Systema Naturae: Annali di Biologia Teorica, No. 1, 1998, pp.41-82.

Il Progetto Embryonics:

Una Macchina Fatta di Cellule Artificiali

Gianluca Tempesti, Daniel Mange, André Stauffer

Logic Systems Laboratory
Swiss Federal Institute of Technology
Lausanne, Switzerland
Email: Nome.Cognome@epfl.ch

Contenuti

Sommario

Per trovare i primi esempi di ispirazione biologica nella concezione di circuiti elettronici è necessario risalire alle origini stesse dell'ingegneria elettronica, con il lavoro di John von Neumann negli anni quaranta. Al suo genio dobbiamo non solo i primi tentativi di definire gli equivalenti elettronici di molti processi biologici fondamentali, ma anche la concezione delle prime macchine auto-replicanti.

Sfortunatamente, la tecnologia del periodo non permise una realizzazione fisica delle macchine di von Neumann, e solo l'introduzione di una nuova generazione di circuiti programmabili negli anni ottanta ha dato nuova vita ai tentativi di sviluppare macchine bio-ispirate. La motivazione che spinge gli ingegneri a cercare nei processi biologici un'ispirazione per la concezione di circuiti elettronici è ovvia, se si considera che gli organismi biologici sono esempi impressionanti di macchine da calcolo massivamente parallele.

In questo articolo vogliamo introdurre il progetto Embryonics (*embryonic electronics*), un tentativo di ispirarsi ai processi ontogenetici che guidano la crescita degli organismi multicellulari per lo sviluppo di nuove metodologie per la concezione di matrici massivamente parallele di processori (le *cellule artificiali*). Le nostre cellule sono processori molto semplici, basati su una identica struttura hardware. Ogni cellula contiene lo stesso programma (il *genoma artificiale*), ma ne esegue parti differenti a seconda della sua posizione all'interno della matrice. Come negli esseri viventi, la presenza di una copia del genoma in ogni cellula permette l'introduzione di processi quali l'auto-replicazione e l'auto-riparazione (cicatrizzazione).

Come in natura, la struttura delle nostre cellule può essere adattata alla funzione che deve eseguire. Questa versatilità è ottenuta, seguendo di nuovo l'esempio della natura, con l'introduzione di *molecole artificiali*, piccoli elementi logici programmabili che possono essere messi insieme per realizzare circuiti complessi. L'introduzione del livello molecolare dà origine a un'organizzazione a tre livelli (organismo, cellula, molecola) che presenta notevoli somiglianze con l'organizzazione degli esseri viventi.

Lo scopo del nostro progetto è di ottenere sistemi complessi basati sull'operazione parallela di molti semplici processori, realizzando dunque l'equivalente elettronico degli organismi multicellulari in natura.

1 Introduzione

L'ispirazione biologica per la concezione di macchine artificiali non è un concetto nuovo: la creazione di creature artificiali simili all'uomo (robot e automi meccanici) è di molto anteriore allo sviluppo dei primi computer. Con l'avvento dell'elettronica, i tentativi di imitare i sistemi biologici passarono dal mondo della meccanica al mondo dell'informazione: dal momento che il substrato fisico dei circuiti elettronici (l'hardware) non è facilmente modificabile, l'ispirazione biologica fu applicata quasi esclusivamente all'informazione (il software).

Il recente sviluppo di nuove tecnologie, quali i circuiti logici programmabili (che introdurremo più avanti) hanno reso la distinzione tra hardware e software molto meno netta, causando una rivalutazione del concetto di hardware bio-ispirato [13][16][30][38]. Il lavoro presentato in questo articolo rappresenta appunto un tentativo di realizzare tale hardware: ispirandosi ai processi ontogenetici che guidano la crescita degli organismi multicellulari, il progetto *Embryonics* (elettronica embrionale) vuole sviluppare una nuova metodologia per la concezione di circuiti elettronici digitali [20][24][25][37].

La motivazione di un tale tentativo dovrebbe essere ovvia, considerando che gli organismi multicellulari sono esempi impressionanti di sistemi massivamente paralleli: le $6x10^{13}$ cellule di un corpo umano, elementi relativamente semplici, lavorano insieme per esibire comportamenti estremamente complessi (tra i quali il più impressionante è, naturalmente, l'intelligenza). Se consideriamo la difficoltà di programmazione dei computer paralleli (una difficoltà che ha causato un declino nella loro popolarità), l'ispirazione biologica potrebbe suggerire degli approcci per trattare il parallelismo massivo nell'elettronica. Inoltre, la sorprendente capacità degli organismi biologici di sopravvivere a danni considerevoli può essere di grande interesse nella concezione di circuiti digitali, la cui crescente complessità sta facendo emergere il problema della tolleranza ai difetti (l'abilità di funzionare nonostante la presenza di difetti nel substrato di silicio di un circuito).

Di queste due caratteristiche maggiori, la tolleranza ai difetti (o auto-riparazione) è probabilmente la più "convenzionale": la concezione di circuiti digitali capaci di tollerare i difetti è un soggetto relativamente ben studiato. Altrettanto non si può dire per quanto concerne l'auto-replicazione, una componente indispensabile per lo sviluppo di macchine multicellulari. Dal momento che la ricerca nel campo delle macchine auto-replicanti è molto scarsa, una quantità considerevole di ricerca innovativa e originale è stata necessaria per integrare questa caratteristica nei nostri sistemi.

Siamo estremamente lieti di poterci rivolgere a un pubblico di biologi. Abbiamo tentato, entro ragionevoli limiti, di minimizzare il contenuto "tecnico" di questo articolo per renderlo (speriamo) più comprensibile. Cominceremo dunque con una breve definizione di alcuni dei termini tecnici usati in questo articolo, per poi introdurre la nostra interpretazione di cellula artificiale, ispirata dal processo biologico dell'ontogenesi da una parte e dal lavoro dei nostri predecessori, e in particolare di John von Neumann [40], dall'altra. Osserveremo poi che, per ottenere sistemi che siano efficienti dal punto di vista dell'ingegneria, è necessario introdurre un equivalente elettronico delle molecole quali elementi costitutivi delle nostre cellule. Concluderemo poi con qualche osservazione sull'ispirazione biologica del nostro sistema e su eventuali sviluppi futuri.

2 Alcuni concetti di base

Non è semplice trattare le propria area di competenza in modo da renderla comprensibile a tutti: spesso si presuppone che i termini e concetti "standard" siano conosciuti. Per evitare questo errore, in questa sezione cercheremo di definire alcuni dei principali termini usati nel resto dell'articolo, nella speranza che una tale spiegazione, necessariamente molto superficiale, sarà di aiuto per meglio comprendere il soggetto dell'articolo[1].

La concezione di un computer è basata su due concetti essenziali: il *software* e l'*hardware*. Una comprensione quasi assoluta della differenza tra questi due concetti è essenziale e non sempre così ovvia come potrebbe apparire a prima vista.

Il software è il mondo dell'informazione, in forma digitale. Per i computer, esistono due tipi principali d'informazione: programmi e dati. I programmi sono sequenze di istruzioni che indicano al processore di un computer la sua funzione. I dati sono il soggetto dell'operazione del processore, e possono rappresentare immagini, suoni, testo, o qualsiasi altro tipo d'informazione.

È comunque cruciale di ricordarsi che *tutto* il software, indipendentemente del suo significato ultimo (suoni, immagini, ecc.), è una sequenza di bit, cioè una sequenza unidimensionale di cifre 0 o 1. Non esiste alcuna differenza intrinseca tra sequenze che rappresentano, per esempio, il programma di elaborazione testi usato per scrivere questo articolo e il testo dell'articolo stesso. È il processore (e quindi l'hardware) che deve *interpretare* il significato delle sequenze e operare di conseguenza.

Il processore, l'esempio più rappresentativo di hardware digitale, è quindi responsabile della manipolazione (*processing*)dei dati: esso riceve una sequenza di bit che corrisponde a una sequenza di istruzioni, la interpreta per determinare la propria funzionalità, e esegue le operazioni indicate sui dati appropriati.

In realtà, però, i processori rappresentano solo una piccola percentuale di tutti i circuiti digitali in uso corrente. La grande maggioranza dei circuiti sono infatti destinati a eseguire (molto rapidamente) una singola specifica operazione. Tali circuiti sono chiamati *application-specific* (dedicati a un'applicazione), mentre i processori sono definiti *general-purpose* (a utilizzo generale).

Mentre il software è un concetto che sta diventando sempre più corrente, i metodi di concezione dei circuiti digitali non sono generalmente conosciuti dal pubblico, e questo articolo non è, ovviamente, una sede adeguata per una descrizione dettagliata. Nonostante ciò, dedicheremo qualche paragrafo a una definizione di alcuni dei termini principali usati in questo articolo.

L'elemento di base dei circuiti digitali è il transistore (*transistor*)[2], che può essere considerato come un semplice interruttore che permette o impedisce il passaggio di corrente elettrica attraverso un filo metallico[3]. L'apertura e la chiusura degli interruttori appropriati permette al filo di avere valore 1 (il voltaggio operativo del circuito, convenzionalmente 5 volt) o 0 (la massa del circuito, 0 volt). Tutti i circuiti digitali sono, a una prima approssimazione, collezioni di interruttori.

---

1 Molti tentativi di volgarizzazione molto più completi sono naturalmente disponibili. Per esempio. [26] è un'introduzione di facile lettura (ma abbastanza completa) alla concezione di hardware digitale, mentre [11] è un testo più "serio" e rigoroso (ma molto più tecnico).

2 L'utilizzo dei termini originali inglesi è molto corrente anche quando termini italiani equivalenti esistono.

3 In questo contesto, un "filo" è una linea di metallo sul substrato di silicio.

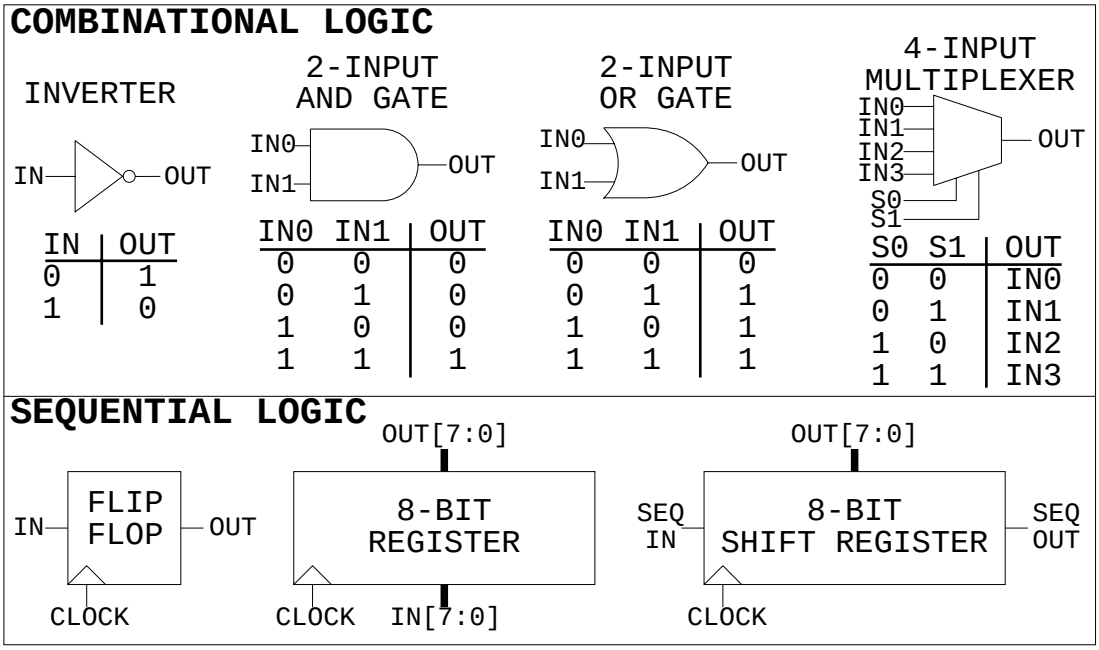


Figura 1: Alcuni elementi logici usati nella concezione di circuiti digitali.

La concezione di circuiti complessi attraverso la manipolazione diretta dei transistori è però estremamente difficile. Di conseguenza, i transistori sono raggruppati per formare *elementi logici* di varia complessità, capaci di implementare semplici funzioni (Figura 1). Tra gli elementi logici più comuni, menzioneremo:

- l'inversore (*inverter*), la cui uscita è l'inverso dell'entrata;
- la porta E (*AND gate*), la cui uscita è 1 se e solo se tutte le entrate sono 1;
- la porta O (*OR gate*), la cui uscita è 0 se e solo se tutte le entrate sono 0;
- il *multiplexer*, la cui uscita assume il valore di una delle entrate, selezionata da una variabile di controllo;
- il *flip-flop* è l'elemento di memoria di base, capace di memorizzare il valore di un singolo bit;
- il registro (*register*) è un gruppo di flip-flop, messi uno accanto all'altro per memorizzare multipli bit;
- il registro a scorrimento (*shift register*) è un tipo particolare di registro in cui tutti i flip-flop sono collegati in modo tale che il valore da memorizzare nel registro entra sequenzialmente da un lato.

Un circuito che non contiene alcun elemento di memoria (flip-flop, registri) è chiamato combinatorio (*combinational*), in caso contrario sequenziale (*sequential*). I circuiti sequenziali sono controllati da un orologio (*clock*), che determina la frequenza alla quale i valori sono caricati negli elementi di memoria.

La concezione di circuiti digitali consiste dunque nel mettere insieme questi elementi per realizzare funzioni che possono essere estremamente complesse. Come abbiamo menzionato, però, la distinzione tra hardware e software sta diventando meno netta in seguito all'introduzione di una nuova generazione di circuiti, chiamati *FPGA* (*Field-Programmable Gate Arrays*, o matrici di porte programmabili sul campo). Questi circuiti, descritti più sotto, sono privi di funzionalità ma possono essere configurati con una sequenza di bit (del software, quindi) per dare loro una qualsiasi struttura (una qualsiasi funzionalità). Gli FPGA, permettendo al software di alterare l'hardware, hanno aperto la strada allo sviluppo di hardware bio-ispirato.

3 Cellule artificiali

Il lavoro presentato in questo articolo tratta dello sviluppo di una nuova metodologia per la sintesi di circuiti digitali, e trae ispirazione da due fonti distinte. La prima, come abbiamo menzionato, è il meccanismo biologico dell'ontogenesi: il comportamento complesso degli organismi naturali è una conseguenza dell'operazione parallela di una moltitudine di semplici elementi, le cellule. La seconda fonte d'ispirazione per il nostro lavoro è il concetto di auto-replicazione di un computer universale, sviluppato da John von Neumann e basato sulla creazione automatica di molteplici copie identiche di una macchina a partire da una sola copia iniziale. Come vedremo, queste due fonti d'ispirazione, analizzate in dettaglio, presentano molti punti in comune.

3.1 Il costruttore universale di von Neumann

John von Neumann può essere considerato come il pioniere dell'hardware digitale bio-ispirato. Un matematico di genio e una delle figure principali nello sviluppo dell'intero settore dell'ingegneria informatica, von Neumann dedicò gli ultimi anni della sua vita a quella che definì la teoria degli automi [40]. La sua ricerca, che fu sfortunatamente interrotta dalla sua morte prematura nel 1957, si ispirava all'analogia tra gli automi artificiali, il principale esempio dei quali è il computer, e gli automi naturali, quali il sistema nervoso, gli organismi in evoluzione, ecc.

Per sviluppare una realizzazione fisica dei sui automi, von Neumann concepì una serie di macchine dotate di molte delle stesse proprietà dei sistemi biologici: evoluzione, apprendimento, auto-replicazione, auto-riparazione, ecc. Il suo approccio era basato sullo sviluppo di macchine auto-replicanti, cioè macchine capaci di produrre copie identiche di se stesse.

3.1.1 Le macchine auto-replicanti di von Neumann

Von Neumann, confrontato alla mancanza di affidabilità dei calcolatori elettronici, cercò nella natura l'ispirazione per la concezione di macchine capaci di tollerare i difetti. I sistemi naturali sono tra i sistemi complessi più affidabili conosciuti dall'uomo, e la loro affidabilità è una conseguenza non di una particolare robustezza delle cellule (o degli organismi) individuali, ma piuttosto della loro estrema ridondanza. Il meccanismo naturale alla base di tale ridondanza è l'auto-riproduzione[4], tanto al livello cellulare (per la sopravvivenza di un individuo) quanto al livello degli organismi (per la sopravvivenza della specie).

Von Neumann, traendo ispirazione dai sistemi naturali, tentò quindi di sviluppare un approccio alla realizzazione di macchine da calcolo auto-replicanti. A tal scopo, immaginò una serie di cinque modelli distinti di auto-riproduzione ([40], pp. 91-99), presentati in cinque conferenze all'Università dell'Illinois nel dicembre 1949:

- Il modello cinematico (*kinematic model*) era il più generale. Comprendeva elementi sia strutturali (sensori, "muscoli", strumenti di taglio, ecc.) sia elettronici (interruttori, memorie, ecc.). Il suo scopo era di definire le basi dell'auto-riproduzione, e non era destinato a una realizzazione pratica.

4 I termini auto-replicazione e auto-riproduzione non sono sinonimi: auto-riproduzione è più appropriata per la riproduzione degli organismi, mentre auto-replicazione riguarda il livello cellulare. In questo contesto, il termine più corretto sarebbe auto-replicazione, ma dal momento che von Neumann favoriva auto-riproduzione, ignoreremo la distinzione.

- Per trovare un approccio all'auto-replicazione che si prestasse più facilmente a un trattamento matematico rigoroso, von Neumann, su consiglio del matematico S. Ulam, sviluppò un modello cellulare (*cellular model*). Questo modello, basato sull'utilizzo degli *automi cellulari*[5], è quello che più si avvicinò a una realizzazione completa, ed è il punto d'inizio di ogni ulteriore ricerca sull'auto-riproduzione.
- Il modello eccitazione-limite-fatica (*excitation-threshold-fatigue model*) era basato sul modello cellulare, ma in esso gli elementi dell'automa cellulare erano sostituiti con elementi simili a neuroni. Von Neumann non definì mai la struttura esatta dei neuroni, ma possiamo immaginare che avrebbero una certa somiglianza a quelli usati oggi nelle reti neurali artificiali [12].
- Per il modello continuo (*continuous model*), von Neumann aveva l'intenzione di usare equazioni differenziali per descrivere il processo di auto-replicazione. Di nuovo, non conosciamo i dettagli precisi di questo modello, ma possiamo supporre che von Neumann volesse definire sistemi di equazioni differenziali per descrivere le proprietà dei suoi neuroni.
- Il modello probabilistico (*probabilistic model*) è il più vago. Sappiamo che von Neumann aveva l'intenzione di introdurre un tipo di automa nel quale le transizioni tra gli stati fossero probabilistiche piuttosto che deterministe. Un tale approccio avrebbe permesso l'introduzione di meccanismi quali la mutazione e quindi del processo di evoluzione degli automi artificiali.

Tra tutti questi modelli, l'unico che von Neumann sviluppò in un certo dettaglio fu il modello cellulare. Dal momento che esso rappresenta la base per il lavoro di tutti i suoi successori (compreso il nostro progetto Embryonics), merita un esame più approfondito.

3.1.2 Il modello cellulare di von Neumann

Nel lavoro di von Neumann, l'auto-reproduzione è sempre presentata come un caso particolare di *costruzione universale* (Figura 2): la sua macchina `Uconstr` può costruire qualsiasi altra macchina `M`, se può accedere alla sua descrizione `D(M)`.

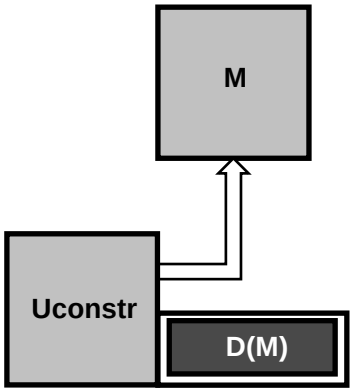


Figura 2: Il costruttore universale `Uconstr` di von Neumann può costruire una qualsiasi macchina `M` a partire dalla sua descrizione `D(M)`.

5 Gli automi cellulari [7][42] sono matrici di elementi, normalmente chiamati cellule (un termine che eviteremo per evitare confusione con le nostre cellule artificiali), che si comportano tutti in modo identico, in funzione del loro *stato*. A intervalli regolari e discreti (*iterazioni*), lo stato di ogni elemento è ricalcolato in funzione del suo stato precedente e di quello dei suoi vicini, sulla base di un serie di *regole di transizione*.

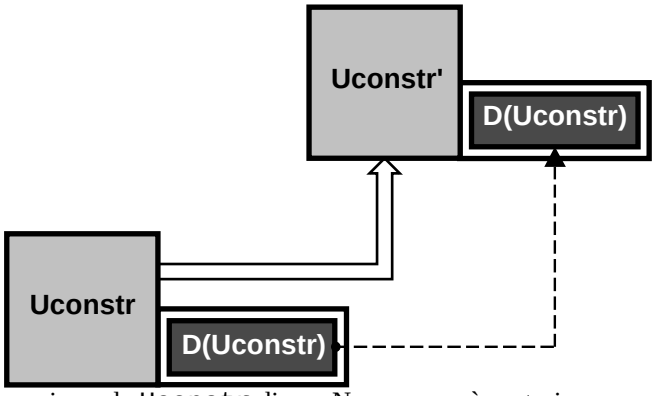


Figura 3: Il costruttore universale `Uconstr` di von Neumann può costruire una copia `Uconstr'` di se stesso a partire dalla sua descrizione `D(Uconstr)`.

Questo approccio fu mantenuto nella concezione del suo automa cellulare, che è quindi molto più di una macchina auto-replicante. La complessità della sua struttura riflette la complessità della sua funziona, ed è basata su tre componenti:

- Una banda di memoria (*memory tape*), che contiene la descrizione (una sequenza unidimensionale di elementi) della macchina da costruire. Nel caso particolare dell'auto-replicazione, la memoria contiene una descrizione `D(Uconstr)` del costruttore universale (Figura 3).
- Il costruttore universale (*universal constructor*), una macchina molto complessa capace di leggere la banda di memoria e di interpretare il suo contenuto.
- Un braccio costruttore (*constructing arm*), controllato dal costruttore universale, usato per costruire la macchina descritta nella banda di memoria. Il braccio si muove nello spazio e assegna uno stato appropriato agli elementi che comporranno la nuova macchina.

La realizzazione del costruttore come automa cellulare non è meno complessa. Ogni elemento ha 29 stati possibili e quindi, dato che lo stato futuro di un'elemento dipende dal suo stato presente e da quello dei suoi quattro vicini immediati, 29^5=20,511,149 regole di transizione sono necessarie per definire completamente il suo comportamento. Considerando che il construttore di von Neumann si estende su approssivativamente 100,000 elementi, possiamo facilmente capire come mai l'implementazione fisica di una tale macchina non fosse (e non sia tuttora) praticamente realizzabile.

Nel corso del progetto Embryonics abbiamo sviluppato un prototipo hardware di un gruppo di elementi dell'automa di von Neumann [5][31]. Attraverso una concezione accurata della struttura hardware degli elementi, siamo riusciti a ridurre considerevolmente la quantità di memoria necessaria a contenere le regole di transizione. Nonostante ciò, il nostro sistema resta un'unità di dimostrazione, dal momento che contiene solo pochi elementi, il minimo necessario per illustrare il comportamento di una piccola parte della macchina intera.

Il costruttore, come lo abbiamo descritto finora, non ha alcuna funzionalità all'eccezione dell'auto-riproduzione. Von Neumann, riconoscendo che una macchina auto-replicante necessita una funzionalità per essere interessante da un punto di vista ingegneristico, postulò la presenza di un computer universale `Ucomp` (in pratica, una macchina di Turing universale [14], un automa capace di ogni calcolo finito) accanto al costruttore universale (Figura 4).

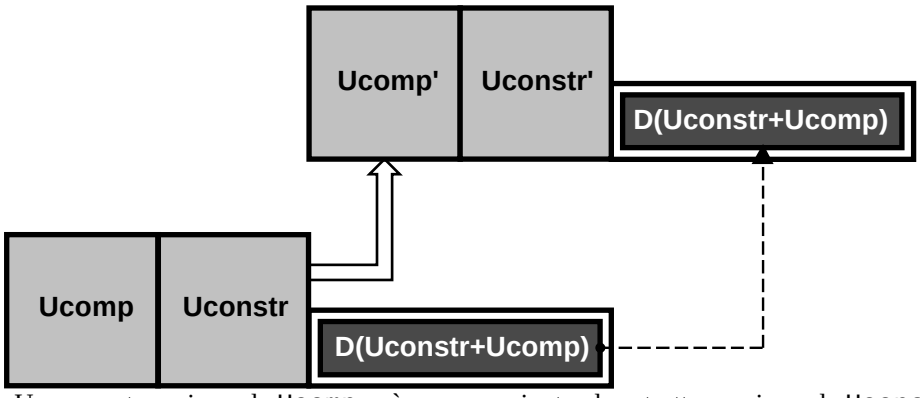


Figura 4: Un computer universale `Ucomp` può essere aggiunto al costruttore universale `Uconstr`, il quale può allora costruire una copia della macchina completa (`Ucomp'+Uconstr'`) dalla sua descrizione `D(Ucomp+Uconstr)`.

3.2 Il progetto Embryonics

Se consideriamo il costruttore universale di von Neumann da un punto di vista biologico, possiamo associare la banda di memoria con il genoma, e quindi il costruttore intero con una singola cellula (il che implica un parallelo tra gli elementi dell'automa cellulare e le molecole). Il costruttore di von Neumann può dunque essere considerato come un organismo unicellulare contenente un genoma, memorizzato sotto forma di una banda di memoria, letto e interpretato dal costruttore universale (la cellula madre) sia per determinare la propria funzionalità sia per dirigere la costruzione di una copia completa (la cellula figlia).

L'approccio usato per la realizzazione di hardware bio-ispirato nel progetto Embryonics è alquanto diverso. Abbiamo deciso di basare i nostri sistemi su cellule molto più semplici e di seguire l'esempio della natura, cercando di ottenere comportamenti complessi con l'operazione parallela di molte cellule. I nostri sistemi sono dunque l'equivalente artificiale di organismi multicellulari.

3.2.1 Macchine multicellulari

Lo sviluppo di un organismo biologico multicellulare coinvolge una serie di processi che guidano la crescita dell'individuo, cioè lo sviluppo di un organismo da una singola cellula madre (lo *zigote*) a un individuo adulto. Lo zigote si divide e ognuna delle due cellule risultanti contiene una copia del genoma (*divisione cellulare*). Questo processo continua (ogni nuova cellula si divide in due altre cellule, e così via) e ogni nuova cellula acquisisce una funzionalità (per esempio, cellula del fegato, dell'epidermide, ecc.) a seconda della sua posizione in relazione alle cellule vicine (*differenziazione cellulare*). Chiamiamo *ontogenesi* l'insieme dei processi che determinano l'evoluzione di un organismo dall'embrione all'individuo adulto.

In questo rispetto, le due fonti d'ispirazione che abbiamo descritto (l'ontogenesi e le macchine di von Neumann) sono differenti in modo fondamentale. Entrambe si basano su un meccanismo d'auto-replicazione per ottenere matrici di elementi che possono essere considerati come processori che eseguono tutti lo stesso programma. Nel caso di von Neumann, però i processori (macchine di Turing universali) sono identici in struttura come in funzionalità, e il fenomeno di differenziazione cellulare non è presente. In natura, le cellule sono diverse tanto in funzionalità quanto in struttura (una cellula dell'epidermide è diversa da una cellula del fegato sia in aspetto che in comportamento), ma ogni cellula è potenzialmente capace di rimpiazzare

qualsiasi altra, dal momento che contiene la descrizione dell'intero organismo (il genoma). La differenziazione cellulare è dunque fondamentale per i sistemi biologici.

In Embryonics, abbiamo sviluppato una soluzione che cerca di integrare i due approcci, basata su un'architettura veramente multicellulare capace sia di divisione che di differenziazione cellulare [20][21][24][37]. Come vedremo, il nostro approccio ci permette non solo di rispettare molte delle definizione di base della biologia, ma anche di sfruttare alcuni dei meccanismi specializzati alla base dello sviluppo ontogenetico di un organismo.

3.2.2 L'organismo artificiale

Per trovare un approccio pratico alla concezione di sistemi di calcolo ispirati all'operazione di organismi biologici multicellulari, abbiamo tentato di identificare le principali caratteristiche di tali organismi:

- In biologia, un organismo è una matrice di cellule che operano in parallelo per generare processi globali (cioè processi che coinvolgono l'organismo intero). Per rispettare l'analogia, il nostro organismo artificiale consisterà anch'esso di una matrice di elementi che operano in parallelo per eseguire un compito globale (un'applicazione).
- In biologia, ogni cellula contiene il genoma, cioè la descrizione dell'organismo, che è interpretato per determinare la funzionalità della cellula. Nel nostro sistema, ogni cellula è un piccolo processore che interpreta e esegue lo stesso programma, il nostro genoma artificiale.
- In biologia, nessuna cellula usa l'intero genoma, ma accede solo alla porzione necessaria per la propria funzione. In modo simile, nessun processore eseguirà tutte le istruzioni del programma, ma userà la propria posizione all'interno della matrice per identificare quale parte del programma è rilevante.

L'ispirazione biologica ci ha quindi portati a definire il nostro organismo artificiale come una matrice di processori di struttura identica (dal momento che ogni cellula deve poter eseguire qualsiasi parte del genoma)che eseguono parti diverse dello stesso programma, in funzione delle proprie coordinate nella matrice.

A prima vista, un tale sistema può sembrare molto inefficiente dal punto di vista della concezione tradizionale dei circuiti digitali: memorizzare una copia dell'intero genoma in ogni processore è inutile, dal momento che ogni processore ne eseguirà solo una parte. D'altro canto, accettare gli inconvenienti dell'ispirazione biologica ci permette di sfruttare i suoi vantaggi. Una delle caratteristiche più interessanti degli organismi biologici è la loro robustezza, una conseguenza di quella stessa ridondanza che sembra così inutile: dal momento che ogni cellula contiene una copia del genoma, può teoricamente rimpiazzare qualsiasi altra cellula. Così se una o più cellule dovessero morire a causa di un trauma (per esempio, una ferita), esse possono essere rigenerate a partire da una qualsiasi cellula viva. Per analogia, se uno o più processori dovessero "morire" (in conseguenza, per esempio, dell'apparizione di un difetto nell'hardware) essi possono essere rimpiazzati da altri processori intatti nella matrice.

La ridondanza introdotta dalla presenza di copie multiple dello stesso programma fornisce quindi un supporto intrinseco all'auto-riparazione, uno degli obiettivi principali del nostro progetto: grazie a una serie di cellule di riserva (cioè cellule che sono inattive durante il funzionamento normale del circuito, ma che sono identiche a tutte le altre e contengono lo stesso programma) possiamo (Figura 5) riconfigurare la

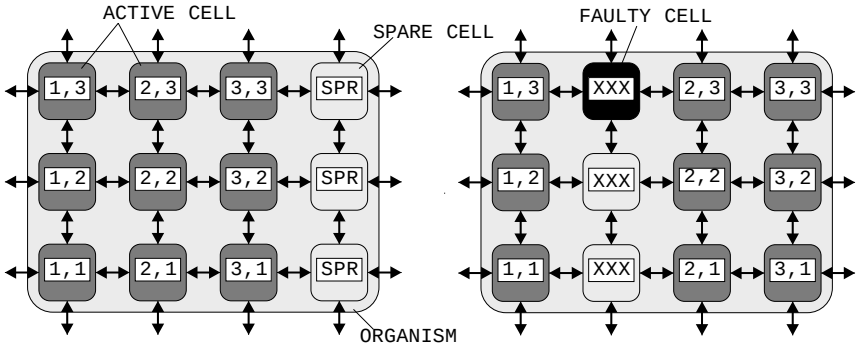


Figura 5: Quando una delle cellule di un organismo muore, la colonna che la contiene è disattivata, e le coordinate sono ricalcolate in tutta la matrice.

matrice in modo da evitare uno o più processori difettosi (naturalmente, come negli esseri viventi, un numero eccessivo di cellule morte causa la morte dell'intero organismo).

Inoltre, se la funzionalità di una cellula dipende dalle coordinate, l'auto-riproduzione ne risulta molto semplificata: calcolando le coordinate in modo ciclico (Figura 6), possiamo ottenere molteplici copie di un organismo con una singola copia del programma (ammesso, naturalmente, che un numero sufficiente di processori sia disponibile). A seconda dell'applicazione e delle necessità dell'utilizzatore, questa caratteristica può essere utile sia per ottenere una migliore performance (multipli organismi che lavorano in parallelo su dati diversi) o una più grande robustezza (le uscite di più processori operanti sugli stessi dati possono essere paragonate per verificare il loro corretto funzionamento).

3.2.3 La cellula artificiale

Tenendo presente le necessità e la struttura degli organismi artificiali, possiamo ora definire le caratteristiche essenziali della nostra cellula artificiale. A livello hardware, tutte le cellule devono essere identiche: dal momento che i nostri organismi devono poter eseguire in modo efficace diverse applicazioni, non possiamo fissare a priori la funzionalità delle nostre cellule. Inoltre, devono poter memorizzare il programma genico con un meccanismo di accesso dipendente dalle coordinate.

La struttura hardware della nostra cellula dovrà dunque poter eseguire una qualsiasi funzione, e sarà compito del genoma e del sistema di coordinate scegliere quale parte del circuito dovrà essere attivata a un dato momento.

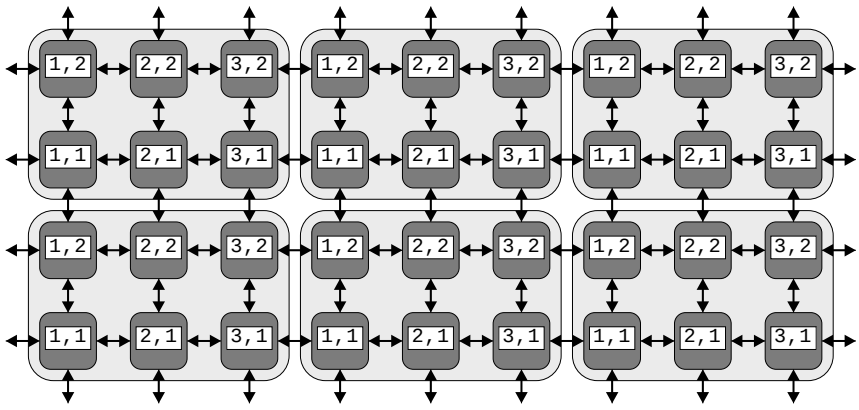


Figura 6: Il calcolo ciclico delle coordinate genera automaticamente copie multiple dell'organismo, a condizione che un numero sufficiente di cellule siano presenti.

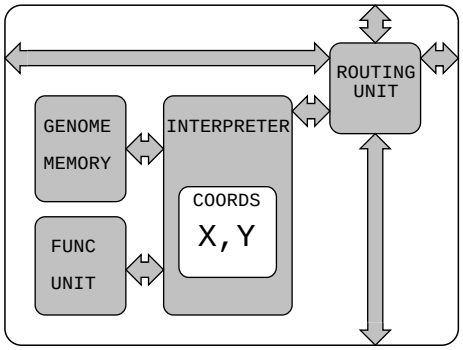


Figura 7: Struttura generale delle nostre cellule artificiali.

Ovviamente, esistono molti possibili approcci alla definizione di una cellula artificiale, ma se dobbiamo mantenere l'analogia con la biologia, essa dovrà necessariamente contenere (Figura 7):

- Una memoria per il genoma, la cui taglia dipende dall'applicazione.
- Un sistema di coordinate [X,Y] che permetta alla cellula di identificare la propria posizione all'interno della matrice, e quindi la propria funzione.
- Un decodificatore per leggere e eseguire il genoma.
- Un'unità funzionale per il trattamento dei dati. A seconda dell'applicazione, essa può contenere una varietà di elementi logici, da un semplice registro a un'unità aritmetica completa e oltre[6].
- Una serie di connessioni comandate da un'unità di scambio.

Per dimostrare le caratteristiche del nostro sistema cellulare, abbiamo sviluppato un prototipo, chiamato MicTree (Figura 8) [21][24], e lo abbiamo usato per realizzare una serie di applicazioni che, se relativamente semplici a causa della taglia limitata del nostro prototipo, sono comunque interessanti in quanto esse esibiscono le proprietà di auto-riparazione (in presenza di cellule di riserva) e di auto-replicazione (se la matrice è sufficientemente grande).

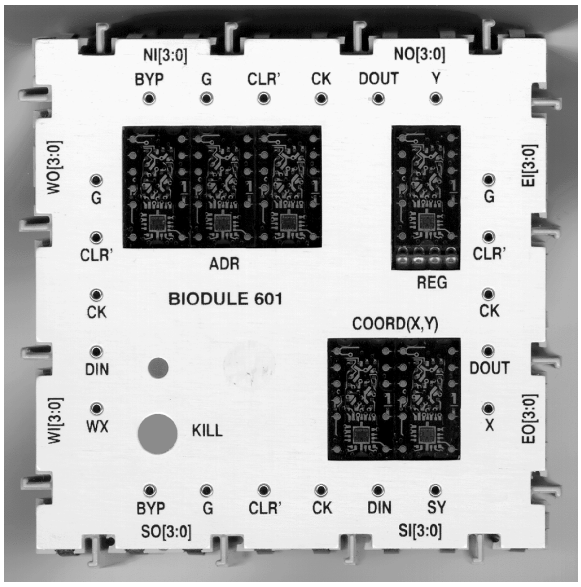


Figura 8: Il Biodule 601, un prototipo che contiene una singola cellula. [Foto: André Badertscher]

6 Mentre l'unità funzionale può teoricamente essere di qualsiasi taglia e complessità, abbiamo già menzionato che uno dei nostri obiettivi è mostrare che è possibile sviluppare sistemi complessi combinando cellule molto semplici.

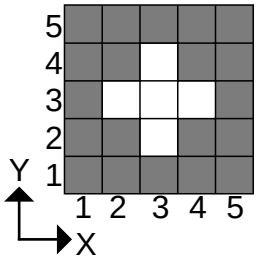


Figura 9: L'organismo *bandiera svizzera*, una matrice di 5x5 cellule.

3.2.4 Un semplice esempio

Per illustrare la metodologia di concezione dei nostri sistemi, possiamo usare un organismo molto semplice che chiameremo, per ragioni che dovrebbero essere apparenti, la *bandiera svizzera* (Figura 9).

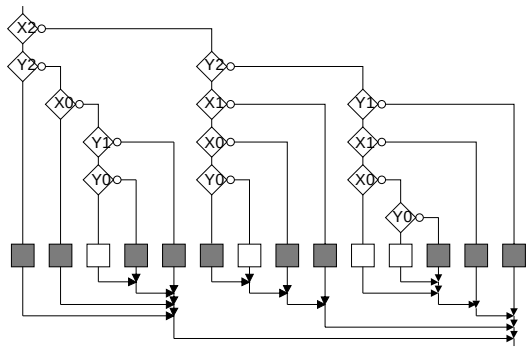
Questo esempio contiene 25 cellule (una matrice 5x5) con due soli tipi di cellula: rosse (in grigio nella figura) e bianche, distribuite a formare una croce (la bandiera della Svizzera, appunto). In un'applicazione reale, naturalmente, le cellule avrebbero comportamenti diversi (per esempio, eseguirebbero operazioni diverse a seconda del loro colore), ma per semplicità supporremo che la funzionalità delle cellule è semplicemente di determinare il proprio colore a seconda delle proprie coordinate all'interno della matrice, dimostrando così la differenziazione cellulare.

La prima parte del programma cellulare consiste dunque nel determinare le coordinate della cellula. La prima cellula (in basso a sinistra) vedendo che non ha vicini né a est né a sud, può fissare le proprie coordinate (`X,Y=1,1`). In seguito, essa incrementa questi valori e li trasmette ai suoi vicini a ovest e a nord, che li usano per calcolare le proprie coordinate (`X,Y=1,2` e `X,Y=2,1`, rispettivamente), li incrementano, e li trasmettono ai propri vicini. Il processo continua finché tutte le cellule hanno trovato le proprie coordinate.

La rappresentazione più semplice della differenziazione cellulare è probabilmente la tavola di verità o *lookup table*, una descrizione lineare del colore di ogni cellula in funzione delle sue coordinate. Se usiamo il codice binario, abbiamo bisogno di tre bit per rappresentare ogni coordinata (e quindi sei bit per paio di coordinate). Possiamo allora usare sei variabili ausiliarie (`X0`, `X1`, `X2`, `Y0`, `Y1`, e `Y2`) per rappresentare le coordinate `X` e `Y` di ogni cellula.

X	X2	X1	X0	Y	Y2	Y1	Y0	FLAG
1	0	0	1	1	0	0	1	
1	0	0	1	2	0	1	0	
1	0	0	1	3	0	1	1	
1	0	0	1	4	1	0	0	
1	0	0	1	5	1	0	1	
2	0	1	0	1	0	0	1	
2	0	1	0	2	0	1	0	
2	0	1	0	3	0	1	1	
2	0	1	0	4	1	0	0	
2	0	1	0	5	1	0	1	
3	0	1	1	1	0	0	1	
3	0	1	1	2	0	1	0	
3	0	1	1	3	0	1	1	
3	0	1	1	4	1	0	0	
3	0	1	1	5	1	0	1	
4	1	0	0	1	0	0	1	
4	1	0	0	2	0	1	0	
4	1	0	0	3	0	1	1	
4	1	0	0	4	1	0	0	
4	1	0	0	5	1	0	1	
5	1	0	1	1	0	0	1	
5	1	0	1	2	0	1	0	
5	1	0	1	3	0	1	1	
5	1	0	1	4	1	0	0	
5	1	0	1	5	1	0	1	

(A)



(B)

Figura 10: L'organismo *bandiera svizzera* come tavola di verità (A) e albero di decisione binario (B).

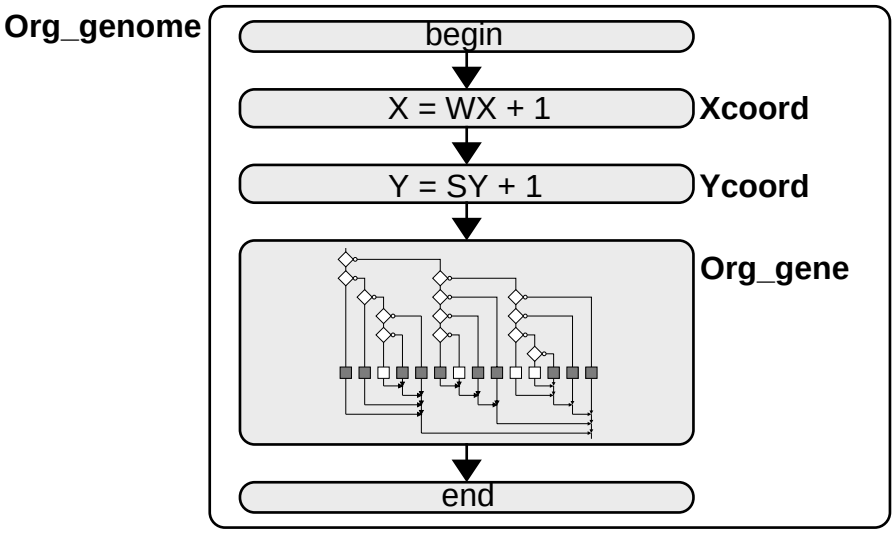


Figura 11: Il programma `Org_genome` eseguito in ognuna delle cellule dell'organismo.

La tavola di verità può essere facilmente trasformata in una forma equivalente, chiamata albero di decisione binario [3][19]. Tali alberi consistono di due tipi di elementi: *elementi di test* romboidali, che selezionano quale di due rami dell'albero sarà percorso dal programma in funzione del valore di una *variabile di test* (se il valore della variabile è 0, il programma percorrerà il ramo identificato da un piccolo cerchio) e *elementi di uscita* quadrati, che assegnano un valore (rosso o bianco) alla cellula. A seconda delle loro coordinate (le variabili di test), cellule diverse percorreranno rami diversi, ottenendo la differenziazione cellulare.

A questo punto, possiamo mettere insieme il nostro genoma completo `Org_genome` (Figura 11), il programma che sarà eseguito all'interno di ogni cellula della matrice. Esso consisterà di tre sub-programmi:

- `Xcoord`, il cui compito è di incrementare la coordinata `WX` inviata dal vicino a ovest e di propagarla verso l'est;
- `Ycoord`, il cui compito è di incrementare la coordinata `SY` inviata dal vicino a sud e di propagarla verso il nord;
- `Org_gene`, che contiene la parte operativa del genoma (in questo caso, l'assegnazione del colore delle cellule, ma normalmente qualcosa di più complicato), eseguito in funzione delle coordinate della cellula.

La presenza di una copia del genoma completo conferisce al nostro organismo *bandiera svizzera* entrambe le proprietà che abbiamo menzionato (Figura 12): può, in presenza di cellule di riserva, auto-ripararsi cicatrizzando le colonne che contengono cellule morte, mentre il calcolo ciclico delle coordinate può generare molteplici copie dell'organismo, se un numero sufficiente di cellule sono disponibili nella matrice.

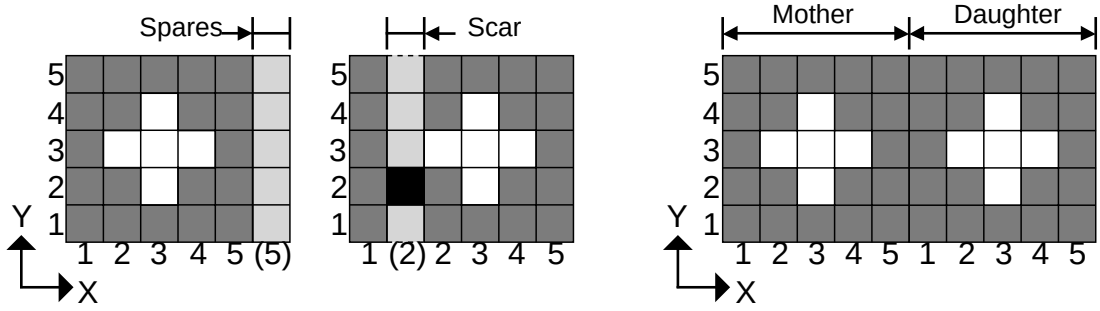


Figura 12: Cicatrizzazione (A) e replicazione (B) dell'organismo *bandiera svizzera*.

4 Molecole artificiali

L'esperienza accumulata nella concezione e uso del nostro prototipo ci ha permesso di stabilire che fissare a priori la taglia e le caratteristiche delle nostre cellule artificiali (per esempio, la taglia della memoria o la struttura dell'unità funzionale) è una limitazione troppo rigida per la realizzazione di sistemi complessi. Per trovare una soluzione a questo problema, abbiamo di nuovo cercato ispirazione nella natura. La struttura delle cellule biologiche varia a seconda della loro funzionalità, una versatilità resa possibile dalla loro struttura molecolare: dal momento che le cellule sono create a partire da elementi più piccoli (le molecole), esse possono cambiare facilmente taglia e funzionalità.

In aggiunta all'equivalente elettronico di una cellula, abbiamo dunque bisogno di definire il concetto di *molecole artificiali* come piccoli elementi elettronici che possono essere messi insieme per realizzare una cellula artificiale. Trovare un equivalente elettronico delle molecole non è, fortunatamente, difficile: un tipo particolare di circuiti, gli *FPGA* (*Field-Programmable Gate Arrays*, o matrici di porte programmabili sul campo), sono perfettamente adatte per assolvere questo compito.

In questa sezione forniremo una breve descrizione della struttura generale di questo tipo di circuiti, per poi introdurre *MuxTree*, l'FPGA che abbiamo sviluppato nel progetto Embryonics per realizzare facilmente le nostre cellule artificiali, e esaminare in dettaglio le due proprietà che abbiamo introdotto per rispettare le esigenze del nostro sistema: l'auto-replicazione e l'auto-riparazione.

4.1 Field-Programmable Gate Arrays

L'ostacolo principale alla realizzazione di hardware ontogenetico è stata finora la necessità dei sistemi bio-ispirati di modificare la struttura dell'hardware per implementare proprietà tali l'auto-replicazione o l'evoluzione. Fino a pochi anni fa tali modifiche erano essenzialmente impossibili: un circuito era concepito e costruito per eseguire una singola applicazione o funzione.

Verso la fine degli anni ottanta, però, un nuovo tipo di circuito fu introdotto sul mercato. Questi circuiti, chiamati FPGA [6][26][39], sono matrici bidimensionali di elementi logici[7] che possono essere *configurati* (cioè programmati via software) per realizzare qualsiasi funzione logica (cioè per implementare qualsiasi circuito logico digitale).

Mentre la struttura esatta e la taglia di un elemento di FPGA possono variare in modo considerevole da un fabbricante a un altro, alcune caratteristiche fondamentali rimangono costanti (Figura 13):

- Ogni elemento può implementare una *funzione programmabile* che normalmente consiste di logica combinatoria più almeno un elemento di memoria (flip-flop) per realizzare circuiti sequenziali. La complessità e la struttura della funzione programmabile possono variare considerevolmente da un tipo di FPGA a un altro.
- Lo scambio di informazione tra gli elementi è realizzata con una serie di *connessioni programmabili*, la cui complessità è anch'essa variabile.

7 Ancora una volta, la terminologia standard è in conflitto con le definizioni usate nel progetto Embryonics: Gli elementi di un FPGA sono convenzionalmente chiamati *cellule*, un termine che eviteremo di usare per non creare confusione con le nostre cellule artificiali.

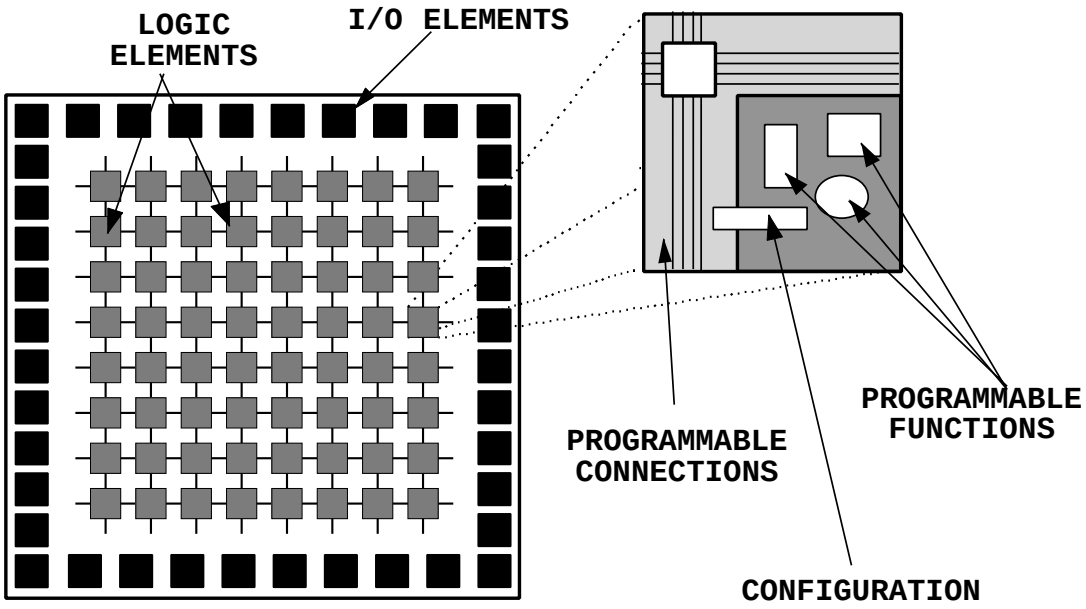


Figura 13: Architettura di base di un FPGA.

- La funzionalità e le connessioni di un elemento sono controllati dalla sua *configurazione*, una sequenza di bit (normalmente memorizzati in un registro) che indica le parti della logica funzionale e le connessioni che devono essere attive. La *sequenza di configurazione* (cioè, la somma delle configurazioni di tutti gli elementi) determina il comportamento globale del circuito.

Una data sequenza di configurazione assegnerà dunque funzioni differenti a ogni elemento e connetterà insieme gli elementi per realizzare il comportamento globale desiderato. Con la configurazione appropriata, un FPGA può realizzare qualsiasi circuito logico digitale, a condizione naturalmente che un numero sufficiente di elementi sia disponibile. Inoltre, nella maggior parte dei casi, gli FPGA sono riprogrammabili, cioè la loro configurazione può essere cancellata e sostituita con un'altra, cambiando così il circuito realizzato dall'FPGA.

Ovviamente, la notevole versatilità degli FPGA ha un prezzo: la rapidità. Circuiti realizzati con un FPGA sono necessariamente molto più lenti (devono cioè operare a una frequenza molto più bassa) di un circuito integrato. In certi casi, però, la versatilità degli FPGA può compensare questo inconveniente, o perché la rapidità non è un fattore essenziale e il circuito necessita alterazioni dinamiche in corso di funzionamento, o perché il vantaggio di avere processori specializzati (spesso paralleli) riesce a compensare la diminuzione della frequenza di funzionamento (per esempio, certe operazioni matematiche che richiederebbero molti cicli d'orologio in un processore a uso generale possono essere eseguite da un processore specializzato in un solo ciclo, anche se più lento).

Il progetto Embryonics spera di sfruttare quest'ultima considerazione, cercando di compensare la lentezza relativa degli FPGA con l'uso di sistemi di calcolo paralleli specializzati per un'applicazione particolare. La riprogrammabilità degli FPGA è infatti la soluzione ideale al problema di realizzare macchine ontogenetiche, dal momento che permette di modificare la struttura hardware di un circuito attraverso una modificazione del software (la sequenza di configurazione).

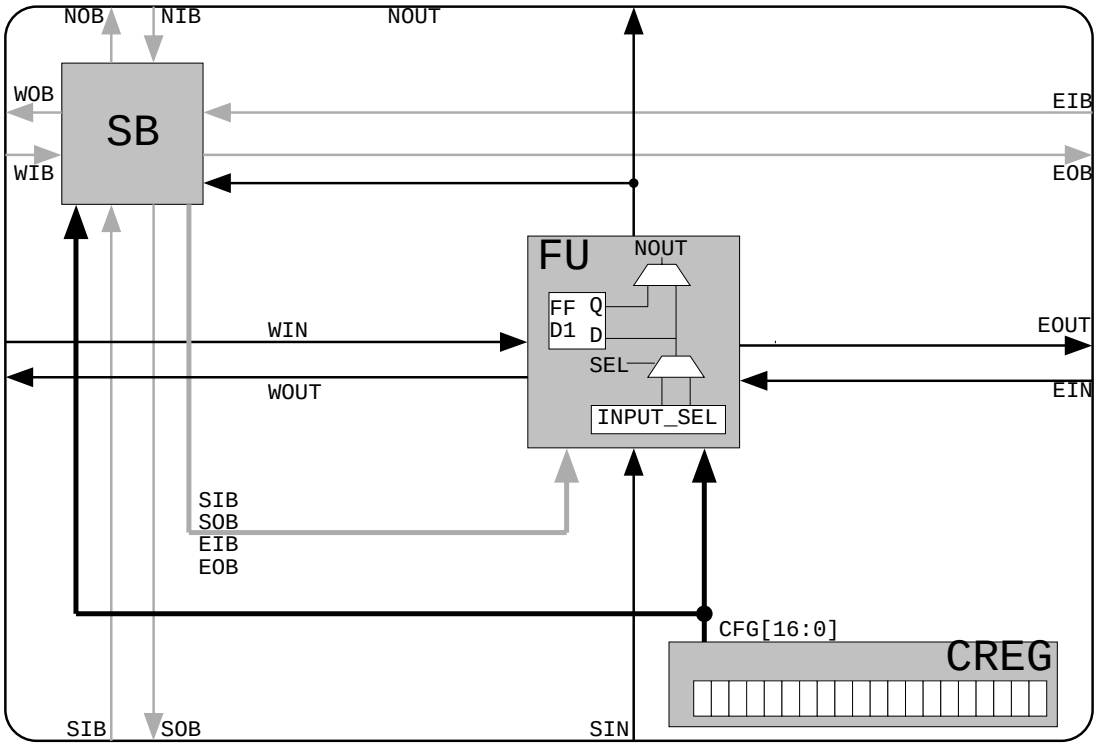


Figura 14: La struttura di un elemento MuxTree.

4.2 Un FPGA per il progetto Embryonics: MuxTree

Come abbiamo visto, qualsiasi FPGA può essere usato, entro ragionevoli limiti, per realizzare qualsiasi circuito logico digitale. Di conseguenza, qualsiasi FPGA può potenzialmente realizzare le nostre matrici di cellule artificiali (le quali, essendo piccoli processori, sono circuiti digitali). Gli FPGA disponibili sul mercato, però, presentano un certo numero di inconvenienti che rendono il loro uso nel progetto Embryonics, se non impossibile, perlomeno difficoltoso[8]. Abbiamo dunque deciso di creare un FPGA concepito specialmente per implementare matrici di cellule artificiali.

L'architettura di un elemento di FPGA può variare in modo considerevole da un circuito a un altro. L'unica esigenza è che deve essere possibile realizzare qualsiasi funzione discreta usando uno o più elementi. Inoltre, è consueto, se non rigidamente obbligatorio, includere della memoria in un elemento per poter realizzare facilmente sistemi sequenziali.

MuxTree [24][35][36][37], l'FPGA che abbiamo sviluppato, rispetta in pieno queste esigenze, ma è inconsueto nel suo *grano*[9], che è particolarmente fino: gli elementi che compongono la matrice bidimensionale dell'FPGA (le nostre molecole) sono particolarmente piccole. Ogni elemento è infatti capace di realizzare la funzione universale di una sola variabile e di memorizzare un singolo bit di informazione.

L'elemento di base del nostro FPGA (Figura 14) consiste di tre sub-circuiti separati: la funzione programmabile (FU), le connessioni programmabili (SB), e il registro di configurazione (CREG).

---

8 Menzioneremo, per esempio, la loro mancanza di omogeneità e la difficoltà di generare sequenze di configurazione per sistemi che si estendono su più circuiti.

9 Un FPGA è a *grano fino* se i suoi elementi sono piccoli, a *grano grosso* se più grandi.

La funzione programmabile è realizzata con un singolo multiplexer a due entrate (il nome MuxTree è una contrazione di *tree of multiplexers*, o albero di multiplexers). Dal momento che il multiplexer è una porta universale (cioè, è possibile realizzare ogni funzione con un numero sufficiente di multiplexers), la prima esigenza per un elemento di FPGA è rispettata. In aggiunta, ogni elemento può memorizzare un singolo bit di informazione in un flip-flop, rispettando così la seconda esigenza.

Per quanto riguarda la rete di connessioni programmabile, un elemento MuxTree contiene due reti separate: una rete fissa a corto raggio per la comunicazione tra elementi vicini e una rete programmabile a lungo raggio per la comunicazione tra elementi più distanti. Quest'ultima è controllata da una *switch box* (SB, per scatola di scambio) che può dirigere l'uscita `NOUT` di un elemento verso i suoi quattro vicini immediati e/o ridirigere segnali in entrata verso le quattro direzioni cardinali.

La funzione e le connessioni di un elemento sono determinate da un configurazione di 17 bit, memorizzata nel registro a scorrimento `CREG`. Questi bit sono sufficienti per configurare sia la funzione programmabile che le reti di connessioni. Tutti i registri di configurazione di tutti gli elementi della matrice sono collegati in modo da formare un solo lungo registro a scorrimento che permette alla sequenza di configurazione (che entra dall'angolo in basso a sinistra della matrice) di propagarsi in tutti gli elementi del circuito.

A prima vista, MuxTree non è necessariamente più adatto alla realizzazione di sistemi bio-ispirati di qualsiasi altro FPGA commerciale. La concezione di un FPGA dedicato, però, ci permette di modificare la sua architettura in modo da rispettare le esigenze particolari del nostro progetto.

Queste esigenze sono relativamente chiare: abbiamo bisogno di un FPGA che possa essere facilmente configurato come una matrice di processori identici (le nostre cellule artificiali, che hanno una identica struttura hardware). In altre parole, esso deve facilitare l'*auto-replicazione*, cioè la generazione di multiple copie identiche delle nostre cellule. In aggiunta, dal momento che il meccanismo di riparazione al livello cellulare è costoso (la morte di un'intera colonna di processori rappresenta una perdita considerevole di risorse), sarebbe estremamente utile avere delle molecole capaci di sopravvivere almeno qualche errore (difetti nel substrato di silicio). Sarebbe quindi molto interessante rendere le nostre molecole capaci di *auto-riparazione*.

4.3 L'auto-replicazione

L'auto-replicazione degli organismi, ottenuta attraverso il calcolo ciclico delle coordinate cellulari, è una conseguenza immediata dell'architettura delle nostre cellule artificiali. Anche per l'auto-replicazione delle cellule abbiamo quindi deciso di includere hardware dedicato all'interno dell'architettura delle nostre molecole. Questo hardware ci permetterà dunque di ottenere copie multiple delle nostre cellule senza eccessiva difficoltà. Sfortunatamente, l'approccio da adottare per la realizzazione di un tale meccanismo non è ovvio, dal momento che la ricerca nel settore dell'hardware auto-replicante è relativamente scarsa.

Il costruttore universale di von Neumann è probabilmente il primo esempio di hardware auto-replicante. Sfortunatamente, la tecnologia degli anni cinquanta non permetteva la realizzazione di macchine così complesse. Di conseguenza, la ricerca in questo settore fu praticamente abbandonata per molti anni. Gli anni ottanta, però, videro una ripresa dell'interesse degli ingegneri per l'ispirazione biologica e la nascita di un nuovo settore di ricerca, chiamato *vita artificiale*.

Christopher Langton, considerato il fondatore di questo campo di ricerca, definì il problema dell'auto'replicazione in modo leggermente diverso, cercando di sviluppare la più piccola macchina capace *esclusivamente* di auto-replicazione (abbandonando quindi il concetto di calcolo e costruzione universali di von Neumann). Il risultato della sua ricerca fu un automa cellulare relativamente semplice, conosciuto con il nome di *Langton's loop* (circolo di Langton) [18]. Questo automa rappresenta il punto d'inizio dei nostri tentativi di sviluppare strutture auto-replicanti.

La prima fase della nostra ricerca fu quindi una fase teorica, basata, seguendo l'esempio dei nostri predecessori, sull'uso di automi cellulari per studiare il problema dell'auto-replicazione, e risultò nella concezione di una famiglia di nuovi automi auto-replicanti molto più versatili e potenti di quello di Langton [28][34][37].

La successiva transizione dagli automi cellulari all'hardware, però, ha richiesto un attento processo di sintesi, dal momento che gli automi cellulari non sono ben adattati a una realizzazione hardware. Abbiamo dunque dovuto identificare i meccanismi di base usati nell'auto-replicazione dei nostri automi e cercare di semplificarli per poterli inserire nel nostro FPGA molecolare.

La chiave in questa transizione è l'osservazione che il meccanismo di auto-replicazione può essere diviso in due fasi distinte: una fase *strutturale*, nella quale lo "scheletro" del nuovo automa è generato nello spazio, e una fase di *configurazione*, nella quale l'informazione contenuta nell'automa originale (cioè la sua funzionalità) è copiata nel nuovo.

Mentre la fase di configurazione, per ragioni pratiche, non è ben adattata al nostro problema, la fase strutturale si rivela essere una soluzione molto efficiente per realizzare l'auto-replicazione nel nostro FPGA. Infatti, se consideriamo la matrice degli elementi FPGA prima dell'arrivo della configurazione come analoga alla matrice di un automa cellulare vuoto, possiamo vedere la fase strutturale dell'auto-replicazione come un meccanismo che divide l'FPGA in blocchi identici di molecole di taglia programmabile (ogni blocco conterrà poi una delle nostre cellule artificiali).

Questo processo di divisione può essere realizzato con un automa cellulare di una semplicità tale che una realizzazione hardware diventa triviale(Figura 15) [33][37]. L'automa dovrà semplicemente trasformare (con un processo che non descriveremo in dettaglio) una sequenza unidimensionale di istruzioni (tale una sequenza di configurazione memorizzata all'esterno del circuito) in una struttura bidimensionale.

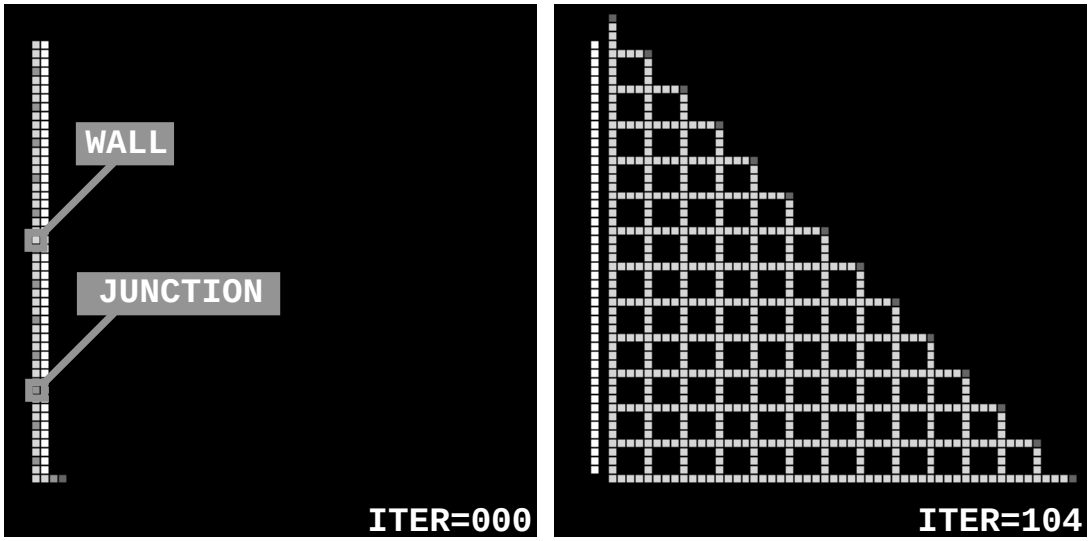


Figura 15: Il costruttore di membrane è un automa cellulare molto semplice capace di dividere lo spazio in blocchi di taglia identica.

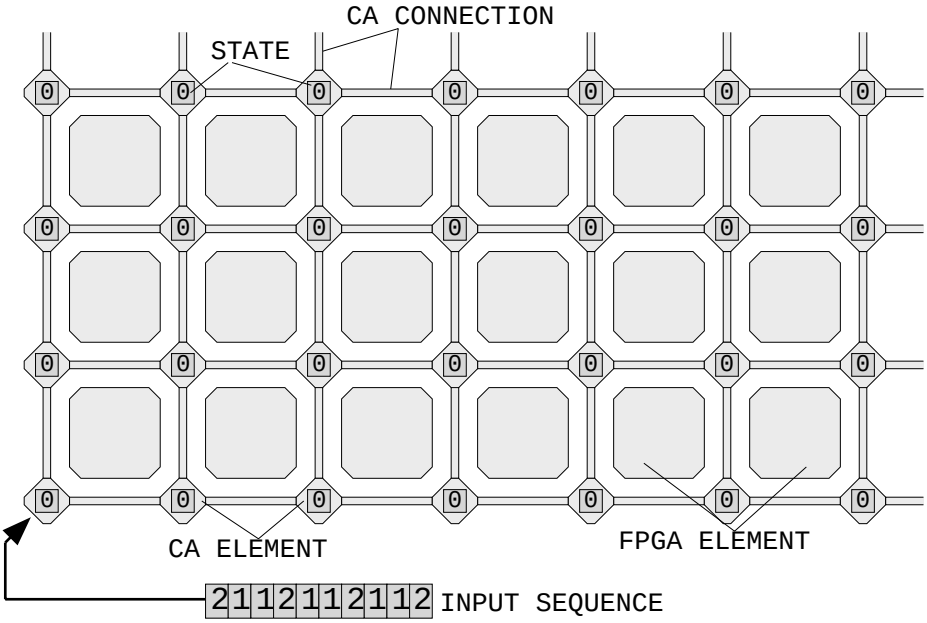


Figura 16: Il costruttore di membrane è inserito tra gli elementi del nostro FPGA.

Al fine di integrare l'automa nel nostro FPGA molecolare, abbiamo inserito i suoi elementi nello spazio tra gli elementi MuxTree (Figura 16). Con l'appropriata sequenza di istruzioni possiamo quindi dividere la matrice in blocchi identici di taglia variabile.

La funzione dell'automa è quindi quella di creare una *membrana cellulare*, che circonderà ognuna delle nostre cellule. Una volta che la membrana è posizionata, il circuito può sfruttarla per dirigere la propagazione della sequenza di configurazione dell'FPGA (Figura 17): dal momento che la configurazione di ogni cellula è identica, possiamo inviare la stessa sequenza in tutti i blocchi in parallelo, ottenendo così automaticamente multiple copie della stessa cellula e realizzando il processo di auto-replicazione desiderato.

Approfittando dell'esperienza accumulata nello sviluppo di automi cellulari auto-replicanti, abbiamo dunque potuto creare un meccanismo d'auto-replicazione per il nostro FPGA. A questo punto, abbiamo rivolto la nostra attenzione alla seconda proprietà bio-ispirata che vorremmo includere nel nostro FPGA: l'auto-riparazione.

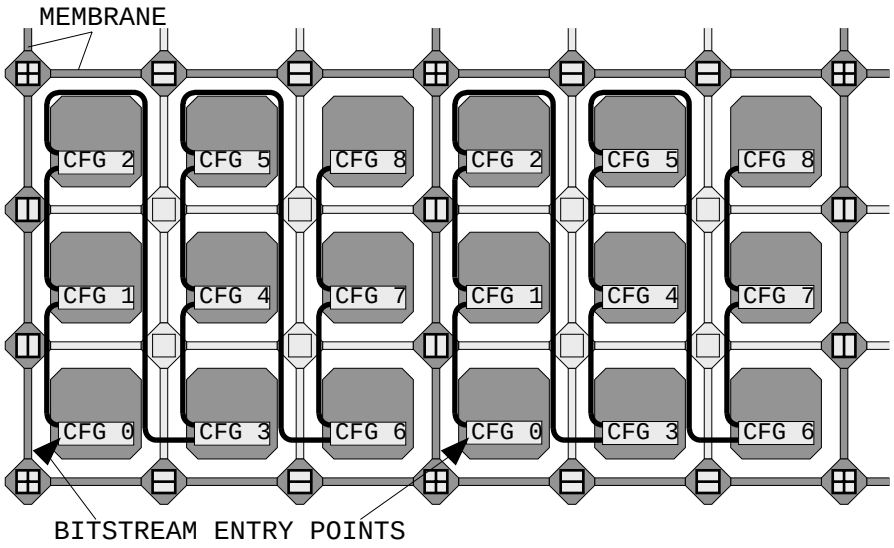


Figura 17: La membrana è usata per dirigere la propagazione della sequenza di configurazione.

4.4 L'auto-riparazione

Lo sviluppo di un meccanismo di auto-riparazione per MuxTree ha posto problemi diversi rispetto all'auto-replicazione: come abbiamo menzionato, l'auto-riparazione è una proprietà molto più "convenzionale" per i circuiti digitali, ed è possibile trovare una quantità considerevole di ricerca su questo soggetto. La concezione di un tale meccanismo non ha dunque richiesto altrettanta ricerca originale. D'altra parte, le esigenze del nostro progetto hanno introdotto notevoli difficoltà tecniche, particolarmente per lo sviluppo di un meccanismo di *auto-test* (cioè di un meccanismo che permetta a MuxTree di accorgersi della presenza di un errore nel circuito e di identificarne la posizione esatta, un requisito indispensabile per l'auto-riparazione).

4.4.1 L'auto-test in MuxTree

Un'abbondante letteratura scientifica è disponibile sul soggetto dell'auto-test nei circuiti digitali in generale [1] e degli FPGA in particolare [2][15][32]. Pur approfittando al massimo di tale ricerca per il nostro progetto, le esigenze particolari del nostro sistema ci hanno impedito l'uso diretto di meccanismi già esistenti.

Tra le esigenze più limitanti, menzioneremo la necessità di determinare non solo la presenza di un errore, ma anche la sua posizione esatta (*fault location*), la completa omogeneità del nostro circuito (necessaria per poter realizzare facilmente reti di processori), che impedisce l'uso di unità di controllo centralizzate, molto comuni nei sistemi di auto-riparazione convenzionali, e soprattutto il nostro desiderio di effettuare il test senza interrompere l'operazione del circuito (*on-line testing*).

L'analisi separata dei tre sub-circuiti di MuxTree ci ha portati a sviluppare l'approccio seguente [35][36][37]:

- Considerando la sua taglia relativamente ridotta (occupa all'incirca il 10% della superficie totale di un elemento), abbiamo deciso di realizzare il test della funzione programmabile usando la *duplicazione* (Figura 18), una tecnica molto comune tanto nella concezione dei circuiti elettronici quanto nei sistemi biologici (per esempio, la doppia elica dell'ADN). Un semplice confronto delle uscite delle due copie della funzione rivelerà così la presenza di un errore. Abbiamo poi dovuto aggiungere una terza copia del flip-flop per assicurare l'uso del valore corretto nell'auto-riparazione.
- Soluzioni al problema di effettuare un test *on-line* delle connessioni in un FPGA esistono, ma richiedono invariabilmente una quantità importante di ridondanza. Anche se non escludiamo la possibilità di introdurre un tale test in futuro, abbiamo stimato che i vantaggi di un tale meccanismo non giustificavano il materiale aggiuntivo per un elemento di taglia ridotta quale MuxTree. Di conseguenza, abbiamo deciso di tralasciare il test delle connessioni nella versione attuale del nostro circuito.
- Il test del registro di configurazione pone problemi simili, ma la sua taglia (approssimativamente 80% della superficie dell'elemento) impone un test almeno parziale e allo stesso tempo impedisce l'uso della duplicazione. Il meccanismo che abbiamo finalmente adottato è basato sull'uso di una *configurazione di test*, inviata in tutti i registri di configurazione prima dell'arrivo della sequenza di configurazione. Questo meccanismo è capace di identificare la presenza di errori nei registri di configurazione con una quantità estremamente ridotta di logica addizionale.

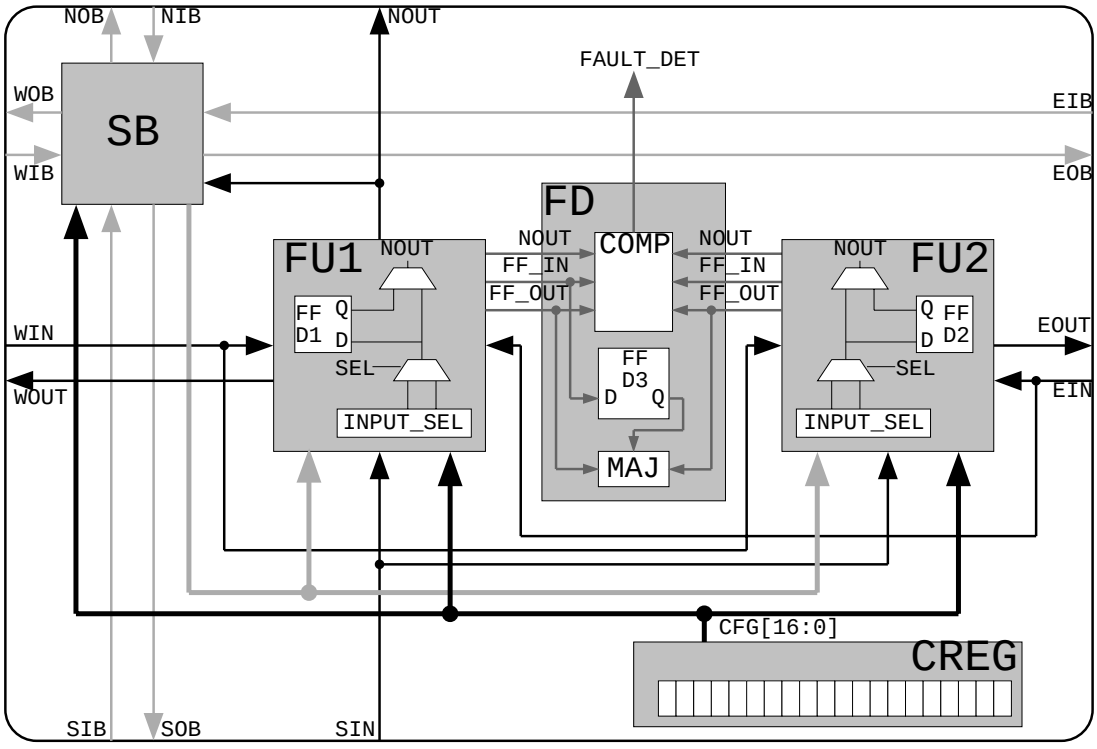


Figura 18: La logica addizionale necessaria per effettuate l'auto-test in un elemento MuxTree.

In conclusione, pur non essendo riusciti a rispettare completamente il nostro desiderio di avere un test in corso di funzionamento (il test del registro ha luogo in una fase separata durante la configurazione del circuito), abbiamo ottenuto un sistema di test estremamente semplice e compatto (l'hardware aggiuntivo rappresenta meno del 20% della superficie totale di un elemento) che, come vedremo, è perfettamente compatibile con il nostro meccanismo di auto-riparazione.

4.4.2 L'auto-riparazione in MuxTree

Come per l'auto-test, esistono numerosi approcci all'auto-riparazione di circuiti digitali basati su matrici bidimensionali di elementi identici [10][17][27]. La maggior parte di essi è basata su due meccanismi: la *ridondanza* e la *riconfigurazione*. Data l'impossibilità di riparare fisicamente l'hardware, la presenza di elementi di riserva è indispensabile (ridondanza), così come un meccanismo che ridiriga le connessioni tra gli elementi per permettere la sostituzione degli elementi difettosi (riconfigurazione). Tali meccanismi sono dunque anche alla base del nostro sistema di auto-riparazione, nonostante le esigenze non-convenzionali del nostro FPGA.

Per implementare la ridondanza in modo efficace, abbiamo sfruttato nostro meccanismo di auto-replicazione. È infatti relativamente semplice modificare la membrana cellulare per definire quali colonne nella matrice conterranno elementi di riserva (Figura 19). Con la semplice aggiunta di uno stato al nostro automa, possiamo non solo limitare la riconfigurazione all'interno di un singolo blocco (una caratteristica desiderabile) ma anche di configurare la tolleranza ai difetti. Infatti, con modifiche triviali alla sequenza di stati che costruisce la membrana possiamo alterare la frequenza di colonne di riserva, e quindi la tolleranza del sistema. Senza modificare la sequenza di configurazione dell'FPGA, che è separata dalla sequenza di stati della membrana (un vantaggio notevole, dato il tempo richiesto per la generazione di una sequenza), possiamo dunque introdurre diversi gradi di ridondanza, da zero (nessuna colonna di riserva) fino a 100% (una colonna di riserva per ogni colonna attiva).

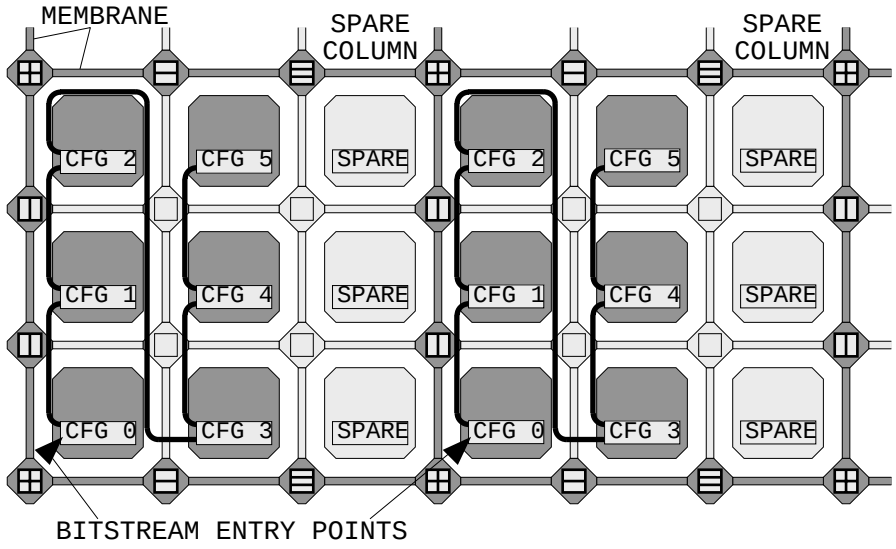


Figura 19: La membrana cellulare può definire la frequenza e la posizione delle colonne di riserva.

Per sfruttare gli elementi di riserva, abbiamo inoltre bisogno di un meccanismo per trasferire l'informazione contenuta in un elemento difettoso (la sua configurazione più il valore memorizzato dal suo flip-flop) a un elemento di riserva.

Il nostro meccanismo di auto-riparazione (Figura 20) si basa sulla riconfigurazione della rete di connessioni e la sostituzione dell'elemento difettoso da parte del suo vicino di destra: la configurazione dell'elemento difettoso e il valore memorizzato nel suo flip-flop sono trasmessi al vicino. La configurazione di quest'ultimo è essa stessa trasmessa verso la destra, e così via fino al raggiungimento di un elemento di riserva.

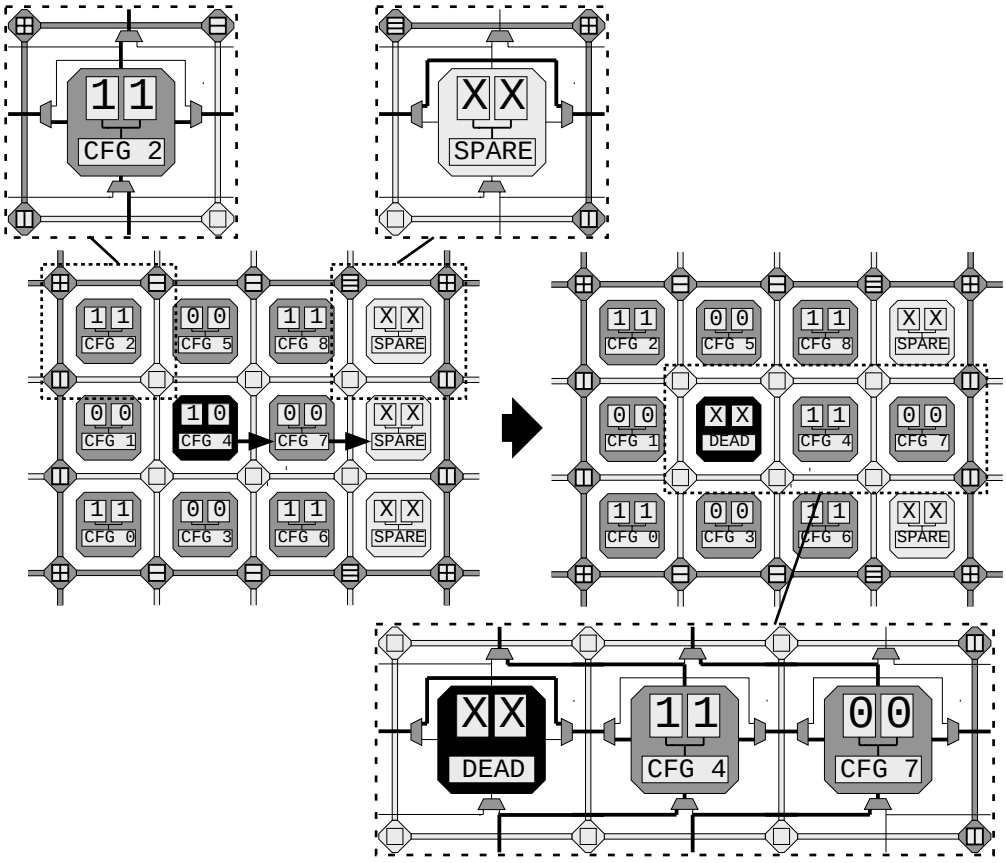


Figura 20: L'informazione memorizzata in un elemento difettoso e negli elementi all sua destra sono spostati fino al raggiungimento di una colonna di riserva.

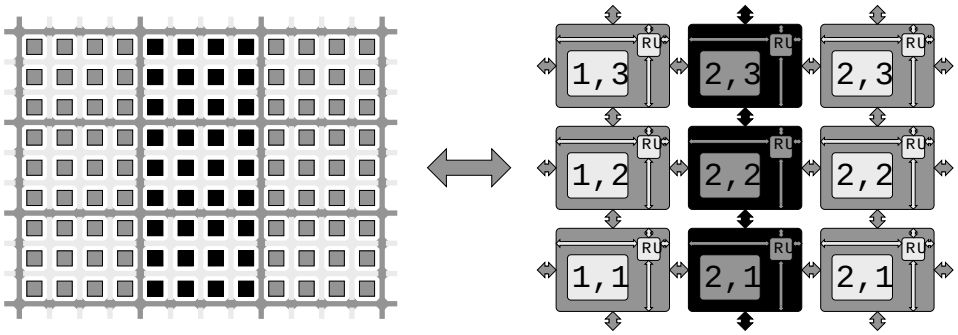


Figura 21: La morte di una colonna di blocchi al livello molecolare corrisponde alla morte di una colonna di cellule al livello cellulare.

Al termine della configurazione, l'elemento difettoso "muore": tutte le connessioni del circuito sono modificate in modo da evitarlo, un'operazione che può essere effettuata molto semplicemente deviando le connessioni nord-sud verso la destra e rendendo l'elemento trasparente alle connessioni est-ovest. La matrice riconfigurata può allora riprendere l'esecuzione dell'applicazione dal punto in cui si era fermata in seguito all'identificazione dell'errore. Quando trova un errore, quindi, il circuito si disattiva per il tempo richiesto dalla riconfigurazione, un po' come un organismo indebolito da una malattia.

4.4.3 MuxTree e MicTree

Il meccanismo di auto-riparazione che abbiamo introdotto nel livello molecolare è dunque molto versatile. Nonostante ciò, esso è necessariamente limitato e può fallire sia per saturazione (se tutti gli elementi di riserva sono usati) sia per l'identificazione di un errore non riparabile (per esempio, un errore nelle connessioni). In tal caso, diventa necessario attivare il meccanismo di auto-riparazione al livello cellulare descritto più sopra.

A tal scopo, abbiamo inserito nel nostro sistema un segnale globale `KILL` che si propaga in una colonna di blocchi e "uccide" tutti gli elementi sul suo cammino. Dal momento che ogni blocco contiene una delle nostre cellule artificiali, una tale operazione equivale a disattivare una colonna di cellule, un evento che fa scattare il calcolo delle coordinate in tutte le cellule del sistema. In altre parole, esso attiva il meccanismo di auto-riparazione al livello cellulare (Figura 21).

La tolleranza ai difetti del nostro sistema è quindi basata non su un singolo meccanismo di auto-riparazione, necessariamente limitato, ma piuttosto su due meccanismi separati che cooperano per evitare che un errore possa causare la distruzione dell'intero sistema.

5 Prospettive biologiche

Per realizzare i nostri sistemi bio-ispirati, abbiamo dovuto scegliere equivalenti elettronici per molti termini e concetti biologici. Spesso, le nostre scelte sono state dettate più dalle esigenze dell'ingegneria che dal nostro desiderio di trarre ispirazione dai sistemi naturali. Nonostante ciò, abbiamo potuto notare che, in molti casi, la soluzione più efficiente dal punto di vista ingegneristico è molto simile alla soluzione adottata dalla natura.

In questa sezione, analizzeremo alcune delle caratteristiche principali del nostro sistema dal punto di vista dell'ispirazione biologica, prendendo nota dei maggiori punti in comune tra le nostre macchine e i loro equivalenti naturali.

5.1 L'ispirazione biologica nella ricerca di von Neumann

Nel 1958, un anno dopo la morte di John von Neumann, due eventi fondamentali occorsero nella storia della biologia molecolare:

1. Francis Crick, uno degli scopritori della struttura a doppia elica dell'ADN, propose quello che chiamò il *dogma centrale* della biologia molecolare: le proteine non sono generate direttamente dai geni, ma necessitano un intermediario, l'ARN [8]. L'ADN (acido deossiribonucleico) contiene l'informazione necessaria per il corretto funzionamento dell'organismo. Per esempio, nel caso di un organismo multicellulare, esso contiene (tra l'altro) le informazioni necessarie per la differenziazione cellulare (e quindi per la crescita dell'organismo da una singola cellula, lo zigote, a un individuo maturo), per la riproduzione e, finalmente, per la morte dell'organismo. Enzimi trascrivono queste informazioni dall'ADN per generare un'altra famiglia di molecole, chiamate acidi ribonucleici (ARN). A questo punto, esse sono tradotte per generare specifiche proteine, cioè le molecole responsabili dell'operazione della cellula. L'ADN è quindi il portatore di informazione, l'ARN è il messaggero, e le proteine sono gli esecutori (con poche eccezioni). Insomma, il dogma centrale afferma che (Figura 22): l'ADN genera l'ARN (processo di *trascrizione*), dopodiché l'ARN genera le proteine (processo di *traduzione*).

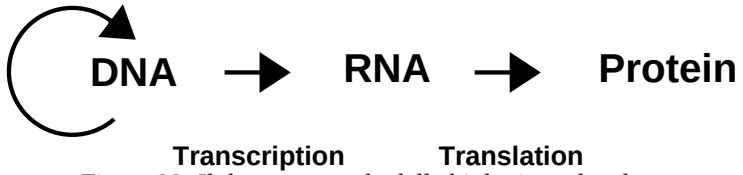


Figura 22: Il dogma centrale della biologia molecolare.

2. Roberts [29] introdusse il nome *ribosomi* per indicate quegli elementi che interpretano l'informazione genetica, cioè traducono la sequenza dell'ARN (una catena unidimensionale di nucleotidi) per produrre la proteina appropriata (una struttura tridimensionale di aminoacidi).

Nel suo provocativo libro *La Teoria Semantica dell'Evoluzione* [4], Marcello Barbieri fa le osservazioni seguenti:

- Il ruolo dei ribosomi nella biologia molecolare non è stato sufficientemente enfatizzato.
- In ogni cellula, la maggior parte dei nucleotidi si occupano della produzione di ribosomi.
- Barbieri afferma che: "... Per la sintesi proteica la natura è arrivata a creare nientemeno che il capolavoro dei suoi sistemi molecolari. I ribosomi sono i suoi gioielli, l'apice di tutta l'ingegneria molecolare che la natura ha investito nella vita."
- Finalmente, Barbieri propone una nuova teoria dell'evoluzione, basata sulla trinità *genotipo-ribotipo-fenotipo* (Figura 23).

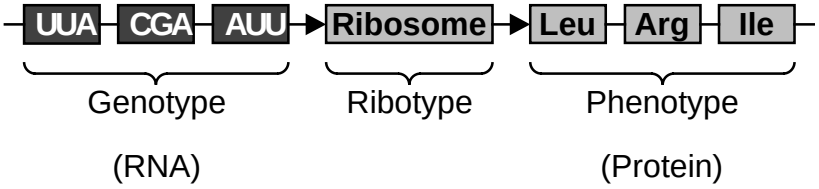


Figura 23: La trinità genotipo-ribotipo-fenotipo.

I ribosomi sono capaci, in generale, di tradurre catene di ARN di qualsiasi lunghezza in proteine e in particolare di interpretare specifiche sequenze di ARN, l'ARN messaggero delle proteine ribosomiali, per produrre una copia esatta del ribosoma stesso, un tipico esempio di auto-replicazione (Figura 24).

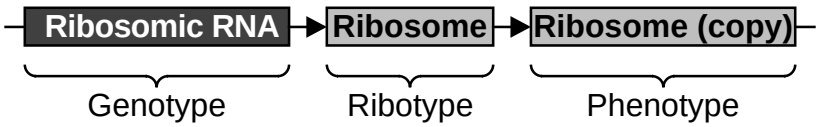


Figura 24: L'auto-replicazione del ribosoma.

La ricerca visionaria di von Neumann [40] negli anni quaranta è anteriore alle pubblicazioni di Crick, Roberts, Barbieri, e altri biologi. Siamo dell'opinione che il suo messaggio fondamentale a proposito dell'auto-replicazione degli automi artificiali risiede nell'architettura del suo costruttore universale, che non è altro se non la versione artificiale del ribosoma biologico. Possiamo allora distinguere la trinità genotipo-ribotipo-fenotipo nell'automa cellulare di von Neumann (Figura 25):

- Il genotipo è la banda di memoria dell'automa, che contiene la descrizione (il genoma) della macchina da costruire;
- Il ribotipo è il costruttore universale stesso;
- Il fenotipo è la costruzione, nello spazio cellulare, della macchina descritta sulla banda.

L'auto-replicazione del costruttore universale ha dunque luogo in modo analogo all'auto-replicazione cellulare in natura: la descrizione (genotipo) memorizzata sulla banda è tradotta da un ribosoma (ribotipo) per generare il nuovo costruttore (fenotipo).

Siamo dunque dell'opinione che il messaggio fondamentale di von Neumann è [22]:

Genotipo + Ribotipo = Fenotipo

Da questo punto di vista, la somiglianza tra le nostre fonti d'ispirazione principali (l'ontogenesi e l'automa di von Neumann) dovrebbe essere apparente: nonostante la mancanza di differenziazione nella macchina di von Neumann, quest'ultima presenta notevoli somiglianze con il meccanismo ontogenetico della divisione cellulare.

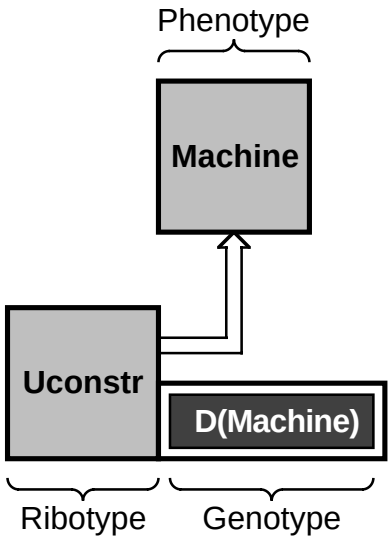


Figura 25: La trinità genotipo-ribotipo-fenotipo nel costruttore universale di von Neumann.

5.2 L'ispirazione biologica in Embryonics

Un essere umano è composto di approssimativamente 60 mila miliardi ($6x10^{13}$) di cellule. In ogni istante, in ognuna di queste cellule, il genoma, un nastro di 2 miliardi di caratteri, è interpretato per produrre le proteine necessarie alla sopravvivenza dell'organismo, un processo che si ripete in continuazione dalla concezione alla morte dell'individuo.

Questo processo, impressionante per complessità e precisione, è basato su processi completamente discreti: la struttura chimica dell'ADN (il substrato chimico del genoma) è una sequenza di quattro basi, indicate con le lettere A (adenina), C (citosina), G (guanina), e T (timina). Ogni gruppo di tre basi è interpretato nella cellula per produrre un aminoacido particolare, uno dei componenti della proteina finale.

La somiglianza tra il genoma naturale e un programma informatico è immediata (ed è infatti uno della motivazioni iniziali del progetto Embryonics): un programma è una sequenza di due stati, indicati con le cifre 0 e 1, divisa in gruppi (le istruzioni) che sono interpretati dai processori per eseguire una data funzione.

Considerando questo parallelo, possiamo vedere come il genoma artificiale sia al centro della metodologia di concezione dei nostri sistemi bio-ispirati. È dunque fondamentale comprendere la sua struttura.

Per evitare confusione, abbiamo finora usato il termine genoma per indicare il programma eseguito in ogni processore. Con l'introduzione del livello molecolare, questa definizione deve essere ampliata. In natura, infatti, il genoma contiene tutte le informazioni genetiche necessarie all'individuo, comprese le istruzioni per la costruzione dell'organismo. Di conseguenza, per essere accurati, il nostro genoma artificiale deve contenere anche le informazioni necessarie alla costruzione del nostro sistema, e in particolare le sequenze di configurazione della membrana cellulare e della matrice molecolare.

Per essere più precisi, la nostra metodologia di concezione di macchine multicellulari richiede tre fasi successive (Figura 26).

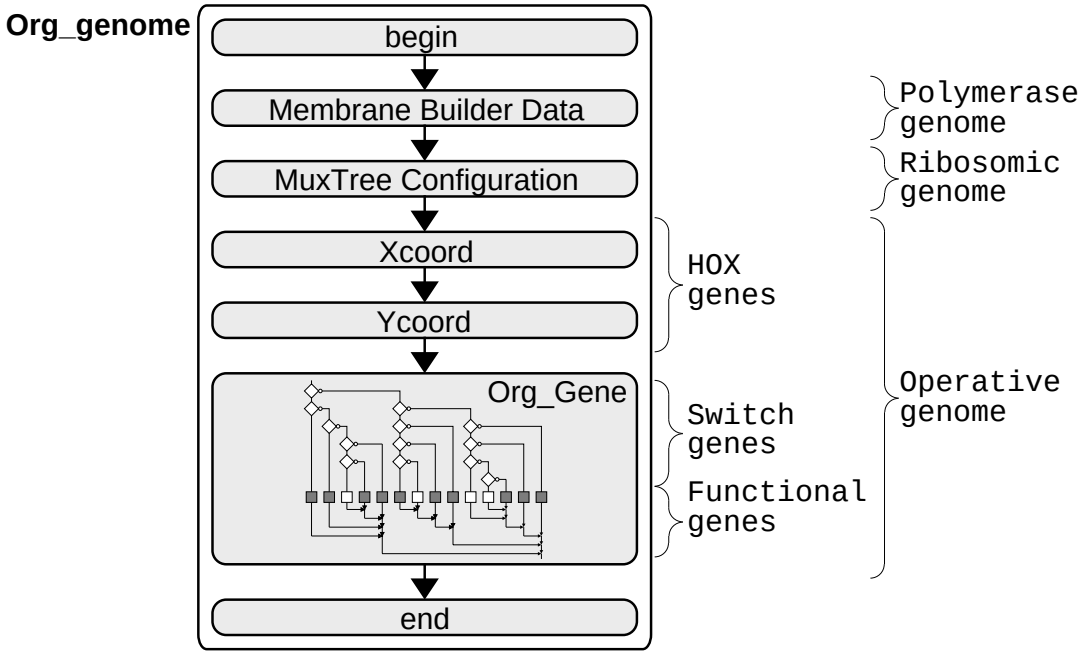


Figura 26: Il completo genoma necessario per la costruzione di un organismo artificiale, comprese le informazioni relative al livello molecolare.

In una prima fase, la descrizione della macchina (la funzionalità desiderata del nostro sistema) è analizzata per determinare la sua realizzazione in una matrice omogenea di cellule. Il software (il programma) e l'hardware (l'architettura della cellula) sono adattate all'applicazione. In termini biologici, questo programma è la parte *operativa* del nostro genoma, ed è costituita di tre sub-programmi:

- I geni di coordinata (*coordinate genes*), incaricati di calcolare le coordinate e le condizioni iniziali della cellula, sono simili agli *omeobox* o *geni HOX* che recenti ricerche hanno rivelato essere responsabili per la definizione dell'architettura generale degli esseri viventi [41]. Nell'esempio della bandiera svizzera presentato più sopra, questi geni corrispondono ai sub-programmi `Xcoord` e `Ycoord`.
- I geni di selezione (*switch genes*) determinano quale parte del genoma deve essere eseguita in funzione della posizione della cellula nella matrice (cioè in funzione delle sue coordinate) [9]. Nel nostro esempio, questi geni corrispondono agli elementi di test dell'albero di decisione binario.
- I geni funzionali (*functional genes*), che realizzano la funzionalità del nostro organismo artificiale, sono equivalenti ai geni della parte codificante di un genoma naturale. Nel nostro esempio, questi geni corrispondono agli elementi di uscita dell'albero di decisione binario (i quali sono estremamente semplici per la bandiera svizzera, ma possono diventare molto complessi per applicazioni reali).

In una seconda fase, l'architettura delle cellule è realizzata usando la matrice di molecole MuxTree. Questa operazione genera la sequenza di configurazione per gli elementi della matrice necessari a implementare una singola cellula (in media, una cellula richiede qualche centinaio di molecole).

Se consideriamo la nostra cellula artificiale come l'equivalente del ribosoma di una cellula naturale (un parallelo che sarà esplorato più in dettaglio nella conclusione), che interpreta la parte operativa del genoma (il genotipo) per eseguire una data operazione (il fenotipo), la sequenza di configurazione può essere vista come la parte ribosomiale del genoma finale. Colonne di riserva possono esservi inserite per aumentare la tolleranza ai difetti del circuito.

Una volta che le dimensioni della cellula (il numero di molecole necessario per la sua realizzazione) sono state definite, possiamo definire, in una terza fase, la sequenza che genera la membrana cellulare. Dal momento che questa informazione permette la creazione di multiple cellule a partire dalla descrizione di una sola, la sequenza può essere considerata equivalente alla parte polimerase del genoma.

A partire della matrice molecolare (la sola componente hardware rimasta nel nostro sistema), la configurazione del sistema procede in direzione opposta:

- la parte polimerase è inserita per creare la membrana che divide le cellule;
- la parte ribosomiale è inviata nel circuito per configurare l'FPGA molecolare e ottenere la struttura hardware delle nostre cellule;
- la parte operativa del genoma è memorizzata in ogni cellula per renderla capace di eseguire l'applicazione voluta.

L'esistenza di queste categorie differenti di geni è la conseguenza di esigenze puramente logiche derivanti dalla concezione del nostro automa multicellulare, e la loro somiglianza a strutture biologiche è, a nostra opinione, una conferma dell'efficienza dei sistemi naturali da un lato e della validità dell'ispirazione biologica del nostro approccio dall'altro.

6 Conclusione

I nostri sistemi, nella loro versione corrente, sono basati su tre livelli di complessità: gli organismi, le cellule e le molecole (Figura 27).

L'organismo artificiale è un sistema di calcolo completo che consiste di una matrice bidimensionale di cellule (la matrice cellulare) che operano in parallelo per eseguire un'applicazione definita dall'utilizzatore. La taglia (il numero di cellule) di un organismo è programmabile e, se un numero sufficiente di cellule sono disponibili, gli organismi si replicano automaticamente. Dal momento che ogni copia dell'organismo ha la stessa funzionalità, questo meccanismo può essere usato per dare al sistema una tolleranza intrinseca agli errori.

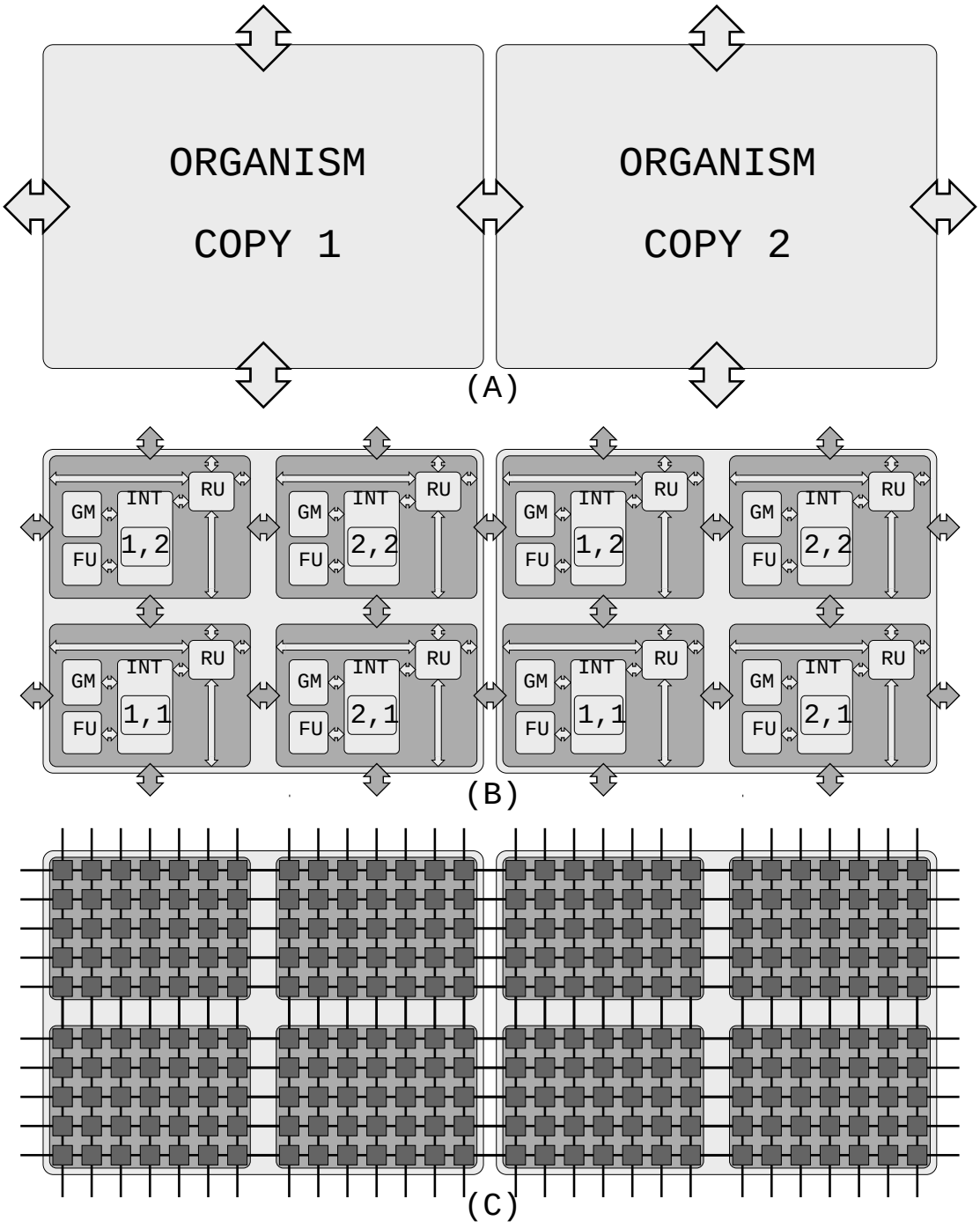


Figura 27: I tre livelli del progetto Embryonics: (A) organismi, (B) cellule e (C) molecole.

Da un punto di vista biologico, i nostri organismi sono dunque esseri multicellulari (contrariamente agli organismi unicellulari di von Neumann). Essi sono, nella maggioranza dei casi, organismi specializzati, dal momento che sono concepiti per un'applicazione specifica (non c'è tuttavia nessun ostacolo intrinseco alla realizzazione di organismi a uso generale). Il meccanismo di replicazione può essere considerato come una garanzia della sopravvivenza della specie, anche se (nella versione attuale) il fenomeno dell'evoluzione non è presente. Con una struttura cellulare adeguata, gli organismi possono essere capaci di apprendimento.

La cellula artificiale è un processore semplice ma universale, capace cioè di eseguire qualsiasi applicazione voluta. La strutture hardware della cellula, realizzata con una matrice bidimensionale di molecole (la matrice molecolare), può essere adattata all'applicazione, così come può esserlo la sua taglia (il numero di molecole necessario alla sua realizzazione), definita da una membrana cellulare. Tutti i processori eseguono lo stesso programma (il genoma artificiale) e selezionano quali istruzioni eseguire in funzione della loro posizione (cioè delle loro coordinate nella matrice cellulare). Dal momento che il genoma è duplicato in ogni cellula e che tutte le cellule hanno la stessa struttura hardware, una cellula morta può essere rimpiazzata da un altra semplicemente ricalcolando le coordinate della matrice, un meccanismo che conferisce al circuito un'ulteriore resistenza agli errori.

Il parallelo tra le nostre cellule artificiali e il loro equivalente biologico è relativamente solido [23][24]. Come nella natura, le nostre cellule sono capaci di moltiplicarsi (divisione cellulare), dal momento che molteplici copie sono generate se un numero sufficiente di molecole sono disponibili, di differenziarsi, dal momento che la loro funzionalità varia a seconda della loro posizione, e di scambiarsi segnali che alterano il loro comportamento. Esse contengono un nucleo (la memoria del genoma), un citoplasma (il processore) e una membrana. La matrice cellulare può inoltre continuare a operare in presenza di una quantità non triviale di difetti con un processo simile a quello biologico della cicatrizzazione.

La molecola è un elemento di FPGA basato su un multiplexer che può essere programmato per realizzare qualunque circuito digitale. La sua struttura regolare e il meccanismo che permette di definire la membrana cellulare lo rendono una piattaforma ideale per la realizzazione di matrici cellulari. Un semplice meccanismo di test e di riconfigurazione permette la sostituzione di molecole morte con molecole di riserva, assicurando così un ulteriore livello di tolleranza ai difetti.

Il parallelo biologico per le nostre molecole artificiali è probabilmente meno solido che per le cellule, ma ciò nonostante esistono numerose caratteristiche comuni. Il registro di configurazione, per esempio, determina la struttura della molecola (attraverso l'attivazione o la disattivazione di parti del circuito). Come in biologia, il numero teorico di molecole differenti è enorme (17 bit permetterebbero 2^{17}=131072 configurazioni differenti), ma in pratica solo un numero relativamente limitato di tutte le configurazioni possibili è usato. Per quanto concerne il meccanismo di auto-riparazione, il parallelo con la biologia è immediato: la struttura a doppia elica dell'ADN è un esempio tipico di duplicazione usata per identificare la presenza di errori, e la ridondanza è un meccanismo estremamente comune nei sistemi naturali.

Visto nel suo complesso, è nostra opinione che il nostro sistema a tre livelli presenta una somiglianza considerevole con i sistemi biologici, una somiglianza che potrebbe in futuro essere ancora aumentata, per esempio con l'introduzione di meccanismi quali l'evoluzione o l'apprendimento al livello degli organismi o delle cellule, o anche con l'aggiunta di un quarto livello (atomico) al sistema.

In ogni caso, prima di introdurre ulteriori cambiamenti fondamentali, una modifica più immediata è necessaria. Infatti, lo scopo principale del progetto, cioè lo sviluppo di un sistema multicellulare paragonabile al costruttore universale di von Neumann, non è stato ancora completamente raggiunto nella versione attuale. Abbiamo infatti visto che l'automa di von Neumann rispetta in pieno il dogma centrale della biologia molecolare (genotipo + ribotipo = fenotipo), mentre altrettanto non si può ancora dire del nostro sistema.

Se consideriamo la struttura della nostra cellula artificiale come il ribotipo, il genoma artificiale come il genotipo, e la funzionalità della cellula come il fenotipo, il sistema sembra rispettare, almeno in apparenza, il dogma centrale[10]. Il punto debole risiede nel processo di auto-replicazione cellulare: mentre il costruttore di von Neumann può generare copie che sono esse stesse capaci di generare ulteriori copie, le nostre cellule sono, per il momento, capaci di auto-replicarsi solo con l'aiuto di una sequenza di configurazione (il genoma polimerase e il genoma ribosomiale) introdotta dall'esterno della matrice.

In altre parole, le nostre cellule non sono capaci di *dirigere la propria auto-replicazione*: sarà dunque necessario sviluppare un meccanismo, situato all'interno delle cellule stesse, capace di generare una o più copie della cellula senza alcun intervento esterno.

Realizzare un tale meccanismo richiederà uno sforzo notevole sia dal punto di vista concettuale che da quello tecnico (dal momento che avrà un effetto su tutti i livelli del nostro sistema), e rappresenta il prossimo obiettivo maggiore del progetto Embryonics: con l'introduzione di cellule capaci di auto-replicarsi senza assistenza esterna, il nostro sistema rispetterà in pieno il dogma centrale della biologia molecolare e diventerà una completa realizzazione multicellulare della macchina di von Neumann.

Ringraziamenti

Siamo molto grati a tutti coloro che hanno contribuito allo sviluppo del nostro progetto. Tra di essi, menzioneremo: Dominik Madon, Eduardo Sanchez, Moshe Sipper, e Jacques Zahnd, del Logic Systems Laboratory dello Swiss Federal Institute of Technology a Losanna, Svizzera, e Serge Durand, Pierre Marchal, e Christian Piguet, del Centre Suisse d'Electronique et de Microtechnique SA a Neuchâtel, Svizzera.

Vogliamo inoltre ringraziare Marcello Barbieri dell'Università di Ferrara per i suoi preziosi suggerimenti e per il suo incoraggiamento al nostro progetto.

Questo lavoro è stato finanziato in parte dai grant 20-42270.94 e 2000-049349.96 della Swiss National Science Foundation.

10 Tra l'altro, è interessante notare come l'evoluzione del nostro sistema (il formato delle istruzioni, la struttura molecolare, ecc.) sia basata sull'evoluzione della struttura delle nostre cellule, e quindi sull'evoluzione del nostro ribotipo, un'osservazione che coincide con la tesi esposta da M. Barbieri [4].

Bibliografia

[1] M. Abramovici, M. A. Breuer, A. D. Friedman [1990]. *Digital Systems Testing and Testable Design*. Computer Science Press, New York, 1990.

[2] M. Abramovici, C. Stroud [1995]. "No-overhead BIST for FPGAs". In *Proc. 1st IEEE International On-Line Testing Workshop*, pp. 90-92, 1995.

[3] S. B. Akers [1978]. "Binary decision diagrams". *IEEE Transactions on Computers*, c-27(6), June 1978, pp. 509-516.

[4] M. Barbieri [1985]. *La Teoria Semantica dell'Evoluzione*. Ed. Boringhieri, Torino, Italy, 1985. Pubblicato in inglese come *The Semantic Theory of Evolution*. Harwood Academic Publishers, Chur, Switzerland, 1985.

[5] J.-L. Beuchat, J.-O. Haenni [1998]. "Von Neumann's 29-State Cellular Automaton: A Hardware Implementation". *IEEE Trans. on Education*. Submitted.

[6] S.D. Brown, R.J. Francis, J. Rose, Z.G. Vranesic [1992]. *Field-Programmable Gate Arrays*. Kluwer Academic Publishers, Boston, 1992.

[7] A. Burks, ed [1970]. *Essays on Cellular Automata*. University of Illinois Press, Urbana, IL, 1970.

[8] F. H. C. Crick [1958]. *On protein synthesis*. Symposia of the Society for Experimental Biology, 12:548-555, 1958.

[9] S.F. Gilbert [1991]. *Developmental Biology*. Sinauer Associates, Inc., MA, 3rd ed., 1991.

[10] F. Hanchek, S. Dutt [1998]. "Methodologies for Tolerating Cell and Interconnect Faults in FPGAs". *IEEE Transactions on Computers*, v. 47, n. 1, January 1998.

[11] J.P. Hayes [1993]. Introduction to Digital Logic Design. Addison-Wesley, Reading, MA, 1993.

[12] M. H. Hassoun [1995]. *Fundamentals of Artificial Neural Networks*. The MIT Press, Cambridge, MA, 1995.

[13] T. Higuchi, M. Iwata, I. Kajitani, H. Iba, Y. Hirao, T. Furuya, B. Manderick [1996]. "Evolvable Hardware and its Application to Pattern Recognition and Fault-Tolerant Systems". In E. Sanchez, M. Tomassini, eds., *Towards Evolvable Hardware*, Lecture Notes in Computer Science, Springer, Berlin, 1996, pp. 118-135.

[14] J. E. Hopcroft, J. D. Ullman [1979]. *Introduction to Automata Theory Languages and Computation*. Addison-Wesley, Redwood City, CA, 1979.

[15] W.K. Huang, F. Lombardi [1996]. "An Approach for Testing Programmable/Configurable Field Programmable Gate Arrays". *IEEE VLSI Test Symposium*, 1996.

[16] J. R. Koza, F. H. Bennett III, D. Andre, M. A. Keane [1996]. "Automated {WYWIWYG} Design of Both the Topology and Component Values of Electrical Circuits Using Genetic Programming". In *Genetic Programming 1996: Proceedings of the First Annual Conference*, The MIT Press, Cambridge, MA, 1996, pp.123-131.

[17] J. Lach, W.H. Mangione-Smith, M. Potkonjak [1998]. “Efficiently Supporting Fault-Tolerance in FPGAs”. *Proc. FPGA 98*, Monterey, CA, February 1998, pp. 105-115.

[18] C. G. Langton [1984]. “Self-Reproduction in Cellular Automata”. *Physica 10D*, pp.135-144, 1984.

[19] D. Mange [1992]. *Microprogrammed Systems: An Introduction to Firmware Theory*. Chapman & Hall, London, 1992.

[20] D. Mange, M. Goeke, D. Madon, A. Stauffer, G. Tempesti, S. Durand [1996]. “Embryonics: A New Family of Coarse-Grained Field-Programmable Gate Array with Self-Repair and Self-Reproducing Properties”. In E. Sanchez, M. Tomassini, eds., *Towards Evolvable Hardware*, Lecture Notes in Computer Science, Springer, Berlin, 1996, pp. 197-220.

[21] D. Mange, D. Madon, A. Stauffer, G. Tempesti [1997]. “Von Neumann Revisited: A Turing Machine with Self-Repair and Self-Reproduction Properties”. *Robotics and Autonomous Systems*, Vol. 22, No. 1, 1997, pp. 35-58.

[22] D. Mange, M. Sipper [1998]. “Von Neumann’s Quintessential Message: Genotype + Ribotype = Phenotype”. *Artificial Life Journal*. Accepted.

[23] D. Mange, M. Sipper, P. Marchal [1998]. “Embryonic Electronics”. Submitted.

[24] D. Mange, M. Tomassini, eds [1998]. *Bio-inspired Computing Machines: Towards Novel Computational Architectures*. Presses Polytechniques et Universitaires Romandes, Lausanne, Switzerland, 1998.

[25] P. Marchal, P. Nussbaum, C. Piguet, S. Durand, D. Mange, E. Sanchez, A. Stauffer, G. Tempesti [1996]. “Embryonics: The Birth of Synthetic Life”. In E. Sanchez, M. Tomassini, eds., *Towards Evolvable Hardware*, Lecture Notes in Computer Science, Springer, Berlin, 1996, pp. 166-197.

[26] Maxfield, Clive [1995]. *Bebop to the Boolean Boogie*. HighText Publications, Solana beach, CA, 1995.

[27] R. Negrini, M. G. Sami, R. Stefanelli [1989]. *Fault Tolerance Through Reconguration in VLSI and WSI Arrays*. The MIT Press, Cambridge, MA, 1989.

[28] J.-Y. Perrier, M. Sipper, J. Zahnd [1996]. “Toward a Viable, Self-Reproducing Universal Computer”. *Physica 97D*, pp.335-352, 1996.

[29] R. B. Roberts, ed [1958]. *Microsomal Particles and Protein Synthesis: Papers Presented at the First Symposium of the Biophysical Society*. Pergamon Press, 1958.

[30] E. Sanchez, D. Mange, M. Sipper, M. Tomassini, A. Perez-Uribe, A. Stauffer [1997]. “Phylogeny, Ontogeny, and Epigenesis: Three Sources of Biological Inspiration for Softening Hardware”. In T. Higuchi, M. Iwata, W. Liu, eds., *Proc. 1st Int. Conference on Evolvable Systems: From Biology to Hardware (ICES96)*, Lecture Notes in Computer Science, vol. 1259, Springer-Verlag, Berlin, 1997, pp. 35-54.

[31] M. Sipper, D. Mange, A. Stauffer [1997]. “Ontogenetic Hardware”. *BioSystems* 44 (1997), pp. 193-207.

[32] C. Stroud, S. Konala, M. Abramovici [1996]. “Using ILA testing for BIST in FPGAs”. *Proc. 2nd IEEE International On-Line Testing Workshop*, Biarritz, July 1996.

[33] A. Stauffer [1997]. “Membrane building and binary decision machine implementation”. *Technical Report 247*, Computer Science Department, EPFL, Lausanne, 1997.

[34] G. Tempesti [1995]. “A New Self-Reproducing Cellular Automaton Capable of Construction and Computation”. *Proc. 3rd European Conference on Artificial Life*, Lecture Notes in Artificial Intelligence, 929, Springer Verlag, Berlin, 1995, pp. 555-563.

[35] G. Tempesti, D. Mange, A. Stauffer [1997]. “A Robust Multiplexer-Based FPGA Inspired by Biological Systems”. *Journal of Systems Architecture: Special Issue on Dependable Parallel Computer Systems*, EUROMICRO, 43(10), 1997.

[36] G. Tempesti, D. Mange, A. Stauffer [1998]. “Self-Replicating and Self-repairing Multicellular Automata”. *Artificial Life*. Accepted.

[37] G. Tempesti [1998]. *A Self-Repairing Multiplexer-Based FPGA Inspired by Biological Processes*. Ph.D. Thesis, Swiss Federal Institute of Technology, Lausanne, 1998.

[38] A. Thompson [1996]. “Silicon Evolution”. In *Genetic Programming 1996: Proceedings of the First Annual Conference*, The MIT Press, Cambridge, MA, 1996, pp. 444-452.

[39] S. Trimberger, ed [1994]. *Field-Programmable Gate Array Technology*. Kluwer Academic Publishers, Boston, 1994.

[40] J. von Neumann [1966]. *The Theory of Self-Reproducing Automata*. A. W. Burks, ed. University of Illinois Press, Urbana, IL, 1966.

[41] J.D. Watson, N.H. Hopkins, J.W. Roberts, J. Argetsinger Steitz, A.M. Weiner [1987]. *Molecular Biology of the Gene*. Benjamin/Cummings, Menlo Park, CA, 4th edition, 1987.

[42] S. Wolfram [1994]. *Cellular Automata and Complexity*. Addison-Wesley, Reading, MA, 1994.