguit realitzar.

- *Estudi de l'estat de l'art:* Mitjançant diferents articles científics s'ha pogut estudiar la situació actual del camp, i comparar diferents enfocaments i tecnologies utilitzades a l'hora de treballar amb generadors de patrons centrals.
- *Simulació de les equacions i sistemes d'equacions mitjançant Matlab:* Els sistemes s'han simulat satisfactòriament, obtenint els resultats indicats als articles.
- *Realització i simulació del disseny VHDL:* S'ha aconseguit realitzar un disseny que segueix les equacions establertes i presenta un comportament satisfactori. No obstant no s'ha aconseguit optimitzar tant com es podria i per tant és possible que no hi càpiga a algunes *FPGAs*. Per tal d'aconseguir aquesta optimització caldria seguir el plantejament explicat mitjançant multiplexors.
- *Connexió o adaptació de la FPGA al robot:* Mitjançant el circuit de buffer ha sigut possible connectar la *FPGA* al bus de comunicació del robot i efectuar una comunicació satisfactòria. No obstant, no s'ha pogut integrar la *FPGA* al robot tal i com es desitjava en un principi, degut a que la placa amb la que es volia treballar (*Cmod S6)* no tenia suficients recursos pel disseny realitzat, i l'altra placa (*Nexys 2*) era massa gran. Així doncs s'ha optat per fer servir la *Nexys 2* per fer les proves, mantenint-la al costat del robot.

  Per adaptar la *Cmod S6* al robot seria necessari optimitzar el codi per tal que hi càpiga el disseny, i llavors realitzar una *PCB* amb aquesta placa per posar-la al robot.

- *Realització de proves amb el robot i la FPGA i ajustament dels paràmetres:* S'ha aconseguit que el robot camini fent servir diferents els diferents patrons de moviment.

Un cop realitzades les proves experimentals amb el robot hexàpode, es comprova que la xarxa neuronal descrita permet efectuar diferents patrons de moviment adequats per a una correcta locomoció. Per tant, l'objectiu general de realitzar una xarxa neuronal funcional ha estat assolit.

Així doncs, s'ha fet servir el disseny totalment paral·lel, amb la *Nexys 2 Spartan-3E*. Aquesta plataforma ha servit per realitzar proves al laboratori però no és adequada per integrar-la al robot.

La complexitat dels sistemes d'equacions involucrats ha limitat en gran part la llibertat o marge per modificar els paràmetres, ja que és molt fàcil que el sistema deixi de funcionar o funcioni erròniament quan es fan servir paràmetres que s'allunyen dels especificats a la bibliografia. L'estudi d'aquests sistemes i obtenció de paràmetres òptims podria ser un tema per un altre treball, però això potser pertanyeria més a un àmbit més matemàtic que d'enginyeria. Com que no hi ha gaires mètodes establerts per al disseny d'aquests sistemes, pot ser un àmbit de recerca interessant de cara a un futur.

Un altre aspecte que es pot millorar és l'aspecte mecànic del robot, ja que afecta directament al funcionament dels sistemes locomotrius implementats. Per exemple, seria convenient substituir els servos per uns de més qualitat.

També es podria dissenyar un sistema que integri aquestes xarxes neuronals i hi introdueixi un element de realimentació fent servir diferents sensors. Una possible estructura seria introduir un microcontrolador que s'encarregui de les decisions de més alt nivell, mentre la FPGA s'encarrega dels càlculs de la xarxa neuronal.

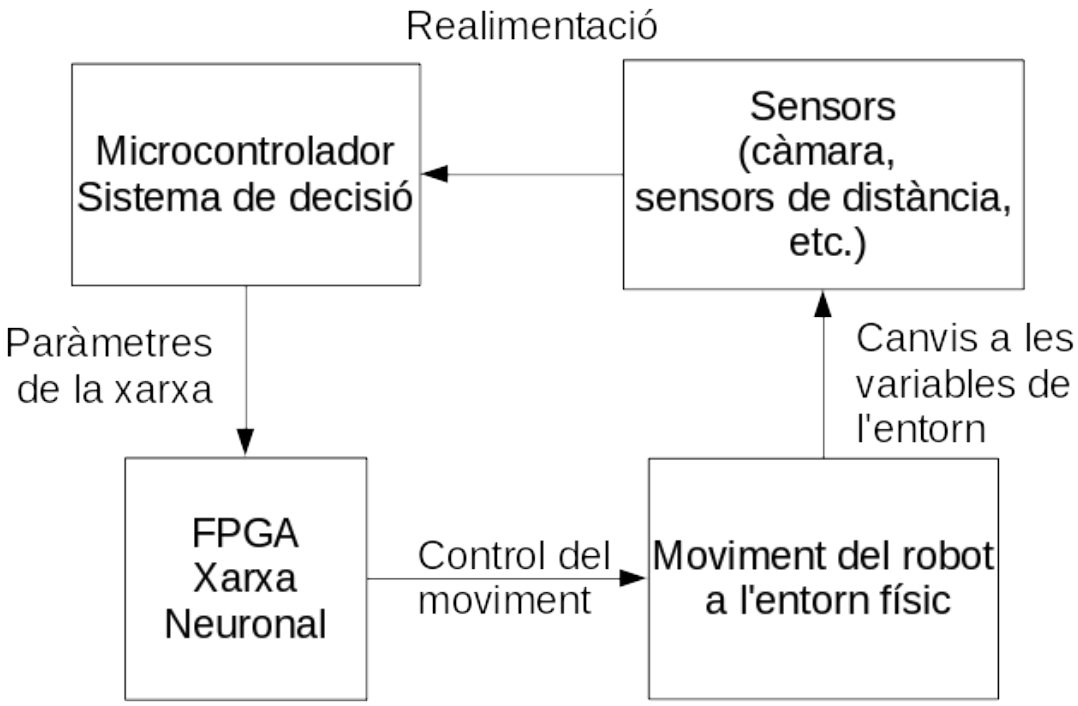

Figura 65: *Estructura de control amb realimentació mitjançant sensors.*

CAPÍTOL 6: BIBLIOGRAFIA

6.1.Referències bibliogràfiques

Atmel. 2015. 8-BIT microcontroller with 4/8/16/32KBytes in-system programmable flash datasheet. http://www.atmel.com/devices/atmega328.aspx?tab=documents (últim accés el Maig, 2016). [1]

Barron-Zombrano, J.H. and Torres-Huitzil, C. 2013. FPGA implementation of a configurable neuromorphic CPG-based locomotion controller. [2]

Barron-Zombrano, J.H. and Torres-Huitzil, C. 2011. Two-phase GA parameter tunning method of CPGs for quadruped gaits. [3]

Digilent. 2011. Digilent Nexys2 Board Reference Manual. http://store.digilentinc.com/nexys-2-spartan-3e-fpga-trainer-board-retired-see-nexys-4-ddr/ (últim accés el Maig, 2016). [4]

Digilent. 2011. Digilent Cmod S6™ FPGA Board Reference Manual. http://store.digilentinc.com/cmod-breadboardable-spartan-6-fpga-module/ (últim accés el Maig, 2016). [5]

Ijspeert, A.J. 2008. Central pattern generators for locomotion control in animals and robots: a review. [6]

IEEE Computer Society. 2009. IEEE Standard VHDL Language Reference Manual. New York, NY 10016-5997. [7]

Microchip. 7/8-Bit Single/Dual SPI Digital POT with Volatile Memory. http://www.microchip.com/wwwproducts/en/MCP4251 (últim accés el Maig, 2016). [8]

Philips Semiconductors. 1988. HCMOS family characteristics. https://www.cl.cam.ac.uk/teaching/2003/DigElec/part2-data.pdf (últim accés el Maig, 2016). [9]

Saborit, A.C. i Román, R.B. 2015. Disseny, construcció I test d'un robot hexàpode. Barcelona: EUETIB. [10]

Scott Larson, 2013. Serial Peripheral Interface (SPI) Master (VHDL). https://eewiki.net/pages/viewpage.action?pageId=4096096 (últim accés el Maig, 2016). [11]

Wikipedia. 2016. Serial Peripheral Interface Bus. https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus (últim accés el Maig, 2016). [12]

6.2.Bibliografia de Consulta

Arduino. Arduino Mini. https://www.arduino.cc/en/Main/ArduinoBoardMini (últim accés el Maig, 2016).

EUETIB. Treballs de Fi de Grau [TFG]. Barcelona: EUETIB. https://www.euetib.upc.edu/els-estudis/treballs-de-fi-de-grau (últim accés el Maig, 2016).

MathWorks. MATLAB Documentation. http://es.mathworks.com/help/matlab/ (últim accés el Maig, 2016)