ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA

DIPARTIMENTO di
INGEGNERIA DELL'ENERGIA ELETTRICA E DELL'INFORMAZIONE
"Guglielmo Marconi"
DEI

CORSO DI LAUREA/LAUREA MAGISTRALE IN INGEGNERIA ELETTRONICA

TESI DI LAUREA
in
Elettronica Dei Sistemi Digitali M E Laboratorio C.I.

Progetto e sintesi di un modulo digitale per il calcolo di prodotti matrice-vettore multilivello

CANDIDATO

Andrea Marrazzo

RELATORE

Prof. Ing. Eleonora Franchi Scarselli

CORRELATORI

Prof. Ing. Antonio Gnudi
Dott. Ing. Alessio Antolini

Anno Accademico
2019/2020

Sommario

Introduzione

Le nuove tecnologie tendono a servirsi dell'ausilio dell'intelligenza artificiale in qualsiasi contesto applicativo, portando con sé la manipolazione di grandi quantità di dati, Big Data, per poter applicare quelli che sono gli algoritmi cardine dell'AI, Machine learning, Deep Learning.

Le tecniche di machine learning, si basano in gran parte sulla risoluzione di problemi di algebra lineare, sistemi di equazioni lineari, per cui fattorizzazioni di matrici o moltiplicazioni di matrici, perciò diventa molto importante l'elaborazione di prodotti matrice-vettore, soprattutto nelle prestazioni di questa tipologia di calcolo.

Ciò comporta un'ardua sfida per sistemi hardware che si trovano ad elaborare enormi quantità di dati e quindi necessitano di altrettanta potenza di calcolo. Complessivamente però le tipologie di calcolo da elaborare tendono a ripetersi, al variare dei dati, questo consente di dirottare il calcolo verso Process Unit esterne alla CPU, verso gli acceleratori come le Graphic Processings Unit (GPUs); queste, inizialmente sviluppate per rendering 3D, si prestano in modo efficace al problema essendo dotate di un ampio numero di ALU, meno potenti di quella di una CPU. Il punto chiave, che spinge a muoversi verso tecnologie differenti, è il costo, in termini di potenza, necessario per trasportare i dati dalla locazione di memoria verso l'unità di calcolo: il target resta quella di scegliere un compromesso tra efficienza energetica e throughput.

A tal proposito si presenta l'in-memory computing (IMC), attraverso questa tipologia di calcolo è possibile raggiungere alti livelli di efficienza energetica, spostando l'unità di calcolo in memoria.

Perciò le prestazioni del prodotto matrice vettore diventano di cruciale importanza, in quanto radicati nei principali algoritmi, e proprio tale punto diventa l'oggetto di questo lavoro di tesi.

La stesura di questa tesi è stata strutturata su quattro capitoli in modo incrementale; il primo capitolo vuole essere un'introduzione all'ambito di interesse, quindi un excursus sullo stato dell'arte dell'in-memory computing e sulle tipologie di memorie in uso ed applicate nel caso in esame.

Nel secondo capitolo si vuole illustrare una prima architettura progettata, la quale vede una matrice con pesi a 2 bit, tre livelli, che si presenta come un aumento di complessità rispetto ad una struttura progettata in precedenza in un lavoro di tesi; all'interno di questo capitolo vengono commentati i risultati delle sintesi effettuate sul modulo.

Nel terzo capitolo viene incrementata la complessità della struttura progettata passando ad una matrice con pesi a 4 bit, analizzandone eventuali criticità e possibili ottimizzazioni energetiche a fronte dei risultati ottenuti dalla sintesi circuitale.

Il quarto capitolo, conclusivo, vuole illustrare una simulazione, e sintesi, della struttura con pesi a 4 bit, ma inserendo dei registri, in un primo momento per memorizzare i valori di uscita dal modulo, e successivamente là dove possibile per ottenere un'ulteriore ottimizzazione energetica.

Lo studio termina confrontando criticamente la soluzione proposta con lo stato dell'arte, aprendo ad eventuali sviluppi futuri.

1. In memory computing

In questo capitolo si vuole introdurre, in maniera molto generica, lo stato dell'arte del campo applicativo nel quale si colloca il lavoro di tesi, al fine di contestualizzare le scelte intraprese ed avere un confronto diretto con altre tecniche concorrenti.

L'ingegneria dell'Informazione è sempre più focalizzata sullo sviluppo di intelligenza artificiale (AI) e quindi su tecniche fondamentali quali reti neurali, machine learning, deep learning, etc. Tutto ciò, per esser reso possibile, porta con sé una grande quantità di calcolo computazionale ed una grande quantità di dati interessati, per i quali tecnologie di Cloud Computing e Edge Computing provano a dare una soluzione efficace [1].

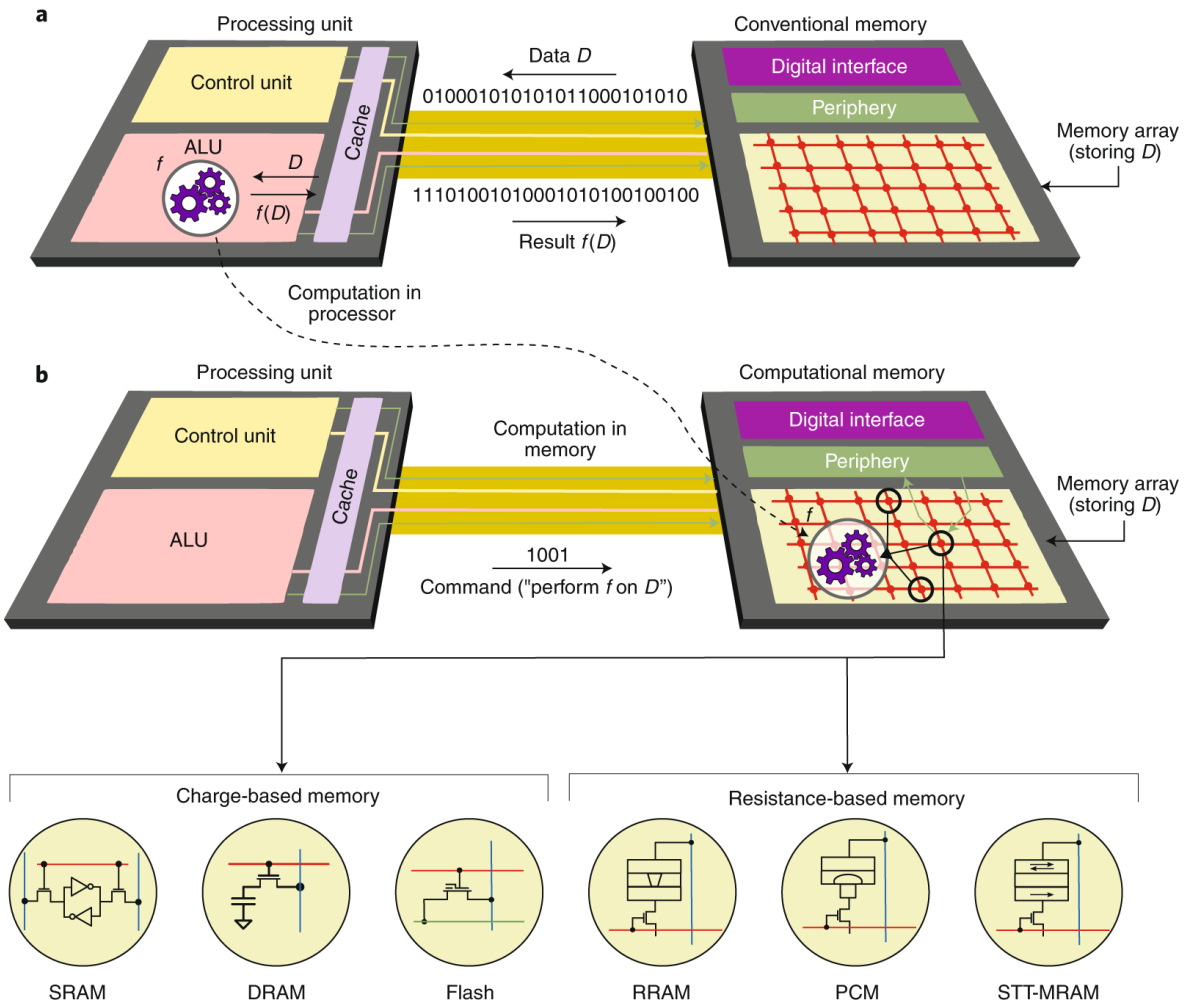


Figura 1: In-memory Computing

Il problema quindi che viene posto è l'esecuzione di un gran numero di operazioni su altrettanti dati [2]; questo può essere semplificato considerato che molte di quelle operazioni sono spesso

ricorrenti, quindi è immediato pensare all'applicazione di acceleratori hardware, GPU GPGPU, che, seppur progettati per rendering 3-D, offrono una grande quantità di potenza di calcolo dirottandola dalla CPU. Se pur efficace questa non è una soluzione efficiente dal momento che architetture di questo tipo devono essere viste anche in un'ottica mobile, là dove il concetto del "low-power" spopola, quindi è necessaria efficienza energetica, se si considera che trasportare i dati dalla memoria all'unità di calcolo ha un costo, e questo costo tende ad aumentare con le dimensioni della memoria, è evidente come strutture con architetture aventi memoria e calcolo separati incorrono in un "memory wall", per cui l'utilizzo di acceleratori digitali comporta un limite sulla quantità di dati spostare, in quanto, anche se l'operazione resta la stessa, i dati interessati cambiano e ciò comporta un nuovo caricamento dei dati in cache.

In figura [3] viene mostrata l'energia richiesta per accedere ad una word di 64 bit in memorie di diverso taglio, ad una tecnologia litografica a 45 nm, in relazione con l'energia necessaria per un operazione di moltiplicazione

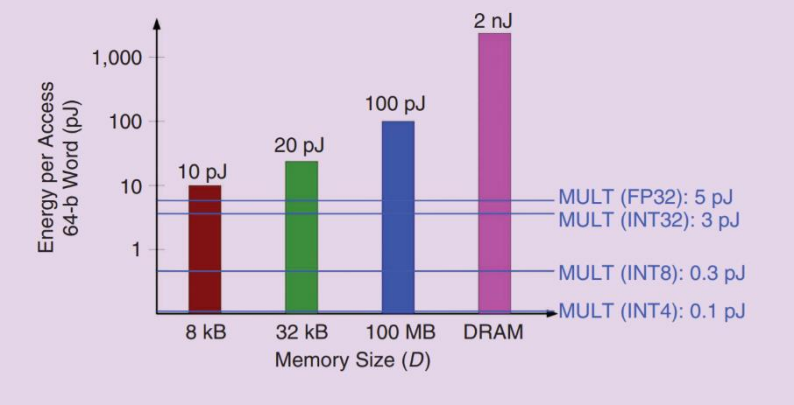


Figura 2: Energia per accesso word

Una possibile soluzione nasce dall'osservazione che una delle operazioni dominanti in digital-signal-processing e nel machine learning è la moltiplicazione matrice vettore (Matrix Vector Multiplication MVM) e che sarebbe possibile ammortizzare il costo computazionale svolgendo direttamente in memoria l'operazione, secondo la tecnica denominata In-Memory Computing (IMC). La tecnica IMC trova spazio sia in realizzazioni analogiche, che verranno descritte nel corso del capitolo, che in realizzazioni digitali, come vedremo.

Entrando nel merito di una struttura digitale IMC, si riporta, come esempio, la soluzione proposta nell'articolo di riferimento [2], dove viene strutturata una matrice di Process Engines (PEs) per effettuare il prodotto matrice vettore (MVM): $\vec{c} = A \times \vec{b}$. Queste PE, come si può notare dalla struttura in figura 3, sono dotate di una piccola porzione di memoria in grado di

memorizzare gli operandi e di spostare il risultato su PE adiacenti ammortizzando efficacemente il movimento di tutti gli operandi precedentemente consultati [4].

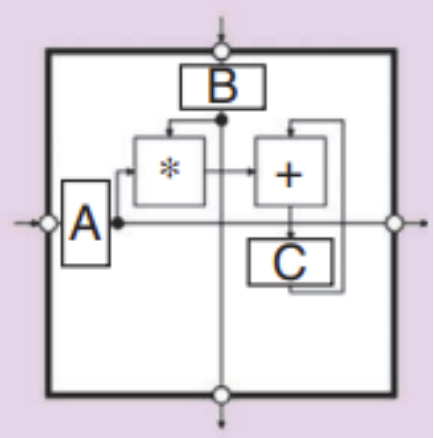


Figura 3: PEs

L'IMC può essere considerata come un'architettura spaziale nella quale le PEs sono delle bit cell: in figura 4 si mostra lo spostamento dei dati, un set di operandi si sposta orizzontalmente lungo l'array, vettore di input, un set può essere memorizzato nel la PEs, elementi della matrice, ed un set può spostarsi verticalmente, vettori di output.

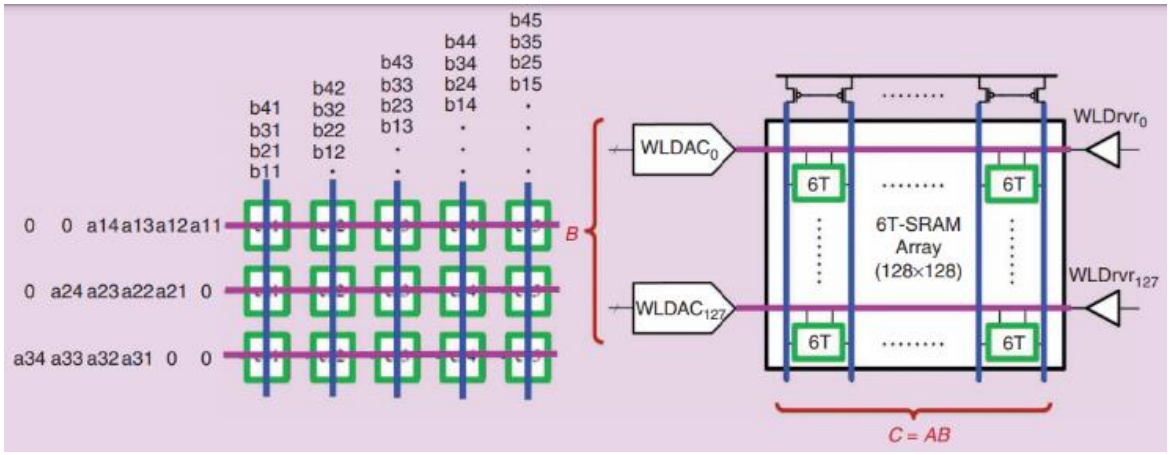


Figura 4:Strutture per il calcolo ed energia necessaria

Analog Computing

L'esigenza di ottimizzare il prodotto matrice-matrice, matrice-vettore, essenziale per tecniche di Edge computing, Deep Learning e Machine Learning, trova spazio nel mondo analogico [5]. Quel che si vuole ottenere è una soluzione completamente analogica che vada a sfruttare le leggi fondamentali dell'elettronica, legge di Ohm e legge di Kirchoff alle maglie e ai nodi [6], per ottenere il calcolo direttamente in memoria. Il modello è intuitivo ed immediato, se si considera la struttura di una memoria: questa è composta da due wire, bit line e word line, le due sono interconnesse tra loro da un transistor il quale permette di memorizzare un valore di tensione, attraverso una differenza di potenziale opportunamente posta sula bit line, una volta attivata la word line corrispondente. Data questa premessa, sulla base della legge di Ohm, è possibile rivedere il modo di intendere gli elementi di suddetta legge [7], se considerassimo la memoria in maniera del tutto analogica, sarebbe evidente che possiamo vedere una matrice di ingressi in tensione V, che si applica al terminale positivo di una matrice di conduttanze G, e infine questi producono un'uscita in corrente I, per cui da ogni singolo transistor a cui è applicato un segnale in tensione otteniamo un'uscita I = GV che, come è possibile notare in figura [8], può essere trasformato in un ingresso in tensione con un semplice integratore, e

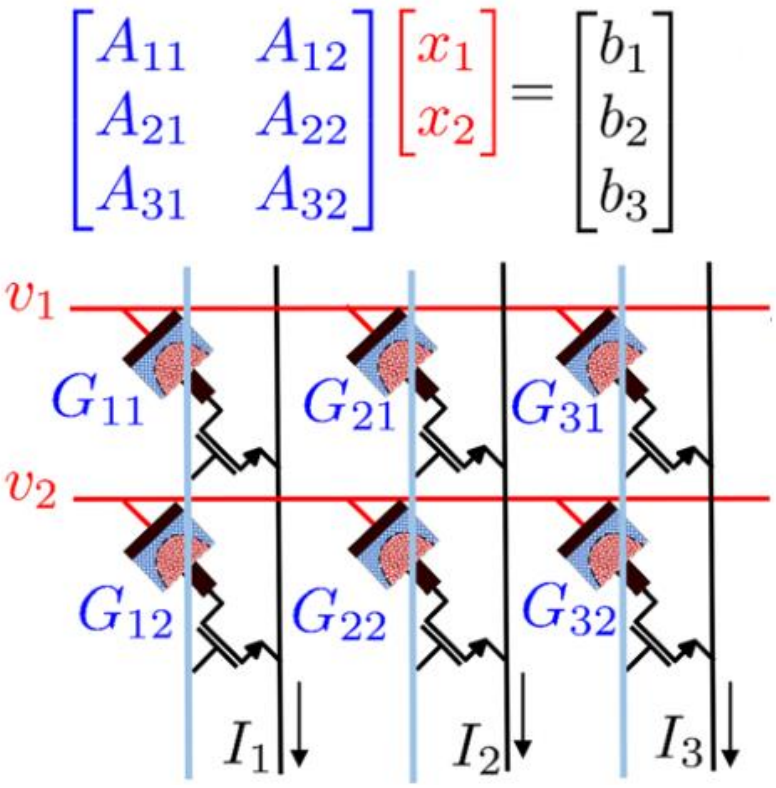


Figura 5: Struttura di una memoria utilizzata nell'in-memory computing

successivamente utilizzato in forma analogica o in forma digitale, utilizzando un opportuno convertitore ADC, a seconda della tipologia del circuito a valle [9], così facendo è possibile

incrementare le performance del calcolo. Si evince dalla figura 5 che, per effettuare questa tipologia di calcolo, è necessario l'accesso a tutta la struttura di memoria, in particolare, se consideriamo una memoria digitale convenzionale, questa avrà la linea, alla quale sono connessi i source del MOS, a massa; per cui, per poter effettuare una lettura di corrente su queste linee, è necessario svincolarle dalla massa.

È necessario introdurre alcune premesse riguardanti la tipologia di memoria in uso, per poter effettuare un ragionamento di questo tipo è indispensabile che la memoria sia in grado di conservare il dato e soprattutto che abbia un meccanismo di lettura non distruttiva; per ottenere un effettivo incremento di efficienza è necessario che sia disponibile l'acceso, sia in lettura che in scrittura, a tutta la memoria in una singola operazione e, considerato che le memorie di questa tipologia sono ottimizzate per un accesso random o comunque ad un numero di indirizzi limitati, si propone come un applicazione diametralmente opposta. Ciò comporta quindi l'utilizzo di memorie convenzionali all'interno di strutture differenti dalle architetture di memoria convenzionali.

Una delle principali problematiche, avverse a questo tipo di tecniche, consiste nel fatto che, le memorie che si vogliono utilizzare sono costruite con l'applicazione di materiali NVM (Non Volatile Memory), questi hanno delle performance ottimizzate per le memorie standard (digitali), quindi un alto rapporto segnale/rumore SNR, per immagazzinare correttamente i dati, e presenta pochi livelli di conduttanza, se non solo due (corrispondenti ai livelli logici alto e basso). Quanto detto evidenzia un'ardua sfida nella costruzione di dispositivi che possano supportare questa tipologia di tecnologie, di seguito vengono introdotte le tipologie di memorie, più diffuse per questo tipo di applicazioni.

Memorie PCM

Le memorie a cambiamento di fase sfruttano la variazione di conduttività dovuta al cambiamento morfologico, di materiali a cambiamento di fase, come i calcogenuri, da una condizione amorfa ad uno stato cristallino [10].

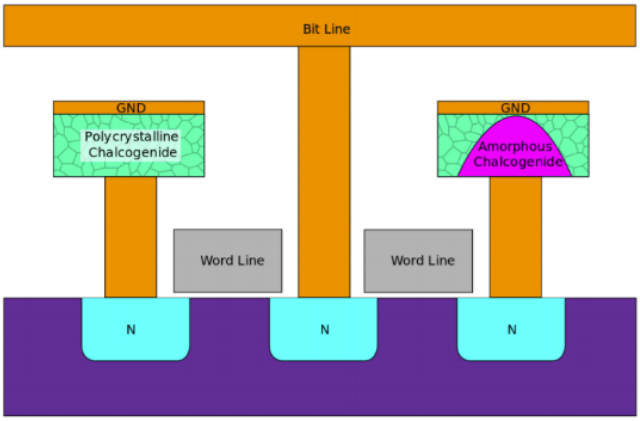


Figura 6:Morfologia cella PCM

La transizione da amorfo a cristallino avviene mediante attivazione termica, sfruttando il riscaldamento per effetto Joule viene imposta un'alta corrente in modo tale da scaldare gran parte del materiale al di sopra della temperatura di cristallizzazione. Per la transizione inversa si immette un impulso di corrente veloce e grande in modo da interrompere bruscamente la transizione e lasciare il materiale in una condizione amorfa [11].

La velocità di transizione tra i due stati determina le prestazioni di scrittura in tecnologia PCM, mentre la lettura avviene mediante misura della resistività a bassa tensione in modo tale da non modificare la struttura del materiale; la resistività cala nel passaggio da amorfo a cristallino, passando da centinaia di MΩ a decine di kΩ.

RRAM & CBRAM

Le Resistive RAM si basano sul valore di resistenza introdotta da un sottile film metal-oxide, materiali commutabili elettricamente, interposto tra due contatti metallici. Il contatto superiore controlla l'infusione di lacune di ossigeno del filamento conduttivo, costituito da ossido sub stechiometrico, la conducibilità è determinata dalla vicinanza del filamento al contatto inferiore [12], ed è controllata dal *growing* o *shrinking*, crescita e assottigliamento, del filamento, è evidente la reversibilità del processo.

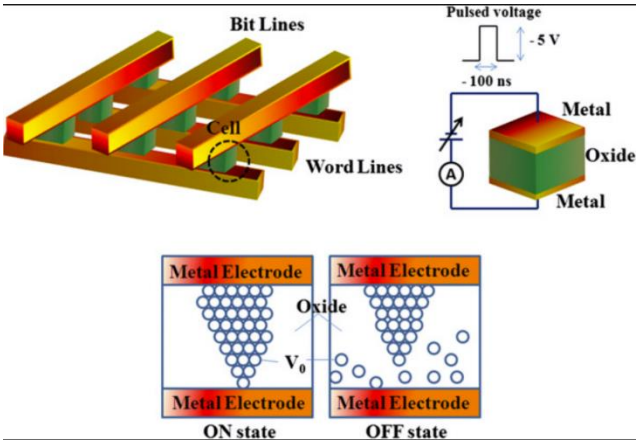


Figura 7: Struttura RRAM

Le CBRAM, Conductive Bridge RAM [13], si basano su un percorso conduttivo formato da ioni metallici mobili, liberi di muoversi attraverso un elettrolita o un dielettrico; tipicamente è una pila formata da un elettrodo inerte, un elettrolita o un dielettrico solido ed un elettrodo elettrochimicamente attivo.

Progetto ricercatori ARCES-STMicroelectronics

Il progetto sviluppato dai ricercatori ARCES-STMicroelectronics, oggetto di un articolo sotto revisione al momento della stesura della tesi [14], ha come obiettivo quello di utilizzare una memoria PCM, in particolare un a ePCM, per applicazioni di analog in-memory computing AIMC. Ricollegandosi al precedente paragrafo, si vuole mostrare il caso di un'applicazione reale di AIMC, con le problematiche che affliggono le memorie PCM, come variabilità della conduttanza e il rumore in bassa frequenza; l'oggetto di questa tesi vuole essere un confronto con questo studio, in modo da offrire una relazione tra due architetture, una analogica ed una digitale.

Coerentemente con quanto descritto nel paragrafo precedente, questo studio intende utilizzare le PCM, che rappresentano il futuro per le Non-Volatile Memories NVMs, per la loro capacità di memorizzare valori '0' '1' cambiando struttura, amorfo-cristallina, ma sfruttandone la variazione della conduttività per applicazioni analogiche, aprendo all'opportunità di avere livelli analogici multipli, necessari per operazioni su celle multilivello. La struttura riportata in figura vuole mostrare come partendo dalla legge di Kirchoff e dalla legge di Ohm, $I_{TOT} = \sum_{i=1}^{N} G_i V_i$, è possibile estrarre una struttura atta all'esecuzione dell'operazione prodotto matrice-vettore, con una matrice di conduttanze **G** di dimensioni MxN ed un vettore **V**

$$\begin{pmatrix} G_{11} & \cdots & G_{1N} \\ \vdots & \ddots & \vdots \\ G_{M1} & \cdots & G_{MN} \end{pmatrix} \begin{pmatrix} V_1 \\ \vdots \\ V_N \end{pmatrix} = \begin{pmatrix} I_{TOT,1} \\ \vdots \\ I_{TOT,M} \end{pmatrix}$$

Si distinguono in figura word line e bit line connesse attraverso il canale di un mos ed una resistenza variabile.

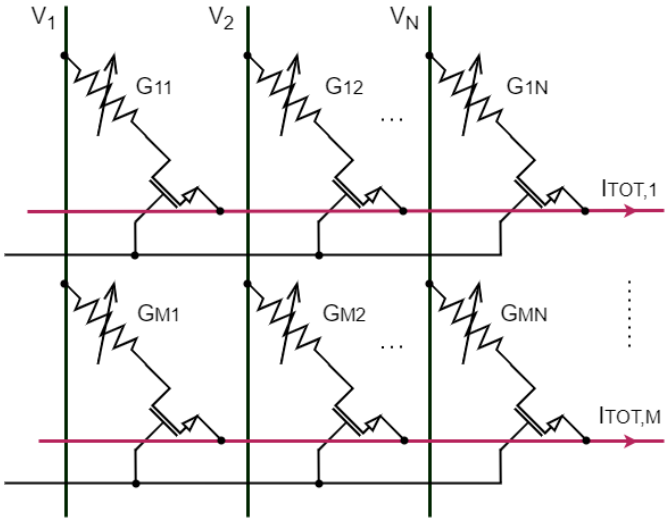


Figura 8 : Schema di un array di una PCM e l'applicazione nell'AIMC

Purtroppo, come si evidenzia nello studio, nella realtà applicativa si incorre in diverse non idealità, che presentano delle evidenti limitazioni al progetto. In primis, come introdotto nel precedente paragrafo, per una soluzione ideale sarebbe necessaria una memoria in grado di fornire l'accesso a tutte le celle in contemporanea, l'utilizzo di una memoria digitale, quale la ePCM, si pone in contrasto dal momento che i decoder di ingresso ed uscita vengono progettati per avere cicli di lettura e scrittura con accesso alla singola word.

Inoltre, contrariamente a quanto previsto dalla figura precedente, la lettura della corrente non può essere fatta dal terminale di source del transistore di indirizzamento che, infatti, nelle memorie digitali è connesso a massa. Per generare una corrente Ii= GiVi e leggerla dal terminale positivo della cella PCM, si può adottare una schema come quello indicato nelle due figure

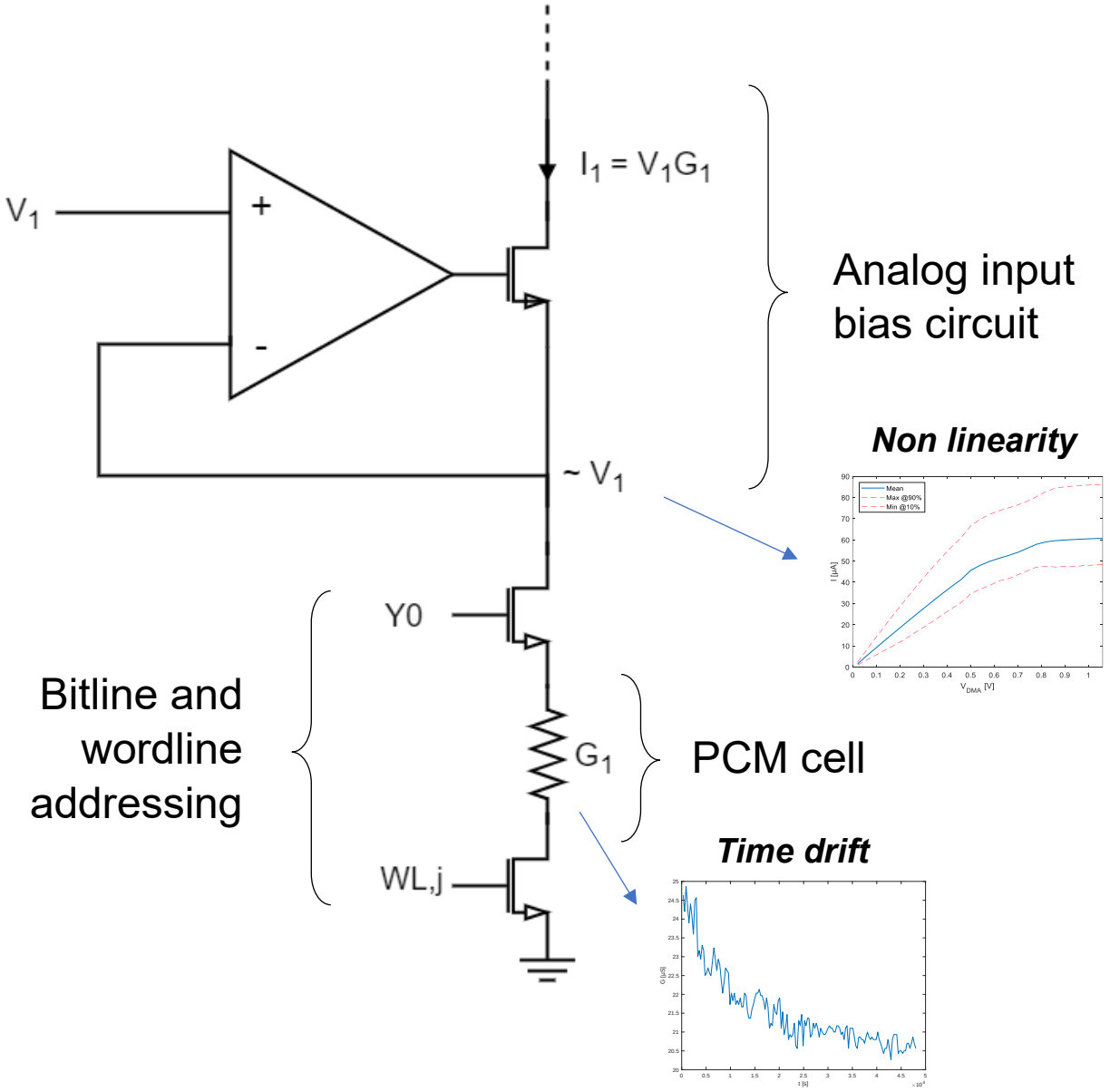


Figura 9 Struttura di riferimento brevetto STM

seguenti (Brevetto ST-19-AG-0705 “Adapting a PCM IP to Computing in memory preserving the cut IP”). La prima mostra come il valore di tensione sulla bit-line possa essere forzato da un amplificatore permettendo di generare la corrente Ìì= GiVi come desiderato. La seconda spiega come connettere le varie bit-line per realizzare un prodotto vettore* vettore per ogni word-line selezionata. Selezionando in sequenza le word-line è possibile realizzare il prodotto matrice (G) per il vettore degli ingressi (V).

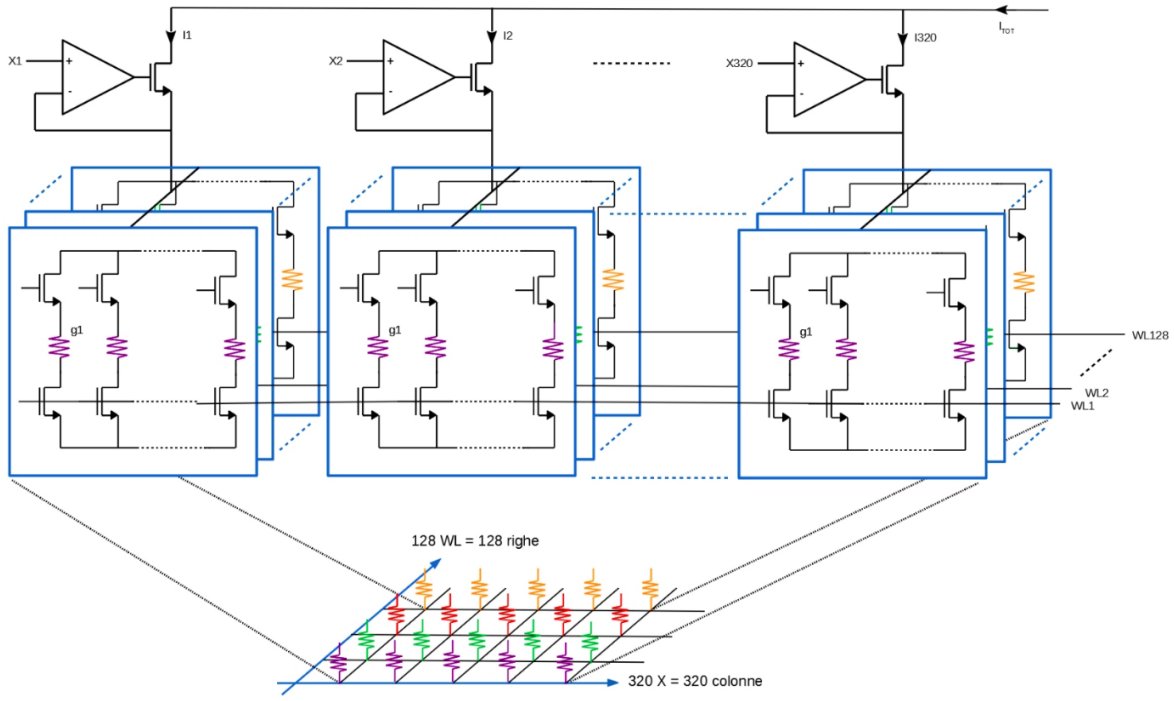
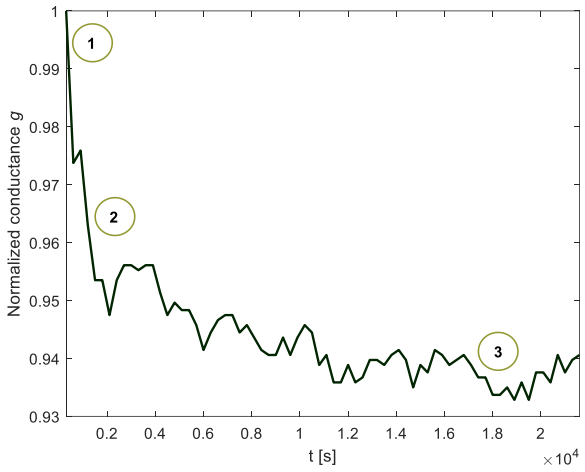


Figura 10

Un secondo problema, puramente analogico, interessa la lettura stessa del dato, essendo progettate come analogiche, questo tipo di memoria è interessato a soli due livelli logici ‘alto’ e ‘basso’, per applicazioni multilivello subentrano variazioni legate all’incertezza del valore iniziale, alla deriva temporale ed al rumore in bassa frequenza (flicker noise), rispettivamente ‘1’ ‘2’ e ‘3’ nella figura seguente, dove è stata simulata la lettura del valore di conduttanza per un tempo limitato.

L’articolo in questione si addentra nello studio delle tecnologie per cercare di minimizzare l’inferenza di queste problematiche, provando a fornire una soluzione funzionale realizzabile.



Tipologie SRAM

Le memorie utilizzate in questo progetto di tesi sono dei moduli SRAM embedded general purpose fornite da STMicroelectronics Le memorie utilizzate, quindi, sono predisposte per il flusso semi-custom a standard cell per la generazione di un circuito integrato dedicato. Al contrario, nella versione analogica dei ricercatori dell'università di Bologna, sono state utilizzate memorie non volatili PCM, per l'appunto analogiche, delle quali si è data una breve introduzione sulle caratteristiche tecnologiche nei paragrafi precedenti.

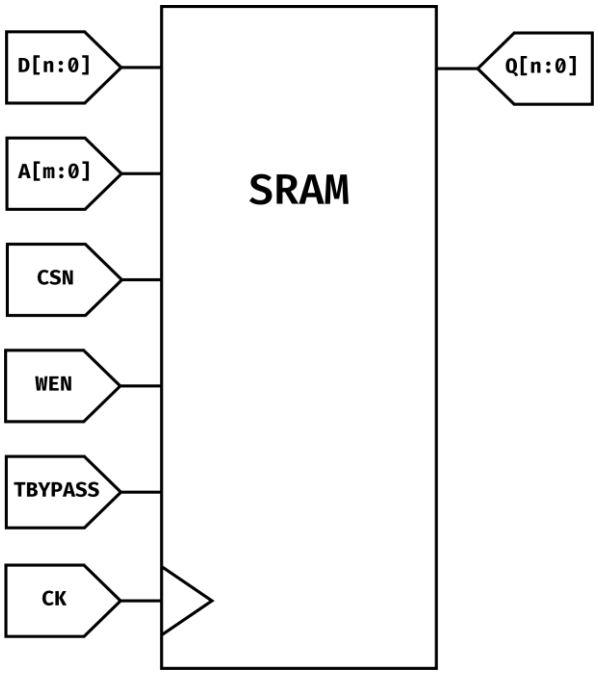


Figura 11: Cella di memoria SRAM

In figura viene mostrato lo schematico del modulo SRAM fornito da STMicroelectronics. Ne segue l'illustrazione dei vari pin di input/output del modulo:

- D[n:0]: dato di ingresso.
- A[m:0]: indirizzo (address) di riga della memoria. Indice di riferimento della word selezionata, sia in lettura che in scrittura.
- CSN: chip select enable. Quando assume il valore logico basso l'accesso a memoria è abilitato. Quando CSN è alto la memoria è bloccata e nessuna operazione può essere effettuata internamente (Standby mode).
- WEN: write enable. Abilita la scrittura o la lettura della memoria. Quando in ingresso assume il valore logico basso la memoria può essere scritta, quando invece assume il valore logico alto, viene configurata per la lettura.

- TBYPASS: consente di copiare (bypassare) la porta D su Q. Non è interessante per questo progetto, quindi è stato imposto il valore costante 0 per disabilitarne la funzione.
- CK: input di clock.
- Q[n:0]: dato di uscita. Genera il contenuto della memoria localizzato all'indirizzo A[m:0].

Per il corretto funzionamento del sistema sarà necessario che la frequenza massima del clock rispetti la nota relazione tra periodo del clock, tempo di setup e percorso critico del circuito. Il tool di sintesi si occuperà di rispettare i tempi di setup e di hold di tutti i segnali elencati, per avere il corretto funzionamento del modulo.

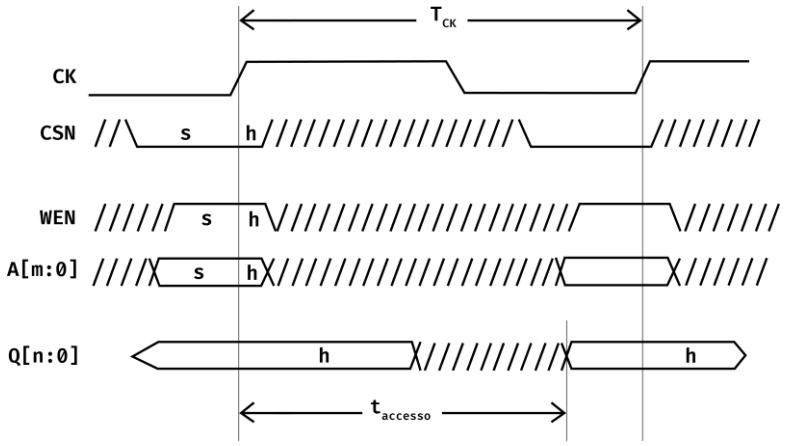


Figura 12: Timing in scrittura

Nella figura precedente viene mostrato il timing delle memorie utilizzate durante un ciclo di scrittura, e durante un ciclo di lettura nella figura successiva, con i relativi tempi di setup e di hold per ogni segnale, indicati rispettivamente con s e h nelle figure.

Durante la fase di scrittura il dato di uscita Q[n:0] rimane stabile durante tutta l'operazione al valore precedentemente letto, senza quindi possibili variazioni dell'uscita durante la scrittura della memoria, che non è pertanto mostrato nel diagramma. Il segnale di CSN deve avere un valore logico basso sui fronti di salita del clock per permettere alla memoria di essere abilitata. Il segnale di scrittura WEN deve essere campionato basso per poter scrivere un dato all'indirizzo A[m:0], poi, nel successivo fronte di clock il dato sarà stato scritto e sarà necessario portare il segnale WEN al valore logico alto per configurare la memoria in lettura ed evitare la sovrascrittura. Al contrario, se si intende scrivere un altro dato ciò che dovrà cambiare al fronte successivo sarà l'indirizzo al quale si scriverà il nuovo dato.

Per quanto riguarda la fase di lettura si avrà il dato Q[n:0] i-esimo in uscita dopo il tempo di accesso ($t_{accesso}$) e rimarrà stabile fintanto che il segnale CSN continua ad abilitare

la memoria al suo regolare funzionamento, WEN rimane al valore alto per consentire la lettura e l'indirizzo A[m:0] resta all'i-esimo valore.

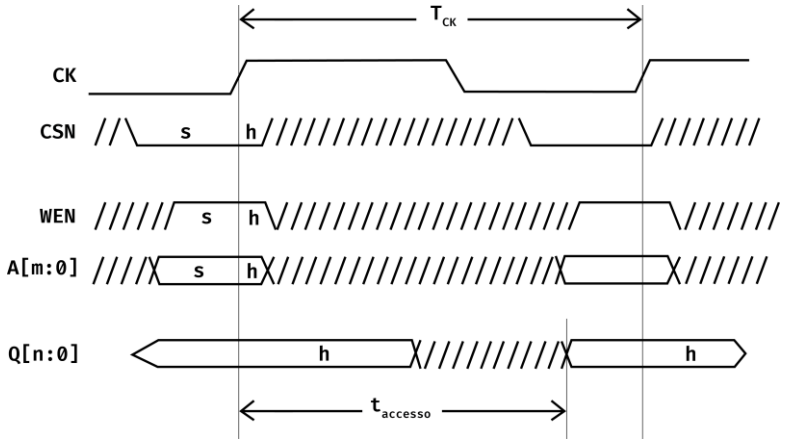


Figura 13: Timing in lettura

2. Progetto di un modulo digitale per il calcolo di prodotti matrici-vettori a tre livelli

La tesi intrapresa vuole offrire un'alternativa digitale, quindi un confronto, al circuito analogico in fase di progettazione da parte del team di ricercatori del laboratorio ARCES, nell'ambito di "in-memory computing". L'obiettivo finale è la progettazione di un'architettura digitale multilivello in grado di effettuare moltiplicazioni, bypassando la CPU, risparmiando tempo, potenza di calcolo e quindi anche energia. Il primo obiettivo affrontato nel lavoro di tesi mira ad ampliare la complessità di un modulo, progettato in un precedente lavoro di tesi, il quale effettuava una moltiplicazione matrice-vettore, fra un vettore di coefficienti a 8 bit e una, matrice di pesi binaria (a due livelli, quindi solo elementi [0,1]). Il primo passo verso una soluzione multilivello è stata quella di aggiungere un bit ai pesi e quindi andarne a considerare anche il segno, ottenendone una rappresentazione in logica ternaria (±1, 0).

Prima di approfondire la descrizione dell'architettura è necessario puntualizzare che, la struttura progettata prevede l'utilizzo di memorie digitali convenzionali, per cui l'accesso in lettura ed in scrittura, dettato dai decoder presenti nella stessa, è vincolato alla singola word line.

Scelte e vincoli progettuali per realizzare il prodotto matrice-vettore

Il prodotto matrice-vettore si presenta dunque come una moltiplicazione di una matrice di pesi, dove ogni elemento è a due bit, e un vettore di campioni; la soluzione scelta propone di distinguere i bit dei pesi in due matrici, una prima che contiene il segno del peso, '0' positivo '1' negativo, e una seconda matrice che è quella del valore 0 o 1 del peso.

Per la memorizzazione delle matrici dei pesi e del vettore di ingresso si sono utilizzati dei banchi di memoriaembedded forniti da STMicroelectronics. Come mostrato in figura, si è scelto di realizzare il vettore colonna C con una memoria da 320 parole ad 8 bit ciascuna. Le parole del vettore C sono interpretate come valori interi, mentre, per la memoria P, che implementa la matrice dei pesi P, si è scelto di utilizzare dei valori binari a singolo bit per descrivere ciascun elemento come per la memoria S, che implementa la matrice del segno dei pesi S.

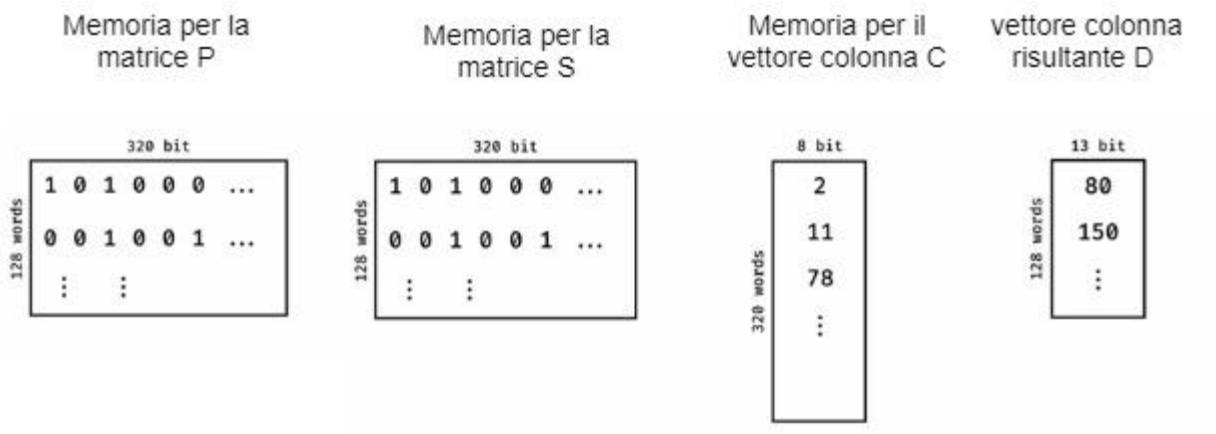


Figura 14: Struttura matriciale sistema

Facendo riferimento alle caratteristiche delle matrici dei coefficienti utilizzati nelle applicazioni di Compressive Sensing nel testbench si è scelto di costituire le righe della matrice P con una probabilità del 20% di valori pari a 1: la maggior parte degli elementi, dunque, avrà valore nullo. Le dimensioni della memoria P sono 128 righe per 320 colonne. Per quanto riguarda il vettore colonna D, sarà costituito da 128 righe, rispettando le regole di match dimensionale tra la matrice ed il vettore. Ciascuna parola del vettore D, per non avere perdita dell'informazione, dovrebbe essere costituita da 17bit:

$$bit\ di\ una\ parola\ del\ vettore\ D = 8bit + log_2(320)\ bit = 8bit + 9bit = 17bit$$

Sia il modulo sviluppato in questo progetto di tesi, che il modulo sviluppato dai ricercatori, basato sul calcolo analogico, avranno un impiego in settori in cui la precisione di calcolo non è la caratteristica maggiormente richiesta, ma dove caratteristiche quali velocità e consumo sono i principali constraints. Considerando applicazioni in cui solo una bassa percentuale di pesi assuma valore diverso da 0, è possibile ridurre ulteriormente i bit con cui rappresentare il vettore di uscita, al costo di una conseguente perdita di informazione, per alleggerire l'unità di calcolo:

$$bit\ di\ una\ parola\ del\ vettore\ D = 13bit$$

Come detto, sono state utilizzate delle memorie per la realizzazione fisica della matrice e del vettore che contengono i dati che devono essere moltiplicati. Prima di effettuare le operazioni di calcolo sarà necessaria la scrittura delle memorie. Si è scelto come vincolo di progetto quello di voler realizzare la scrittura della memoria C e tutte le operazioni di moltiplicazione tra la memoria dei pesi (P e S) ed il vettore C in un intervallo di tempo di circa 5 ms.

La scrittura delle memorie P e S faranno comunque parte del progetto, ma il tempo di scrittura non verrà considerata in questo vincolo progettuale. Le operazioni che devono essere svolte, quindi, sono 320 per la scrittura della memoria C e successivamente 320 · 128 per effettuare le somme di prodotti. Ciò impone un vincolo sul periodo di clock, ipotizzando di effettuare una operazione ogni periodo di clock:

$$(320 + 320 \cdot 128) \cdot T_{CLK} \simeq 5\ ms$$

si ottiene quindi un periodo di clock pari a:

$$T_{CLK} \simeq 100\ ns$$

che impone una specifica sulla frequenza di clock pari a:

$$f_{CLK} \simeq 10\ MHz$$

Effettuate queste scelte progettuali, la scelta dell'architettura ha come obiettivo quello di minimizzare il consumo del modulo rispettando i constraints imposti, svolgendo quindi la moltiplicazione matrice-vettore nel modo più efficiente possibile.

La memoria dei campioni e la memoria dei pesi e segno

Sono state progettate e descritte per il flusso di progetto digitale a Standard Cells da STMicroelectronics delle memorie SRAM, descritte nella tesi di Davide Molino e brevemente richiamate nel capitolo precedente, con diversi tagli. Sono state selezionate quelle le cui dimensioni meglio rispettavano le scelte progettuali.

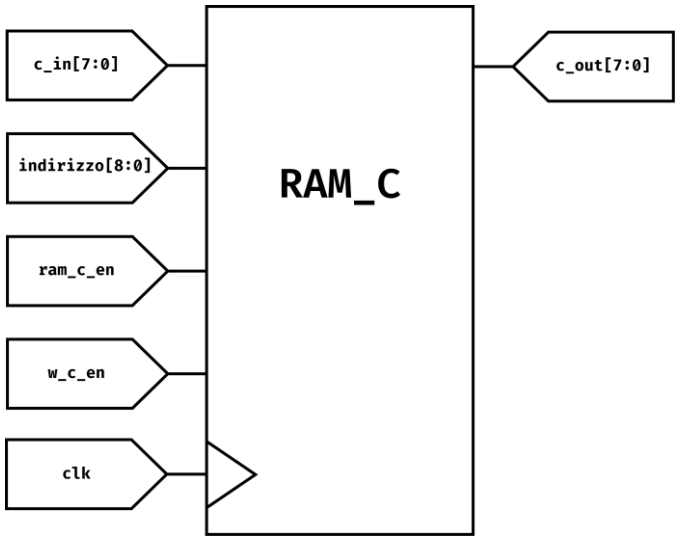


Figura 15: Modulo RAM_C

Per la memoria C si è utilizzata una SRAM, denominata RAM_C, con 320 righe e parole da 8bit , che calza esattamente le dimensioni volute.

Sono stati ridefiniti i pin di input ed output per mappare le esigenze: i dati in ingresso saranno dei campioni provenienti da una acquisizione esterna, c_in[7:0], per questo motivo in questa progetto spesso ci si riferirà a questa memoria con il nome di campioni. L'indirizzo A[m:0] della memoria verrà chiamato semplicemente indirizzo[8:0] e saranno necessari 9bit per permettere di esprimere le 320 posizioni possibili (log2(320) = 8.32, quindi 9bit). Per quanto riguarda i pin CSN, WEN e CK sono stati solo modificati i nomi in ram_c_en, w_c_en e clk rispettivamente. Il pin TBYPASS è stato posto ad un valore logico basso durante tutto il funzionamento del circuito, pertanto non viene mostrato.

Per la memoria P invece la SRAM scelta ha 128 righe e 64 colonne, quindi è stato necessario utilizzare cinque moduli in parallelo per ottenere le 320 colonne richieste. Si analizza un singolo modulo, denominato RAM_P_n [fig. B] e poi si vedrà la configurazione in parallelo di cinque moduli.

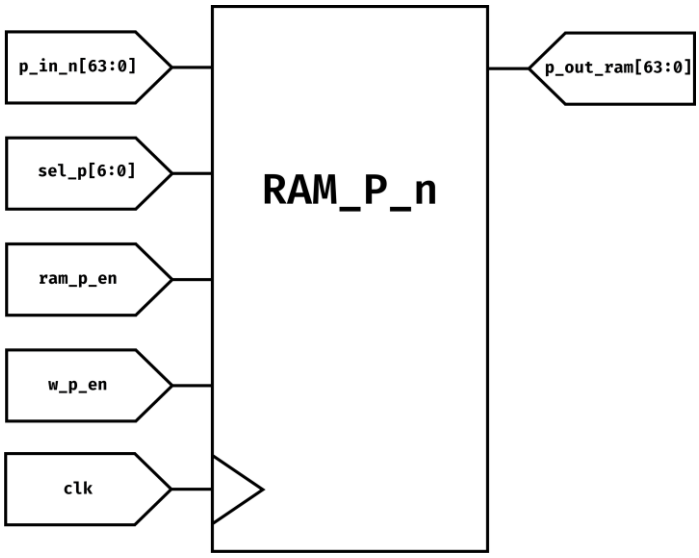


Figura 16: Modulo RAM_P_n

Ciascuna parola in ingresso sarà composta da 64bit, p_in_n[63:0], e rappresenta i valori che

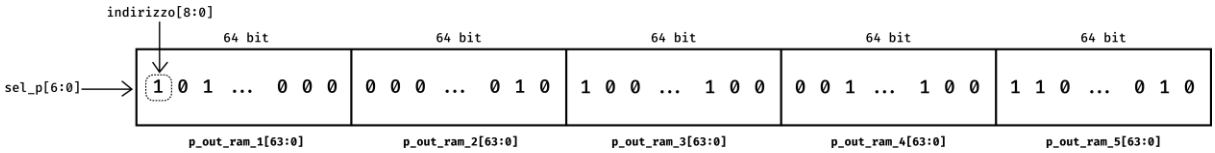


Figura 17: Struttura word

saranno moltiplicati per 64 campioni della RAM_C. Ci si riferirà a loro con il termine pesi, per

indicare il fatto che i campioni verranno pesati (moltiplicati) per questi valori. I segnali CSN, WEN e CK vengono denominati rispettivamente in ram_p_en, w_p_en e clk, analogamente alla memoria dei campioni. L'indirizzo di riga della memoria invece, sel_p[6:0], dovrà descrivere le 128 posizioni possibili: saranno necessari quindi 7bit per permettere di esprimere ciò (log2(128) = 7, quindi 7 bit). Come detto, è stato necessario utilizzare cinque moduli di RAM_P_n per costituire la memoria desiderata, sono state denominate RAM_P_1, RAM_P_2, RAM_P_3, RAM_P_4 e RAM_P_5. Ogni memoria riceve in parallelo gli input sel_p[6:0], ram_p_en, w_p_en e clk, mentre i segnali dei dati in ingresso, da p_in_1[63:0] a p_in_5[63:0], differiscono perché ognuna di loro memorizzerà un segmento del vettore da 320 elementi che si vuole avere. Inoltre, si vuole leggere in sequenza i singoli bit di una riga della memoria P formata da 320 elementi. Così come per i dati in ingresso, anche per quelli di uscita, ciascuna delle cinque memorie contribuisce fornendo un segmento di 64 elementi alla riga d'uscita della memoria P. Sarà necessario disporre di un modulo che, attraverso un indirizzo di colonna, indirizzo[8:0] in figura, seleziona da quale memoria delle cinque bisogna prelevare il dato di uscita, e qual è il bit richiesto tra i 64 elementi di riga della singola memoria identificata.

Per fare ciò è stato utilizzato un multiplexer, come mostrato in [fig. D]. Il segnale di controllo del multiplexer sarà lo stesso indirizzo di riga, indirizzo[8:0], della RAM_C. Si ricorda che il prodotto tra una riga della RAM_P e la colonna di campioni si esegue moltiplicando ciascun

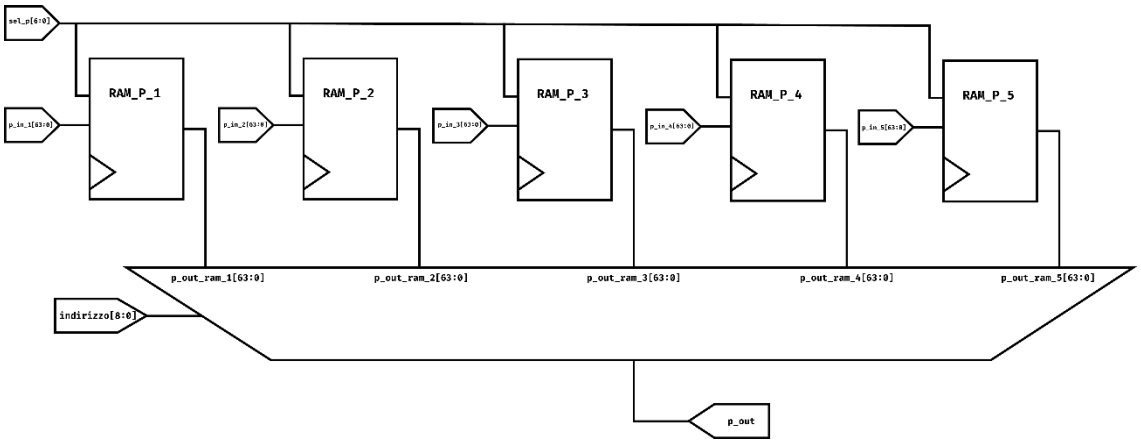


Figura 18: Struttura Interna RAM_P

peso nella i-esima posizione con il corrispettivo campione nella stessa posizione i, fino ad esaurire tutti gli elementi della riga selezionata sulla RAM_P e, dopodiché sommando i vari prodotti. Si noti come il segnale indirizzo[8:0] deve in ogni istante di calcolo combaciare per tutte le memorie.

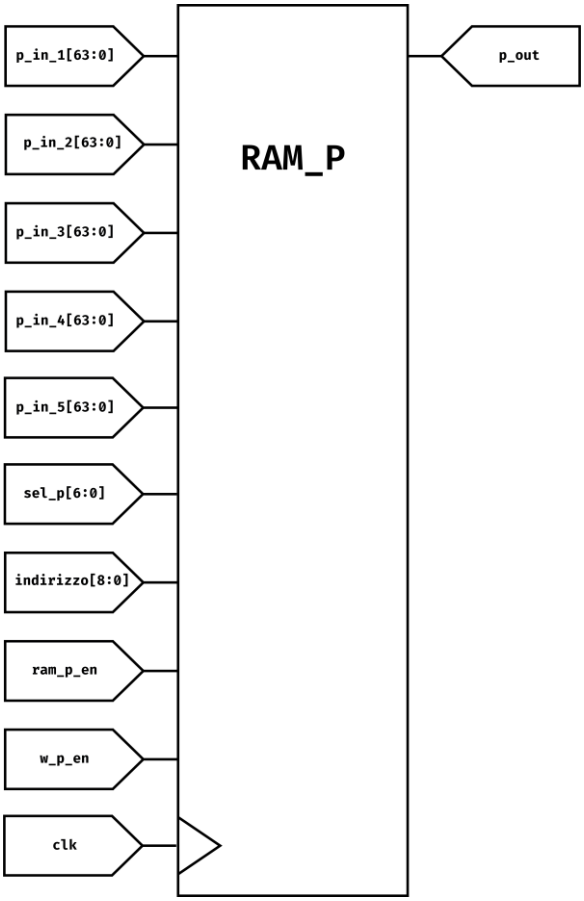


Figura 19: Modulo RAM_P

Questo segnale quindi identifica le 320 posizioni disponibili nelle righe della RAM_C, mentre nella RAM_P, identifica le 320 posizioni possibili tra le colonne del vettore in uscita dalla matrice P. Si arriva, a questo punto, alla descrizione di un modello compatto per rappresentare la RAM_P [fig. E], memoria dei pesi, nella quale in uscita si ha un singolo bit utilizzando due indirizzi: uno di riga, sel_p[6:0], ed uno di colonna, indirizzo [8:0].

L’architettura descritta fino a questo punto è quella proposta da Davide Molino nella quale i pesi sono binari. La rappresentazione ternaria dei pesi richiede, come detto, di aggiungere un bit di segno per ogno valore di pesi. Per la memorizzazione dei bit di segno il ragionamento è perfettamente uguale a quello della matrice P, quindi saranno utilizzate sempre SRAM a 128 righe 64 colonne, e sono richiesti cinque moduli in parallelo. Il singolo modulo verrà denominato RAM_S_n, si vedranno cinque moduli parallelo.

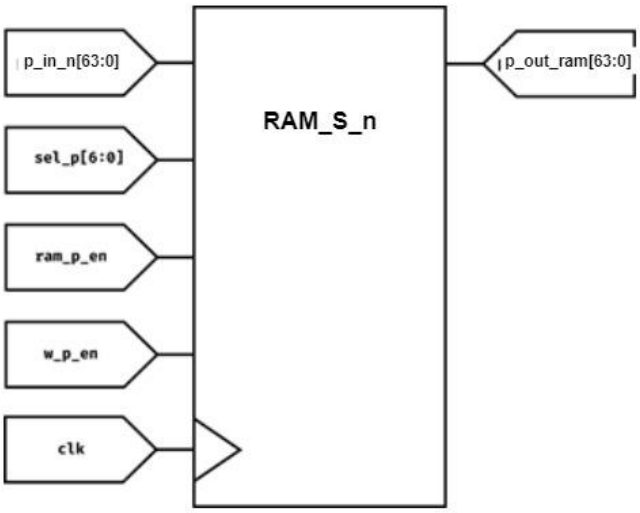


Figura 20: Struttura RAM_S_n

Ciascuna parola in ingresso sarà composta da 64 bit, S_in_n[63:0], e rappresenta i valori che saranno utilizzari per scegliere il segno corretto dei campioni corrispondenti della RAM_P. Ci si riferirà a loro con il termine segni, per indicare il fatto che i campioni avranno segno positivo o negativo a seconda del valore dell’elemento ad essi relativo. Dovendo la matrice S essere letta e scritta contemporaneamente alla matrice P si è scelto di denominare i segnali CSN, WEN e CK rispettivamente in ram_p_en, w_p_en e clk, come per la RAM_P_n, l’indirizzo della riga sarà anch’esso sel_p[6:0] per lo stesso ragionamento fatto in precedenza; saranno utilizzati cinque moduli, come già detto, e i segnali di ingresso seguiranno la stessa logica precedente. Anche in questo caso troveremo un multiplexer, che sarà lo stesso usato nella matrice P, il quale selezionerà, in base all’indirizzo, i segni memorizati nella memoria corretta. Si finisce quindi anche per questo modulo, come in P, alla descrizione di un modulo compatto per rappresentare

RAM_S, memoria dei segni, avente in uscita un singolo bit identificato da un indirizzo riga , sel_p[6:0], ed uno di colonna, indirizzo [8:0].

Organizzazione generale dell'architettura

Volendo evettuare il prodotto matrice-vettore, l'architettura che si è scelto di utilizzare parte da un modello su cui sono state effettuate alcune scelte architetturali per riuscire a ridurre i consumi durante le operazioni di calcolo, facendo leva su alcune caratteristiche peculiari del prodotto matrice-vettore che si vuole realizzare. L'operazione di moltiplicazione sfrutta le caratteristiche del prodotto per 0/1, che ci consente di effettuare delle scelte architetturali vantaggiose ai fini del consumo, si basa sull'accumulo del valore del singolo campione, a seconda del valore del peso, il quale andra ad abilitare o meno la lettura di RAM_C, scelta già introdotta nella tesi di Davide Molino, la scelta progettuale per la gestione del segno è stata quella di introdurre un multiplexer, nel quale s_out_ram sarà il segnale di selezione, e selezionerà il valore diretto di c_out_ram o c_out_ram, ottenuto sommando 1 al complemento a 1 di c_out_ram stesso.

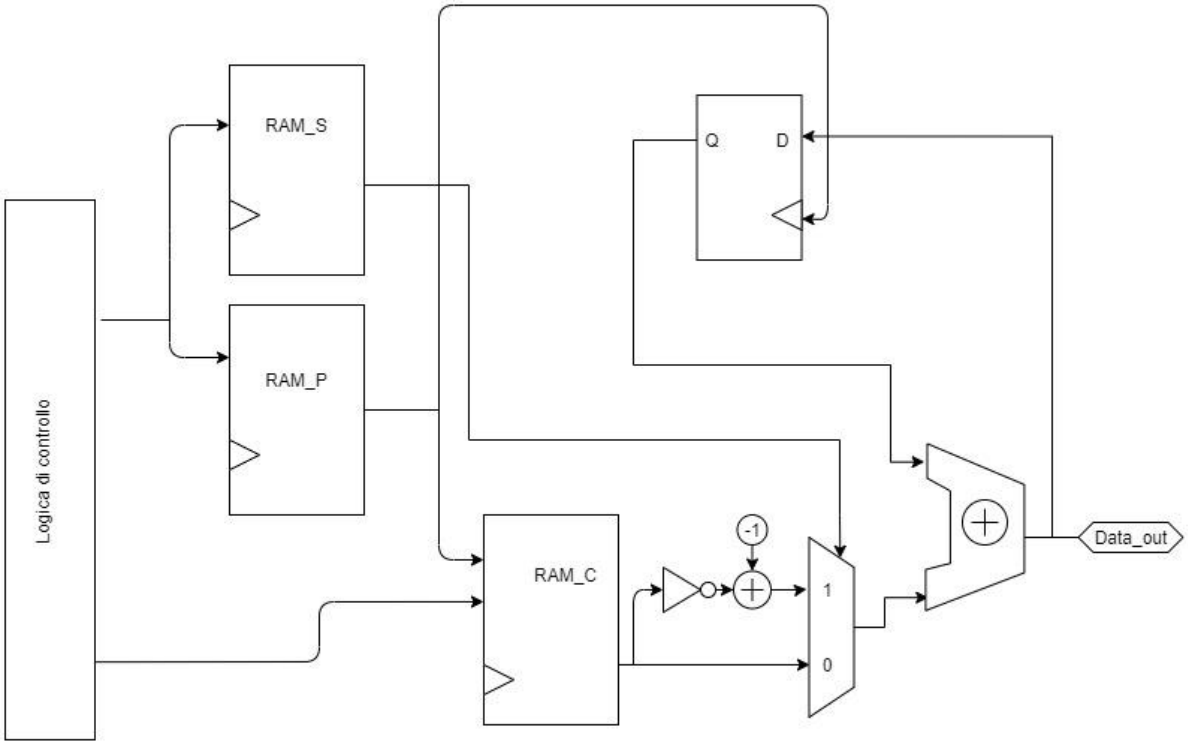


Figura 21: Schema generale

La logica di controllo si occupa come in precedenza della generazione dei segnali di controllo che coordinano le operazioni. Si leggeranno in sequenza i bit d'uscita della RAM_P e della RAM_S, si possono distinguere due casi in base al valore del bit che si ha: peso pari a 1 o peso

pari a 0. Nel caso in cui il valore del peso sia 1, il funzionamento dell'architettura deve portare ad un accumulo del campione. Ciò che accade è che il pin ram_c_en, ovvero la porta che abilita la memoria, sarà controllata dal segnale di p_out in forma negata. In questo modo, avendo un peso dal valore logico alto, il pin ram_c_en assumerà il valore logico basso, la RAM_C sarà abilitata al regolare funzionamento ed aggiornerà la propria uscita con il valore del campione che deve essere accumulato. Anche l'accumulatore avrà una abilitazione selettiva: si sfrutterà un modulo di clock gating per abilitare il clock del registro in retroazione. Per fare ciò la porta di enable del clock gating sarà controllata del valore del p_out, questa volta in forma vera. In questo caso il segnale di enable del clock gating, essendo p_out pari ad 1, sarà anch'esso ad un valore logico alto ed attiverà il registro in retroazione al regolare funzionamento. Quindi il sommatore dell'accumulatore sommerà il valore precedentemente memorizzato nel registro con il campione appena fornito dalla RAM_C, selezionato tra il valore positivo e negativo.

Il vantaggio sostanziale sta nel fatto che, nel caso in cui si presenta un peso p_out pari a 0, il segnale ram_c_en, della RAM_C, sarà pari ad 1 e la memoria dei campioni verrà dunque disabilitata. Si ricorda che in questa condizione operativa l'uscita non viene aggiornata e mantiene l'ultimo dato letto. Il clock gating del registro avrà un segnale di enable pari a 0, quindi disabiliterà il clock che non permetterà l'aggiornamento del valore memorizzato. Il sommatore vedrà dunque ad un suo ingresso il valore congelato del registro in retroazione, nel suo altro ingresso lo stesso valore che aveva al ciclo di somma precedente. Entrambi gli ingressi del sommatore pertanto non subiranno variazioni, quindi non ci sarà commutazione dei segnali interni.

Dato che i pesi di una riga della memoria P sono per l'80% pari a 0 queste scelte architetturali hanno portato a vantaggi in termini di consumo, dal momento che hanno ridotto la potenza dinamica evitando di aggiornare il registro in retroazione quando non necessario ed evitando di leggere parimenti la RAM_C quando il campione verrà pesato per uno 0.

La scelta di introdurre una memoria che andasse a raccogliere i valori relativi al segno ha introdotto necessariamente un raddoppio dei moduli SRAM, il che, è immediato, comporta un incremento di area necessaria per il floor plan, un raddoppio intuitivamernte, ed ancora un raddoppio di potenza.

Descrizione RTL del sistema

Nel presente capitolo si descrive l'architettura digitale realizzata in questo progetto di tesi, la quale consiste in un modulo digitale descritto a livello RTL in System Verilog che realizza i prodotti matrice-vettore seguendo l'architettura mostrata precedentemente per minimizzare la potenza dissipata. Si partirà con la presentazione dei vari moduli che caratterizzano il sistema, fino ad arrivare ad un modulo top level che verrà utilizzato per simulazioni funzionali nell'ambiente ModelSim ed una sintesi con tecnologia CMOS a 90nm per ottenere delle stime di potenza.

RAM_C

Nel caso delle memorie, come già ampiamente discusso nella tesi di Davide Molino, sono stati forniti dei modelli da STMicroelectronics, tramite una relativa descrizione RTL in Verilog. Del modulo C090D_ST_SPREG_320x8m4_CU5MAP_mr_pc, utilizzato per la RAM_C e del modulo C090D_ST_SPREG_128x64m4_CU5MAP_mr_pc, utilizzato per la RAM_P non verrà mostrata la descrizione RTL, poiché è stata fornita da STMicrolectronics solo una descrizione ai terminali soggetta a clausola di riservatezza al solo scopo di simulazione funzionale e stima di potenza del sistema in cui è utilizzata.

Il modulo RAM_C, già introdotto da Davide Molino. si limita ad essere una istanza del modello C090D_ST_SPREG_320x8m4_CU5MAP_mr_pc fornito. Come già introdotto nel capitolo precedente, i pin sono stati mappati per soddisfare le esigenze progettuali. Con il codice System Verilog che segue si realizza esattamente il modello circuitale introdotto nel capitolo 3 in figura

Module RAM_C

```
module RAM_C (clk, c_in, indirizzo, ram_c_en, w_c_en,
c_out_ram);
input logic clk;
input wire w_c_en, ram_c_en;
input logic [7:0] c_in;
input wire [8:0]  indirizzo;
output logic [7:0] c_out_ram;
C090D_ST_SPREG_320x8m4_CU5MAP_mr_pc  ram_c_ST
(.A(indirizzo), .CK(clk), .CSN(ram_c_en), .D(c_in), .Q(c_ou
t_ram), .TBYPASS(1'b0), .WEN(w_c_en));
```

RAM_P

Per la RAM_P, come già introdotto nella tesi di Davide Molino, sono state stanziate cinque moduli STMicroelectronics C090D_ST_SPREG_128x64m4_CU5MAP_mr_pc per poter costituire la memoria con le dimensioni richieste di 128 righe e 320 colonne. Il multiplexer MUX_P_OUT permette di leggere in sequenza l'uscita dalla memoria utilizzando indirizzo [8:0] come segnale di controllo per discriminare qual è il bit richiesto dei 320 disponibili in uscita dai cinque moduli di memoria connessi in parallelo. Attraverso questi moduli si realizza la struttura mostrata precedentemente in figura .Segue il codice System Verilog che implementa quanto appena descritto.

Module RAM_P

module MUX_P_OUT (p_out_ram_1, p_out_ram_2, p_out_ram_3, p_out_ram_4,
p_out_ram_5, indirizzo, p_out_ram);

input logic [63:0] p_out_ram_1;
input logic [63:0] p_out_ram_2;
input logic [63:0] p_out_ram_3;
input logic [63:0] p_out_ram_4;
input logic [63:0] p_out_ram_5;
input logic [8:0] indirizzo;
output logic p_out_ram;

```
always_comb begin
 if (indirizzo >= 0 & indirizzo <= 63)
   p_out_ram = p_out_ram_1 [indirizzo];
 if (indirizzo >= 64 & indirizzo <= 127)
   p_out_ram = p_out_ram_2 [indirizzo-64];
 if (indirizzo >= 128 & indirizzo <= 191)
   p_out_ram = p_out_ram_3 [indirizzo-128];
 if (indirizzo >= 192 & indirizzo <= 255)
   p_out_ram = p_out_ram_4 [indirizzo-192];
 if (indirizzo >= 256 & indirizzo <= 319)
   p_out_ram = p_out_ram_5 [indirizzo-256];
 if (indirizzo > 319)
   p_out_ram = 0;
end
endmodule
module RAM_P (clk, p_in_1, p_in_2, p_in_3, p_in_4, p_in_5, sel_p, indirizzo, ram_p_en,
w_p_en, p_out_ram);
input wire clk;
input logic [63:0] p_in_1;
input logic [63:0] p_in_2;
input logic [63:0] p_in_3;
input logic [63:0] p_in_4;
input logic [63:0] p_in_5;
input wire [6:0] sel_p;
input logic [8:0] indirizzo;
input wire ram_p_en, w_p_en;
output logic p_out_ram;
wire [63:0] p_out_ram_1;
wire [63:0] p_out_ram_2;
wire [63:0] p_out_ram_3;
wire [63:0] p_out_ram_4;
wire [63:0] p_out_ram_5;
```

```
C090D_ST_SPREG_128x64m4_CU5MAP_mr_pc RAM_P_1 (.A(sel_p), .CK(clk),
.CSN(ram_p_en), .D(p_in_1), .Q(p_out_ram_1), .TBYPASS(1'b0), .WEN(w_p_en));

C090D_ST_SPREG_128x64m4_CU5MAP_mr_pc RAM_P_2 (.A(sel_p), .CK(clk),
.CSN(ram_p_en), .D(p_in_2), .Q(p_out_ram_2), .TBYPASS(1'b0), .WEN(w_p_en));

C090D_ST_SPREG_128x64m4_CU5MAP_mr_pc RAM_P_3 (.A(sel_p), .CK(clk),
.CSN(ram_p_en), .D(p_in_3), .Q(p_out_ram_3), .TBYPASS(1'b0), .WEN(w_p_en));

C090D_ST_SPREG_128x64m4_CU5MAP_mr_pc RAM_P_4 (.A(sel_p), .CK(clk),
.CSN(ram_p_en), .D(p_in_4), .Q(p_out_ram_4), .TBYPASS(1'b0), .WEN(w_p_en));

C090D_ST_SPREG_128x64m4_CU5MAP_mr_pc RAM_P_5 (.A(sel_p), .CK(clk),
.CSN(ram_p_en), .D(p_in_5), .Q(p_out_ram_5), .TBYPASS(1'b0), .WEN(w_p_en));

MUX_P_OUT mux_p_out (p_out_ram_1, p_out_ram_2, p_out_ram_3, p_out_ram_4,
p_out_ram_5, indirizzo, p_out_ram);

endmodule
```

RAM_S

Per il progetto si rende necessario l'utilizzo di ulteriori blocchi di memoria per la memorizzazione della matrice S, quindi, analogamente a quando detto per le RAM_P nella tesi di Davide Molino,per la RAM_S sono state stanziate cinque moduli STMicroelectronics C090D_ST_SPREG_128x64m4_CU5MAP_mr_pc per poter costituire la memoria con le dimensioni richieste di 128 righe e 320 colonne. Il multiplexer MUX_S_OUT permette di leggere sequenzialmente l'uscita dalla memoria s utilizzandoil segnale indirizzo [8:0] come segnale di controllo per discriminare qual è il bit richiesto dei 320 disponibili in uscita dai cinque moduli di memoria connessi in parallelo. Attraverso questi moduli si realizza la struttura

mostrata precedentemente in figura [fig. L]. Segue il codice System Verilog che implementa quanto appena descritto.

Module RAM_S

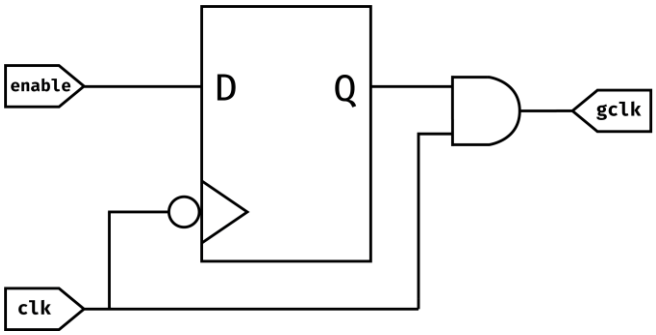
```
module MUX_S_OUT (s_out_ram_1, s_out_ram_2, s_out_ram_3, s_out_ram_4, s_out_ram_5, indirizzo, s_out_ram);

input logic [63:0] s_out_ram_1;

input logic [63:0] s_out_ram_2;

input logic [63:0] s_out_ram_3;

input logic [63:0] s_out_ram_4;

input logic [63:0] s_out_ram_5;

input logic [8:0] indirizzo;

output logic s_out_ram;

always_comb begin

 if (indirizzo >= 0 & indirizzo <= 63)

   s_out_ram = s_out_ram_1 [indirizzo];

 if (indirizzo >= 64 & indirizzo <= 127)

   s_out_ram = s_out_ram_2 [indirizzo-64];

 if (indirizzo >= 128 & indirizzo <= 191)

   s_out_ram = s_out_ram_3 [indirizzo-128];

 if (indirizzo >= 192 & indirizzo <= 255)

   s_out_ram = s_out_ram_4 [indirizzo-192];

 if (indirizzo >= 256 & indirizzo <= 319)

   s_out_ram = s_out_ram_5 [indirizzo-256];

 if (indirizzo > 319)

   s_out_ram = 0;

end

endmodule

module RAM_S (clk, s_in_1, s_in_2, s_in_3, s_in_4, s_in_5, sel_p, indirizzo, ram_p_en, w_p_en, s_out_ram);

input wire clk;

input logic [63:0] s_in_1;
```

```
input logic [63:0] s_in_2;
input logic [63:0] s_in_3;
input logic [63:0] s_in_4;
input logic [63:0] s_in_5;
input wire [6:0] sel_p;
input logic [8:0] indirizzo;
input wire ram_p_en, w_p_en;
output logic s_out_ram;
wire [63:0] s_out_ram_1;
wire [63:0] s_out_ram_2;
wire [63:0] s_out_ram_3;
wire [63:0] s_out_ram_4;
wire [63:0] s_out_ram_5;
C090D_ST_SPREG_128x64m4_CU5MAP_mr_pc RAM_S_1 (.A(sel_p), .CK(clk),
.CSN(ram_p_en), .D(s_in_1), .Q(s_out_ram_1), .TBYPASS(1'b0), .WEN(w_p_en));
C090D_ST_SPREG_128x64m4_CU5MAP_mr_pc RAM_S_2 (.A(sel_p), .CK(clk),
.CSN(ram_p_en), .D(s_in_2), .Q(s_out_ram_2), .TBYPASS(1'b0), .WEN(w_p_en));
C090D_ST_SPREG_128x64m4_CU5MAP_mr_pc RAM_S_3 (.A(sel_p), .CK(clk),
.CSN(ram_p_en), .D(s_in_3), .Q(s_out_ram_3), .TBYPASS(1'b0), .WEN(w_p_en));
C090D_ST_SPREG_128x64m4_CU5MAP_mr_pc RAM_S_4 (.A(sel_p), .CK(clk),
.CSN(ram_p_en), .D(s_in_4), .Q(s_out_ram_4), .TBYPASS(1'b0), .WEN(w_p_en));
C090D_ST_SPREG_128x64m4_CU5MAP_mr_pc RAM_S_5 (.A(sel_p), .CK(clk),
.CSN(ram_p_en), .D(s_in_5), .Q(s_out_ram_5), .TBYPASS(1'b0), .WEN(w_p_en));
MUX_S_OUT mux_s_out (s_out_ram_1, s_out_ram_2, s_out_ram_3, s_out_ram_4,
s_out_ram_5, indirizzo, s_out_ram);
endmodule
```

ARITMETICA

Il modulo ARITMETICA, già definito nella tesi di Davide Molino, realizza la vera e propria l'operazione di moltiplicazione matrice-vettore. In esso è presente l'accumulatore, il modulo di clock gaiting per abilitare e disabilitare il clock del registro in retroazione dell'accumulatore, un modulo REGISTRO_EN che serve ad introdurre un ritardo previsto per avere il giusto timing tra i segnali, ed un modulo chiamato SEND_OUT, in uscita dall'accumulatore. In realtà

quest'ultimo modulo è anch'esso un registro, la cui funzione è campionare il valore in uscita dall'accumulatore, alla fine di ognuna delle 320 somme di prodotti tra una riga di pesi ed i campioni, e mantenere il dato in memoria fino alla fine delle somme successive, per poi, a quel



enable	clk	gclk
0	0	0
0	1	0
1	0	0
1	1	1

Figura 22: Modulo REGISTRO_EN

punto, campionare il nuovo dato, risultato della nuova somma di prodotti. Il tutto fino alla fine delle operazioni di calcolo. In questo modo, un calcolatore esterno può prelevare comodamente il dato che resta disponibile per 320 cicli di clock, ovvero il tempo di moltiplicazione tra una riga di RAM_P e la colonna dei campioni. Sostanzialmente, se il modulo sta elaborando la i-esima riga della memoria dei pesi, in uscita, il dato calcolato disponibile è il dato precedentemente calcolato con la riga (i-1)-esima della RAM_P.

Il segnale gclk, in uscita dal modulo CLK_ENABLE_ACCUMULATORE, è il segnale che comanda il registro posto in retroazione dell'accumulatore. Dalla figura si nota come il segnale di enable, che durante le operazioni di calcolo sarà il valore del peso p_out, subisce una operazione di and logico con il segnale di clock: se il segnale enable vale 0, a prescindere del valore del clock in quell'istante, il gclk viene forzato ad un valore logico basso. Al contrario, quando il segnale di enable è 1 il segnale gclk sarà libero di valere 0 se il clock vale 0, 1 se il segnale di clock vale 1. In questo caso il segnale di clock è libero di agire imponendo il suo andamento al pin d'uscita, con un ritardo calcolato dovuto al latch introdotto.

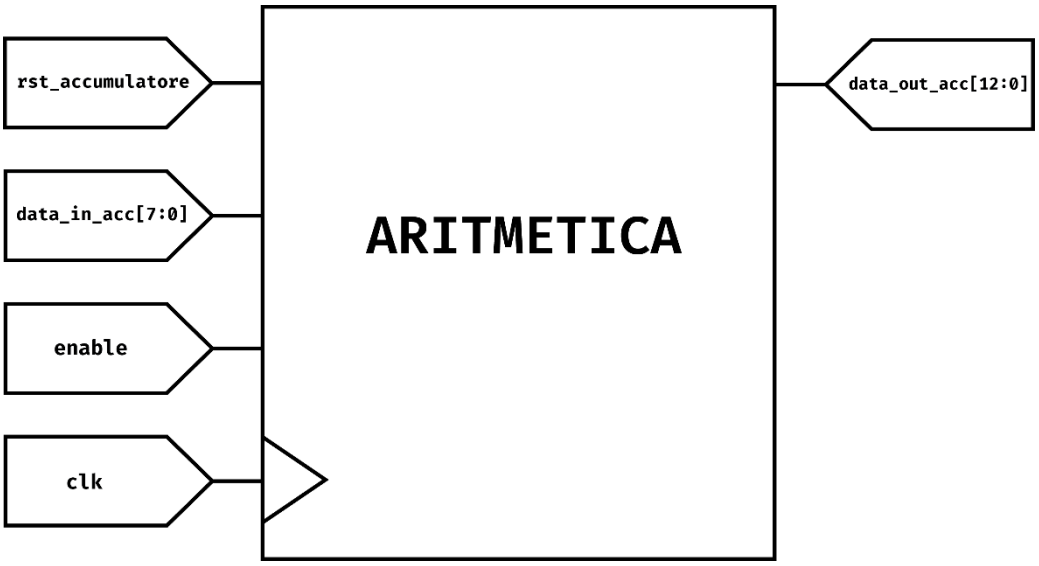


Figura 23: Modulo ARITMETICA

I singoli moduli descritti vengono stanziati in un modulo di gerarchia superiore, il modulo ARITMETICA [fig. P]. Segue il codice RTL in System Verilog che implementa quanto appena descritto. È stato tenuto conto che i segnali in causa, rispetto a quelli descritti da Davide Molino, contengono il segno, quindi è stata utilizzata la sintassi corretta dove evidenziato.

Module ARITMETICA

```
module REGISTRO_EN (clk, D, Q);
input logic clk;
input logic D;
output logic Q;
reg Q_int;
always_ff @ (posedge clk) begin
  Q_int = D;
end
assign Q = Q_int;
endmodule

module CLK_ENABLE_ACCUMULATORE (clk, enable, gclk);
input clk, enable;
output gclk;
wire enable_in = (enable);
```

```
reg enable_latch;
always @(clk or enable_in)
 if (~clk)
   enable_latch <= enable_in;
assign gclk = (clk & enable_latch);
endmodule

module ACCUMULATORE (gclk, rst_accumulatore, data_in_acc, data_out_acc_int);
input logic signed [8:0] data_in_acc;
input logic gclk, rst_accumulatore;
output logic signed [12:0] data_out_acc_int;
logic signed [12:0] data_out_acc_tmp;
always @(posedge gclk, posedge rst_accumulatore) begin
        if (rst_accumulatore)
                data_out_acc_tmp = 13'd0;
        else
                data_out_acc_tmp = data_out_acc_tmp + data_in_acc;
end
assign data_out_acc_int = data_out_acc_tmp;
endmodule

module SEND_OUT (rst_accumulatore, data_out_acc_int, data_out_acc);
input logic rst_accumulatore;
input logic signed [12:0] data_out_acc_int;
output logic signed [12:0] data_out_acc;
logic [12:0] data_out_acc_tmp;
always @(posedge rst_accumulatore) begin
                data_out_acc_tmp = data_out_acc_int;
end
assign data_out_acc = {13{data_out_acc_tmp}};
endmodule
```

```
module ARITMETICA (clk, rst_accumulatore, enable, data_in_acc, data_out_acc);
input wire clk, rst_accumulatore;
input logic enable;
input logic signed [8:0] data_in_acc;
output logic signed [12:0] data_out_acc;
wire gclk, enable_reg;
wire [12:0] data_out_acc_int;
REGISTRO_EN registro_en (clk, enable, enable_reg);
CLK_ENABLE_ACCUMULATORE clk_enable_accumulatore (clk, enable_reg, gclk);
ACCUMULATORE accumulatore (gclk, rst_accumulatore, data_in_acc, data_out_acc_int);
SEND_OUT send_out (rst_accumulatore, data_out_acc_int, data_out_acc);
endmodule
```

CONTATORI

In questa sezione verranno analizzati due moduli, come realizzato da Davide Molino, che si occupano sia della generazione degli indirizzi di memoria, attraverso un meccanismo di conteggio, sia della generazione di segnali di controllo per la macchina a stati finiti. Entrambi i moduli dialogheranno proprio con la macchina a stati per capire quando inizierà il conteggio e quando sarà terminato, durante ogni fase operativa del modulo digitale.

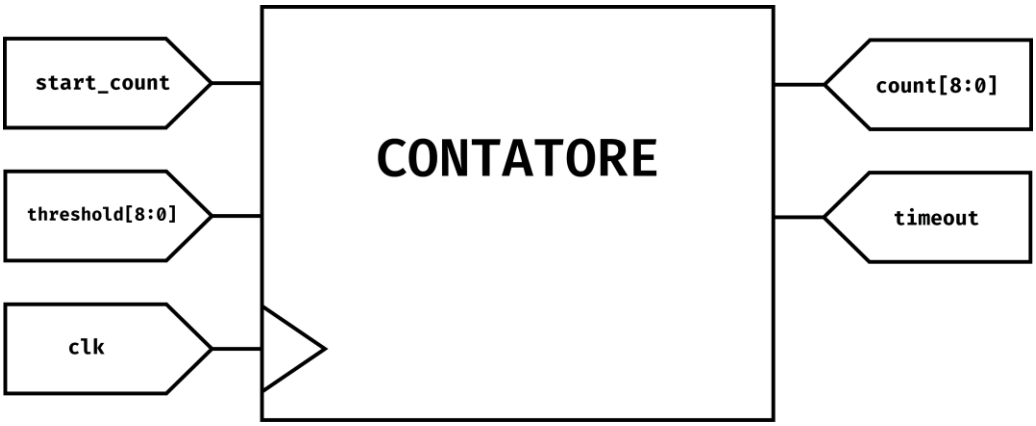


Figura 24: Modulo CONTATORE

Il primo modulo descritto è il modulo CONTATORE. Esso ha tre pin di ingresso e due di uscita. Il pin start_count può assumere il valore logico alto o basso: durante tutto il tempo in cui il suo valore è pari ad 1 il modulo è abilitato a contare, fino a quando il valore di start_count non viene forzato a 0, o fino a quando non si raggiunge il valore di soglia imposto attraverso un altro pin di input threshold [8:0]. Raggiunto questo valore, il segnale di timeout si porta al valore 1 ed

indica la fine dell'operazione. Il segnale count [8:0] è il valore del conteggio che viene incrementato di una unità decimale ogni fronte di salita del clock, se il segnale start_count vale 1.

È bene distinguere le fasi operative nelle quali ci si trova perché il funzionamento di questo modulo cambierà in funzione di esse. Quando ci si trova nella fase di scrittura delle memorie esso si occupa di generare gli indirizzi di riga nei quali vengono scritti i dati. Il segnale sel_p [6:0], indirizzo di riga della RAM_P e RAM_S, partirà da un valore nullo, per scrivere i pesi nella prima riga della memoria ed arriverà fino al valore 127, che indica l'ultima riga della matrice dei pesi. Il segnale count [8:0] farà esattamente ciò: iniziata la scrittura varrà 0 e poi sarà incrementato ogni istante di clock fino al valore 127. Valore imposto dalla macchina a stati sul pin di threshold [8:0]. Si nota come i bit utilizzati per esprimere sel_p [6:0], non combaciano dimensionalmente con i bit del conteggio count [8:0]. I due bit in più sono dovuti al fatto che lo stesso modulo viene utilizzato anche per la generazione dell'indirizzo della RAM_C, dove, per descrivere 320 posizioni, saranno necessari 9 bit.

Nel caso di sel_p [6:0] sono stati collegati i soli sette bit necessari per esprimere le 127 posizioni. Per quanto riguarda la scrittura della RAM_C, invece, ciò che accade è del tutto analogo alla scrittura dei pesi, eccezione fatta per il segnale di threshold [8:0], che dovrà permettere al count [8:0] di arrivare ad un valore di 319 per completare le operazioni di scrittura.

Durante la fase di calcolo, nella quale si effettuano le somme di prodotti, il segnale sel_p [6:0] non sarà generato dal modulo CONTATORE, il segnale threshold [8:0] del modulo CONTATORE sarà stabilmente al valore che consente il conteggio da 0 a 319 per permettere al segnale indirizzo [8:0] di scorrere tra le righe della RAM_C e tra le colonne delle memorie RAM_P e RAM_S.

Figura 25: Modulo INCREMENTRATORE

Il modulo INCREMENTATORE verrà utilizzato solo durante le operazioni di calcolo, proprio per generare il segnale sel_p [6:0], durante il resto delle operazioni il suo segnale di reset, rest_inc, posto ad un valore logico alto, disabiliterà il modulo. Il suo unico altro pin di ingresso è *incrementa*: lo si utilizza come segnale per scegliere quando incrementare il valore inc [7:0].

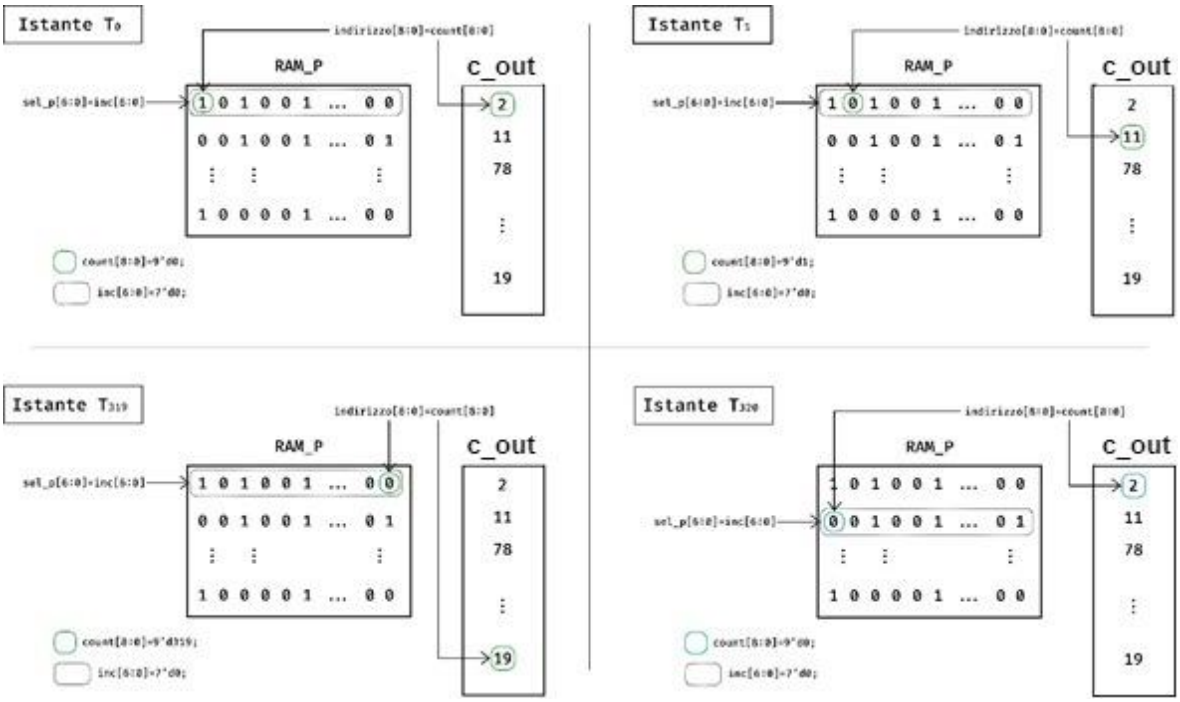


Figura 26: Descrizione scorrimento memorie

Inoltre, quando il segnale inc [7:0] raggiungerà il valore decimale 128, il segnale done_calcola assumerà il valore logico alto e segnalerà la fine delle operazioni di calcolo. Anche qui si può notare un'incongruenza tra i bit del segnale sel_p [6:0] ed i bit del suo generatore inc [7:0]. In realtà il segnale assegnato sarà inc [6:0], e il bit in più è utilizzato per poter far arrivare il valore dell'incremento a 128, esprimibile con almeno 7bit (log2(128) bit = 7bit), necessario per l'allineamento con gli altri segnali.

Come già detto, i moduli CONTATORI si occupano della generazione degli indirizzi di memoria. In figura [fig. L] è mostrato lo scorrimento degli indirizzi durante la fase di calcolo. Nell'istante T0 sono iniziate le operazioni di calcolo: è stato selezionato il primo indirizzo di riga della RAM_P e RAM_S, ed i bit iniziano ad essere letti in sequenza. Il primo peso selezionato verrà confrontato con il primo campione, e il bit di S selezionerà il segno corretto. A questo punto, nell'istante T1 il valore di sel_p [6:0] non varierà, mentre il modulo CONTATORE aggiungerà uno al valore di count [8:0], che si passerà dal valore decimale 0 al valore decimale 1. Il valore di inc [6:0] resterà stabile fino all'ultima coppia peso-campione all'indirizzo 319, come mostrato all'istante T319. A questo punto, sarà stata completata la moltiplicazione tra la riga 0 dei pesi e dei segni e la colonna dei campioni ed il segnale timeout

del CONTATORE, assumerà il valore 1. All'istante successivo, T320, il valore incrementa, del modulo INCREMENTATORE, sarà pari ad uno ed il valore di inc [6:0] verrà incrementato di

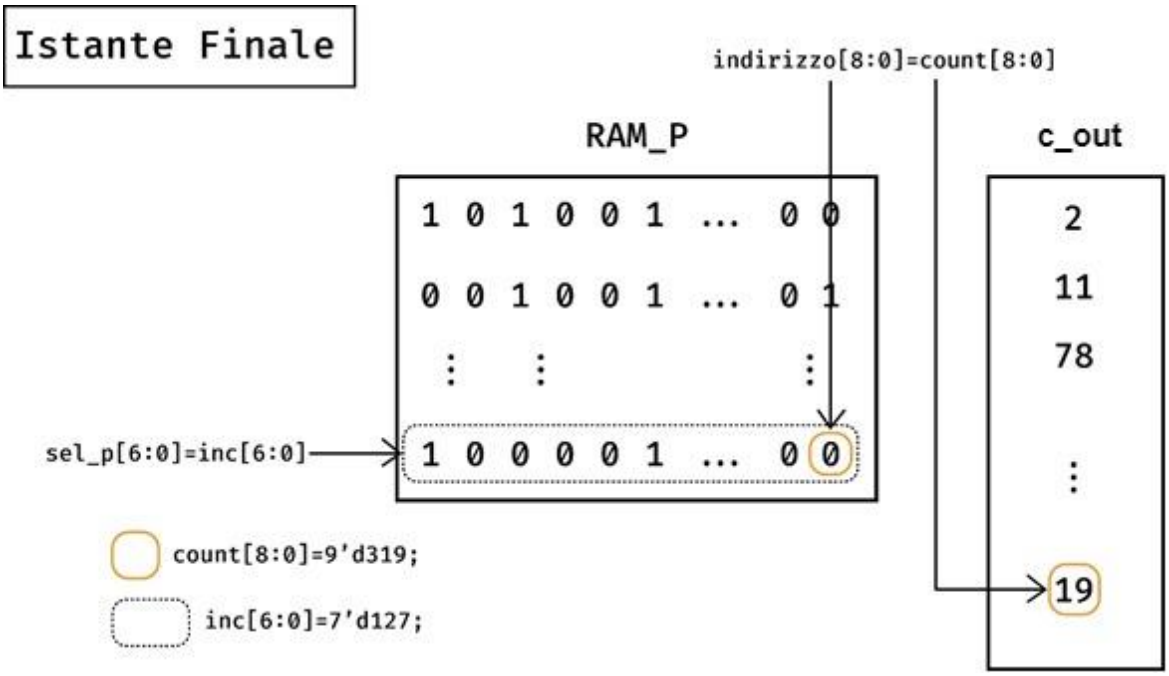


Figura 27: Descrizione istante finale

una unità decimale. Ciò porterà il valore di sel_p [6:0] a scorrere e selezionare la seconda riga della RAM_P e della RAM_S. Ripartiranno da qui le operazioni di moltiplicazione tra la riga 1 della RAM_P e della RAM_S e la colonna dei campioni. Le operazioni andranno avanti con questo meccanismo fino a quando non si arriverà all'ultimo elemento nell'ultima riga della RAM_P e della RAM_S. Come mostrato in figura, il valore di sel_p [6:0] sarà pari a 127 ed il valore di indirizzo [8:0] a 319. A questo punto, dopo aver pesato anche l'ultimo campione, il segnale done_calcola, in uscita dall'INCREMENTATORE, si porterà ad un valore logico alto per segnalare alla macchina a stati la fine della moltiplicazione tra RAM_P e il valore RAM_C col segno definito da RAM_S.

Segue il codice RTL in System Verilog che implementa il modulo CONTATORI, che al suo interno contiene i due moduli appena descritti.

Module CONTATORI

```
module CONTATORE (clk, start_count, threshold, count, timeout);
input logic clk, start_count;
input logic [8:0] threshold;
output logic timeout;
output logic [8:0] count;
```

```
logic [8:0] count_int;
always_ff @(posedge clk, negedge start_count) begin
	if (~start_count)
		count_int <= 9'd0;
	else if (timeout)
		count_int <= 9'd0;
	else
		count_int <= count_int + 9'd1;
end
always_comb begin
timeout <= (count == threshold) ? 1'b1 : 1'b0;
end
assign count = count_int - 9'd1;
endmodule
module INCREMENTATORE (incrementa, rst_inc, inc, done_calcola);
input logic incrementa, rst_inc;
output logic [7:0] inc;
output logic done_calcola;
logic [7:0] inc_int = 8'd0;
always_ff @ (posedge incrementa, posedge rst_inc) begin
	if (rst_inc)
		inc_int <= 8'd0;
	else
		inc_int <= inc_int + 8'd1;
end
always_comb begin
	done_calcola <= (inc == 8'd128) ? 1'b1 : 1'b0;
end
assign inc = inc_int;
endmodule
```

FSM

Il modulo FSM rappresenta il cuore della logica del circuito in quanto gestisce ogni condizione operativa dei singoli moduli visti nei paragrafi precedenti, dialogando sia con loro che con il mondo esterno. La struttura di base è stata definita nella tesi di Davide Molino, opportunamente modificata per adattarla al caso d'uso.

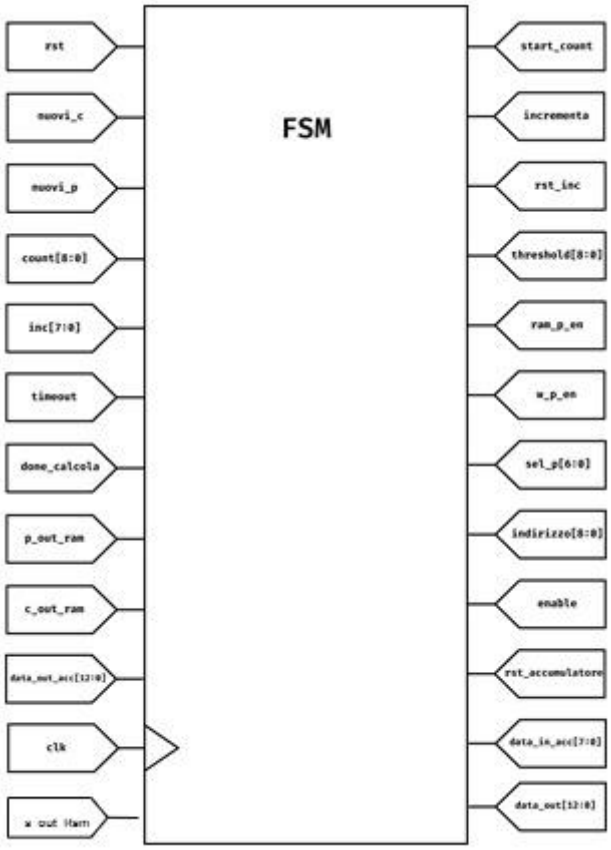


Figura 28: Modulo FSM

I segnali nuovi_c, nuovi_p e rst, sono l'interfaccia della macchina a stati con il mondo esterno. Il segnale rst serve per effettuare un reset del registro di stato e tornare, in qualsiasi momento, nello stato di RIPOSA. Si può notare dal modulo in figura, che tutti gli altri segnali di input del modulo FSM provengono dagli output dei moduli precedentemente introdotti. Mentre, i pin di uscita della macchina a stati sono i segnali di controllo che bisogna generare per avere il corretto funzionamento di tutti gli altri moduli descritti. Il pin data_out [12:0], che rappresenta un elemento del vettore risultante dalla moltiplicazione di una riga di pesi per la colonna di campioni, sarà disponibile direttamente come uscita dal modulo FSM e non dall'accumulatore stesso. Si è scelto di fare ciò per avere maggiore controllo dei segnali durante le varie fasi

operative. In figura sotto è mostrato il grafo degli stati che si è scelto di realizzare con la macchina a stati finiti. Verranno analizzati uno alla volta per comprendere il comportamento dei vari moduli, coordinati dalla FSM, al variare degli stati.

Si parte dallo stato RIPOSA, che rappresenta la condizione di disattivazione della maggior parte dei moduli. Il CONTATORE avrà il segnale di reset attivo, quindi il segnale count [8:0] sarà una stringa di zeri, il segnale start_inc varrà 0, quindi anche l'INCREMENTATORE sarà disabilitato. L'ACCUMULATORE avrà il clock disabilitato ed entrambe le memorie avranno il segnale di enable, ram_p_en e ram_c_en, disabilitato. Per entrare in questa condizione di

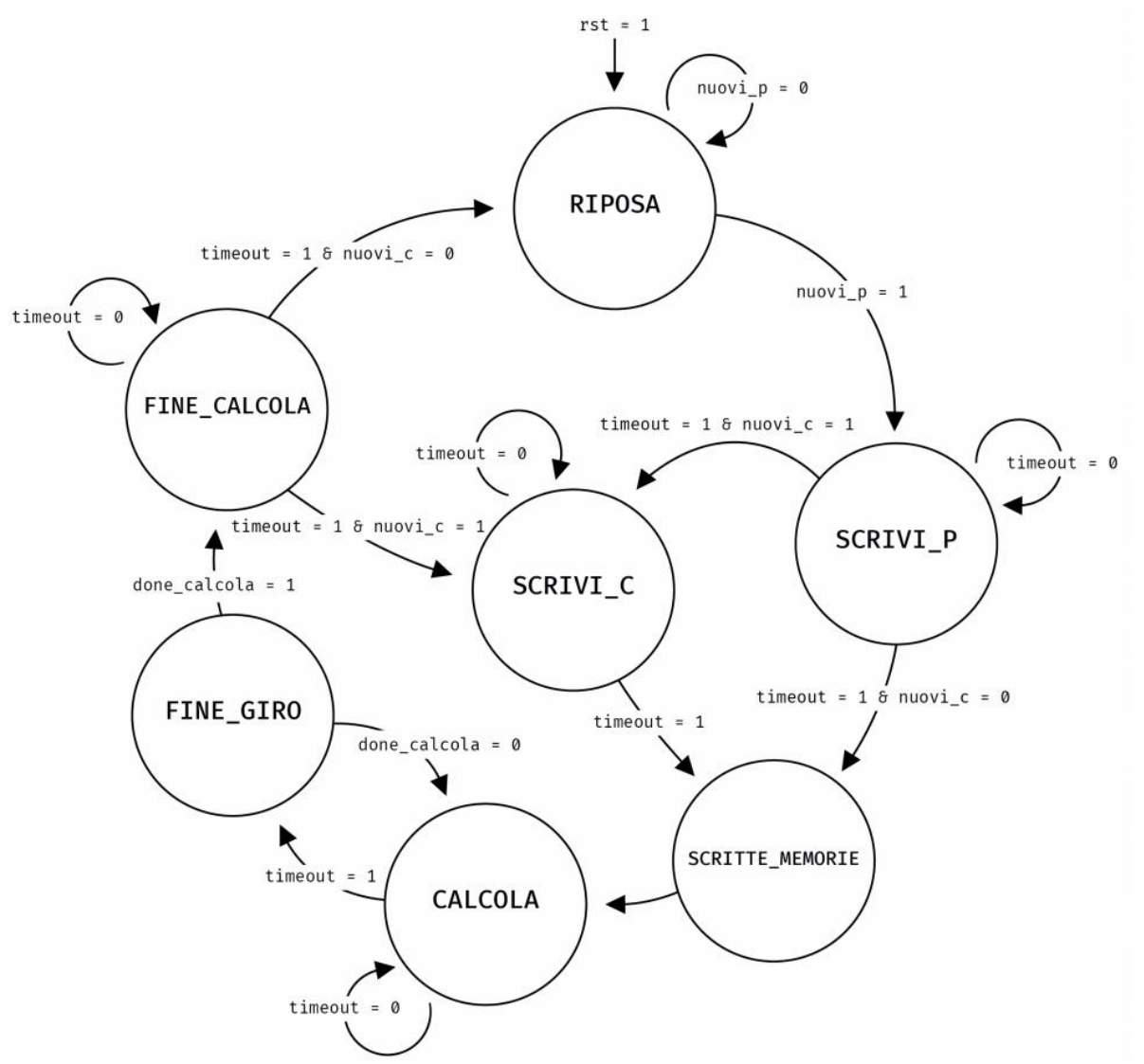


Figura 29: Modalità passaggio stati

funzionamento ci sono due possibilità: un reset forzato, sincrono, tramite il segnale rst di

sistema, quando esso assume il valore 1, oppure il termine delle operazioni di calcolo segnalato dal passaggio al valore logico alto del segnale done_calcola. Il circuito rimarrà in questa condizione di funzionamento fino a quando il segnale nuovi_p non assumerà il valore logico alto.

Nel momento in cui il segnale nuovi_p diventa pari ad 1, si passa allo stato SCRIVI_P, nel quale il circuito campionerà i valori dei cinque pin dei pesi in ingresso e dei cinque pin dei segni in ingresso, che verranno memorizzati rispettivamente nella RAM_P e nella RAM_S. In questa condizione di funzionamento, il modulo ARITMETICA, la RAM_C e il modulo INCREMENTATORE continueranno ad essere disabilitati, mentre il CONTATORE inizierà a contare, generando il segnale sel_p [6:0] per la RAM_P, che abilitata, memorizzerà i dati nei rispettivi indirizzi ricevuti. Quando l'operazione di scrittura sarà terminata, il CONTATORE genererà il segnale di timeout che avvertirà la FSM dell'avvenuta operazione.

A questo punto il segnale nuovi_c discriminerà lo stato successivo: se non saranno disponibili nuovi campioni, nuovi_c pari a 0, si entrerà in uno stato transitorio chiamato SCRITTE_MEMORIE. Se, al contrario, saranno disponibili nuovi campioni, si entrerà in uno stato analogo al precedente, nel quale però, avverrà la scrittura della RAM_C, chiamato SCRIVI_C. Al termine del conteggio, questa volta pari a 319, il dispositivo si configurerà egualmente nello stato di SCRITTE_MEMORIE. Questo stato dura un solo periodo di clock e sarà necessario per permettere l'allineamento di tutti i segnali interni. Gli stati descritti finora vengono riportati dal lavoro di tesi di Davide molino, la modifica sostanziale, effettuata per adattare al caso il codice, risiede nello stato CALCOLA dove viene introdotta la selezione del segno.

Nello stato di CALCOLA, terminata la fase di scrittura delle RAM, i moduli saranno pronti ad iniziare le operazioni di moltiplicazione matrice-vettore. Si entrerà a questo punto in un loop di

calcolo che rappresenta la fase più lunga ed energicamente dispendiosa per il modulo. Come spiegato in precedenza, si tratta di 128 righe della matrice RAM_P, ciascuna con 320 elementi, che vengono moltiplicate, una alla volta, per la colonna di campioni: in questa situazione la FSM assocerà al segnale ram_c_en, pin per abilitare la memoria dei campioni, il valore negato di p_out. In questo modo, iniziata la lettura delle memorie coordinate dai CONTATORI, come visto nel paragrafo precedente, gli accessi in memoria della RAM_C saranno effettuati solo quando un peso letto avrà valore 1. Inoltre, il segnale p_out in forma vera, durante le operazioni di calcolo, sarà anche il segnale che abilita il clock dell'ACCUMULATORE. Il segnale c_out riporta il campione memorizzato in RAM_C, del quale ne viene calcolato il valore con segno negativo "-c_out", sommando 1 allo stesso in complemento ad 1, il segno verrà selezionato a seconda del valore 0/1 in RAM_S utilizzando un multiplexer. Tornando al funzionamento del modulo nello stato di CALCOLA, per effettuare ciascuna operazione tra una riga di pesi, e segni, e la RAM_C sarà necessario che il segnale indirizzo [8:0] scorra su tutti i 320 elementi delle memorie: di questo si occuperà il modulo CONTATORE. Terminate le operazioni tra una riga di pesi e la colonna di campioni, il segnale di timeout porterà il circuito nello stato FINE_GIRO. A questo punto l'INCREMENTATORE aumenterà di 1 il valore decimale di sel_p [6:0], e dopodiché, ricominceranno le somme di moltiplicazioni per la riga di pesi successiva.

Terminate le 128 righe della matrice RAM_P, il segnale done_calcola sarà pari ad 1, perché l'INCREMENTATORE avrà raggiunto il suo massimo. Si uscirà quindi dal loop di calcolo, transitando nello stato FINE_CALCOLA. Si resterà in questo stato per ulteriori 320 periodi per mantenere disponibile l'ultimo elemento del vettore colonna, data_out [12:0], per eventuali campionatori esterni.

Dopodiché se non ci saranno nuovi campioni, le operazioni termineranno e il circuito entra nello stato di RIPOSA, altrimenti se si dovesse avere il valore logico alto per il segnale di

nuovi_c si tornerà nello stato SCRIVI_C, nel quale avverrà una sovrascrittura della memoria con i nuovi campioni disponibili e, successivamente, riprenderanno le operazioni di calcolo moltiplicando la matrice dei pesi per la nuova colonna di campioni.

Segue la descrizione RTL in System Verilog del modulo FSM.

Module FSM

```
module FSM (clk, rst, nuovi_c, nuovi_p, count, timeout, done_calcola, inc, p_out_ram,
c_out_ram, s_out_ram, data_out_acc, start_count, threshold, incrementa, rst_inc,
ram_p_en, w_p_en, sel_p, indirizzo, ram_c_en, w_c_en, enable, rst_accumulatore,
data_in_acc, data_out);

input logic clk, rst, nuovi_c, nuovi_p;


input logic [8:0] count;
input logic [7:0] inc;
input logic timeout, done_calcola;
input logic s_out_ram;
input logic p_out_ram;
input logic [7:0] c_out_ram;
input logic [12:0] data_out_acc;
output logic start_count, incrementa, rst_inc;
output logic [8:0] threshold;
output logic ram_p_en, w_p_en;
output logic [6:0] sel_p;
output logic [8:0] indirizzo;
output logic ram_c_en, w_c_en;
output logic enable, rst_accumulatore;
output logic signed [8:0] data_in_acc;
output logic [12:0] data_out;
reg segno;
reg ritardo;
```

```
enum {RIPOSA, SCRIVI_C, SCRIVI_P, SCRITTE_MEMORIE, CALCOLA, FINE_GIRO,
FINE_CALCOLA, FINE} state, next_state;
always_comb begin
case (state)
RIPOSA:
begin
 if (nuovi_p)
 next_state = SCRIVI_P;
 else
 next_state = RIPOSA;
end
SCRIVI_P:
begin
 if (~timeout)
 next_state = SCRIVI_P;
 else if (timeout & nuovi_c)
 next_state = SCRIVI_C;
 else if (timeout & ~nuovi_c)
 next_state = SCRITTE_MEMORIE;
end
SCRIVI_C:
begin
 if (timeout)
 next_state = SCRITTE_MEMORIE;
 else
 next_state = SCRIVI_C;
end
SCRITTE_MEMORIE:
begin
 next_state = CALCOLA;
end
CALCOLA:
```

```
begin
 if (timeout)
 next_state = FINE_GIRO;
 else
 next_state = CALCOLA;
end
FINE_GIRO:
begin
 if (done_calcola)
 next_state = FINE_CALCOLA;
 else
 next_state = CALCOLA;
end
FINE_CALCOLA:
begin
if (~timeout)
next_state = FINE_CALCOLA;
else if (timeout & ~nuovi_c)
next_state = RIPOSA;
else if (timeout & nuovi_c)
next_state = SCRIVI_C;
end
default: next_state = RIPOSA;
endcase
end
always_ff @ (posedge clk) begin
   if (rst)
     state <= RIPOSA;
   else
     state <= next_state;
end
always_comb begin
```

```
case (state)
RIPOSA:
begin
  start_count = 0;
  incrementa = 0;
  rst_inc = 1;
  threshold = 9'd0;
  ram_p_en = 1;
          w_p_en = 0;
  sel_p = 7'd0;
  indirizzo = 9'd0;
  ram_c_en = 1;
  w_c_en = 0;
  rst_accumulatore = 1;
  data_in_acc = 9'd0;
  enable = 0;
  data_out = 13'dx;
end
SCRIVI_C:
begin
  start_count = 1;
  incrementa = 0;
  rst_inc = 0;
  threshold = 9'd319;
  ram_p_en = 1;
  w_p_en = 0;
  sel_p = 7'd0;
  indirizzo = count;
  ram_c_en = 0;
  w_c_en = 0;
  rst_accumulatore = 1;
  data_in_acc = 9'd0;
```

```
 enable = 0;
 data_out = 13'dx;
end
SCRIVI_P:
begin
 start_count = 1;
 incrementa = 0;
 rst_inc = 0;
 threshold = 9'd127;
 ram_p_en = 0;
 w_p_en = 0;
 sel_p = count [6:0];
 indirizzo = 9'd0;
 ram_c_en = 1;
 w_c_en = 0;
 rst_accumulatore = 1;
 data_in_acc = 9'd0;
 enable = 0;
 data_out = 13'dx;
end
SCRITTE_MEMORIE:
begin
 start_count = 1;
 incrementa = 0;
 rst_inc = 0;
 threshold = 9'd319;
 ram_p_en = 0;
 w_p_en = 1;
 sel_p = 7'd0;
 indirizzo = 9'd0;
 ram_c_en = 0;
 w_c_en = 1;
```

```
  rst_accumulatore = 1;
  data_in_acc = 9'd0;
  enable = 0;
  data_out = 13'dx;
end
CALCOLA:
begin
  start_count = 1;
  incrementa = 0;
  rst_inc = 0;
  threshold = 9'd320;

  ram_p_en = 1;
  w_p_en = 0;
  sel_p = inc [6:0];
  indirizzo = count;
  ram_c_en = ~p_out_ram;
  w_c_en = 1;
  rst_accumulatore = 0;
  ritardo <= s_out_ram;
  segno <= ritardo;
  data_in_acc = segno ? -c_out_ram : c_out_ram;
  enable = p_out_ram;
  data_out = data_out_acc;
end
FINE_GIRO:
begin
  start_count = 0;
  incrementa = 1;
  rst_inc = 0;
  threshold = 9'd320;
  ram_p_en = 0;
```

```
  w_p_en = 1;
  sel_p = inc [6:0];
  indirizzo = 9'd0;
  ram_c_en = ~p_out_ram;
  w_c_en = 1;
  rst_accumulatore = 1;
  data_in_acc = 9'd0;
  enable = 0;
  data_out = data_out_acc;
end
FINE_CALCOLA:
begin
  start_count = 1;
  incrementa = 0;
  rst_inc = 1;
  threshold = 9'd319;
  ram_p_en = 1;
          w_p_en = 0;
  sel_p = 7'd0;
  indirizzo = 9'd0;
  ram_c_en = 1;
  w_c_en = 0;
  rst_accumulatore = 0;
  data_in_acc = 9'd0;
  enable = 0;
  data_out = data_out_acc;
end
default:
begin
  start_count = 0;
  incrementa = 0;
  rst_inc = 1;
```

```
 threshold = 9'd0;
 ram_p_en = 1;
        w_p_en = 0;
 sel_p = 7'd0;
 indirizzo = 9'd0;
 ram_c_en = 1;
 w_c_en = 0;
 rst_accumulatore = 1;
 data_in_acc = 9'd0;
 enable = 0;
 data_out = 13'dx;
end
endcase
end
endmodule
```

CIRCUITO

Il sistema viene descritto in un modulo complessivo, denominato CIRCUITO, che costituisce il top level del sistema. Esso è stato utilizzato per la simulazione nell'ambiente ModelSim e

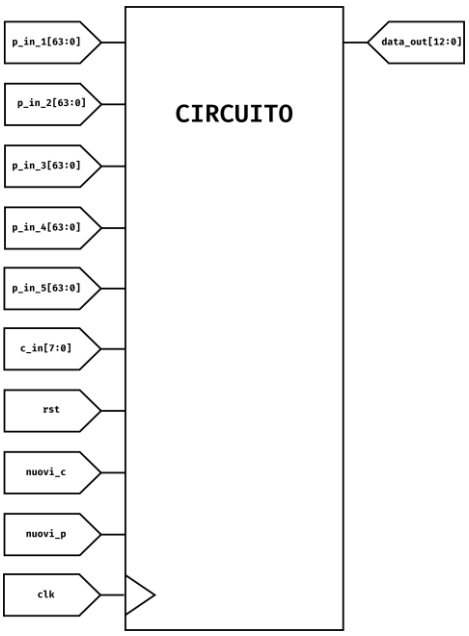


Figura 30: Modulo CIRCUITO

per la sintesi con celle di libreria STMicroelectronics. Questo modulo presenta delle differenze rispetto a quello introdotto nella tesi di Davide Molino per il corretto instradamento dei segnali e per introdurre le nuove memorie utilizzate nel circuito.

Module CIRCUITO

```
module CIRCUITO (clk, rst, nuovi_c, nuovi_p, c_in, p_in_1, p_in_2, p_in_3, p_in_4, p_in_5,
s_in_1, s_in_2, s_in_3, s_in_4, s_in_5, data_out);

input wire clk, rst, nuovi_c, nuovi_p;

input logic [7:0] c_in;

input logic [63:0] p_in_1;

input logic [63:0] p_in_2;

input logic [63:0] p_in_3;

input logic [63:0] p_in_4;

input logic [63:0] p_in_5;

input logic [63:0] s_in_1;

input logic [63:0] s_in_2;

input logic [63:0] s_in_3;

input logic [63:0] s_in_4;

input logic [63:0] s_in_5;

output logic [12:0] data_out;

wire start_count, incrementa, rst_inc, timeout, done_calcola;

wire [8:0] count, threshold;

wire [7:0] inc;

wire ram_p_en, w_p_en, w_c_en, ram_c_en;

wire [6:0] sel_p;

wire [8:0] indirizzo;
```

```
wire [7:0] c_out_ram;

wire p_out_ram;

wire s_out_ram;

wire rst_accumulatore, enable;

wire [8:0] data_in_acc;

wire [12:0] data_out_acc;

FSM fsm (clk, rst, nuovi_c, nuovi_p, count, timeout, done_calcola, inc, p_out_ram,
c_out_ram, s_out_ram, data_out_acc, start_count, threshold, incrementa, rst_inc,
ram_p_en, w_p_en, sel_p, indirizzo, ram_c_en, w_c_en, enable, rst_accumulatore,
data_in_acc, data_out);

CONTATORE contatore (clk, start_count, threshold, count, timeout);

INCREMENTATORE incrementatore (incrementa, rst_inc, inc, done_calcola);

RAM_P ram_p (clk, p_in_1, p_in_2, p_in_3, p_in_4, p_in_5, sel_p, indirizzo, ram_p_en,
w_p_en, p_out_ram);

RAM_S ram_s (clk, s_in_1, s_in_2, s_in_3, s_in_4, s_in_5, sel_p, indirizzo, ram_p_en, w_p_en,
s_out_ram);

RAM_C ram_c (clk, c_in, indirizzo, ram_c_en, w_c_en, c_out_ram);

ARITMETICA aritmetica (clk, rst_accumulatore, enable, data_in_acc, data_out_acc);

endmodule
```

Soluzione circuitale finale

Qui di seguito in figura viene riportato l'intero schema circuitale comprensivo di tutti i moduli precedentemente illustrati.

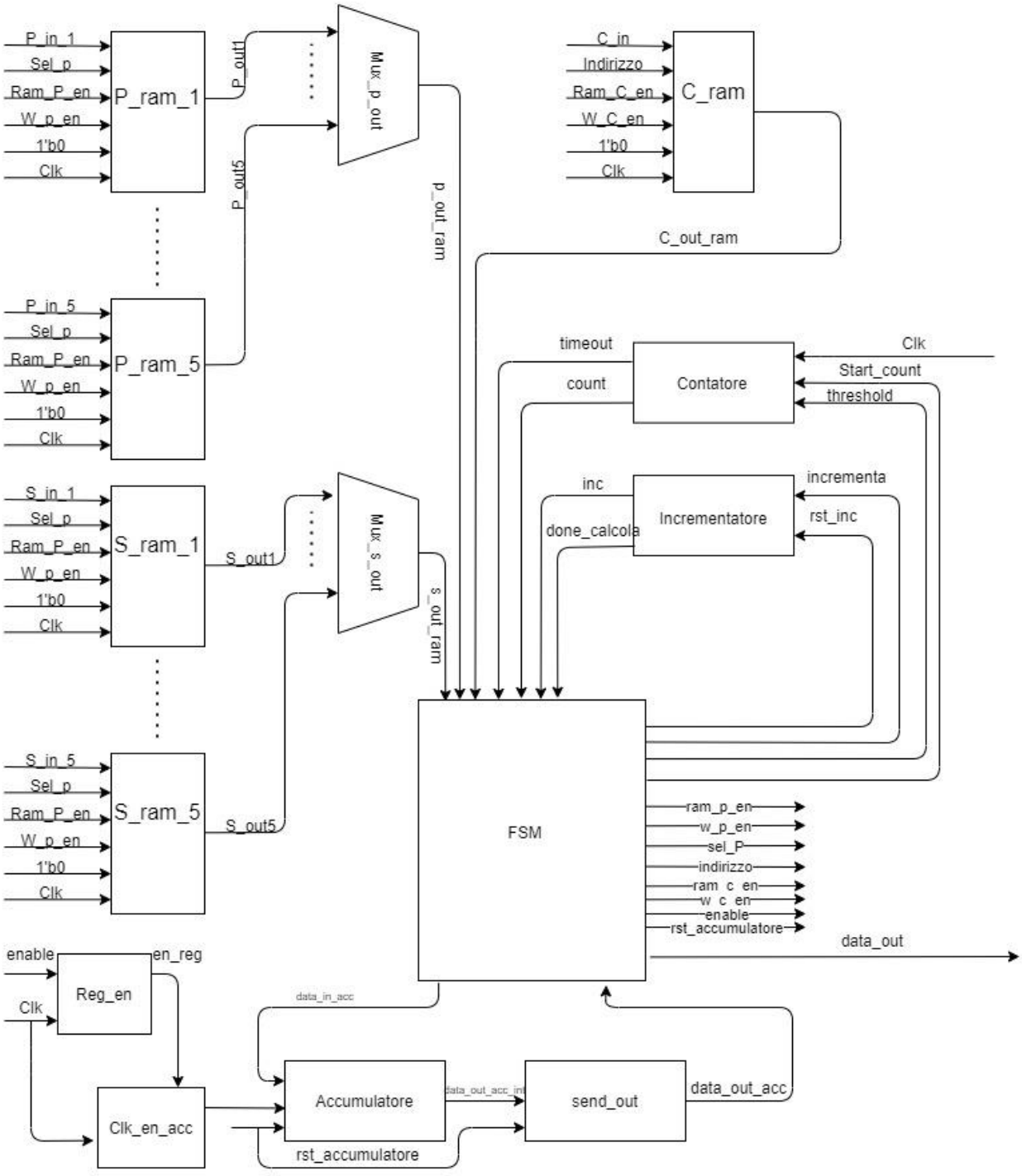


Figura 31: Schema generale

Generazione dei dati in ingresso

Per realizzare un testbench che effettui una simulazione funzionale verosimile, è necessario fornire al modulo top level dei segnali in ingresso altrettanto verosimili. I segnali che devono essere forniti al modulo, oltre ai segnali logici clk, rst, nuovi_c, nuovi_p, sono i segnali c_in [7:0], p_in_1[63:0], p_in_2[63:0], p_in_3[63:0], p_in_4[63:0], p_in_5[63:0].

Per quanto riguarda i campioni in ingresso, ci saranno 320 parole da 8bit ciascuna. Per i pesi, invece, saranno necessari 5 vettori, ciascuno da 64bit per descrivere un'intera riga della RAM_P costituita da 320 elementi.

Per fare ciò si è utilizzato uno script in MATLAB che ha automatizzato la generazione dei vettori richiesti rispettandone le caratteristiche.

MATLAB Script

```
C = randi([0,255], 320, 1);

P_decimal = randi(100, 1, 320);

P = (P_decimal<20);
data_out = P * C;
```

La prima riga del codice rappresenta la generazione di un vettore c_in [7:0]. Il comando randi genera una matrice con una colonna e 320 righe, ovvero un vettore colonna. Gli elementi del vettore saranno dei numeri interi decimali casuali compresi tra 0 e 255, che rappresenta proprio il massimo numero esprimibile in binario con 8bit.

Per quanto riguarda le righe dei pesi è stata considerata una condizione di reale applicazione, nella quale si è calcolato solo una percentuale del 20% composta da pesi pari ad 1, tutti gli altri saranno elementi nulli. Si è utilizzato quindi lo script per generare una riga della RAM_P rispettando queste caratteristiche: P_decimal sarà un vettore con 320 colonne composta da numeri interi decimali casuali minori o uguali a 100. P, invece, è la conversione di P_decimal in binario, nella quale al posto di ogni elemento viene sostituito il valore falso, ovvero 0, nel caso in cui l'elemento sia maggiore di 20, oppure 1, nel caso di condizione verificata, cioè il valore dell'elemento minore di 20. In questo modo si è generato un vettore riga nel quale solo il 20% degli elementi sono 1, tutti gli altri 0.

L’ultima riga del codice, infine, è stata utilizzata per effettuare una verifica sui valori in uscita: data_out effettua la moltiplicazione tra i due vettori generati in precedenza, P e C. In questo modo, confrontando i risultati ottenuti dallo script in MATLAB con quelli mostrati nella simulazione, si è potuto confermare il corretto funzionamento del modulo.

Testbench

Il testbench strutturato, già introdotto nella tesi di Davide Molino, a cui sono stati opportunamente aggiunti i nuovi segnali, è stato utilizzato sia per le verifiche funzionali del circuito che per le operazioni di sintesi. Nel testbench si è istanziato il modulo top level CIRCUITO e si è scelto di pilotare i suoi pin di ingresso per eseguire le operazioni mostrate nel diagramma in figura. Si parte con la scrittura della RAM_P con il set di pesi, poi segue una scrittura di un primo set di campioni. A questo punto sarà effettuata la prima operazione di calcolo tra le due memorie appena scritte.

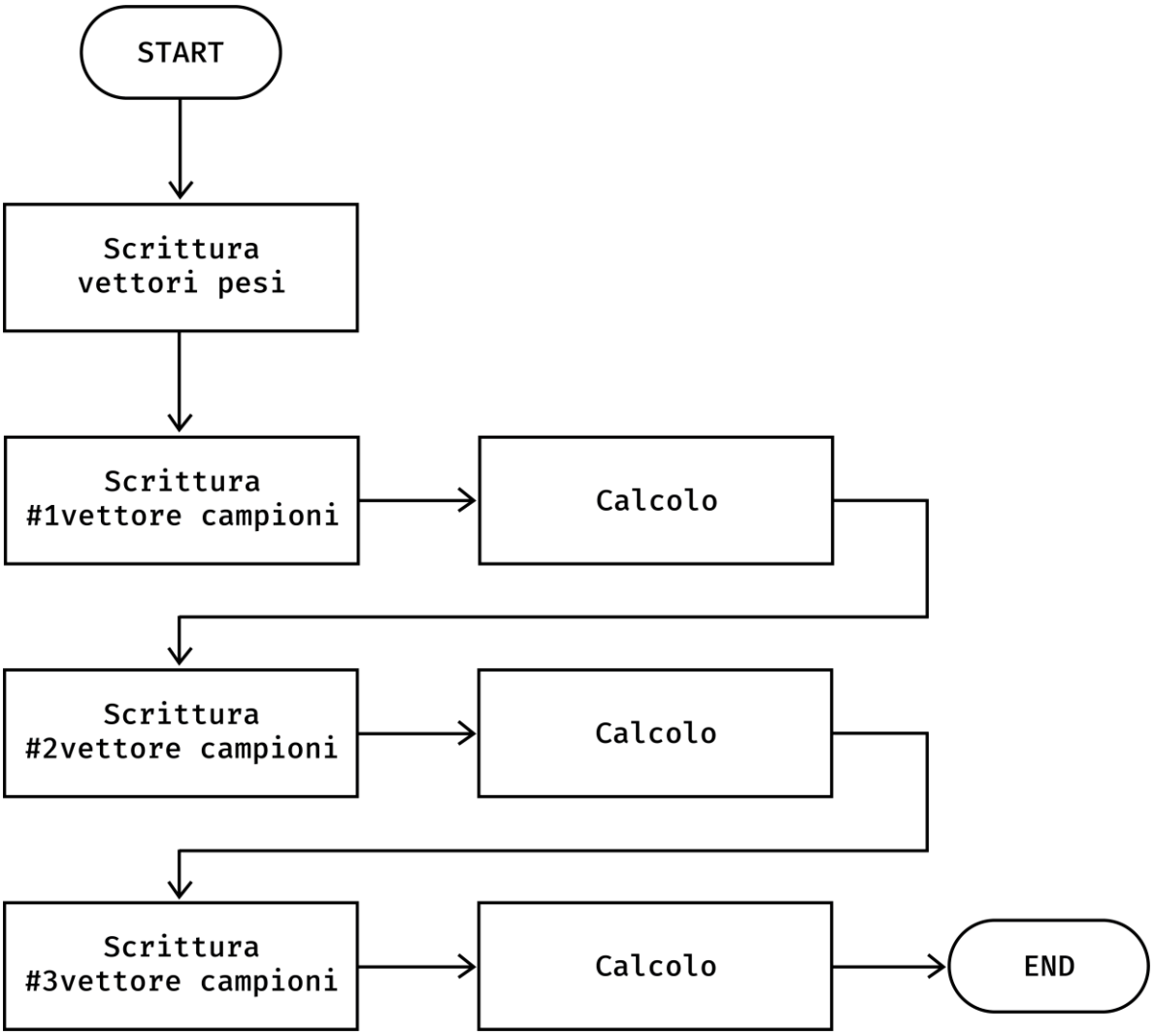


Figura 32: Descrizione testbench

Terminata questa fase, si procederà con la scrittura ed il calcolo di altri due set di campioni differenti tra loro. Si è scelto di effettuare tre operazioni di calcolo per poter stimare meglio il consumo. Il tool di sintesi effettua una media nel periodo, quindi includere un numero maggiore di operazioni porta ad una stima del risultato più accurata.

Segue una parte del codice System Verilog che descrive il testbench.

TB_CIRCUITO

```
module TB_CIRCUITO ();

reg clk, rst, nuovi_c, nuovi_p;

reg [7:0] c_in;
```

```
reg [63:0] p_in_1;

reg [63:0] p_in_2;

reg [63:0] p_in_3;

reg [63:0] p_in_4;

reg [63:0] p_in_5;

reg [63:0] s_in_1;

reg [63:0] s_in_2;

reg [63:0] s_in_3;

reg [63:0] s_in_4;

reg [63:0] s_in_5;

wire [12:0] data_out;

CIRCUITO circuito (clk, rst, nuovi_c, nuovi_p, c_in, p_in_1, p_in_2, p_in_3, p_in_4, p_in_5,
s_in_1, s_in_2, s_in_3, s_in_4, s_in_5, data_out);

always

begin

        clk = 1; #50;

        clk = 0; #50;

end

initial begin

rst = 1;

nuovi_c = 0;

nuovi_p = 0;

c_in = 8'd0;

p_in_1 = 64'd0;
```

```
p_in_2 = 64'd0;

p_in_3 = 64'd0;

p_in_4 = 64'd0;

p_in_5 = 64'd0;

s_in_1 = 64'd0;

s_in_2 = 64'd0;

s_in_3 = 64'd0;

s_in_4 = 64'd0;

s_in_5 = 64'd0;

#550;

rst = 0;

nuovi_c = 0;

nuovi_p = 1;

#170;

//1

p_in_1 =
64'b1000000001001000010000000000000000000000010000001000000000001000;

p_in_2 =
64'b0010000000000000001010000110100100000010100010100001000000010000;

p_in_3 =
64'b1100010100100000000001000000000000000000000000010001101110001000;

p_in_4 =
64'b0000000011000000000000000100100000000000000000000010000001100011;

p_in_5 =
64'b0000000100001000000000001000100010000010010001000010000000010000;
```

s_in_1 =
64'b0000000001001000010000000000000000000001000001000000000001000;

s_in_2 =
64'b0010000000000000010100001001001000001010001010000100000010000;

s_in_3 =
64'b1100010100100000000001000000000000000000000001000000111000100 0;

s_in_4 =
64'b0000000011000000000000000000000000000000000000010000011000 11;

s_in_5 =
64'b0000000000001000000000001000100010000000000001000010000000010000;
#100;

//2

...

//127

p_in_1 =
64'b0100000100000000000000000000000011000001000010010011000000000000;

p_in_2 =
64'b0010000100000001100000000100100000000000001001011100000000000010;

p_in_3 =
64'b0000000000000010000010010100000001101110001010100000000001000001;

p_in_4 =
64'b0100100100000111000000000000110000000000110000000000000001000000;

p_in_5 =
64'b0001101000000000000000000010000000000000000010000000100000000010;

s_in_1 =
64'b0100000100000000000000000000000011000001000010010011000000000000;

s_in_2 =
64'b0010000100000000000000000000000000000000000000000000000000000010;

```
s_in_3 =
64'b0000000000000000000000000000000000000000000010101000000000001000001;

s_in_4 =
64'b0100100000000000000000000000000000000000011000000000000001000000;

s_in_5 =
64'b0000000000000000000000000000000000000000000000000000000000000000;

nuovi_c = 1;

#100;

nuovi_p = 0;

p_in_1 = 64'd0;

p_in_2 = 64'd0;

p_in_3 = 64'd0;

p_in_4 = 64'd0;

p_in_5 = 64'd0;

s_in_1 = 64'd0;

s_in_2 = 64'd0;

s_in_3 = 64'd0;

s_in_4 = 64'd0;

s_in_5 = 64'd0;

#100;

//CAMPIONI 1

c_in = 8'd24; #100;

c_in = 8'd3; #100;

...

c_in = 8'd106; #100;
```

```
c_in = 8'd52; #100;

c_in = 8'd0;

nuovi_c = 0;

#4166400;

nuovi_c = 1;

#200;

//CAMPIONI 2

c_in = 8'd71; #100; c_in = 8'd12; #100;

...

c_in = 8'd78; #100; c_in = 8'd180; #100;

c_in = 8'd0; nuovi_c = 0;

#4166400;

nuovi_c = 1;

#200;

//CAMPIONI 3

c_in = 8'd28; #100;

 c_in = 8'd37; #100;

...

c_in = 8'd82; #100;

 c_in = 8'd209; #100;

c_in = 8'd0;

 nuovi_c = 0;

end

endmodule
```

Simulazione su ModelSim

La simulazione, come visto dal testbench, inizia con una fase di inizializzazione di tutti i segnali di input del modulo circuito. Dopo 600 ns inizia la fase di scrittura della memoria RAM_P e RAM_S, come indicato dallo stato della FSM.

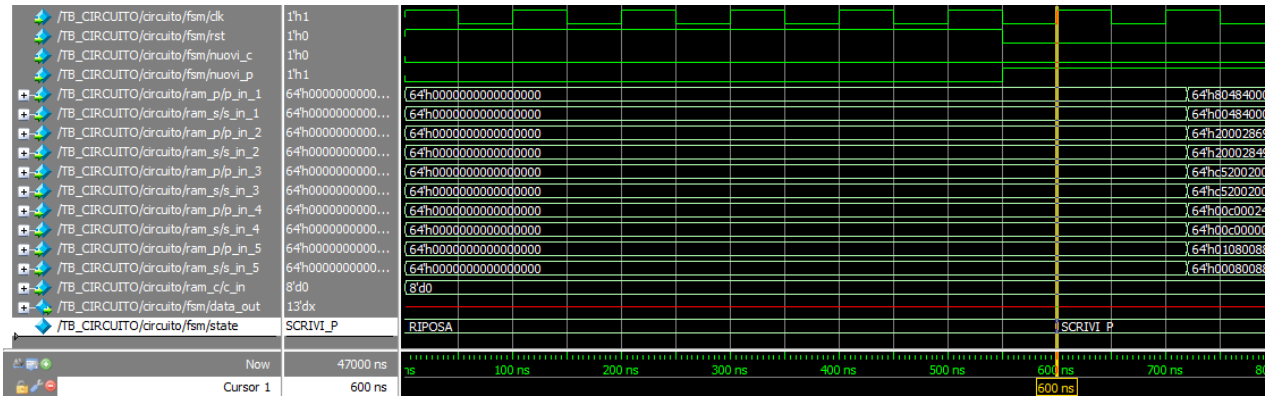


Figura 33: Inizio fase scrittura pesi

All'istante 13500 ns la scrittura dei vettori dei pesi e dei segni sarà terminata e si potrà passare alla scrittura della RAM_C. Tale operazione durerà per 320 periodi di clock che corrisponde ad un tempo di 32000 ns.

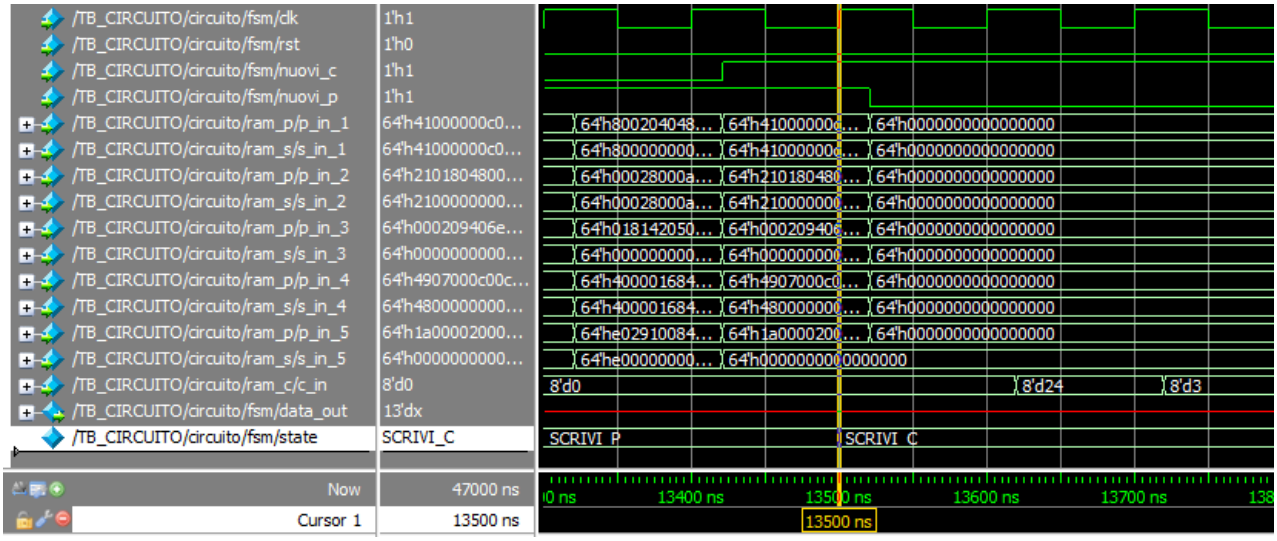


Figura 34: Inizio fase scrittura campioni

All'istante 45600 ns [fig. MS3] sarà ultimata la scrittura della memoria dei campioni e si passerà per lo stato transitorio di SCRITTE_MEMORIE, che nel periodo successivo porta all'inizio delle operazioni di calcolo. Si può notare che è stato necessario un periodo di clock in più rispetto a quanto previsto per la scrittura della RAM_C: infatti il tempo trascorso è 32100 ns, contro i 32000 ns attesi. Ciò è dovuto al fatto che è necessario un periodo di clock all'inizio

delle operazioni per permettere la commutazione dei segnali interni in modo tale da configurare tutti i moduli nella giusta modalità operativa. Evitando questa operazione di ritardo si rischierebbe di scrivere o leggere un valore errato.

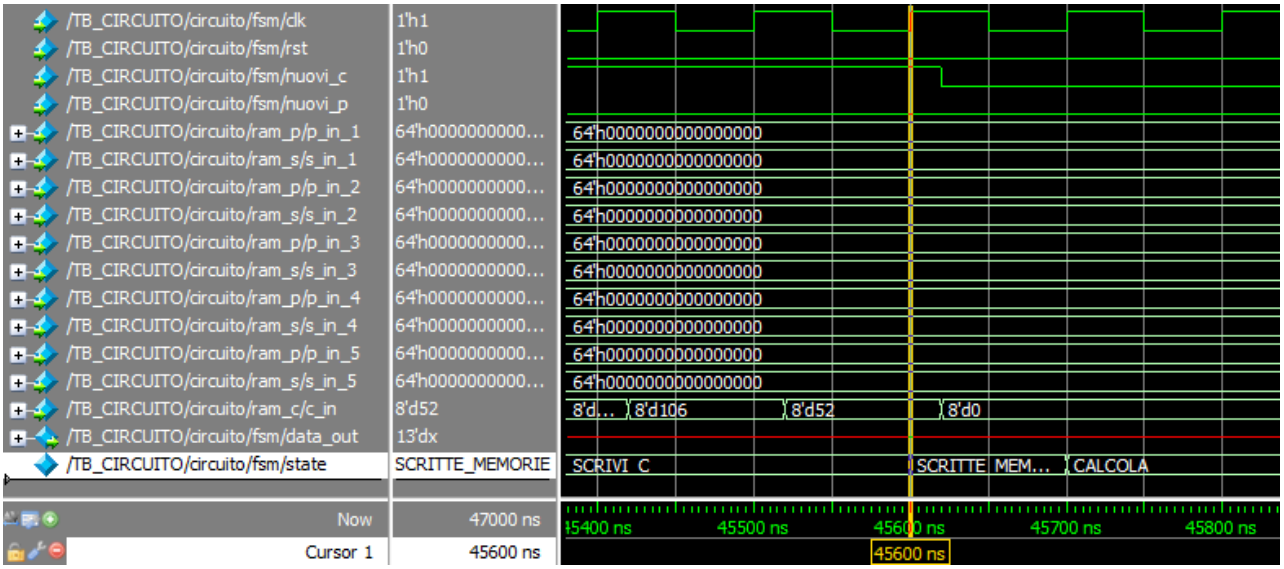


Figura 35: Fine fase scrittura

A questo punto si entra nello stato CALCOLA ed inizierà, all'istante 45700 ns, il prodotto tra la prima riga della RAM_P e la colonna di campioni. All'istante 77800 ns [fig. MS4] sarà terminato il calcolo del primo dato in uscita, corrispondente proprio alla prima riga di pesi. Si finirà nello stato transitorio FINE_GIRO. Da questo momento, fino alle fine delle operazioni con la seconda riga della RAM_P, quindi per 320 periodi di clock, il dato calcolato resterà disponibile sul pin di uscita data_out [12:0]. Nel periodo di clock successivo riprenderà il loop di somme di prodotti fino all'istante 4180000ns, nel quale tutte le righe dei pesi saranno stati moltiplicate per i campioni.

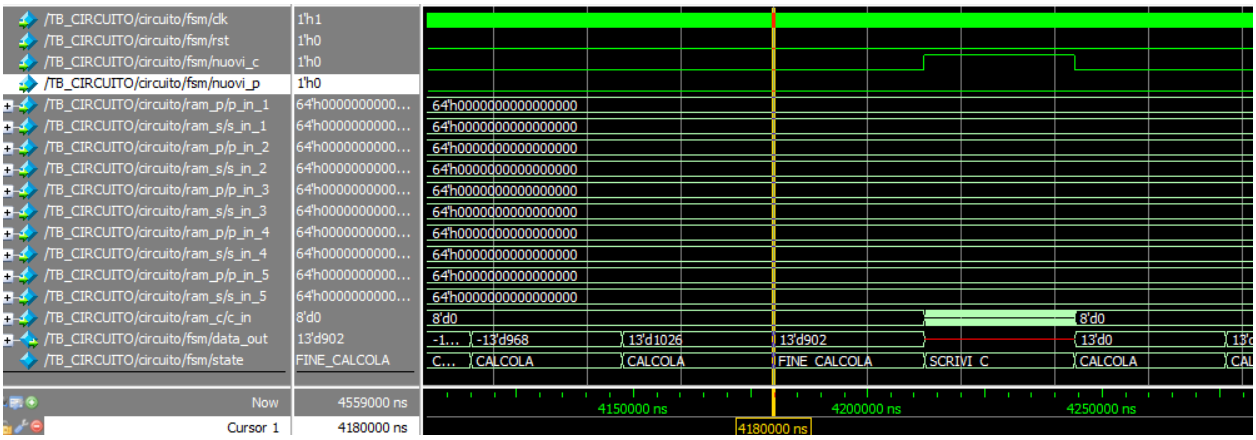


Figura 36: Fine calcolo prima word

Il segnale di done_calcola avvertirà la macchina a stati, la quale deciderà che FINE_CALCOLA sarà lo stato successivo. Finite le operazioni di calcolo per un set di campioni si era deciso di voler scrivere nuovamente la RAM_C ed effettuare nuovi calcoli tra la RAM_P e i nuovi dati della RAM_C: ciò è proprio quello che succede nei periodi successivi, si procede ad una nuova scrittura della memoria dei campioni e si riprenderanno le operazioni di calcolo. Il sistema andrà avanti in questo modo, fornendo dei risultati diversi, essendo cambiato il set dei campioni, fino alla fine delle moltiplicazioni con tutte le righe della RAM_P. Poi nuovamente un terzo set di campioni verrà scritto e si eseguiranno le operazioni. Dopodiché, terminate anche queste ultime operazioni, si entrerà nello stato di RIPOSA ed il test di simulazione si conclude.

Sintesi e Power Analysis del sistema

Si vogliono illustrare i vincoli scelti per effettuare la sintesi del sistema in ambiente STMicroelectronics; dopodiché si procede con una Power Analysis tenendo conto del testbench, quindi di una switching activity del circuito in un caso reale.

Vincoli di sintesi

Il processo di sintesi richiede, in primo luogo, di specificare una frequenza di funzionamento del clock di sistema fMAX, in maniera tale che il tool possa sintetizzare l'architettura rispettando la nota relazione tra periodo del clock, tempo di setup e percorso critico del circuito.

$$T_{CLK} \geq t_{REG} + t_{RC,\,max} + t_{SU} = 1/f_{MAX}$$

Come introdotto nel terzo capitolo si è scelto di realizzare le operazioni di scrittura della RAM_C e le operazioni di calcolo, moltiplicazione tra un set di pesi memorizzati nella RAM_P e i campioni con segno da RAM_S, in 5ms. Da ciò si era ricavata una specifica sulla frequenza di clock pari a: $f_{CLK} \simeq 10\ MHz$. Nel processo di sintesi si è scelto di utilizzare questa f_{CLK} come frequenza massima di funzionamento del clock di sistema f_{MAX}. Si è posta dunque $f_{MAX} = 10\ MHz$. Il tool di sintesi farà in modo di soddisfare la relazione imposta dal tempo di setup e la relazione imposta dal tempo di hold, con questa frequenza scelta.

Il risultato di maggiore interesse è il valore della potenza statica e dinamica consumata dal sistema. Per questo motivo si è scelto un corner operativo di caso peggiore, dal punto di vista della potenza statica per evitare di effettuare una sottostima.

Nel processo di sintesi, la tensione di alimentazione è stata ridotta del 10% rispetto al valore nominale, che rappresenta un peggioramento in termini di prestazioni di timing, mentre la

temperatura è stata posta ad un valore più alto del nominale, con conseguente aumento del consumo di potenza. Quindi, la tensione di alimentazione ha un valore di 1.08 V, contro gli 1.2 V nominali delle celle di libreria utilizzate nella sintesi: ciò causerà una minore velocità del circuito dato che le celle di libreria sono progettate per funzionare al valore nominale di alimentazione. Per quanto riguarda la temperatura il valore scelto è stato 125 °C, che rappresenta la peggiore condizione di funzionamento per il consumo di potenza statica. Imponendo queste condizioni, si è quindi in grado di verificare il corretto funzionamento del sistema nelle condizioni più sfavorevoli, ottenendo così una maggiore affidabilità delle stime finali.

La tecnologia delle celle utilizzate è stata la C090D a 90 nm e sono state scelte le celle di libreria STMicroelectronics COREL_H. Questa scelta è stata dovuta al fatto che i transistor utilizzati in queste celle sono a lunghezza di canale non minima. L'utilizzo di transistori con lunghezza di canale minima causa un aumento della corrente di leakage, essendo i transistor a canale più corto, che si traduce in un aumento della potenza statica.

La potenza statica è infatti determinata dalla corrente sottosoglia dei transistori che cresce in modo esponenziale al calare del valore della tensione di soglia.

$$\begin{aligned} I_{DS} &= \beta \left(\frac{k_B T}{q}\right)^2 \exp\left(\frac{q(V_{GS}-V_T)}{n k_B T}\right)\left[1-\exp\left(-\frac{qV_{DS}}{k_B T}\right)\right] \\ &\simeq \beta \left(\frac{k_B T}{q}\right)^2 \exp\left(-\frac{qV_T}{n k_B T}\right) \end{aligned}$$

dove l'espressione da considerare è quella nella seconda riga valida per Vgs= 0 e Vds maggiore di 4-5 volte la tensione termica (Kb T(q)).

La tensione di soglia ha l'espressione seguente dove e il parametro è inversamente proporzionale alla lunghezza di canale.

$$V_T = \Phi_{MS} + 2\phi_F + \gamma\sqrt{2\phi_F - V_{BS}} - \eta V_{DS}$$

L'obiettivo finale è una potenza statica ridotta per cui è preferibile usare delle celle di libreria a lunghezza di canale maggiore, il che porterebbe un peggioramento in termini di velocità,

accettabile considerando uno Slack di circa 63 ns, ma un miglioramento in termini di potenza statica.

Infine, l'ultimo vincolo imposto nel tool di sintesi è stato un vincolo d'area: il circuito sarà sintetizzato in un riquadro 700 x 1000µm.

Risultati del processo di sintesi

Il blocco è stato sintetizzato con un totale di 895 celle di libreria, di cui 778 combinatorie, 50 sequenziali e 67 buffer, mentre nel lavoro di tesi di Davide Molino erano risultate 837 celle di libreria, 588 combinatorie, 54 sequenziali e 195 buffer

L'area del blocco stimata per le celle di libreria è di 469640.09 μm^2, circa il doppio rispetto alla soluzione a due livelli, che era 243341.19 μm^2, circa 500 µm di lato, in cui i moduli di memoria occupano la maggior parte dell'area del dispositivo. In figura è mostrato il place and route del circuito. Sono presenti piste di collegamento per una lunghezza totale stimata di 536419.75 µm anch'essi raddoppiati rispetto ai 260377.84 µm precedenti.

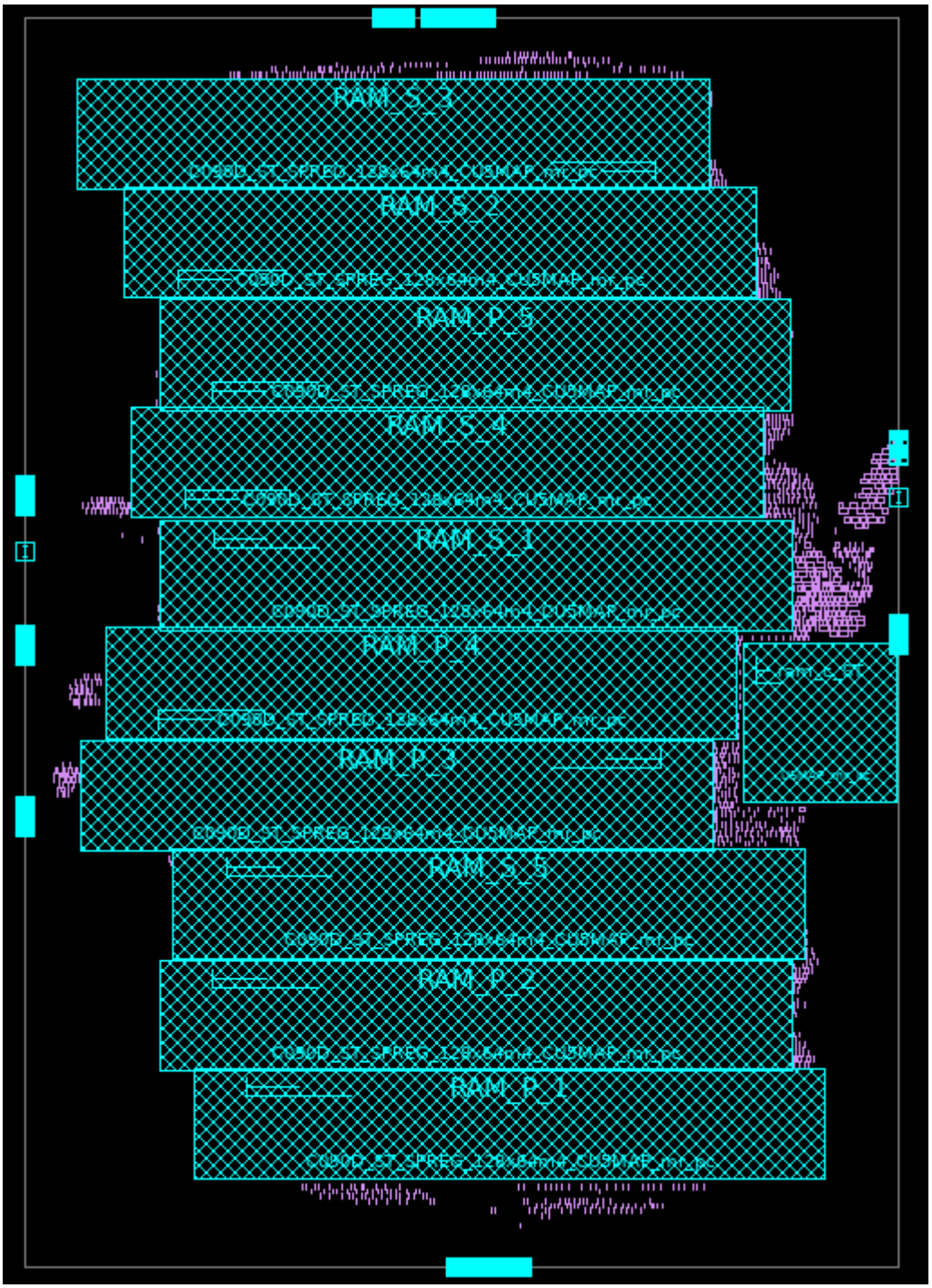


Figura 37: Schema circuito 3 livelli sintetizzato

Il percorso critico del circuito è localizzato nella FSM, dal momento che si trova a dover configurare il dispositivo per le operazioni di scrittura della RAM_P. In questo percorso il tempo di propagazione dei dati è di 3.76 ns, come previsto maggiore dei 2.23 ns della soluzione illustrata nel precedente lavoro di tesi, che corrisponde a una frequenza massima di clock pari a 265.95 MHz, nettamente superiore alla specifica di progetto. In questo percorso critico, inoltre, si ha uno Slack (definito come la differenza tra il ritardo massimo e il periodo di clock imposto) pari a 63.18 ns, minore dei 66.09 ns ma rispetta ampiamente i vincoli. Essendoci un

valore positivo di Slack, significa che, appunto, i segnali stanno arrivando con molto anticipo, rispetto al vincolo sul periodo di clock, ai nodi, ovvero il dispositivo potrebbe adattarsi ad un clock molto più veloce, appunto di 268.95 MHz.

Il tool di sintesi effettua in questa fase anche una prima stima di potenza, non basata però sull'attività del testbench. La switching activity, di conseguenza non sarebbe considerata con la giusta attività che avrà nel funzionamento reale, ma attraverso dei vettori di input casuali e imponendo una switching activity dei nodi interni pari al 10%. In questa fase, è stata pertanto stimata una potenza totale di 1.0928 mW, di cui 1.0539 mW potenza dinamica, una cui stima più accurata viene fornita di seguito, la potenza totale stimata nel lavoro del collega era di 589.4 µW. Si ricorda il modello usato dal programma di power analysis per stimare il consumo.

Il coefficiente α rappresenta la switching activity del nodo, cioè il numero di commutazioni nel periodo di clock. Una stima esatta di questo parametro può essere fatta solo dopo una simulazione significativa del reale funzionamento del circuito.

$$P_{Static} = \sum_{cells} P_{static,i}$$

$$P_{dyn} = \frac{1}{T_{CK}} \left[\sum_{cells} E\mathrm{int}_i \bullet \alpha_i + \sum_{nets} V_{DD}^2 \bullet \alpha_{0->1,i} \bullet Cext_i \right]$$

Power Analysis

Per ottenere una stima della potenza più precisa è stata eseguita una Power Analysis con il tool Prime Time messo a disposizione da STMicroelectronics. Il tool richiede di isolare un intervallo di tempo del testbench del quale si vuole tener conto per poter effettuare una media sulla potenza dissipata. In questo modo, il tool di stima della potenza può tenere conto della switching activity di ogni nodo del circuito indotta da un reale funzionamento del sistema, che viene descritto appunto dal testbench. Facendo riferimento alla struttura del testbench si è considerato

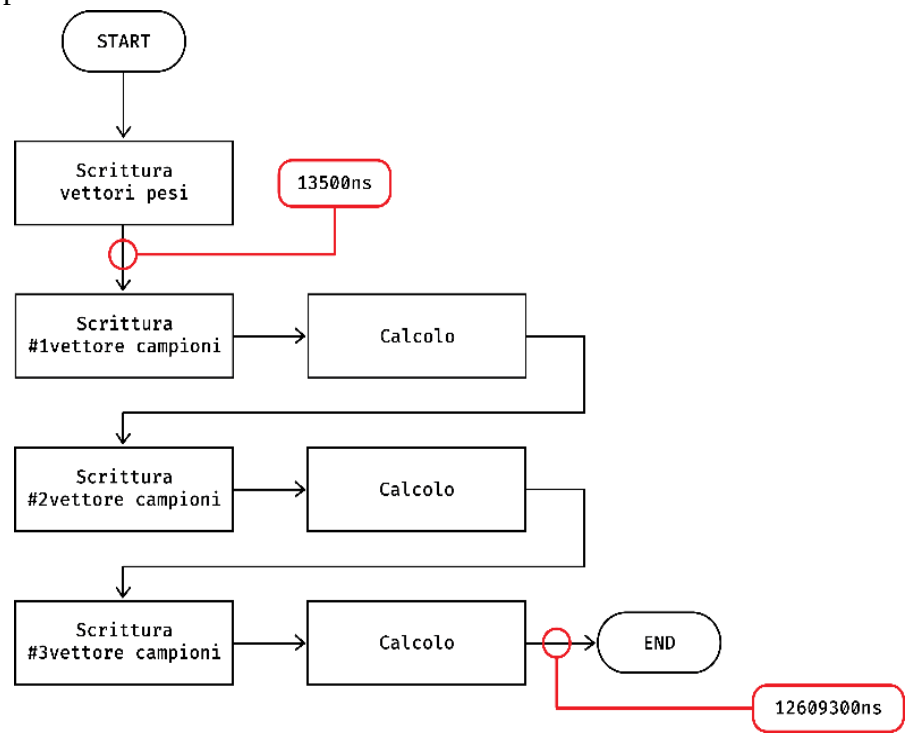


Figura 38: Istanti di sintesi

come istante iniziale l'istante in cui inizia la scrittura della RAM_C. Si è dunque esclusa la scrittura della memoria dei pesi.

Il motivo di questa scelta deriva dal fatto che, nell'ottica di confronto con la tecnologia di calcolo analogico, la fase di scrittura sarà esclusa dal computo energetico.

Successivamente, il testbench procederà con le operazioni di calcolo e scrittura, come spiegato in precedenza, fino all'istante finale, in cui saranno terminati i prodotti tra i di pesi e il terzo vettore dei campioni. In questo modo il tool salverà la switching activity simulata nel funzionamento imposto dal testbench. Per questo motivo, rispetto alla precedente stima della potenza effettuata dopo il place and route, ci si aspetta un valore inferiore di potenza dinamica, in quanto molti nodi del circuito, in quanto facenti parte di sottosistemi spesso disattivati, presenteranno una switching activity inferiore. Si è infatti passati dal precedente valore di 1.0539 mW ad un valore di 11 µW (un fattore 100 di riduzione) del contributo di switching

della potenza dinamica (quello cioè associato alla commutazione delle porte di uscita dei vari blocchi) che è dominata dal contributo interno (294 uW). In particolare, le commutazioni interne ai bocchi di memoria, conseguenti alle operazioni di indirizzamento per la lettura e la scrittura determinano il contributo maggiore. La memoria di pesi è composta da righe nelle quali solo il 20% di elementi ha valore diverso da zero, sarà molto alta la probabilità, durante la lettura in sequenza dei pesi, che i nodi interni dei moduli non abbiano transizioni basso-alto o alto-basso, essendoci due o più zeri consecutivi. Ciò significa che la switching activity ha avuto una riduzione rispetto al caso precedente, il che porta ad una conseguente riduzione della potenza dinamica. Questi risultati sono stati possibili grazie alle scelte architetturali finalizzate al risparmio di potenza effettuate in questo progetto.

Nella tabella sottostante sono riportati i valori di potenza stimati dal tool di Power Analysis. I valori sono espressi in µW.

Gerarchia	Potenza dinamica interna	Potenza dinamica di switching	Potenza statica	Potenza totale
Incrementatore	0.00639	0.00119	0.0157	0.0232
Contatore	2.27	0.70	0.0277	2.99
Ram_C	15	0.107	1.44	16.5
Ram_P	137	4.64	18.6	161
Ram_S	137	4.09	18.6	160
Aritmetica	1.47	0.367	0.0425	1.88
Fsm	0.683	1.05	0.0312	1.77
Circuito	294	11	39	344

Figura 39:Tabella consumi

Le simulazioni sono state effettuate direttamente sul dispositivo utilizzante la sospensione delle Ram_P e Ram_S quando non utilizzati.

Confronto dei valori di potenza

Si vogliono mettere a confronto la soluzione precedentemente proposta, a due livelli, con quella finora illustrata, a tre livelli. I valori sono espressi in µW.

Modulo Circuitale	Potenza dinamica Interna	Potenza di switching dinamica	Potenza Statica	Potenza totale
CIRCUITO due livelli	156	9,63	20,2	186
CIRCUITO tre livelli	294	11	39	344

Figura 40: Tabella confronto

Intuitivamente si è considerato un raddoppio di area e di potenza, dalla valutazione dei risultati ottenuti dalle diverse simulazioni l'intuizione è stata confrontata, difatti dai risultati, introdotti nelle tabelle seguenti, si nota come l'area necessaria per il place and route del dispositivo sia circa raddoppiata, merito delle memorie Ram_S, e per lo stesso motivo anche il consumo di potenza, avendo il doppio delle memorie da alimentare contemporaneamente.

Ottimizzazione dell'architettura a tre livelli

Una volta analizzati i valori estratti dalle simulazioni, si è pensato di introdurre un'ottimizzazione scendendo più a basso livello, analizzando il principio su cui sui basa la somma di numeri negativi in complemento a due. Nell'architettura precedentemente illustrata, si è scelto di definire una struttura la quale nel momento in cui un peso avente valore "1", ha segno negativo, quindi anch'esso con valore "1", viene accumulato il valore del campione con segno negativo, il quale viene ottenuto tramite complemento ad uno dello stesso al quale viene sommato "1". Sulla base di quanto detto finora si può considerare quel "1" sommato semplicemente come il carry in dell'accumulatore, andando così ad eliminare un sommatore che, date le premesse, risulterebbe superfluo. Non ci si aspettano grandi miglioramenti in termini di area e consumo dal momento in cui, come già detto precedentemente, gran parte dell'area e del consumo di potenza è da attribuire alle memorie e non ai gate dediti al calcolo effettivo.

Dalla sintesi del dispositivo la situazione ne risulta peggiorata, imponendo un certo tipo di struttura al programma di sintesi, questo non sceglie autonomamente il tipo di architettura ottima per la realizzazione della funzione, ma viene forzato da delle scelte progettuali introdotte dal progettista, che dimostrano avere caratteristiche peggiori.

I valori di potenza estratti dalla sintesi di questa seconda soluzione vengono riportati nella seguente tabella, dove si evidenzia il confronto con la precedente architettura. I valori sono espressi in µW.

Modulo Circuitale	Potenza dinamica Interna	Potenza dinamica di switching	Potenza Statica	Potenza totale
CIRCUITO $_{\text{tre livelli}}$	294	11	39	344
CIRCUITO $_{\text{opt}}$	295	16	39	350

Figura 41: Confronto soluzioni

3. Progetto di un modulo digitale per il calcolo di prodotti matrici-vettori a 4 bit

L'oggetto di questo capitolo è un'architettura, analoga a quanto detto nel precedente capitolo ed a quanto descritto nella tesi di Davide Molino [15], ma che preveda l'ampliamento delle dimensioni del peso esprimendolo a 4 bit, si passa quindi da una struttura a 3 livelli ad una a 2^4 livelli. Per poter effettuare questa modifica architetturale è opportuno esporre le premesse matematiche che interessano il prodotto matrice-vettore è che hanno portato ad effettuare alcune scelte progettuali importanti. Nella stesura del capitolo si è scelto di mantenere le diciture "peso" e "campione" anche se i due operandi non verranno propriamente utilizzati come tali, ma piuttosto come moltiplicando e moltiplicatore, per mantenere una scrittura congruente si è preferito lasciarli invariati.

Il caso preso in esame si riferisce alle reti neurali dove, per l'appunto, il processo di quantizzazione a 4 bit, sta prendendo largo spazio diventando quasi uno standard. Volendo possiamo considerare il calcolo dei prodotti con accumulo relativo alla singola riga della matrice dei pesi, come un singolo neurone al quale vengono apposti come ingressi gli elementi della singola word line della matrice e del vettore dei campioni, restituendone in uscita il risultato.

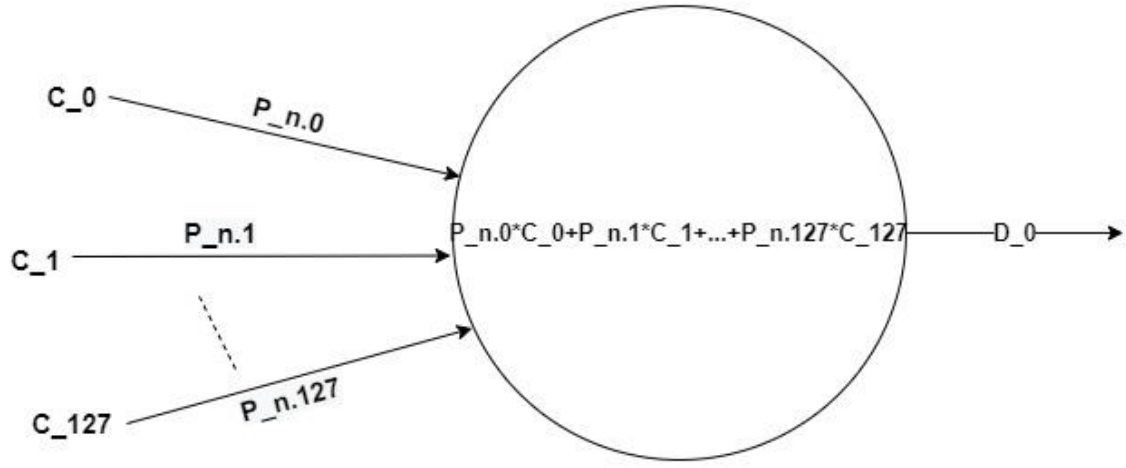


Figura 42: Singolo neurone

Seguendo il ragionamento ad ogni shift della word line corrisponde un neurone, quindi avremo una rete come quella illustrata nella figura seguente.

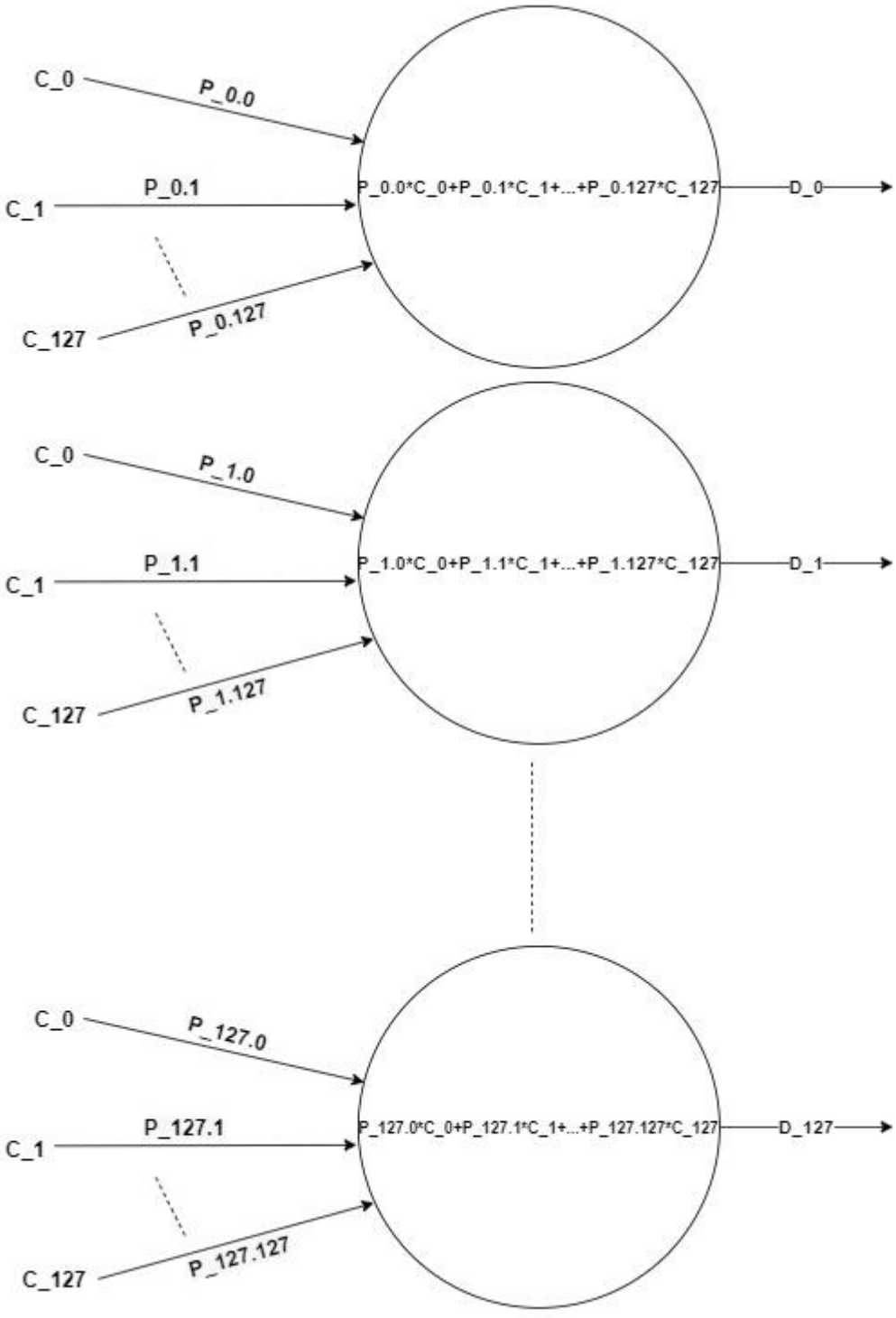


Figura 43: Rete neurale

Il prodotto tra una matrice ed un vettore prevede che il numero di colonne della matrice siano identiche al numero di righe del vettore, tale punto implica che, se nel caso precedente la matrice dei pesi era composta da 128 righe per 320 colonne, quindi 128x320 bit complessivi, e la relativa matrice dei campioni era composta 320 righe ad 8 bit, ora ritroviamo il numero dei bit della matrice dei pesi quadruplicato, quindi, per ottenere la stessa quantità di pesi, disposti su una matrice questa dovrà essere composta da 320*4 pesi per ognuna delle 128 righe. Una soluzione di questo tipo richiederebbe l'impiego di un numero maggiore di memorie e, dal momento che già nell'architettura precedente si vedeva necessaria l'applicazione di dieci banchi RAM, occupando dimensioni inaccettabili, un riquadro di $1\ cm^2\ c.ca$, a tal punto diventa necessario fissare un limite soprattutto in vista dell'obiettivo finale, il consumo energetico oltre che l'area.

Si è scelto quindi un valore di capacità limite per la matrice dei pesi, questo è stato fissato alla capacità utilizzata nell'architettura a tre livelli, che vedeva:

$$2^7\ word * 2^9\ bit * 2^1\ bit\ peso = 2^{17} bit\ totali$$

(con una capacità maggiore di quella strettamente necessaria per memorizzare 128 * 320 pesi a due bit, essendo stati obbligati a utilizzare 128* 512 pesi a due bit). Fissato questo limite è stata scelta la dimensione di capacità per la matrice dei pesi nell'architettura con pesi a 4 bit:

$$2^2\ bit\ peso * 2^7\ word * 2^7\ pesi = 2^{16} bit\ totali$$

Per cui in numero dei pesi scende da 320 a 128, conseguentemente anche il numero di campioni dovrà essere ridotto da 320 a128, per cui avremo un vettore campioni 128x8.

Nella figura sottostante viene illustrata la dimensione delle matrici e dei vettori in uso.

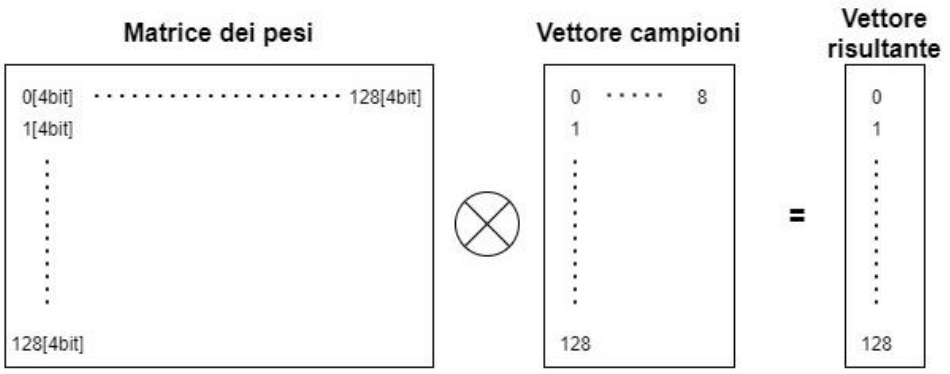


Figura 44: Dimensione strutture matriciali

Per quanto riguarda il vettore risultante le dimensioni dovranno essere necessariamente maggiori rispetto al caso precedente in quanto l'operazione, che vedeva soltanto l'accumulo di campioni ad 8 bit, più bit di segno, adesso si struttura come una vera e propria moltiplicazione a più bit con accumulo, tra un peso a 4 bit ed un campione a 8 bit. Nella figura sottostante viene riportata la struttura della moltiplicazione.

Per contenere il risultato, ottenuto come riportato in figura, sarà necessario che ciascuna parola del vettore D abbia una lunghezza di 19 bit:

$$bit\ per\ parola\ del\ vettore\ D = 4\ bit\ + 8\ bit\ + \log_2 128\ = 19\,\text{bit}$$

Dove 128 sono il numero di operazioni di somma che vengono effettuate, 4 bit del peso 8 bit del campione, sommati in quanto il risultato di un prodotto binario avrà una dimensione data dalla somma delle dimensioni di moltiplicando e moltiplicatore. Resta ancora valido l'assunto per applicazioni di "compressive-sensing" che soltanto il 20% degli elementi della matrice dei

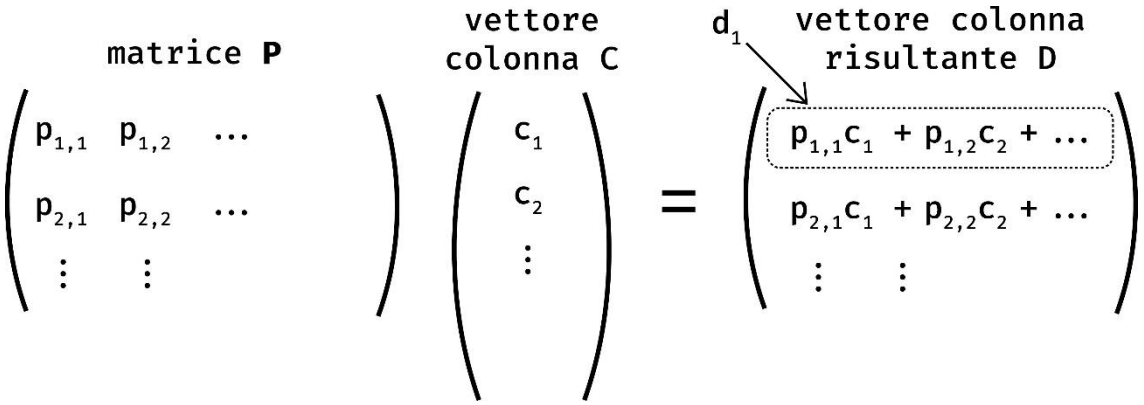


Figura 45: Descrizione operazioni matematiche

pesi abbia un valore diverso da 0, per cui il numero delle operazioni viene ridotto e come tale sarà possibile ridurre le dimensioni delle word del vettore D:

$$bit\ per\ parola\ del\ vettore\ D = 4\ bit\ + 8\ bit\ + \log_2(128 * 0.2)\ = 17\ bit$$

Volendo rispettare il vincolo precedentemente imposto, cioè di effettuare tutte le operazioni in *5ms* sarà necessario rivalutare il numero di operazioni svolte in questo periodo, quindi la scrittura di C e le operazioni di moltiplicazione tra C e P. Avremo quindi 128 cicli necessari per la scrittura di C e successivamente 128 * 128 cicli per il calcolo del prodotto. Questo si traduce in un vincolo sul periodo di clock, che andremo a valutare:

$$(128 + 128 \cdot 128) \cdot T_{CLK} \simeq 5\ ms$$

Ottenendo un clock pari a:

$$T_{CLK} \simeq 300\,ns$$

Quindi una specifica di frequenza di:

$$f_{CLK} \simeq 3\,MHz$$

Quindi si nota un rallentamento rispetto alla precedente architettura dal momento in qui vengono effettuate un numero minore di operazioni nello stesso periodo temporale, se volessimo mantenere la stessa frequenza, 10 MHz, dovremmo effettuare le operazioni in:

$$100\,ns * (128 + 128 * 128) = t$$

$$t \approx 1.5\,ms$$

Confrontando la riduzione del numero di operazioni effettuate, con la riduzione del periodo, il risultato risulta congruente. Si passa, per le operazioni, da $(320 + 128 * 320) = 41280$ a $(128 + 128 * 128) = 16512$, quindi una riduzione di 2.5 volte, mentre per il periodo da $5\,ms$ a $1.5\,ms$ quindi 3.3 volte minore, coerente con quanto detto, per cui la specifica in frequenza è stata mantenuta a 10MHz.

Memorie pesi e campioni

Le modifiche architetturali appena descritte si riflettono sulle memorie che dovranno andare a memorizzare pesi e campioni, subendo delle variazioni in dimensione e disposizione; sono state configurate, come nel caso precedente, da STMicroelectronics, delle memorie SRAM, ma con dei tagli differenti da quelli delle precedenti applicazioni.

Per la memoria C è stata utilizzata una SRAM, denominata RAM_C, con un taglio 128x8m4, 128 word a 8 bit con parallelismo 4.

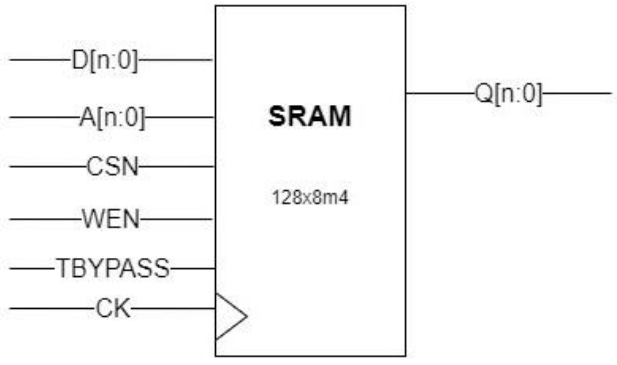


Figura 46: Modulo RAM_C

Sono stati definiti i pin di input e output analogamente a quanto fatto precedentemente, dal momento che a variare sono state solo le dimensioni, mentre i segnali di ingresso e di uscita restano invariati.

Per quanto riguarda la matrice P è stata utilizzata lo stesso tipo di SRAM 128x64m4, dello stesso taglio, denominata anch'essa RAM_P, in questo caso però si vedrà necessaria l'applicazione di 8 banchi in parallelo. Saranno quindi azionati dagli stessi segnali, come ampliamente descritto nel precedente capitolo, a differenza che ora il dato sarà formato da 4 bit e non più da 1 bit, quindi vedremo nei 64 bit di ciascuna word 16 pesi e non più 64, nessuna variazione nel ciclo di scrittura che vedrà arrivare 64 bit, diversamente avverrà in lettura dove i bit dovranno essere distinti e selezionati in gruppi da 4 bit, coerentemente alla posizione con la quale sono stati scritti. Quindi la scrittura avverrà in sincrono sui diversi banchi ma una riga per volta per ogni singolo banco di memoria, indirizzata opportunamente da un contatore che scorrerà, analogamente a quanto descritto nella precedente architettura, da 0 a 127, con un segnale *sel_p* [6:0] che indicherà l'indirizzo della word della memoria sulla quale l'ingresso verrà memorizzato.

È nella fase di lettura che risiede una modifica sostanziale, in particolare nel multiplexer, questo andrà a selezionare il peso da riportare in uscita non solo in base alla word selezionata ma dovrà anche distinguere i 4 bit del singolo peso all'interno dell'array binario memorizzato, in figura sottostante è riportato lo schema delle connessioni tra i banchi e con il multiplexer, con relativi segnali di ingresso e di uscita.

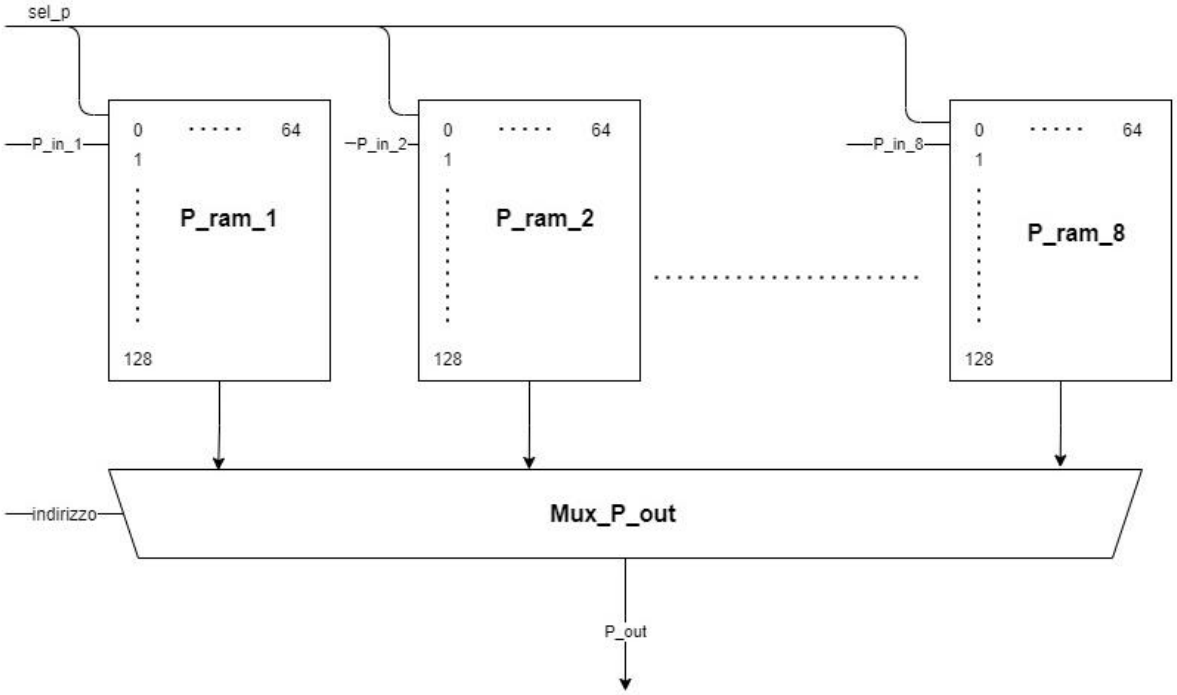


Figura 47: Struttura moduli RAM_P

Per selezionare il singolo peso, una volta attivata la word, viene applicato un secondo contatore, anch'esso da 0 a 127, questo messo in ingresso come segnale *indirizzo* [6:0], sarà il segnale di selezione del multiplexer che andrà a indicare quale sarà il peso a 4bit da riportare in uscita tra i 128 pesi da 4 bit memorizzati nei differenti banchi. Quindi in lettura il segnale *sel_p* [6:0] verrà incrementato solo dopo aver letti tutti i pesi di una riga, variando il segnale *indirizzo* [6:0], il quale andrà a selezionare anche la word della memoria dei campioni che dovrà necessariamente combaciare.

Architettura

L'architettura progettatata mira ad effettuare una moltiplicazione tra due numeri, si inizia descrivendo uno schema di partenza, riportato in figura, dove si possono distinguere le due memorie, RAM_P e RAM_C, descritte in precedenza, la logica di controllo, il moltiplicatore ed il circuito di accumulo dei prodotti.

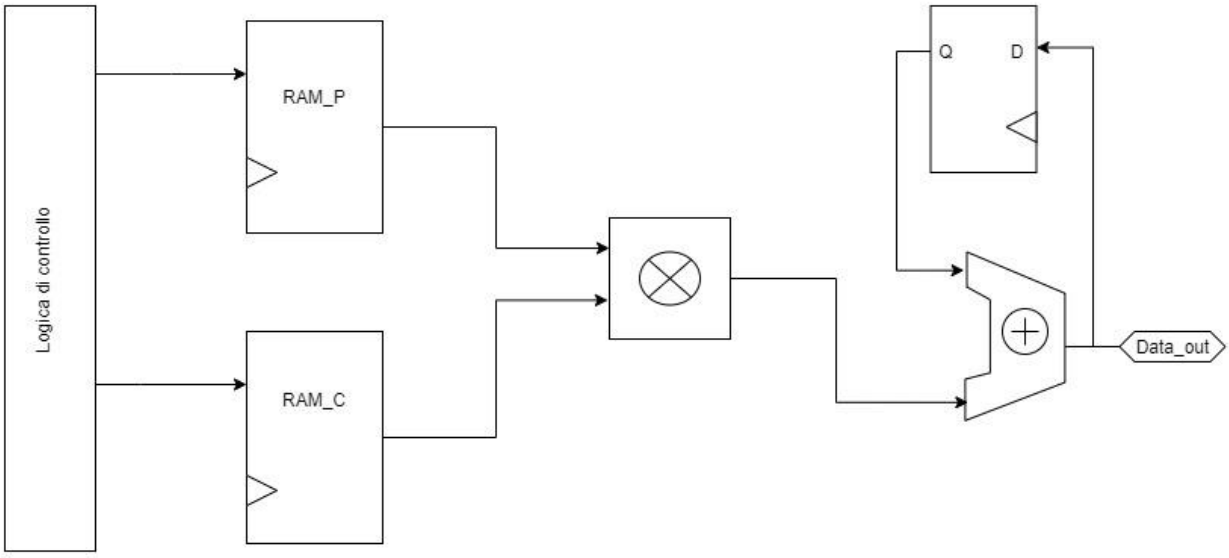


Figura 48: Schema versione non ottimizzata

La logica di controllo di quest'architettura non differisce da quella introdotta nel caso precedente, la sua funzione resta quella di generare gli indirizzi necessari al corretto funzionamento e, successivamente in fase di lettura, a richiamare il corretto indirizzo per selezionare pesi e campioni rispettivamente da RAM_P e RAM_C.

Il moltiplicatore non effettuerà altro che il prodotto tra moltiplicatore e moltiplicando che vengono introdotti direttamente dall'uscita delle memorie, quindi peso e campione selezionati dalla logica di controllo.

Versione ottimizzata

Così come proposto il sistema non presenta nessun tipo di ottimizzazione energetica, le memorie restano sempre attive sia in lettura che in scrittura, il moltiplicatore resta attivo indifferentemente che sia in elaborazione o che sia in stallo. A tal proposito sono state effettuate quindi una serie di scelte progettuali al fine di ridurre l'attività delle varie componenti, nel momento in cui questi non devono necessariamente essere attivi, in modo quindi da ridurre il consumo energetico.

Differentemente dal caso precedente, nel quale l'elaborazione era basata sul singolo bit memorizzato nella matrice dei pesi, il quale andava ad abilitare o meno l'accumulatore, in questo caso i bit in questione sono quattro per cui sarà necessario effettuare una valutazione analoga, ma sul NOR tra i singoli bit del peso; il ragionamento può essere perfettamente iterato anche per quanto riguarda i campioni.

Analizzando la struttura delle matrici risulta evidente che, per un'ottimizzazione energetica, il focus è da porre nella matrice dei pesi, RAM_P, poiché per quanto detto nel precedente capitolo, gli elementi memorizzati in essa sono solo per il 20% diversi da zero, ciò implica che l'attivazione del moltiplicatore, ma anche la lettura del campione memorizzato in RAM_C possa essere condizionata dal valore del peso, per cui solo nel caso in cui il NOR tra i bit del peso risulti avere valore '0', come premesso nella descrizione delle memorie, viene abilitata la lettura del campione corrispondente. Dal momento che il campione verrà fornito in uscita da RAM_C al ciclo di clock successivo all'attivazione, si vede necessario l'inserimento di un registro sul path di *p_out_ram* in modo da moltiplicare correttamente il peso e il campione corrispondenti allo stesso indirizzo.

Quanto detto fin ora non esclude che anche il valore del campione possa essere anch'esso nullo, ma un condizionamento di RAM_P dal valore di RAM_C non risulterebbe una scelta efficace, visto che gran parte dei valori saranno non nulli; in più sarebbe impossibile da introdurre nell'architettura visto il previo collegamento con RAM_P. Il condizionamento dal valore di RAM_C è stato introdotto nel blocco di attivazione del circuito di moltiplicazione ed accumulo, in modo tale che questo si attivi solo nel momento in cui entrambi gli operandi, moltiplicando e moltiplicatore, risultino non nulli.

Da come si può vedere nello schema riportato in figura, il segnale di enable del moltiplicatore è ottenuto dall'AND degli OR, valutati bit a bit, di *p_out_ram* e di *c_out_ram*, quindi il segale di *enable* risulterà essere '1' solo nel caso in cui almeno un bit, per entrambi, peso e campione, sarà diverso da '0', andando così ad attivare il modulo atto al calcolo del prodotto che verrà poi successivamente accumulato; nel caso invece il segnale di *enable* sia '0' non è necessario che venga calcolato alcun prodotto e conseguentemente questo non verrà accumulato, per cui il modulo non verrà attivato.

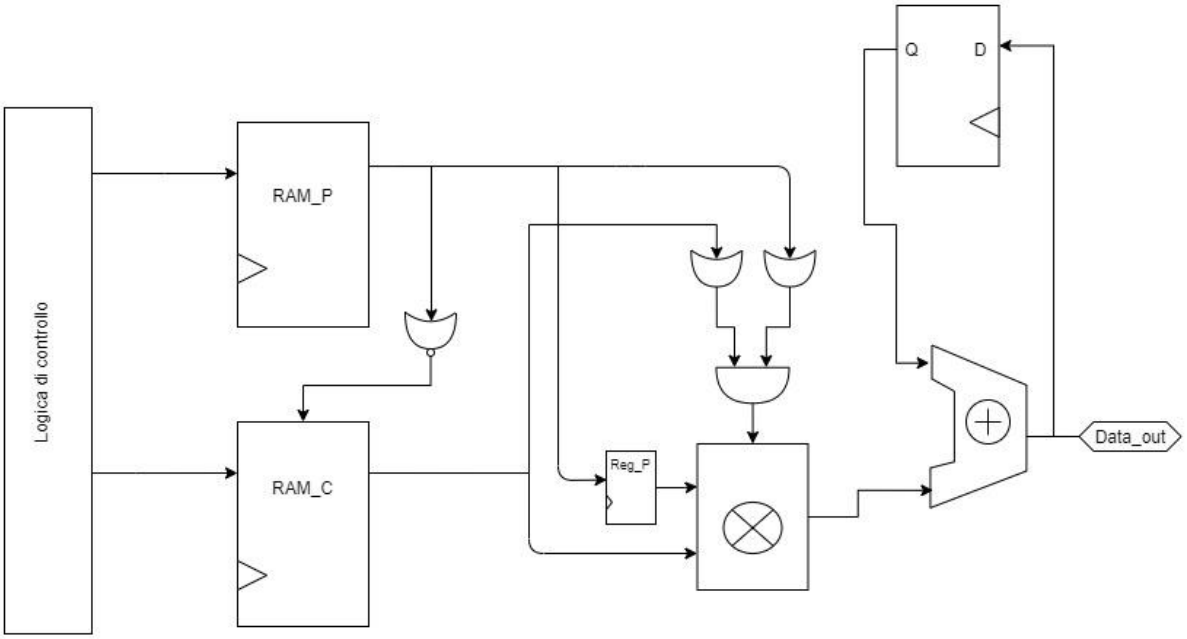


Figura 49: Schema versione ottimizzata

Una soluzione di questo tipo comporta una sostanziale riduzione della potenza necessaria per effettuare il calcolo, dal momento che per almeno l'80%, valore minimo in quanto non siamo in grado di valutare la percentuale di campioni nulli, il circuito di calcolo resterà in stallo e la memoria non verrà attivata.

RTL sistema

Nel precedente capitolo ci si è soffermati sulla struttura dell'architettura digitale che vuole essere l'oggetto principale di questo lavoro di tesi, evolutasi da una struttura, in primo luogo a due livelli e successivamente a tre. Il progetto consiste in un modulo digitale descritto a livello RTL in linguaggio SystemVerilog, ripartito in diversi moduli che verranno illustrati singolarmente nello svolgimento di questo paragrafo, per poi seguire con le simulazioni funzionali, servendosi, anche in questo caso, dell'ambiente HDL ModelSim, e terminare con la sintesi del modulo con tecnologia CMOS a 90 nm per ottenere delle stime di potenza da confrontare con le precedenti soluzioni.

RAM_C

Le modifiche introdotte hanno richiesto l'impiego di un taglio differente per la realizzazione della memoria dei campioni che, pur restando a 8 bit, è stata ridotta in numero di word, da 320 a 128; questo particolare taglio è stato fornito da STMicroelectronics, tramite una relativa descrizione RTL in linguaggio Verilog, in modo da poterla introdurre all'interno del modulo.

Del modulo C090D_ST_SPREG_128x8m4_CU5MAP_mr_pc, utilizzato per la RAM_C e del modulo C090D_ST_SPREG_128x64m4_CU5MAP_mr_pc, utilizzato per la RAM_P non verrà mostrata la descrizione RTL, poiché è stata fornita a solo scopo di simulazione funzionale e stima di potenza.

Il modulo RAM_C si limita ad essere una istanza del modello C090D_ST_SPREG_320x8m4_CU5MAP_mr_pc fornito, del tutto analogo al modulo precedentemente applicato, a tal punto si omette la descrizione del codice SystemVerilog il quale sarebbe ridondante.

RAM_P

Diversamente invece per la RAM_P, dove il taglio delle SRAM utilizzate in precedenza è risultato essere adatto al tipo di applicazione, ma a dover variare è stata invece la disposizione e il numero di banchi utilizzati. Per memorizzare correttamente i pesi sono stati stanziati otto moduli STMicroelectronics C090D_ST_SPREG_128x64m4_CU5MAP_mr_pc necessari a formare le 128 righe per 512 colonne, divise per i 4 bit di ogni singolo peso, 128 pesi per word. Qualche piccola variazione è stata introdotta all'interno del multiplexer MUX_P_OUT dove il segnale di selezione è stato ridotto di due bit, *indirizzo* [6:0], necessario a discriminare 128 valori e non più 320, ordinati in 4 bit; ne segue il codice SystemVerilog del modulo multiplexer in quanto, il modulo RAM_P, differisce da quello utilizzato nella precedente architettura, solo nel numero di memorie instanziate.

```
module MUX_P_OUT (p_out_ram_1, p_out_ram_2, p_out_ram_3, p_out_ram_4, p_out_ram_5,
p_out_ram_6, p_out_ram_7, p_out_ram_8, indirizzo, p_out_ram);

input logic [63:0] p_out_ram_1;

input logic [63:0] p_out_ram_2;

input logic [63:0] p_out_ram_3;

input logic [63:0] p_out_ram_4;

input logic [63:0] p_out_ram_5;

input logic [63:0] p_out_ram_6;

input logic [63:0] p_out_ram_7;

input logic [63:0] p_out_ram_8;
```

```
input logic [6:0] indirizzo;
output logic signed [3:0]p_out_ram;
always_comb begin
case (indirizzo)
        7'd0 : p_out_ram = p_out_ram_1[3:0];
                        ……….
        7'd15 : p_out_ram = p_out_ram_1[63:60];
        7'd16 : p_out_ram = p_out_ram_2[3:0];
                        ……….
        7'd31 : p_out_ram = p_out_ram_2[63:60];
        7'd32 : p_out_ram = p_out_ram_3[3:0];
                ……….
        7'd47 : p_out_ram = p_out_ram_3[63:60];
        7'd48 : p_out_ram = p_out_ram_4[3:0];
                        ……….
        7'd63 : p_out_ram = p_out_ram_4[63:60];
        7'd64 : p_out_ram = p_out_ram_5[3:0];
                        ……….
        7'd79 : p_out_ram = p_out_ram_5[63:60];
        7'd80 : p_out_ram = p_out_ram_6[3:0];
                                ……….
        7'd95 : p_out_ram = p_out_ram_6[63:60];
        7'd96 : p_out_ram = p_out_ram_7[3:0];
                        ……….
```

```
        7'd111 : p_out_ram = p_out_ram_7[63:60];
        7'd112 : p_out_ram = p_out_ram_8[3:0];
                ..........
        7'd127 : p_out_ram = p_out_ram_8[63:60];
        default : p_out_ram = 4'dx;
    endcase
  end
endmodule
```

ARITMETICA

Nel modulo ARITMETICA risiedono tutte le operazioni di calcolo, sia la moltiplicazione che l'accumulo dei prodotti, quindi il modulo di clock gating, che abilita o disabilita il clock del modulo di multiply and accumulate, due registri, REG_EN e P_REG, utilizzati per introdurre un ritardo e gestire il timing tra i segnali provenienti da moduli differenti e un modulo SEND_OUT applicato nella messa in uscita del valore finale dei 128 valori accumulati, un registro del tutto analogo a quello precedentemente descritto per il caso a tre livelli.

Il modulo di clock gating viene attivato, o disattivato, da un segnale *enable* proveniente dalla FSM, questo risulta essere '1' solo nel caso in cui entrambi i valori uscenti dalle memorie, *p_out_ram* e *c_out_ram* sono non nulli, per questo ne viene calcolato, nella FSM, l'OR bit a bit, il quale risulta essere '1' solo nel caso in cui almeno un bit che lo compone sia '1', tra i risultati dei due OR viene effettuato un AND di modo tale che sia soddisfatto l'obiettivo di abilitare il MAC solo nel caso in cui c'è la certezza che i due operanti abbiano valore non nullo, come riportato nella tabella di verità mostrata in tabella.

|P_out_ram	|C_out_ram	Enable	Clk	Gclk
0	0	0	0	0
1	0	0	1	0
1	1	1	0	0
1	1	1	1	1
0	1	0	0	0

Figura 50: Tabella di verità

In tabella sono stati riportati anche la tabella di verità inerente al segnale gclk, in uscita dal modulo CLK_ENABLE_MAC che espleta la stessa funzione del CLK_ENABLE_ACCUMULATORE della precedente architettura.

Se il modulo MAC viene abilitato da *c_out_ram* e *p_out_ram*, è da considerare che anche il modulo RAM_C viene abilitato da *p_out_ram*, ciò implica che il segnale *c_out_ram* sarà fornito con un ciclo di clock in ritardo, si vede quindi necessario introdurre un ulteriore registro sul percorso di *p_out_ram*, il quale fornirà il segnale *p_out* consentendo il corretto timing tra i due segnali.

L'operazione di moltiplicazione ed accumulo avviene in contemporanea quindi, attraverso una variabile temporanea aggiornata ad ogni ciclo di clock, condizionato dal segnale di *enable*, e scaricata una volta che la lettura della riga viene completata ed il risultato viene fornito in uscita.

Di seguito viene introdotto il codice SystemVerilog inerente alle modifiche apportate per effettuare il calcolo così come è stato descritto.

Module ARITMETICA

```
module REG_EN (clk, D, Q);

input logic clk;

input logic D;

output logic Q;

reg Q_int;

always_ff @ (posedge clk) begin

        Q_int = D;

end

assign Q = Q_int;

endmodule


module P_REG (clk, D, Q);

input logic clk;
```

```
input logic signed [3:0] D;
output logic signed [3:0] Q;
reg signed [3:0]Q_int;
always_ff @ (posedge clk) begin
        Q_int = D;
end
assign Q = Q_int;
endmodule

module CLK_ENABLE_MAC (clk, enable, gclk);
input clk, enable;
output gclk;

wire enable_in = (enable);
reg enable_latch;
always @(clk or enable_in)
        if (~clk)
                enable_latch <= enable_in;
assign gclk = (clk & enable_latch);
endmodule

module MAC (gclk, rst_mac, p_out, c_out_ram, data_out_acc_int);
input logic signed [3:0] p_out;
input logic signed [7:0] c_out_ram;
```

```
input logic gclk, rst_mac;
output logic signed [16:0] data_out_acc_int;

logic signed [16:0] data_out_acc_tmp;
always @(posedge  gclk, posedge rst_mac) begin
        if (rst_mac)
                data_out_acc_tmp <= 17'd0;
        else
                data_out_acc_tmp <= data_out_acc_tmp + (c_out_ram * p_out);
        end
assign data_out_acc_int = data_out_acc_tmp;
endmodule

module SEND_OUT (rst_mac, data_out_acc_int, data_out_acc);
input logic rst_mac;
input logic signed [16:0] data_out_acc_int;
output logic signed [16:0] data_out_acc;

logic [16:0] data_out_acc_tmp;
always @(posedge rst_mac) begin
        data_out_acc_tmp = data_out_acc_int;
end
assign data_out_acc = {17{data_out_acc_tmp}};
endmodule
```

```
module ARITMETICA (clk, rst_mac, enable, c_out_ram, p_out_ram, data_out_acc);

input wire clk, rst_mac;

input logic enable;

input logic signed [7:0] c_out_ram;

input logic signed [3:0] p_out_ram;

output logic signed [16:0] data_out_acc;

wire enable_reg, gclk;

wire [3:0] p_out;

wire [16:0] data_out_acc_int;


REG_EN reg_en (clk, enable, enable_reg);

CLK_ENABLE_MAC clk_enable_mac (clk, enable_reg, gclk);

P_REG p_reg (clk, p_out_ram, p_out);

MAC mac (gclk, rst_mac, p_out, c_out_ram, data_out_acc_int);

SEND_OUT send_out (rst_mac, data_out_acc_int, data_out_acc);

endmodule
```

CONTATORI

Il modulo CONTATORE ed il modulo INCREMENTATORE sono rimasti invariati. rispetto alla struttura descritta nel precedente capitolo, in quanto le rispettive strutture e funzioni restano immutate, Naturalmente essendo variate le dimensioni delle matrici, quindi il numero di pesi e campioni da “scorrere”, sono state modificate le soglie. Per quanto riguarda il modulo incrementatore, dal momento che il numero di word non è variato la soglia è rimasta fissata a 128, diversamente per il modulo CONTATORE, la cui soglia viene fissata all’interno della FSM, ma in questo caso dovrà contare da ‘0’ a ‘127’, quindi avremo un contatore a 7 bit, data

l’assenza di modifiche architetturali il codice SystemVerilog, già introdotto nel precedente capitolo, non viene ripetuto

FSM

Il modulo FSM contiene la macchina a stati responsabile della logica alla base di tutta l’architettura, la struttura non presenta molte variazioni rispetto a quella impiegata nell’architettura precedente, gli stati impiegati sono rimasti invariati, così come anche i segnali di ingresso-uscita, che determinano il passaggio tra questi.

Negli stati RIPOSA, SCRIVI_P, SCRIVI_C sono state opportunamente corrette le dimensioni dei bus, i quali dovranno andare a leggere e scrivere segnali con dimensioni differenti, sono stati modificati i segnali *threshold* inerenti al contatore, che come descritto nei precedenti paragrafi è stata ridotta, per il resto gli altri segnali restano invariati.

Anche in questa architettura sarà lo stato SCRITTE MEMORIE ad attivare le memorie, mettendo a disposizione il dato nel primo ciclo di clock nello stato CALCOLA, il contatore non verrà abilitato in questo stato ma in quello successivo, in quanto non richiede un ciclo di clock per avviarsi, indicando così l’indirizzo corretto nel multiplexer, il quale, vedendo le memorie opportunamente inizializzate, potrà selezionare il peso corretto. Analogamente il segnale *count*, il quale diventa proprio il segnale *indirizzo*, abiliterà la word line della memoria RAM_C, selezionando il campione; è opportuno ribadire che il peso e il campione avranno lo stesso indirizzo.

La principale variazione risiede nello stato CALCOLA, non più il fulcro del calcolo, che come descritto nel precedente paragrafo, è stato infatti introdotto nel modulo ARITMETICA, ma resta lo stato responsabile del coordinamento tra il modulo atto al conteggio, e quindi all’assegnazione degli indirizzi, e il modulo ARITMETICA. Per di più sarà in questo stato che verrà generato il segnale di *enable*, seguente la logica descritta dalla tabella di verità introdotta nel precedente paragrafo. Il condizionamento della RAM_C da *p_out_ram* è stato inserito in questo stato; in figura viene riportata la tabella di verità, la quale mostra come l’abilitazione della memoria sia condizionata dal NOR, bit a bit, tra i singoli bit di *p_out_ram*, da specificare che, come visto precedentemente, la memoria è attiva quando il segnale di ingresso ram_c_en è basso.

P_out_ram	Ram_c_en
0000	1
0001	0
…	…
1111	0

Figura 51: Tabella di verità ram_c_en

Di seguito viene riportato il codice SystemVerilog inerente allo stato CALCOLA, dal momento che gli altri stati risultano pressoché invariati, se non nelle dimensioni dei bus.

Module FSM

:
:

```
CALCOLA:

begin

    start_count = 1;

    incrementa = 0;

    rst_inc = 0;

    threshold = 7'd127;

    ram_p_en = 1;

    w_p_en = 0;

     sel_p = inc [6:0];

    indirizzo = count ;

    ram_c_en = (~|p_out_ram);

    w_c_en = 1;

     rst_mac = 0;

     enable = ((|p_out_ram)&(|c_out_ram));

    data_out = data_out_acc;
```

end

:
:

CIRCUITO

Il modulo complessivo, denominato sempre CIRCUITO, presenta variazioni legate al corretto instradamento dei segnali, non essendo stati introdotti nuovi moduli. Questo modulo è stato simulato nell'ambiente ModelSim e su questo è stata effettuata la sintesi con celle di libreria STMicroelectronics. Il codice SystemVerilog non presenta modifiche architetturali, a tal proposito si è scelto di ometterlo, in quanto chiaro riferimento al codice descritto nel precedente capitolo. Un'ultima nota riguardante i codici: è esplicito che se nella precedente architettura una scelta progettuale è stata quella di dividere i due bit, con il quale si andavano ad esprimere peso e segno, in due locazioni di memoria differenti identificate da un indirizzo univoco, in questa architettura si è scelto di mantenere i bit dei pesi in un'unica locazione di memoria, per cui vi è un solo modulo di memoria RAM_P contenente le celle di memoria necessarie

Simulazione finale

Per la verifica del corretto funzionamento dell'architettura completa è stata effettuata una simulazione ModelSim, sul modulo CIRCUITO, il quale comprende tutti i moduli precedentemente descritti. Nello svolgimento del capitolo verranno analizzati i risultati ottenuti dallo stimolo del sistema utilizzando un testbench scritto ad hoc per la struttura da simulare, nel quale verranno inseriti sia pesi che campioni, in forma vettoriale, ed andremo a studiare i segnali di uscita dal modulo.

Generazione dei dati in ingresso

Per una corretta stesura del testbench, che rispettasse le premesse introdotte nei precedenti paragrafi, è stato necessario costruire le matrici in modo coerente a quanto detto. I segnali di ingresso saranno *c_in [7:0], p_in_1 [63:0], p_in_2 [63:0], p_in_3 [63:0], p_in_4 [63:0], p_in_5 [63:0], p_in_6 [63:0], p_in_7 [63:0], p_in_8 [63:0]*. La struttura dovrà vedere due matrici, Pesi e Campioni, entrambe con valori contenenti il bit di segno.

Particolare attenzione è stata posta sulla matrice dei pesi, composta da elementi, rappresentati a 4 bit, nulli per l'80%, per far ciò si è dovuto scrivere un nuovo script Matlab, la cui idea di

base era quella di creare due matrici, della stessa dimensione, una di soli valori binari 0/1, ma che rispettasse la scelta di avere il 20% di valori ‘1’e i restanti ‘0’, la seconda matrice invece programmata per essere una matrice di numeri random con valori compresi tra ‘-8’ e ‘7’, consono con la rappresentazione binaria con segno a 4 bit. Moltiplicando gli elementi delle matrici aventi stesso indice, e riscrivendo i risultati una terza matrice, utilizzando un for loop, è stato possibile ottenere il risultato previsto. Per la matrice dei campioni è stato utilizzato uno script analogo al precedente, per formare un vettore di numeri casuali, andando però a variare gli estremi tra cui andare a generare i valori random tra ‘-128’ e ‘127’, per coerenza con la rappresentazione a 8 bit con segno, di seguito riportato lo script Matlab.

Per ottenere i vettori da poter inserire nel testbench è stato effettuato un reshape ed una conversione in binario per poter organizzare il codice SystemVerilog efficientemente

```
C = randi ([-128,127], 320, 1);
Peso = randi (100,128,128);
Peso_bin = (Peso<20);
P_dec = randi ([-8,7],128, 128);
for i =1:128
    for j=1:128
        p_pesata (i, j) = Peso_bin(i,j)*P_dec(i,j);
    end
end
P_bin = dec2bin(p_pesata,4);
P_Select = P_bin(:,5:8) ;
P = reshape(P_Select,128,64,[]);
```

Come si può notare le prime tre righe dello script risultano totalmente analoghe allo script utilizzato per il testbench della precedente architettura; successivamente viene generata una matrice 128x128 di valori casuali che, attraverso i due for loop, utilizzati per scorrere le righe

e le colonne della matrice, viene moltiplicata per la matrice binaria. La matrice risultante avrà il 20% di elementi non nulli ma in forma decimale; verranno quindi convertiti in binario in rappresentazione a 4 bit. Per questa conversione Matlab gestisce l'overflow ampliando il numero di bit, per cui i numeri negativi vengono comunque rappresentati a 8 bit, motivo per il quale è stato necessario introdurre una selezione del vettore andando quindi ad estrarre i 4 LSB, che infine vengono riorganizzati in matrici 128x64, data l'organizzazione delle memorie a disposizione.

Testbench

La struttura del testbench è stata realizzata per le verifiche funzionali e per la sintesi del modulo complessivo CIRCUITO, istanziato all'interno del testbench.

Si è scelto di lasciare invariato l'ordine di scrittura e calcolo, generando opportunamente matrici e vettori utilizzando Matlab, come descritto nel paragrafo precedente. Quindi ci sarà una prima scrittura dei pesi e dei campioni, si procede con una fase di calcolo, al termine di questa avverrà una riscrittura dei campioni e quindi un nuovo calcolo. Complessivamente i campioni verranno scritti tre volte, con conseguente calcolo. Il codice SystemVerilog del testbench non viene introdotto di seguito in quanto del tutto analogo a quello utilizzato in precedenza, in questo caso avremo otto ingressi *p_in* mentre non avremo ingressi *s_in*, del resto la struttura del testbench resta analoga.

Simulazione ModelSim

Per completezza vengono riportate alcune figure riguardanti la simulazione del codice su ModelSim con relativi riferimenti temporali, considerando sempre che la transizione tra i vari stati è rimasta invariata. La simulazione viene avviata, con il segnale *nuovi_p* e con i relativi segnali *p_in*, dopo 600 ns, passando dallo stato RIPOSA a SCRIVI_P, al termine del quale, istante 1340 ns, passerà in SCRIVI_C, come evidenziato nella figura seguente.

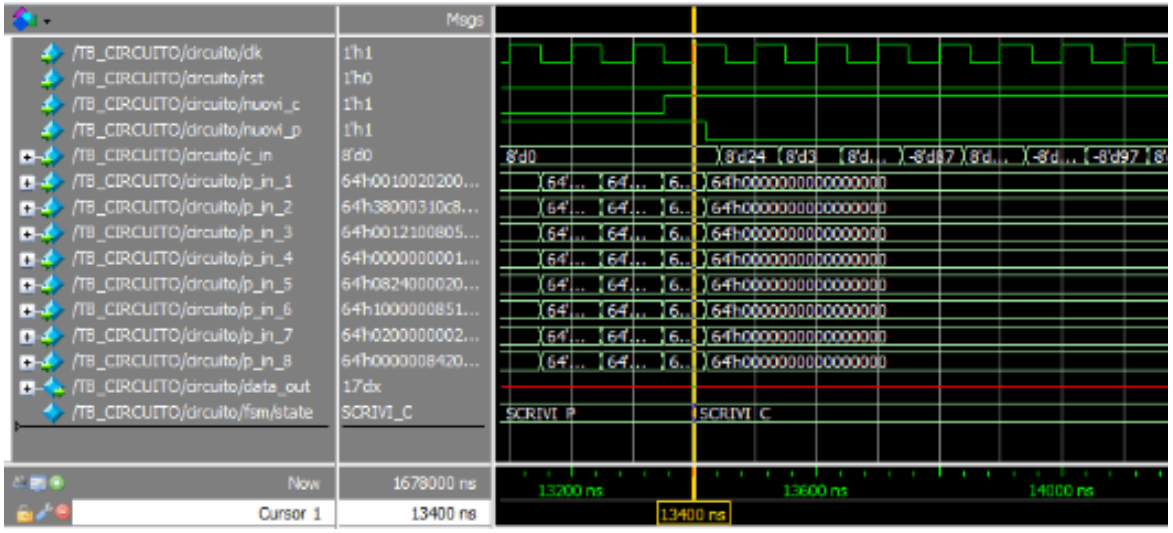


Figura 52: Inizio scrittura campioni

Da qui il sistema passerà, per un ciclo di clock, necessari ad inizializzare le memorie, come detto in precedenza, allo stato SCRITTE MEMORIE, per poi procedere con il calcolo, ed al termine di ogni giro nello stato FINE GIRO verrà posto in uscita l'elemento del vettore di uscita D relativo alla word.

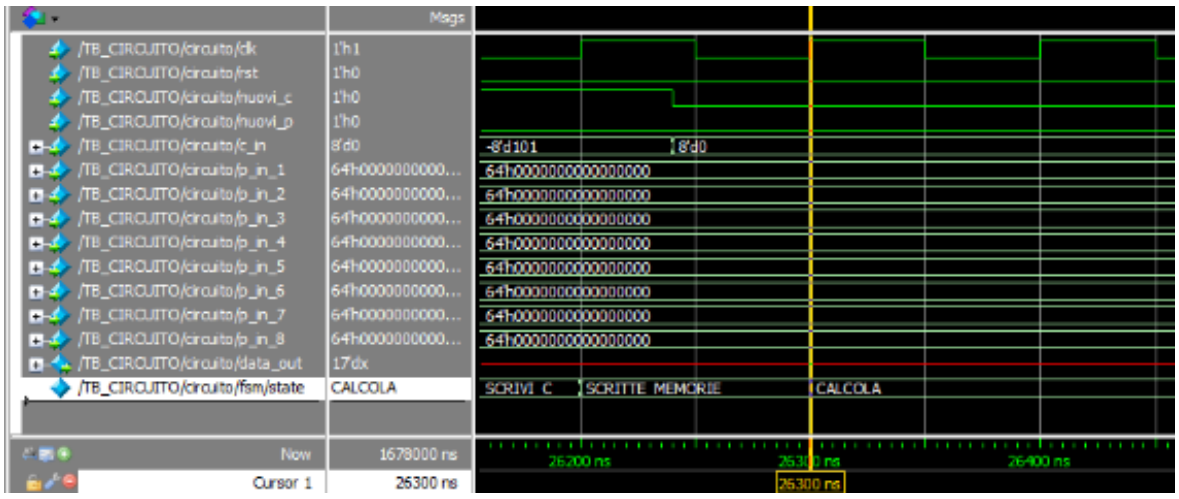


Figura 53: Inizio fase di calcolo

Al termine del calcolo di tutti gli elementi di uscita il sistema passa in FINE CALCOLA, da questo stato attende una nuova introduzione dei campioni, con un nuovo fronte di *nuovi_c*, per riprendere il calcolo, altrimenti ritorna in RIPOSA, e la simulazione si conclude.

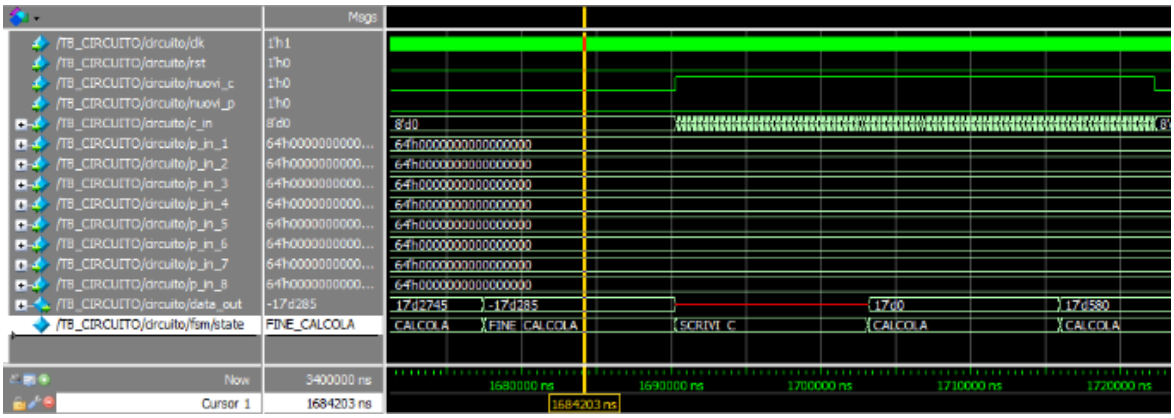


Figura 54: Fine fase di calcolo

Sintesi e power analysis del sistema

Una volta effettuata la corretta verifica delle funzionalità del sistema, come descritto nel precedente paragrafo, si è passati alla sintesi dell'architettura e successivamente alla power analysis, per aver un riscontro dell'efficienza di quest'ultima soluzione. Al fine di un confronto di risultati coerenti si è scelto di mantenere la stessa frequenza di clock dell'architettura a tre livelli, $f_{clk} = 10\,MHz$ difatti, da come si evince dalle simulazioni su ModelSim si ha un periodo di $T_{clk} = 100\,ns$. Questo periodo, quindi, rispetta il vincolo di effettuare la scrittura dei campioni e le operazioni di moltiplicazione tra questi e il set di pesi in un tempo di 1,5 ms, coerentemente con quanto detto per la sintesi della struttura a tre livelli, sarà compito del tool di sintesi rispettare i vincoli di set-up e hold dettati dalla relazione con la frequenza scelta.

Al fine di poter avere due soluzioni perfettamente confrontabili, sono stati scelti gli stessi constranit fissati per il progetto a tre bit, quindi, per quanto riguarda la tensione di alimentazione, coerentemente con la precedente architettura, il valore è stato ridotto del 10% a 1,08 V, con conseguenze analoghe, descritte nel precedente capitolo. La temperatura di funzionamento resta fissata a 125 °C, per mantenere le condizioni peggiorative di funzionamento, come fatto in precedenza (in particolare per la potenza statica). La tecnologia delle celle utilizzate resta la C090D a 90 nm e sono state scelte le celle di libreria STMicroelectronics COREL_H.

Risultati processo di sintesi

È doveroso premettere, al fine di un corretto confronto tra le soluzioni, che, in seguito alle scelte progettuali effettuate, l'architettura a 4 bit dispone di un numero di banchi di memoria inferiori, otto piuttosto che dieci, questo si riflette necessariamente in una minor porzione di silicio occupata da memorie, conseguentemente con un minor consumo di potenza relativo alle memorie.

Il blocco digitale è stato sintetizzato in 1204 celle, ripartite rispettivamente in 843 celle combinatorie, 66 celle sequenziali, 264 celle di buffer e 31 di inverter. L'area stimata del blocco è di 374581.59 μm^2 (circa 620 µm * 620 µm), per lo più occupata da memorie, le piste di collegamento hanno una lunghezza totale stimata di 431310.62 µm. In figura di seguito viene mostrato il place and route del circuito.

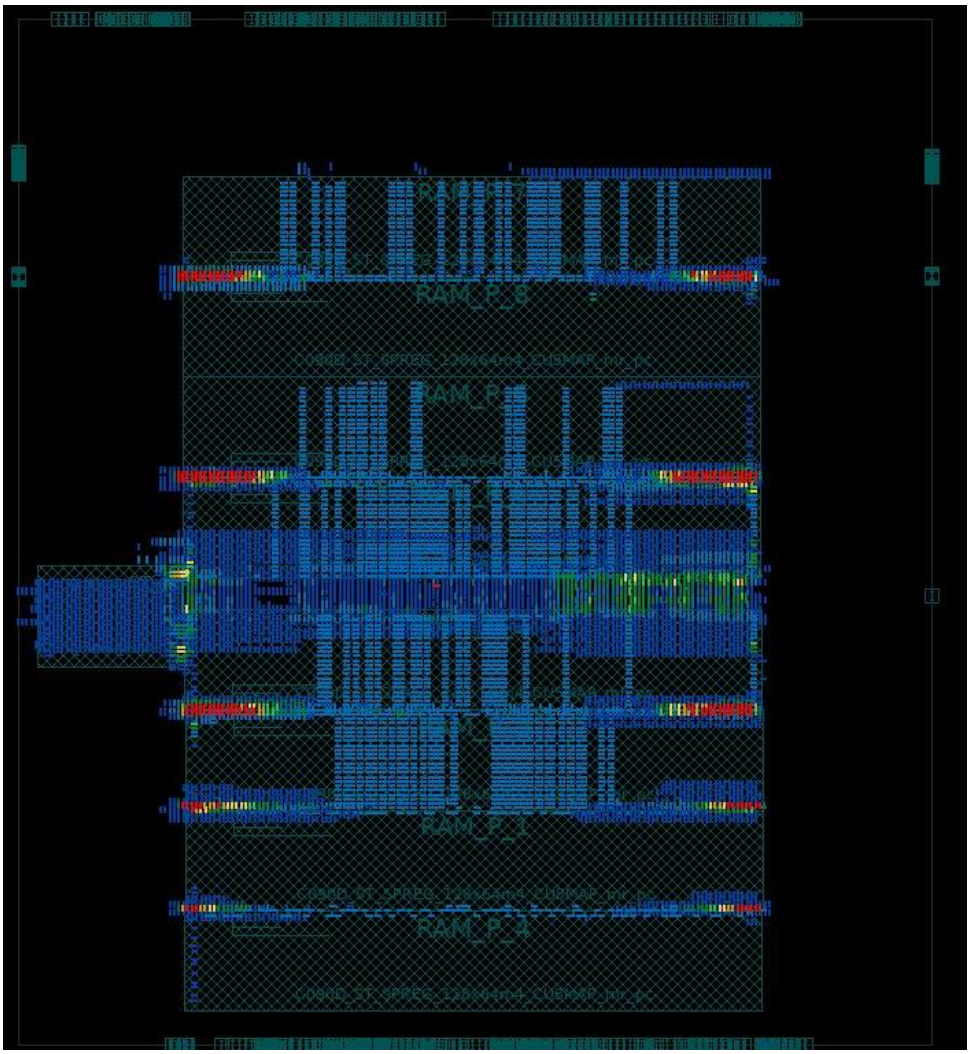


Figura 55: Place and route architettura 4bit

Analogamente al caso precedente il percorso critico del circuito è localizzato nella FSM, dal momento che si trova allo stesso modo a configurare il dispositivo per le operazioni di scrittura

della RAM_P. In questo percorso il tempo di propagazione dei dati è di 2.88 ns, che corrisponde a una frequenza massima di clock pari a 347.22 MHz, nettamente superiore alla specifica di progetto. In questo percorso critico, inoltre, si ha uno Slack pari a 64.69 ns, per cui anche in questo caso i segnali stanno arrivando con molto anticipo, rispetto al vincolo sul periodo di clock, ai nodi, ovvero il dispositivo potrebbe adattarsi ad un clock molto più veloce, appunto di 347.22 MHz.

La stima di potenza estratta in questa fase della sintesi, non basata sul testbench, ma con vettori di input casuali con una switching activity del 10%, distanti da quella che sarebbe una soluzione reale. Si stimano 930.4 µW di potenza totale di cui 871.7 µW di potenza interna, 31.7 µW di switching power e 27.1 µW di static power. Di seguito il risultato di una stima di potenza più accurata.

Power Analysis

Al fine di ottenere una stima di potenza più accurata è stata effettuata una Power Analysis, analogamente a quanto fatto per il progetto precedente, con il tool Prime Time messo a disposizione da STMicroelectronics. Il tool richiede di isolare un intervallo di tempo del testbench del quale si vuole tener conto per poter effettuare una media sulla potenza dissipata. Gli istanti considerati, per estrarre l'intervallo, sono stati l'inizio della fase di scrittura dei campioni, che avviene 13400 ns dall'inizio della simulazione, fino al termine della fase di calcolo del prodotto matrice vettori, all'istante 1677400 ns, dall'inizio della simulazione, analogamente a quanto fatto in precedenza viene esclusa la scrittura dei pesi in memoria.

A questo punto avremo una simulazione che segue un'applicazione reale, andando a stimolare il circuito sintetizzato con il testbench con il quale è stato stimolato. Questo introdurrà degli stimoli funzionali e non casuali sulle porte, il che significa che, ad esempio, la porta del registro di enable del modulo aritmetica verrà attivata quando necessario, seguendo il corretto funzionamento della struttura, e non con una switching activity, dettata apriori, del 10%. Si passa cossi ad un consumo di potenza di switching dinamica di 13.1 µW rispetto ai 27.1 µW precedenti. Questo risultato, analogamente a quanto visto in precedenza, è attribuibile al fatto che la memoria dei pesi ha valori diversi da 0 solo per il 20% del totale, riducendo quindi la probabilità di avere transizioni basso alto, questo si riflette su una riduzione della switching activity e quindi una riduzione della potenza dinamica. Di seguito si riportano in tabella i risultati dei singoli moduli, espressi in µW.

Gerarchia	Potenza dinamica interna	Potenza di switching dinamica	Potenza statica	Potenza totale
Incrementatore	0.0142	0.00388	0.0135	0.0316
Contatore	1.77	1.11	0.0178	2.90
Ram_C	19.1	0.22	1.04	20.3
Ram_P	230	9.21	25.8	265
Aritmetica	3.57	1.16	0.0826	4.81
Fsm	0.596	1.43	0.0253	2.05
Circuito	255	13.1	27	295

Le simulazioni sono state effettuate direttamente sul dispositivo utilizzante la sospensione delle Ram_P e Ram_S quando non utilizzati.

È evidente che area e consumo sono dominati dalle memorie e che la scelta di rappresentare i pesi con 4 bit comporti un aumento significativo rispetto alla rappresentazione puramente binaria.

Il numero minimo di bit con cui rappresentare i pesi nelle applicazioni di AI è argomento di ricerca e funzione del campo applicativo. Una memoria che permetta di memorizzare più livelli nella stessa cella rappresenterebbe sicuramente un grande vantaggio.

Confronto soluzioni

Nel mettere in relazione le diverse architetture bisogna premettere che queste si pongono come un percorso evolutivo, dalla prima architettura ad 1bit, fino all'ultima a 4 bit. È evidente che si avrà una crescita sia in termini di consumo di potenza e occupazione di potenza, ma soprattutto in termini di prestazioni, in quanto a crescere è anche la complessità dei moduli, i quali passeranno dal dover moltiplicare un bit per 8 bit, con relativo accumulo, poi 2 bit per 8 bit, ed infine 4 bit per 8 bit con accumulo del risultato. Anche se le scelte progettuali hanno portato a ridimensionare le dimensioni delle matrici, le diverse architetture hanno visto un contenuto informativo sempre crescente, aumentando la precisione riducendone la perdita di informazione con cui può essere codificato il peso.

È opportuno puntualizzare che, per ottenere un confronto realistico tra le diverse architetture progettate, è necessario omettere dal ragionamento le memorie, in quanto, per soddisfare le scelte progettuali effettuate, sono state scelte con tagli differenti. Per l'appunto i moduli da confrontare direttamente saranno ARITMETICA e FSM, in quanto i moduli CONTATORE e INCREMENTATORE, hanno la stessa funzione, se non una variazione sulla soglia, il che si riflette su una piccola variazione, ma davvero impercettibile.

Per quanto riguarda la FSM, i cui risultati per il confronto di potenza vengono riportati nella tabella seguente: il valore più evidente è quello relativo alla potenza di switching dinamica che si riflette sul totale. È infatti da ricordare che, pur se l'assegnazione del segno al campione non viene più effettuata, in questo modulo vi è il path relativo all'enable della memoria dei campioni e del modulo ARTIMETICA, questo non avviene più per connessione diretta del valore del peso, che nel caso a tre livelli era composto da un solo bit 0/1, ma è composto, questo comporta l'introduzione di porte logiche per espletare la stessa funzione che avranno una propria switching activity e si rifletterà proprio sulla potenza dinamica. Per completezza in tabella sono riportati anche i valori della soluzione a due livelli, progettata nel lavoro di tesi di Davide Molino.

Modulo Circuitale	Potenza dinamica Interna	Potenza di switching dinamica	Potenza statica	Potenza totale
FSM $_{\text{due livelli}}$	0,563	1,53	0,0254	2,12
FSM $_{\text{tre livelli}}$	0.683	1.05	0.0312	1.77
FSM $_{\text{4 bit}}$	0.596	1.43	0.0253	2.05

Per quanto riguarda l'area occupata nella soluzione a 4 bit si ha un'occupazione, da parte del modulo FSM, di 522.4576 μm^2, rispetto ai 665.1456 μm^2 della soluzione a tre livelli.

Il consumo di potenza del modulo ARITMETICA, i cui valori vengono riportati in tabella, la quale comprende anche i valori di potenza della soluzione a due livelli per completezza, vede un consumo più che raddoppiato sia di potenza statica che di potenza dinamica e quindi di potenza totale, valori in linea con quanto previsto. Si ricorda infatti che nelle architetture precedenti il modulo ARITMETICA era stato ottimizzato applicando il valore del peso come segnale di attivazione dell'accumulatore, per l'architettura con pesi a 4 bit invece; oltre al modulo di accumulo è presente un moltiplicatore ad 8 bit che giustifica un tale aumento di potenza.

Modulo Circuitale	Potenza dinamica Interna	Potenza di switching dinamica	Potenza statica	Potenza totale
ARITMETICA due livelli	1,23	0,287	0,0419	1,56
ARITMETICA tre livelli	1.47	0.367	0.0425	1.88
ARITMETICA 4 bit	3.57	1.16	0.0826	4.81

Per quanto riguarda l'area occupata nella soluzione a 4 bit si ha un'occupazione, da parte del modulo ARITMETICA, di 4663.7024 μm^2, rispetto ai 2017.3888 μm^2 della soluzione a tre livelli, raddoppio giustificato dalla presenza di quel moltiplicatore in più, quindi coerente con quanto detto finora.

4. Ottimizzazione mediante l'utilizzo di registri

Le architetture, illustrate nei precedenti paragrafi pongono tutti gli elementi, pesi e campioni, in memoria, questo implica occupazione di silicio e consumo di memoria, come si evince dalla power analysis. È possibile effettuare un'ottimizzazione andando a sostituire le memorie con dei registri là dove i dati da memorizzare possono definirsi temporanei.

Guardando all'architettura a 4 bit, al fine di apportare ulteriori migliorie, si è scelto di provare ad introdurre, in un primo momento dei registri di uscita (una forma di memorizzazione per questi è comunque necessaria e questo modulo rappresenta un aumento di funzionalità piuttosto che una ottimizzazione), e in seguito di sostituire la memoria dei campioni con una serie di registri. Da questo tipo di ottimizzazioni ci si aspetta una riduzione di potenza e di area, senza trascurare il fatto che l'occupazione del silicio e il consumo sono dominati dalle memorie dei pesi, nettamente più ingombranti, in quanto memorizzano un numero maggiore di bit.

Per il compimento del primo passo si sono resi necessari un numero di registri uguale al numero di elementi del vettore risultati D, quindi 128 registri a 17 bit, in modo tale da mantenere il dato senza perdita di precisione. Per una valutazione effettiva è stata effettuata una sintesi ed una power analysis, ci si aspetta un aumento di consumo e di area occupata del circuito totale, seppur minima rispetto agli altri moduli, dovuta proprio all'aggiunta di un ulteriore modulo *Reg_D*. Di seguito si riportano i risultati della power analysis in tabella.

Modulo Circuitale	Potenza dinamica Interna	Potenza di switching dinamica	Potenza statica	Potenza totale
CIRCUITO $_{4\ bit\_reg\_out}$	257	19.7	32.6	309
Reg_D $_{4\ bit\_reg\_out}$	2.07	1.26	5.51	8.84

Come previsto vi è un aumento di consumi dovuto all'aggiunta del modulo Reg_D, che comprende i registri di uscita, i cui risultati vengono riportati nella tabella precedente.

Supponendo di utilizzare più moduli come questo in cascata per realizzare una rete neurale a più livelli, è probabile che il risultato andrebbe troncato e rappresentato con meno bit; la soluzione di memorizzazione dei risultati proposta rappresenta quindi una stima di caso peggiore.

Seguendo un ragionamento del tutto analogo al precedente si è scelto di sostituire la memoria RAM_C con un modulo Reg_C, contenente i registri atti al mantenimento del valore dei campioni per tutta la durata del calcolo, si avranno quindi 128 registri ad 8 bit; questa si è valutata come una scelta lecita dal momento in cui si considerano i valori della matrice dei pesi fissi, e si vanno a variare i campioni, reinserendo il vettore in memoria al termine dell'operazione. Per la progettazione di questo modulo, al fine di mantenere una continuità con i segnali già presenti nell'architettura, ci sono due multiplexer, uno per l'input, in grado di assegnare il segnale di ingresso, in base all'indirizzo, su uno dei 128 registri, nominati C_0, C_1,…,C_127, ed un secondo multiplexer di output, il quale analogamente associa al segnale di uscita *c_out_ram* l'uscita del registro selezionato dall'indirizzo. I due multiplexer sono attivati da due segnali di attivazione, uno in lettura e l'altro in scrittura, rispettivamente *r_c_en* e *r_c_en*. Dall'implementazione di questo registro ci si aspetta una riduzione di area occupata e consumo di potenza rispetto all'utilizzo della SRAM utilizzata per memorizzare i campioni, per constatare ciò è stata effettuata una power analysis, di cui i dati estratti vengono riportati nella seguente tabella.

Modulo Circuitale	Potenza dinamica Interna	Potenza di switching dinamica	Potenza statica	Potenza totale
CIRCUITO $_{4}$ bit_regC	240	22.7	32.6	295

Confrontando i valori con l'architettura avente solo i registri in uscita, è evidente una riduzione sostanziale della potenza interna, al costo della potenza dinamica, che registra un leggero aumento avendo introdotto un gran numero di registri, ed un multiplexer in più, che si riflettono su una maggiore switching activity, la potenza statica resta invariata.

Particolare attenzione è da porre nella sostituzione dei registri al posto del banco di memoria RAM_C; i valori risultanti dalla power analysis vengono introdotti nella seguente tabella, espressi in µW.

Modulo Circuitale	Potenza dinamica Interna	Potenza di switching dinamica	Potenza statica	Potenza totale
RAM_C	19.1	0.22	1.04	20.3
Reg_C	2.07	5.46	0.99	8.52

È evidente come la potenza interna cali sostanzialmente da RAM_C a Reg_C, la potenza dinamica aumenta leggermente, coerentemente con l'aumento della switching activity, mentre la potenza statica resta pressoché invariata, si conferma una scelta opportuna dal momento che si ha una riduzione sulla potenza totale di circa il 60%, confermata dal fatto che i campioni risultano essere gli elementi dell'operazione che variano più frequentemente e quindi necessitano di più cicli di scrittura, rispetto alla matrice dei pesi che resta invariata per tutta la durata dell'operazione.

Per entrambe le architetture in cui sono stati utilizzati i registri, il modulo più dispendioso in termini di area ed energia resta RAM_P, quindi si conferma essere un elemento limitante per tutte le versioni dell'architettura, presentate in questa tesi, al di fuori di qualsivoglia tentativo di ottimizzazione.

5. Conclusioni

In conclusione, si vuole fornire un confronto generico tra le architetture descritte all'interno di questo lavoro di tesi, come previsto abbiamo registrato un aumento di potenza coerente con l'incremento del numero di bit del peso.

Il consumo, così come l'area, sono dominati dalle memorie, concetto condiviso in tutte le tre soluzioni, bisogna però tenere in considerazione che ad un aumento di consumo e di area corrisponde, in ogni caso, ad un aumento in termini di prestazioni. Nella tabella di seguito riportata, vengono riassunti i valori estratti dalla power analysis di ogni architettura, considerando che il numero di celle di memoria nell'architettura ad 1 bit è di cinque, dieci in quella a 2 bit, ed otto nella soluzione con pesi a 4 bit, come ampiamente discusso nei paragrafi precedenti.

Modulo Circuitale	Potenza dinamica Interna	Potenza di switching dinamica	Potenza statica	Potenza totale
CIRCUITO $_{1\ bit}$	156	9,63	20,2	186
CIRCUITO $_{2\ bit}$	294	11	39	344
CIRCUITO $_{4\ bit}$	255	13.1	27	295
CIRCUITO $_{4\ bit\_regC}$	240	22.7	32.6	295

Da quanto risulta dalle diverse sintesi e power analysis si evince che, una soluzione analogica, risulta essere una scelta migliore in termini di performance; questo perché una soluzione digitale necessita di un modulo che effettui l'operazione rispetto al caso analogico, nel quale l'operazione viene effettuata sfruttando leggi fisiche dell'elettronica e senza la necessita di moduli esterni. Si consideri sempre che una soluzione di questo tipo prevede l'utilizzo di memorie digitali convenzionali le quali effettuano una lettura, e scrittura, in memoria interessando la singola word line, per cui non è possibile optare per livelli di parallelismo maggiori.

I valori di potenza riportati in tabella sono relativi alla potenza media per ciclo di clock; effettuando un'analisi in termini di energia possiamo valutare l'energia dissipata per singolo

clock, per l'esecuzione dei calcoli relativi ad una riga di pesi, un nodo, e l'energia per l'esecuzione di tutte le operazione, di tutto il layer.

$$E_{clk} = \frac{P_{dyn}}{f} = \frac{(240 + 23)\mu w}{10MHz} = 26.3\ \rho J$$

L'energia per ciclo di clock risulta essere un valore di grandi dimensioni, questo, dal momento che per le operazioni di moltiplicazione e somma dei pesi per i campioni impiegano 128 cicli di clock, si riflette su l'energia per singolo nodo della rete:

$$E_{node} = E_{clk} * 128\ cicli = \ 3.36\ nJ$$

Per completare il ragionamento, l'energia per effettuare il calcolo sui 128 layer, dal momento che l'accesso in memoria avviene per singola word line, si ottiene:

$$E_{layer} = E_{node} * 128 = 431\ nJ$$

Da questa analisi si evince che una soluzione di questo tipo è adatta per applicazioni general purpose ma che interessano una matrice ridotta; questo perché, l'architettura progettata, vuole proporsi come una soluzione che utilizza memorie digitali standard, senza accedere alla struttura stessa della cella di memoria; in più, essendo una soluzione digitale, necessita di moduli aggiuntivi atti all'esecuzione delle operazioni stesse.

In conclusione, si può affermare che, per applicazioni che interessino grandi matrici di dati, e vettori, la soluzione più efficiente risulta essere quella analogica; inoltre, nello state dell'arte, sono presenti soluzioni più performanti, le quali, andando a modificare la struttura della cella di memoria, riescono a raggiungere livelli di parallelismo più alti, accedendo, sia in lettura che in scrittura, a tutte le word line contemporaneamente.

Ringraziamenti

Questo traguardo per me rappresenta la fine di una fase della vita e l'inizio di una nuova, che si affaccerà verso sfide più grandi dalle quali ne deriveranno soddisfazioni altrettanto grandi, si spera, quindi è doveroso per me ringraziare chi finora ha contribuito, direttamente o indirettamente, al raggiungimento di questa meta, perché forse ci ha creduto più di me.

Purtroppo, molte tra queste persone non possono essere qui presenti, ma quando potremo festeggeremo come si deve in sicurezza.

Un sentito ringraziamento alla professoressa Eleonora Franchi Scarselli per il supporto prestatomi per la stesura della tesi e per il suo particolare interessamento al mio lavoro, la sua cura nei dettagli ha permesso di effettuare un'analisi approfondita ed esaustiva del caso in esame, al professor Antonio Gnudi, al Dott. Ing. Alessio Antolini per il tempo dedicatomi nelle analisi e nelle sintesi, ma soprattutto per aver annullato le difficolta legate all'effettuare un lavoro di tesi in tempo di pandemia e per le chiacchierate informali.

Vorrei ringraziare i miei genitori che hanno saputo capire le mie esigenze ed hanno fatto più del loro possibile per assecondarle, hanno saputo intuire le mie ansie e preoccupazioni che, soprattutto in questo anno, si sono fatte più vive e accese, e spero che verranno ripagate con quante più soddisfazioni potrò dargli.

Ringrazio Alessando ed Emanuele, punto fisso per confronti imparziali, ma anche fonte di amore fraterno incondizionato, frutto di un legame che nessuno scontro potrebbe infrangere; se oggi sono questa persona lo devo anche a loro, crescere insieme vuol dire questo, le foto in camera e gli istanti di vita insieme me lo ricordano costantemente.

Grazie a Giovanna, la persona per la quale cerco di dare il meglio di me, ma alle volte si trova ad aver a che fare con parti peggiori, grazie per esser sempre lì ad aspettarmi e a confortarmi, soprattutto quando starmi affianco diventa difficile, grazie per interpretare i miei sguardi e i miei stati d'animo, quando a volte neanche io riesco farlo.

Ringrazio i miei zii e parenti che mi hanno cresciuto, a loro devo i primi insegnamenti di vita, i primi affetti cari, grazie per riuscir a farmi sentire il vostro affetto anche in questo periodo di distanza e distaccamento.

Grazie ai miei cugini, primi amici, primi compagni, solo la casa di nonna è testimone di tutte le risate, di tutti gli scherzi, gli screzi ed anche di tutti i danni che abbiamo potuto fare finora.

Grazie agli Another Band, Rehtona per gli amici, solo chi ha mai fatto parte di un gruppo sa che tipo di legame si instaura, grazie per le emozioni che solo la musica può dare, per i momenti in sala prove, per l'adrenalina di suonare dal vivo, che purtroppo oggi ci manca fin troppo, grazie per aver condiviso tutto questo con me.

Grazie a Giuseppe per la nostra amicizia oltre le distanze, 23 anni e non sentirli, anche per il suo impegno sul "fronte" in questo momento particolare.

Grazie ai colleghi che hanno condiviso con me questi anni di formazione, accrescendoli di esperienze oltre lo studio, aprendomi a nuove realtà e nuovi punti di vista; in particolare ringrazio Alberto, Andrea, Antonio, Enrico e Francesca, con cui ho condiviso gli ultimi due anni.

Grazie ai compagni delle due ruote, per avermi fatto avvicinare a questo sport che mi porta a scoprire i meandri più incontaminati della nostra terra, ed anche ad avere qualche cicatrice in più.

Grazie agli HiperPing per avermi fatto staccare dalla monotonia di queste giornate infinite di pandemia, e per avermi spronato alla competitività.

Grazie a Giuseppe, Nathan, Gabriele, Luigi, Ilaria, Enza, Erica, per le serate, per le chiacchiere, per le escursioni, per i momenti insieme che ci hanno portati ad intrecciare dei bei rapporti.

Grazie a tutte le persone che ho avuto modo di conoscere in questi anni, e che hanno contribuito ad essere ciò che sono.

6. Bibliografia

[1] A. Sebastian, M. L. Gallo, R. Khaddam-Aljameh e E. Eleftheriou, «Memory devices and applications for in-memory computing,» *Nat. Nanotechnol.*, pp. 529-544, 2020.

[2] N. Verma, H. Jia, H. Valavi, L. Y. Tang, M. Ozatay, L.-Y. Chen, B. Zhang e P. Deaville, «In-Memory Computing, Advances and Prospect,» *IEEE SOLID-STATE CIRCUITS MAGAZINE*, pp. 43-55, 2019.

[3] I. Hubara, M. Courbariaux, D. Soudry, R. ElYaniv e Y. Bengio, «Quantized neural networks: Training neural networks with low,» *J. Mach.Learning Res*, vol. 18, n. 1, p. 6869–6898, 2017.

[4] N. P. Jouppi, «"In-datacenter performance analysis of a tensor processing unit,» *Proc. 44th Annu. Int. Symp. Computer ASrchitectuere*, pp. 1-12, 2017.

[5] W. HAENSCH, T. GOKMEN e R. PURI, «The Next Generation of Learning Hardware: Analog Computing,» *PROCEEDINGS OF THE IEEE*, vol. 107, n. 1, pp. 108-122, 2019.

[6] K. Steinbuch, «Die lernmatrix,» *Kybernetik*, vol. 1, n. 1, pp. 36-45, 1961.

[7] C. Lehmann, M. Viredaz e F. A. Blayo, «A generic systolic array building block for neural networks with on-chip learning,» *IEEE Trans. Neural Netw*, vol. 4, n. 3, pp. 400-407, 1993.

[8] A. Sebastian, M. L. Gallo, G. W. Burr, S. Kim, M. BrightSky e E. Eleftheriou, «Tutorial: Brain-inspired computing using phase-change memory devices,» *JOURNAL OF APPLIED PHYSICS*, 2018.

[9] G. W. Burr, «Experimental demonstration and tolerancing of a large-scale neural network (165,000 synapses), using phase-change memory as the synaptic weight element,,» *IEDM Tech. Dig.*, p. 29.5.1–29.5.4, 2014.

[10] S. Kim, «A phase change memory cell with metallic surfactant layer as a resistance drif stabilizer,» *IEDM Tech. Dig.*, p. 30.7.1–30.7.4., 2013.

[11] W. W. Koelmans, A. Sebastian, V. P. Jonnalagadda, D. Krebs, L. Dellmann e E. Eleftheriou, «Projected phase-change memory devices,» *Nature Commun.*, vol. 6, n. 8181, 2015.

[12] R. Waser, R. Dittmann, G. Staikov e K. Szot, «Redox-based resistive switching memories—Nanoionic mechanisms, prospects, and challenges,» *Adv. Mater.*, vol. 21, n. 25-26, p. 2632–2663, 2009.

[13] D. Jana, «Conductive-bridging random access memory: Challenges and opportunity for 3D architecture,» *" Nanoscale Res. Lett.*, vol. 10, n. 188, 2015.

[14] A. Antolini, E. Franchi Scarselli, A. Gnudi, M. Carissimi, M. Pasotti, P. Romele e R. Canegalli, «Characterization and programming algorithm of a phase change memory cells for analog in-memory computing.,» *MDPI*, in revisione.

[15] D. Molino, *Progetto e sintesi di un modulo digitale per il calcolo di prodotti matrice/vettori*, Bologna, 2020.

7. Indice delle figure