UNIVERSITÀ DEGLI STUDI DI
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in Ingegneria Informatica

ELABORATO ASDI

Anno Accademico 2022/2023

Candidato
PAOLO RUSSO
matr.M63001426

Contents

Chapter 1

Traccia

Progettare, implementare in VHDL e simulare la seguente architettura. Un sistema è composto da 2 nodi, A e B. A possiede una ROM con 8 locazioni di 6 bit ciascuna. All'avvio del sistema A invia a B, mediante protocollo di handshaking, il contenuto della propria ROM, una locazione alla volta. Ricevuta una stringa da A, l'unità B divide la stringa in un operando S1 di 4 bit (i più significativi) e un operando S2 di 2 bit e calcola il valore S1 mod S2 (resto della divisione intera fra S1 e S2) utilizzando una macchina aritmetica implementata in maniera strutturale a partire da un componente sottrattore descritto in maniera behavioral. Il risultato dell?operazione viene salvato in una memoria.

Chapter 2

Descrizione Soluzione

Inazitutto abbiamo due nodi in cui è prevista un hanshake con comunicazione parallela. Partiamo dal suddividere il nostro problema in tanti piccoli sottoproblemi. Per quanto riguarda il nodo A, andiamo ad implementare , una macchina che al proprio interno è dotata di una memoria ed un registro. Per quanto riguarda il nodo B, uno dei punti fondamentali è capire come implementare il nostro sottrattore. Poichè viene richiesto , un sottrattore comportamentale , che sia strutturale, l' idea è quello di andare a valutare il comportamento del singolo bit con il possibile riporto associato. Partendo da questo elemento, mediante un for spaziale,come fatto in altre macchine come il ripple carry ,andiamo ad generare il nostro componente sotrattore, implementando una macchina puramente combinatoria. Attraverso questo componenete, andando a definire un oppurtuna logica di comando cerchiamo di calcolare S1 mod S2 attraverso delle sottrazioni iterative.Pertanto,appena il

valore di S2 risulta maggiore di S1 ci fermiamo ed salviamo il risultato all'interno di una Ram. Pertanto il nostro sottrattore:

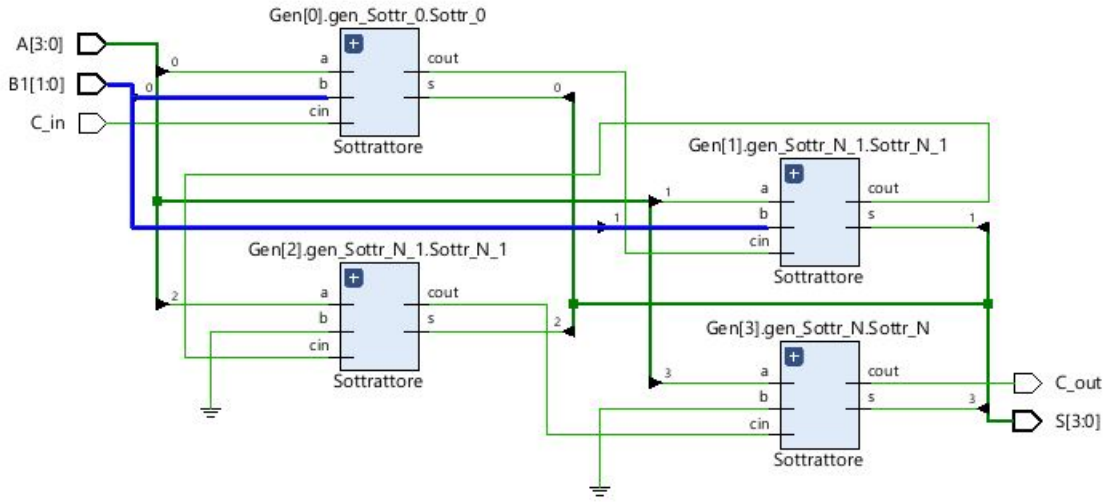

Figure 2.1: Sottrattore

Chapter 3

NODOA

3.1 Descrizione

Il nodo A, è una macchina molto semplice. Dopo aver effettuato l'handshake con il nodo B, preleva il dato dalla sua memoria e lo invia.Pronto per inviare il prossimo dato. Per quanto riguarda la parte Operativa, avremmo dunque, una memoria da cui prelevo il dato, ed un registro Rx, con cui andiamo ad inviare il dato dopo aver effettuato l'handshake. Il nodo A, sarà operativo , appena sarà alto il segnale di Start, ed opererà fintantochè, non verranno letti tutti i dati dalla memoria.

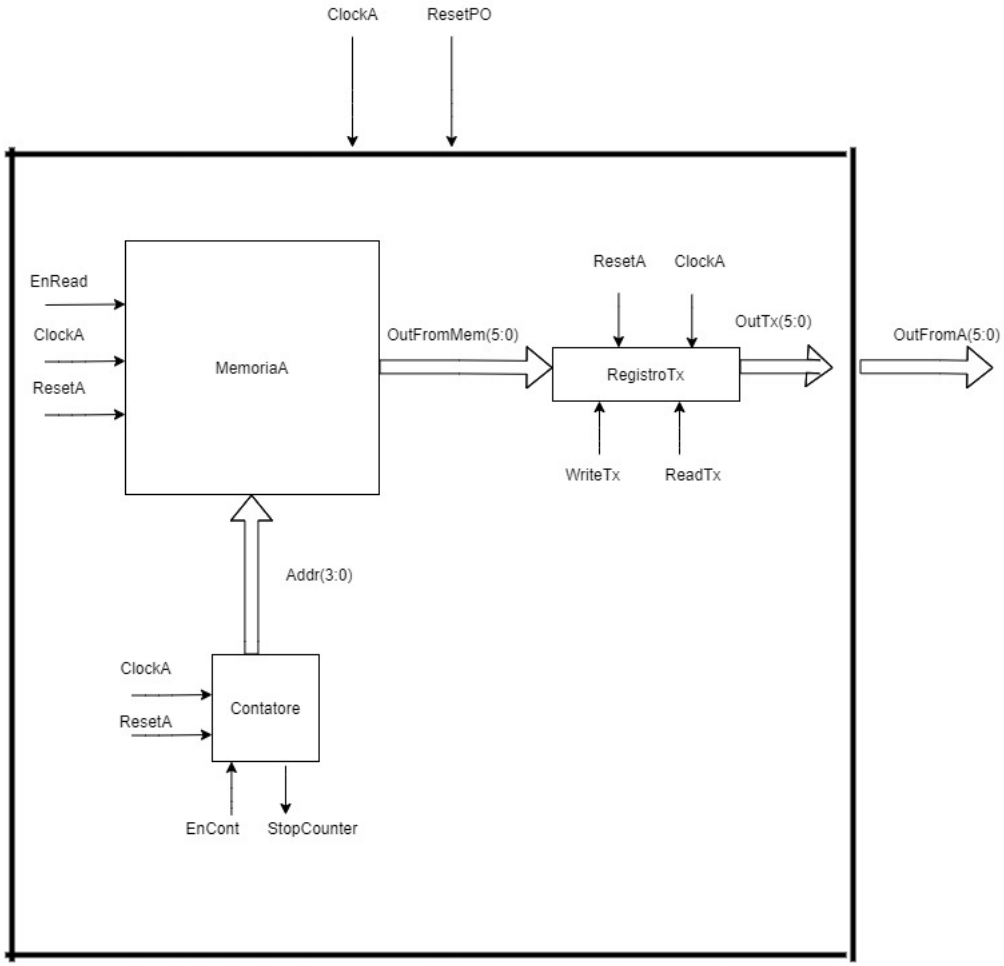

Figure 3.1: ParteOperativa

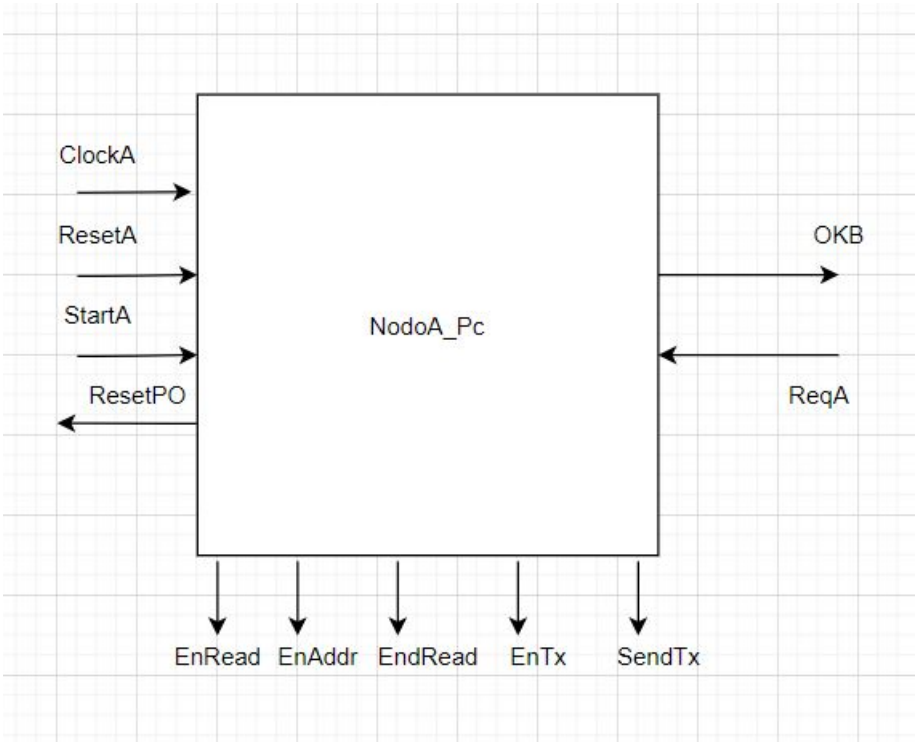

Figure 3.2: ParteControllo

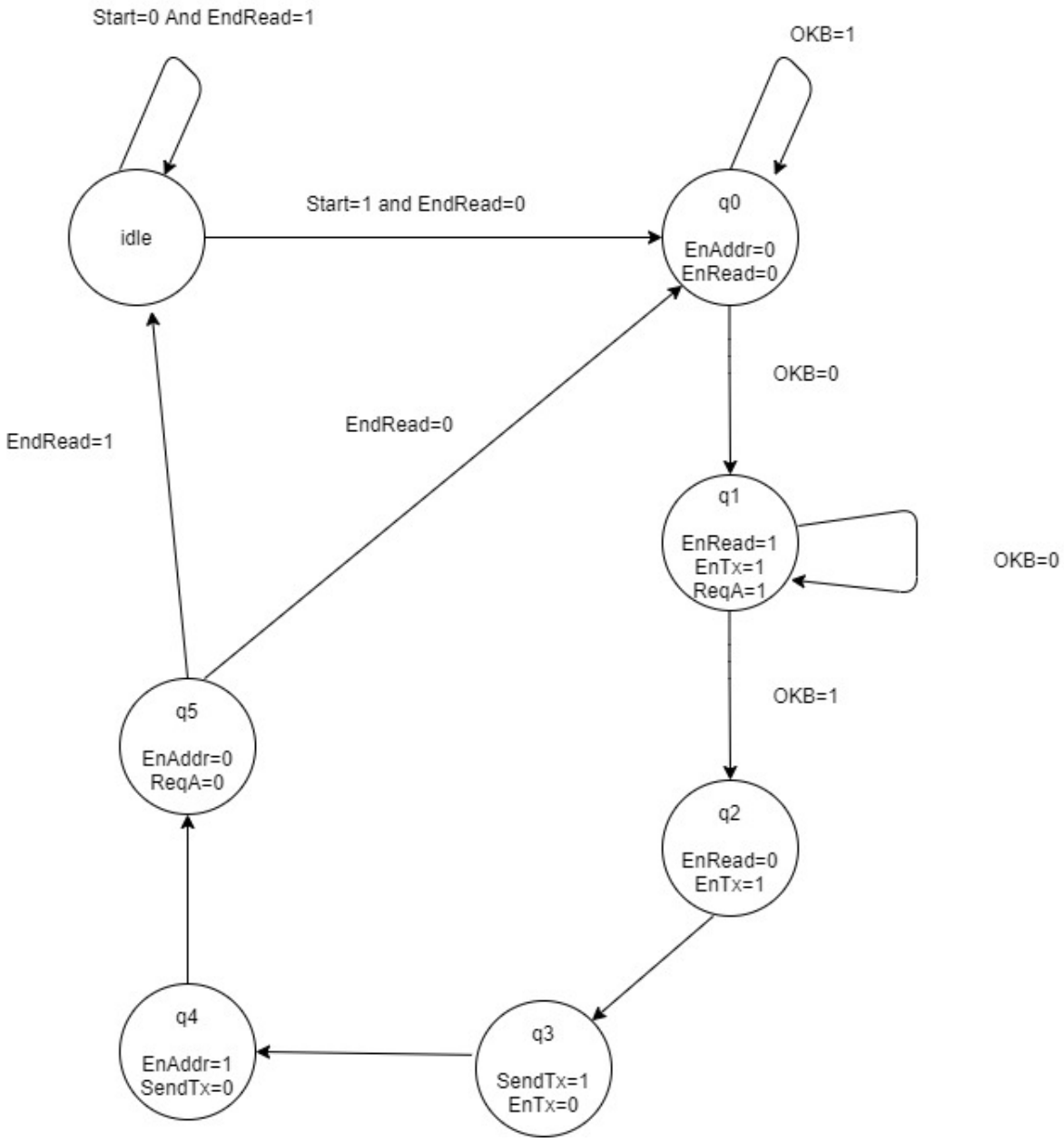

Figure 3.3: AutomaNodoA

Chapter 4

NodoB

4.1 Descrizione

Per quanto riguarda il Nodo B, dopo aver definito il nostro sottrattore strutturale a partire da N sottrattori comportamentali, andiamo a progettarlo. Per quanto riguarda la parte operativa, è previsto un registro RX, il quale riceve il dato dal nodo A(chiaramente dopo che è stato effettuato l'handshake), presentado una duplice funzione, sia di stabilizzare il dato appena ricevuto, sia quello di dividere la stringa in un operando S1 di 4 bit dove considero i più significativi e un operando S2 di 2 bit. Quindi per esempio se avrò in Rx il valore 010010, mi aspetto come S1=0100 ed S2=10. L'uscita di questo Registro Rx, dunque viene separata e salvata negli appositi registri RS1 ed RS2. Per quanto riguarda la stringa Rs1 entrerà prima di arrivare nel registro d'appartenenza in un MUX, poichè il registro appena citato durante

le sottrazioni iterative verrà usato come registro accumulatore, pertanto abbiamo bisogno inazzitutto di precaricarlo al valore indicato dal registro Rx. Analogo è il discorso per quanto riguarda,l'uscita , infatti fintantochè andiamo ad eseguire queste sottrazioni il dato in uscita da Rs1, deve andare in ingresso al sottrattore.Solo al termine delle operazioni di sottrazione verrà salvato nella rispettiva locazione di memoria. Pertanto abbiamo bisogno di un demux, che mi pilota la direzione del mio segnale in uscita da Rs1. In questo schema di progettazione è stato di deciso, di valutare che le operazioni di confronto , tra il valore sottratto OutRs1 ed OutRs2, sarà valutato dalla rete di controllo stessa.Una possibile alternativa , era l'uso di un comparatore, in cui la rete di controllo avrebbe valutato lo stato dei due segnali e agito di conseguenza. Dopo aver precaricato il nostro valore, andiamo ad effettuare le noste sottrazioni. Il ciclo iterativo per definire il risultato durerà fintantochè S2>S1. Dopodichè,attenderemo il prossimo valore da processore. Il sistema ritornerà nello stato di riposo non appena saranno processati tuttti i dati e scritti in memoria.

Analizziamo l'automa, e le operazioni in ogni stato:

- **Idle** - Il nodo B deve essere sempre pronto all'esecuzione di una richiesta di A, quindi ha bisogno di questo stato che ad ogni colpo di clock controlla se ha ricevuto una richiesta, in questo stato non vengono generati segnali di controllo.

- **Q0:** - Lo stato che si occupa di rispondere alle richieste di A e

si pone in ricezione, quindi avremo:

  - **OkA:** viene messo alto per rispondere ad A e quindi realizzare il primo fronte di salita del handshake.

- **Q1** - Questo stato si occupa di salvare il dato appena ricevuto dal nodo A.

  - **WriteRx:** posto alto per salvare il valore nel registro RX.

- **Q2** - Questo stato predispone la lettura dal registro Rx, in cui andiamo a ricarvare le stringhe S1 ed S2, quindi avremo:

  - **WriteRx:** viene posto basso.
  - **ReadRx:** viene posto alto.

- **Q2wait** - In questo stato andiamo a salvare S1 ed S2 nei registri Rs1 ed RS2:

  - **WriteS1,WriteS2:** vengono posti alti.
  - **ReadRx:** viene posto basso.

- **Q3** - In questo stato disabilito la recezione dei registri Rs1 ed Rs2:

  - **WriteS1,WriteS2:** vengono posti bassi.

- **Q4** - Stato cruciale, adiamo a valutare OutS1 ed OutS2 dopo che abbiamo effettuato la sottrazione per valutare il dafarsi.

  - **ReadS1,ReadS2:** abilito Rs1 ed Rs2 in lettura, per mandare i due dati nel sottrattore.
  - **SelMux:** abilitando il selettore del mux , instrado l'informzione dal sottrattore al registro Rs1.

- **Q5a** - Per essere in questo stato vuol dire che S1>=S2, pertando mi predispongo per una nuova sottrazione.Dove al colpo di clock successivo ritorno nello stato Q3.
  - **ReadS1,ReadS2:** li pongo a zero.
  - **WriteS1:** salvo il nuovo valore, che è stato generato dalla sottazione.

- **Q5** - Per essere in questo stato vuol dire che S1<S2.
  - **ReadS1,ReadS2:** li pongo a zero.
  - **WriteS1:** scrivo in Rs1 l'ultimo valore maggiore o uguale a zero ottenuto dalla sottazione.

- **Q6** - Leggo il valore del modulo finale , per predisporlo in ingresso alla memoria.
  - **ReadS1:** lo pongo alto.
  - **WriteS1:** lo pongo basso.
  - **SelDMux:** lo pongo alto in maniera tale da instradare il valore verso la memoria.

- **Q7** - Scivo il dato in memoria.
  - **EnWrite:** lo pongo alto.
  - **ReadS1:** lo pongo basso.
- **Q8** - Aggiorno l'indirizzo di memoria, pilotato da un contatore. E chiudo l'hanshake con il nodo A , ritornando in idle, per la ricezione di un nuovo valore.
  - **EnWrite:** lo pongo basso.
  - **EnAddr:** lo pongo alto e lo azzererò in idle.
  - **OKA:** lo pongo basso, realizzando il fronte di discesa del segnale.

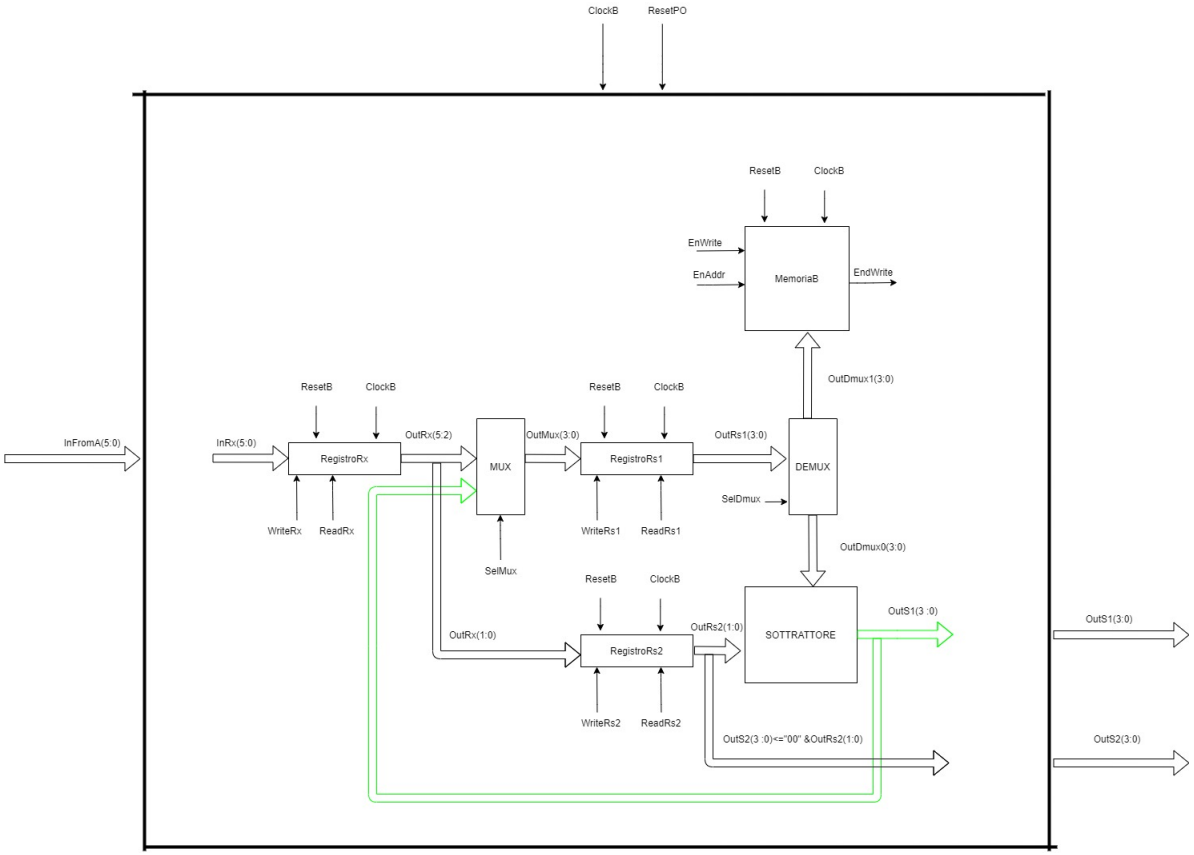

Figure 4.1: Parte Operativa B

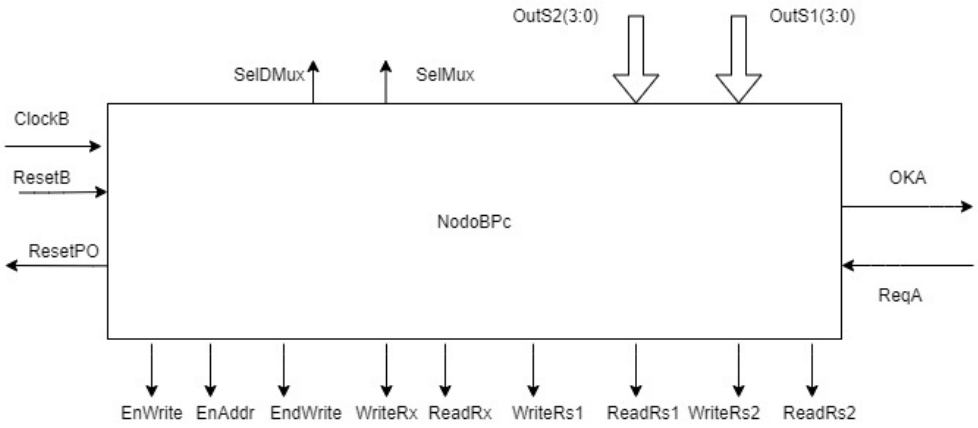

Figure 4.2: Parte Controllo B

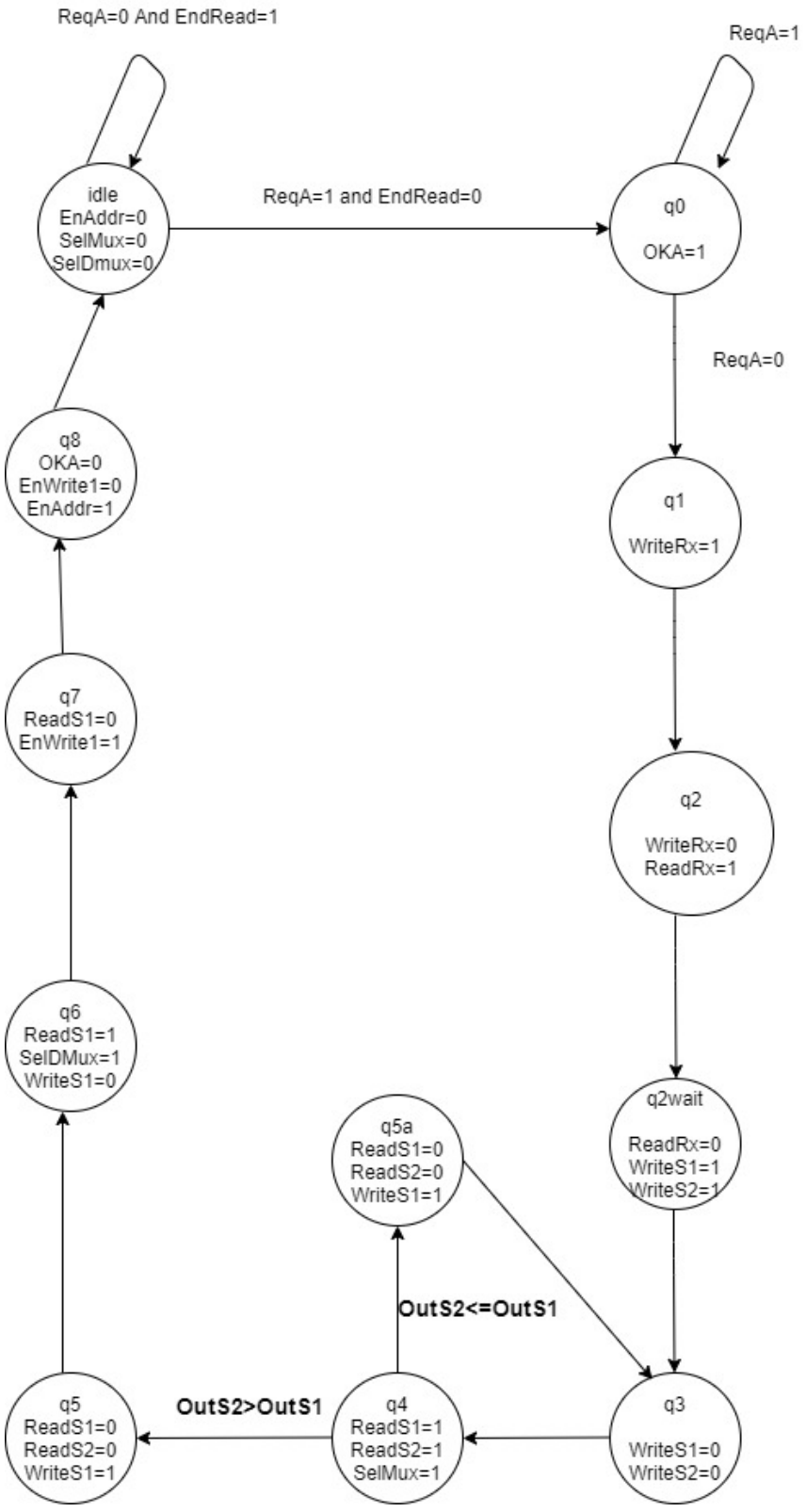

Figure 4.3: Automa B

Chapter 5

Sistema Complessivo

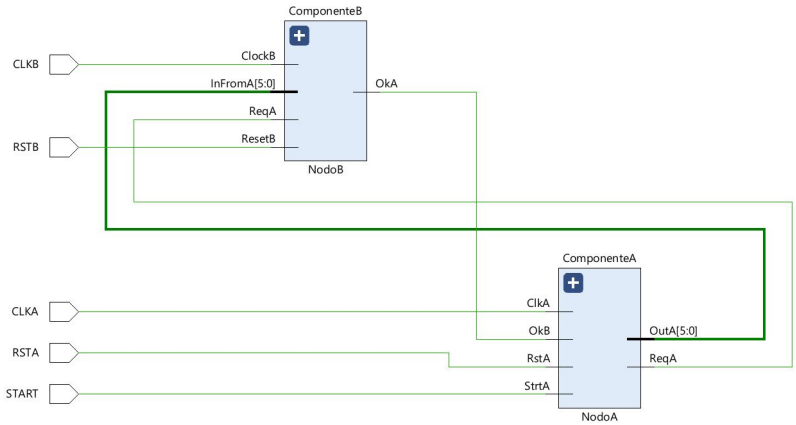

Figure 5.1: Sistema Complessivo

5.1 TestBench

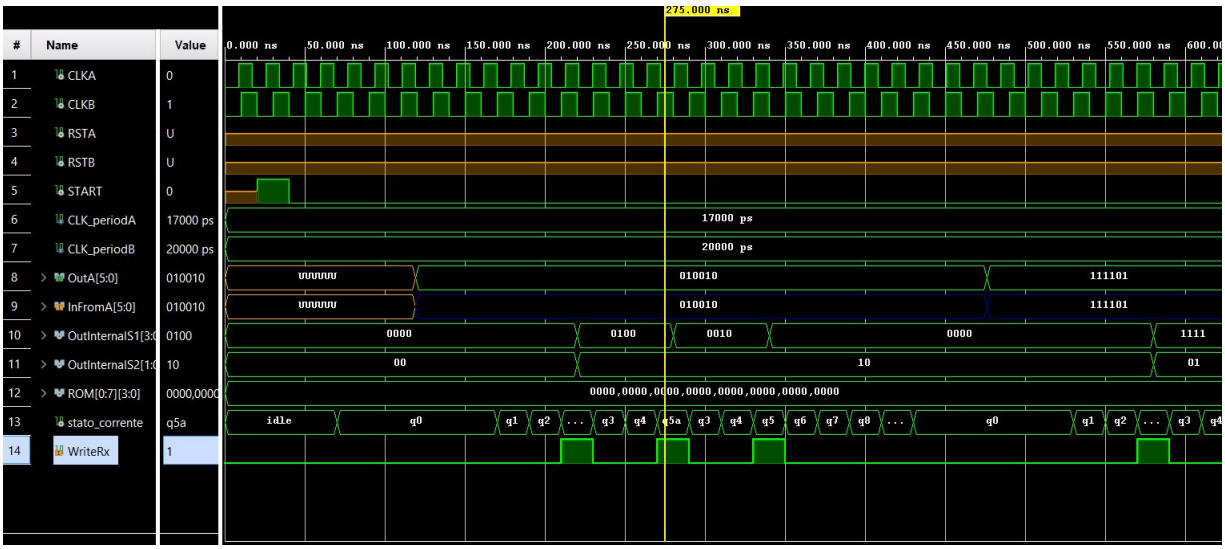

Figure 5.2: TestBench1

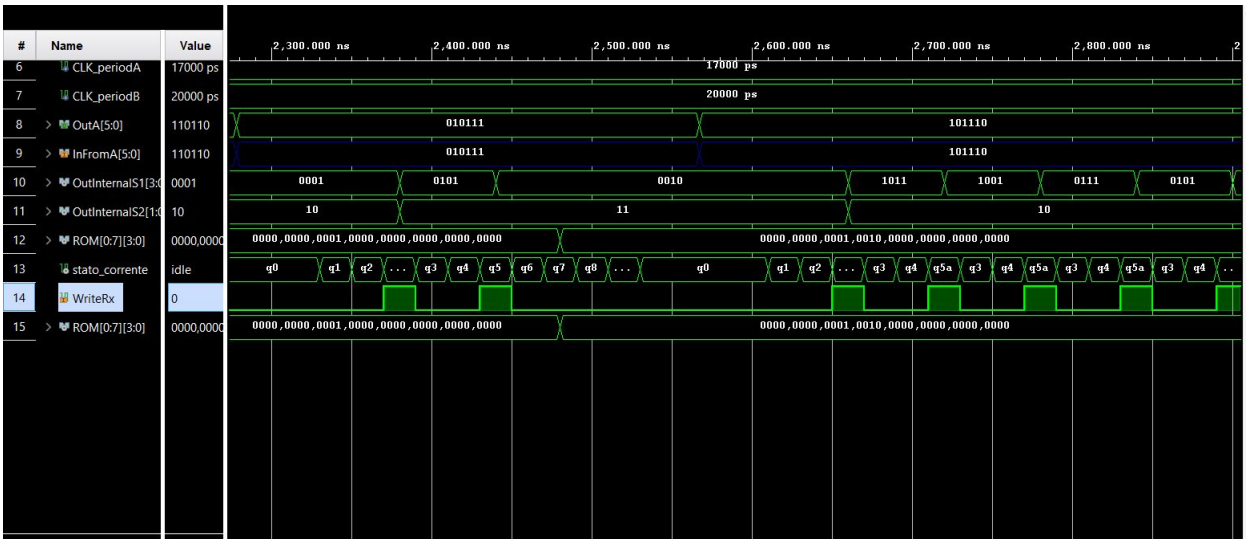

Figure 5.3: TestBench2

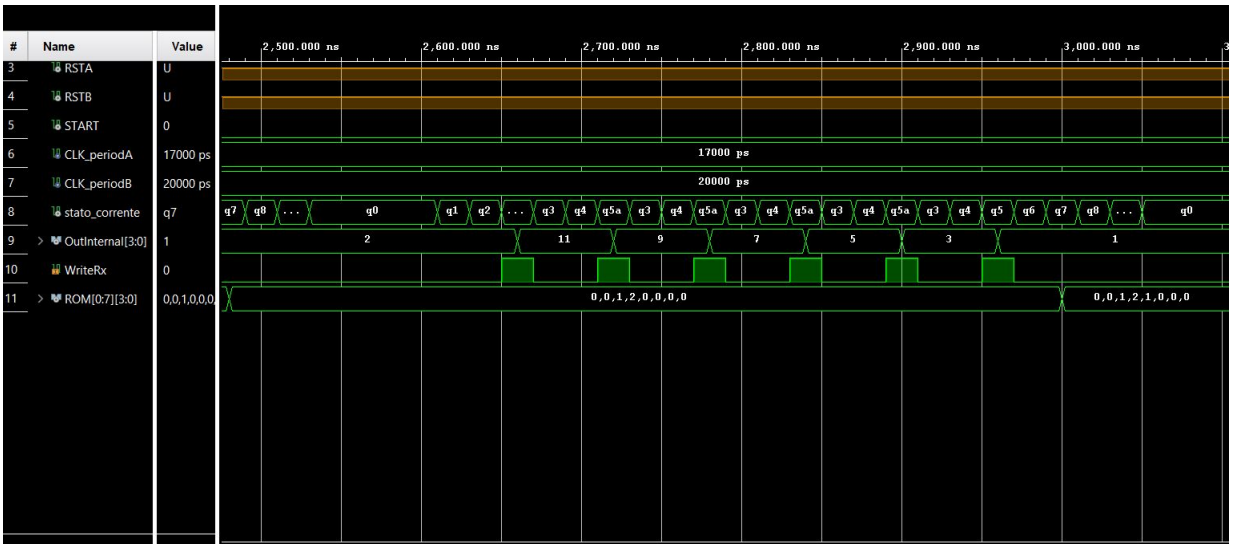

Figure 5.4: TestBench3,valori in decimale

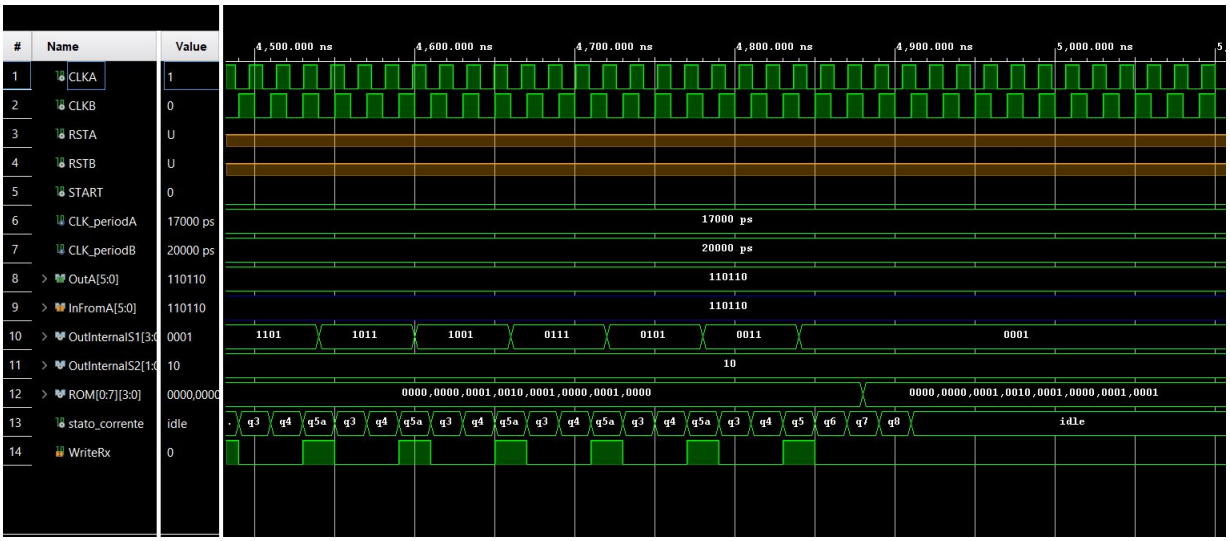

Figure 5.5: TestBench4

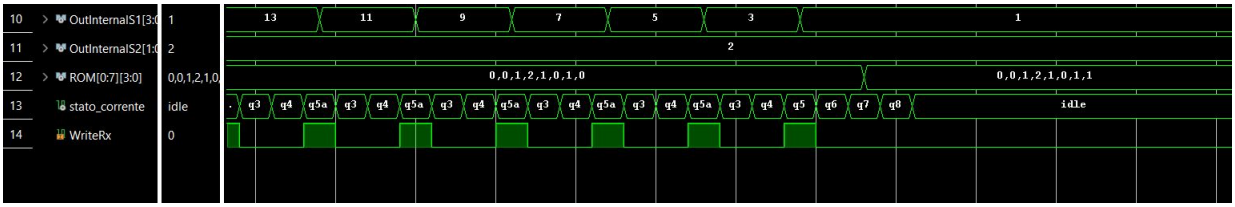

Figure 5.6: TestBench5