



POLITECNICO DI TORINO

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA
ELETTRONICA

**SISTEMI ELETTRONICI
A BASSO CONSUMO**

Progetto 6.2

Atzeni Giacomo 199741
Busignani Fabio 197883

August 3, 2013

Indice

Introduzione	3
Software utilizzato	4
1 Specifiche di interfaccia	5
1.1 Features	5
1.2 Descrizione dei segnali	6
1.3 Timing Diagram	8
1.3.1 Ricezione	8
1.3.2 Trasmissione	9
1.3.3 Reset	9
1.3.4 Ondemand	10
2 Architettura interna	12
2.1 Interfaccia lato sensore	12
2.1.1 Shift Register	12
2.1.2 FSM 1	14
2.1.3 FSM 2	16
2.2 Interfaccia lato controllo	19
2.2.1 Memoria FIFO	19
2.2.2 FSM 1	20
2.2.3 FSM 2	22
3 Codifica dei segnali	24
4 Simulazioni	27
5 Clock Gating	32
5.1 Cenni Teorici	32
5.2 Applicazione del clock gating alla nostra interfaccia	33

5.2.1	Applicazione del clock gating al lato controllo	33
5.2.2	Applicazione del clock gating al lato sensore	37
6	Conclusioni	40

Introduzione

Il rumore è la principale causa di malfunzionamento nei moderni circuiti elettronici digitali. Mentre fino a qualche anno fa si poteva dire che i circuiti digitali fossero immuni (o quasi) al rumore, oggigiorno la situazione non è più così: lo *scaling-down* tecnologico ha comportato non solo una forte diminuzione delle dimensioni dei circuiti integrati, ma uno scalamento anche delle tensioni di alimentazione. Tutto questo ha dunque degradato i margini di immunità al rumore aprendo un mondo nel campo della ricerca che stà cercando di porvi rimedio per poter continuare questo avanzamento tecnologico.

Tuttavia, il nostro progetto non rappresenta uno studio diretto né delle problematiche correlate al rumore, né dei metodi per porvi rimedio. Esso infatti si va ad inserire all'interno di un progetto di monitoraggio del rumore in un chip e ha lo scopo di collegare la parte di controllo alla parte di calcolo, quest'ultima connessa poi coi vari sensori che sono presenti all'interno del chip stesso.

Specifiche di progetto

Per la realizzazione di questo progetto non sono state fornite delle vere e proprie specifiche, ma più che altro si può parlare di linee guida che hanno vincolato le nostre decisioni nelle varie fasi.

La richiesta più stringente che ci è stata fornita è stata quella di realizzare un'interfaccia sincrona che abbia il minor numero di pin esterni possibili, questo si traduce essenzialmente in un vincolo sull'utilizzo di un protocollo di comunicazione che sia di tipo seriale e non parallelo.

Altre scelte e vincoli che sono sorti in fase di progettazione, anche per via di specifiche richieste avanzate dai progetti realizzati parallelamente e che si

intende mettere in comunicazione, verranno esposte nelle pagine successive.

Software utilizzato

Al fine di realizzare questa interfaccia sono stati utilizzati due software differenti:

- ***Modelsim***, per la simulazione dei sorgenti VHDL, nonchè la valutazione della *switching activity* associata ai segnali interni;
- ***Synopsys***, per la sintesi del progetto e per la stima di potenza del nostro sistema.

Capitolo 1

Specifiche di interfaccia

1.1 Features

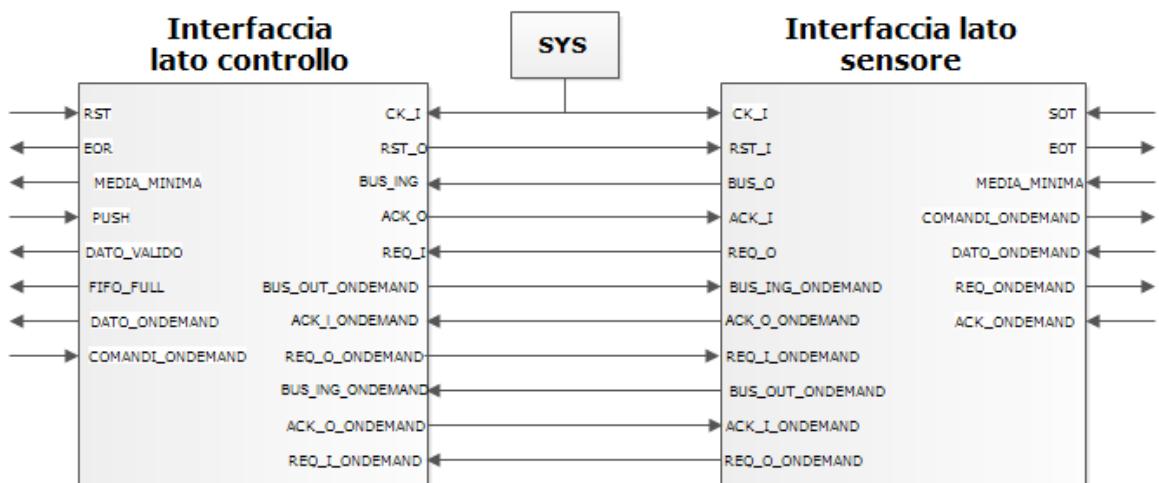


Figura 1.1: Connessione standard dell'interfaccia.

La (Fig.1.1) mostra la connessione standard dell'interfaccia a cui si farà riferimento quando si illustreranno i *timing diagram*, sono stati messi in evidenza dunque anche i nomi dei segnali.

Sempre dalla stessa figura si possono ricavare importanti decisioni prese in fase di progetto e qui esposte.

Clock Come si può notare per il segnale di clock si è scelta l'ipotesi che esso venga fornito esternamente. Tale decisione è coerente con la scelta fatta

circa la distribuzione del clock stesso, che deve essere un processo globale a livello di sistema e non locale a livello di singolo blocco.

Reset Può risultare utile che l’interfaccia di controllo abbia la possibilità di inviare all’interfaccia di calcolo, quella lato sensore, il comando di reset. Tale comando funge da reset di interfaccia, resettando sia la macchina a stati che si occupa del controllo della comunicazione sia i suoi registri interni.

Canali di trasmissione/ricezione Ogni canale di comunicazione prevede tre segnali, due sono dedicati all’*handshake* mentre il terzo è dedicato alla comunicazione seriale. Come si può notare, il numero di canali di trasmissione/ricezione è tre, questo perchè per la stabilità del sistema di controllo è richiesto che un valore medio circa la misura del rumore sia inviato periodicamente. Quindi un canale che va dal lato sensore al lato controllo è costantemente occupato.

Gli altri due canali invece servono uno per inviare le richieste di particolari misure dal controllo all’interfaccia e l’altro per rispondere a tali richieste fornendo il dato corretto.

L’interfaccia descrive quindi una comunicazione *full duplex*, le motivazioni che ci hanno portato a fare questa scelta, non sono da ricercare nelle migliori prestazioni che essa può comportare ma, piuttosto, nel fatto che un bus bidirezionale causa mediamente un maggior consumo di potenza, dovuta a:

- *stato di alta impedenza*, possibilità di causare una caduta di tensione non digitale;
- *collisioni*, oltre a determinare un errore nella trasmissione (recuperabile soltanto con un algoritmo di correzione degli errori, che necessita di un *payload*), causa anche picchi di consumo non trascurabili.

1.2 Descrizione dei segnali

In questa sezione descriviamo i segnali utilizzati nella nostra interconnessione. Iniziamo tenendo in considerazione la parte dell’interfaccia al lato controllo.

CK_I L’ingresso di clock, fornito dal sistema a livello globale, tutti gli eventi all’interno dell’interfaccia sono sincronizzati dallo stesso segnale di clock.

RST_O Uscita che comanda il reset di tutti i registri e macchine a stati dell’interfaccia.

BUS_ING, BUS_ING_ONDEMAND Ingresso seriale dell’interfaccia, nel primo caso la lunghezza del dato ricevuto è di 3 bit, mentre nel secondo caso è di 6 bit.

ACK_O, ACK_O_ONDEMAND Uscita di *acknowledge*, asserita in risposta di una richiesta di ricezione dati.

REQ_I, REQ_I_ONDEMAND Ingresso di *request*, se attivo, indica che l’altro oggetto vuole inviarci qualcosa.

BUS_OUT_ONDEMAND Uscita seriale, il dato trasmesso ha lunghezza pari a 6 bit.

ACK_I_ONDEMAND Ingresso di *acknowledge*, se viene asserito vuol dire che l’altro oggetto sta ricevendo e io continuo con la trasmissione.

REQ_O_ONDEMAND Uscita di *request*, viene asserita qualora si voglia inviare una parola.

RST Ingresso fornito dall’unità di controllo e rappresenta il segnale di reset che sarà propagato anche alla parte d’interfaccia posta sul lato opposto.

EOR Segnale di uscita che indica la fine della ricezione del dato *MEDIA_MINIMA*.

MEDIA_MINIMA Uscita con parallelismo pari a 3 bit, indica il valore di media minima.

PUSH Ingresso, va asserito per poter caricare nella coda dei comandi da inviare quello presentato attualmente all’ingresso *COMANDI_ONDEMAND*.

DATO_VALIDO Uscita, quando asserita indica all’unità di controllo che il dato precedentemente richiesto attraverso la trasmissione di un comando è disponibile all’uscita *DATO_ONDEMAND*.

DATO_ONDEMAND Uscita a parallelismo pari a sei bit dove vengono caricati i dati richiesti dall’unità di controllo per renderli disponibili alla stessa unità.

COMANDI_ONDEMAND Ingresso a parallelismo pari a sei bit pilotato dall’unità di controllo che lo usa per caricare i comandi che devono essere trasferiti all’unità di calcolo posta al lato sensore.

Partendo da queste conoscenze dei segnali al *lato controllo*, la quasi totalità dei segnali che caratterizzano l’altro lato risultano facilmente intuibili e quindi, per evitare ridondanze, verranno descritti solamente due segnali:

REQ_ONDEMAND Segnale d'uscita, viene asserito una volta che è stato ricevuto il comando proveniente dall'unità di controllo e caricato su *COMANDI_ONDEMAND*. Il segnale viene negato solo una volta che l'unità di calcolo ha asserito *ACK_ONDEMAND*.

ACK_ONDEMAND Segnale d'ingresso pilotato dall'unità di calcolo, indica all'interfaccia che il dato richiesto è stato caricato sull'ingresso *DATO_ONDEMAND* e che è dunque pronto ad essere trasferito.

1.3 Timing Diagram

Di seguito vengono riportati i *timing diagram* che evidenziano le varie forme d'onda nelle differenti operazioni.

1.3.1 Ricezione

Il *timing diagram* inerente alla ricezione è mostrato in (Fig.1.2), dove si possono notare alcune proprietà.

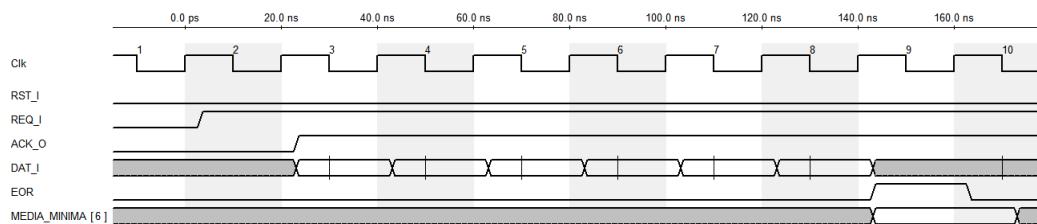


Figura 1.2: Timing diagram della ricezione.

Innanzitutto si nota che per quanto riguarda i segnali di *acknowledge* e *request* anziché renderli validi a livello, sono stati resi validi alla transizione, ossia l'asserzione di uno di questi segnali è dato dall'inversione del suo valore logico. Questa scelta, ininfluente dal punto di vista funzionale, ha però una discreta valenza dal punto di vista energetico in quanto si dimezzano le commutazioni legate a queste due linee e quindi si diminuisce la potenza dinamica associata alla carica/scarica della capacità equivalente dei due fili. Una seconda proprietà che viene messa in evidenza è che attivata la linea di *request*, il ricevitore, al colpo di clock successivo, ha il compito di asserire la linea di *acknowledge*, con lo scopo di fornire a chi trasmette un *feedback* positivo. Si nota infine che chi riceve campiona il segnale sul *bus* in corrispondenza del fronte di discesa del clock, questo viene fatto perché chi trasmette scrive il dato sul fronte positivo del segnale di sincronismo e quindi

questa scelta ci permette di evitare che il dato rimanga scritto per 2 colpi di clock.

1.3.2 Trasmissione

La trasmissione è facilmente intuibile a partire dal *timing diagram* precedente, tuttavia, per completezza, riportiamo in (Fig.1.3) le forme d'onda della trasmissione del valore di media minima che avviene dall'interfaccia lato sensore a quella lato controllo.

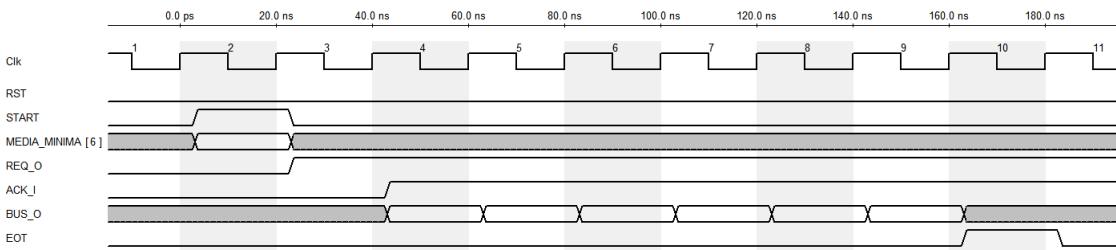


Figura 1.3: Timing diagram della trasmissione.

In (Fig.1.4) si mostra il *timing diagram* della trasmissione del dato qualora il *feedback* legato all'*acknowledge* non sopraggiungesse.

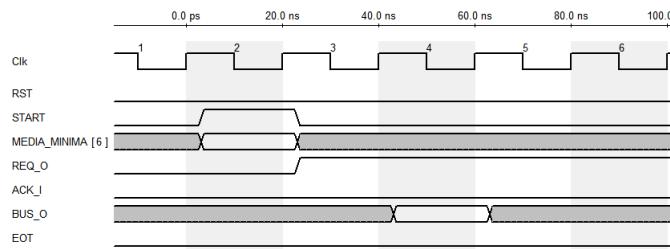


Figura 1.4: Timing diagram della trasmissione con la mancata asserzione dell'acknowledge.

Osservando la (Fig.1.4) si può notare che, per come è stato definito il protocollo, il trasmettitore invia comunque il primo *bit*, poi, prima di inviare il secondo va a verificare lo stato dell'*acknowledge* e, qualora non fosse stato asserito, solo allora blocca la trasmissione.

1.3.3 Reset

Continuiamo illustrando il *timing diagram* in una ricezione interrotta dal segnale di reset in ingresso (Fig.1.5).

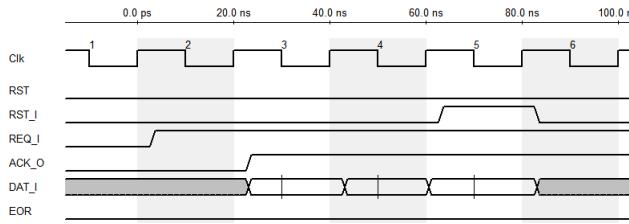


Figura 1.5: Timing diagram della ricezione interrotta dal comando di reset.

Come è stato precedentemente detto, questo segnale non si limita a resettare la macchina a stati che si occupa di coordinare l'intera interfaccia ma comporta pure il reset dei registri interni ad essa.

1.3.4 Ondemand

Concludiamo questa sezione esponendo il *timing diagram* che caratterizza una comunicazione *ondemand* vista dalla parte di interfaccia connessa all'unità di calcolo.

Come si evince dalla (Fig.1.6), una volta ricevuto il dato esso viene fornito in ingresso all'unità di calcolo e viene asserito il segnale *REQ_ondemand* per indicare la presenza di un nuovo comando. L'unità di calcolo campionerà poi il comando e risponderà alla richiesta impiegando un numero di colpi di clock a noi sconosciuto. Una volta che ha calcolato il dato richiesto lo rende disponibile all'ingresso *Dato_ondemand* dell'interfaccia e, per indicare che ciò è avvenuto asserisce per un colpo di clock il segnale *ACK_ondemand*. Prezzo atto di ciò, solo allora l'interfaccia negherà il segnale di *REQ_ondemand*, riportandolo al valore logico basso. A questo punto, si procede con la trasmissione del dato fornito. La trasmissione può essere o a tre bit o a sei bit, tale decisione viene presa sulla base del comando precedentemente inviato e memorizzato all'interno dell'interfaccia.

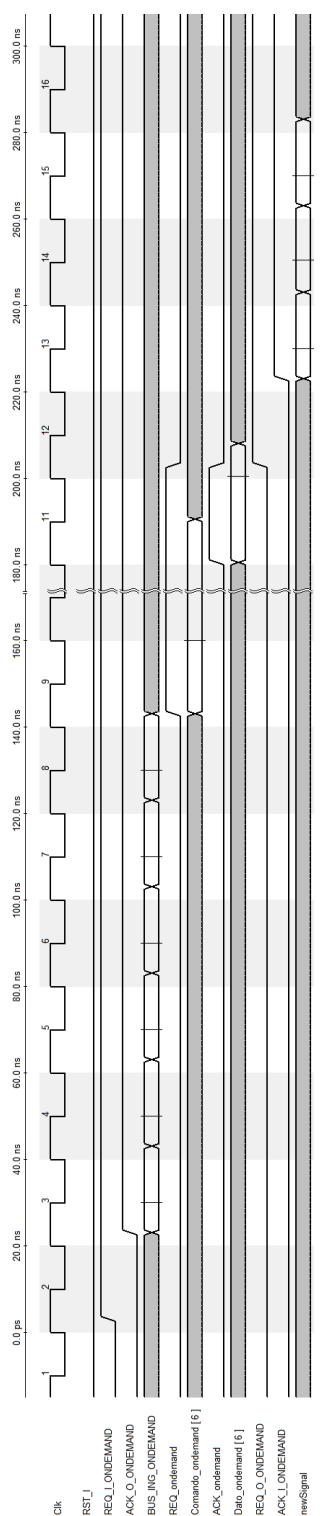


Figura 1.6: Timing diagram della comunicazione ondemand.

Capitolo 2

Architettura interna

Dopo aver descritto il protocollo di comunicazione tra i due lati dell’interfaccia, occorre illustrare l’architettura interna dell’interfaccia stessa e introdurre i segnali con la quale essa è in grado di interfacciarsi con l’unità di controllo e con quella di calcolo.

2.1 Interfaccia lato sensore

Questo lato dell’interfaccia si trova nella parte di calcolo, ossia la parte connessa coi vari sensori presenti nel chip ed incaricata di elaborare le varie misure effettuate.

L’architettura che la descrive è illustrata nello schema a blocchi di (Fig.2.1), dove per non appesantire troppo la figura non sono stati trascritti tutti i nomi dei segnali: ogni registro e latch presente viene connesso ad una macchina a stati mediante due segnali che sono l’*enable* e il *reset*.

Descriviamo ora la funzione che ogni blocco di (Fig.2.1).

2.1.1 Shift Register

Una conseguenza della scelta di trasmettere e ricevere i dati in modo seriale è stata quella di dover introdurre degli *shift registers*.

Per quanto riguarda la fase di ricezione, ogni qualvolta un bit viene ricevuto dal bus questo viene immagazzinato in un *flip flop* dello *shift register*, mentre i restanti bit ricevuti precedentemente vengono shiftati di una posizione ogni volta. In questo modo, alla fine della trasmissione, lo *shift register* contiene il dato completo e ordinato dal *MSB* al *LSB*. Dopodichè il dato è pronto per essere trasferito nel *latch* e successivamente fornito in ingresso all’unità di

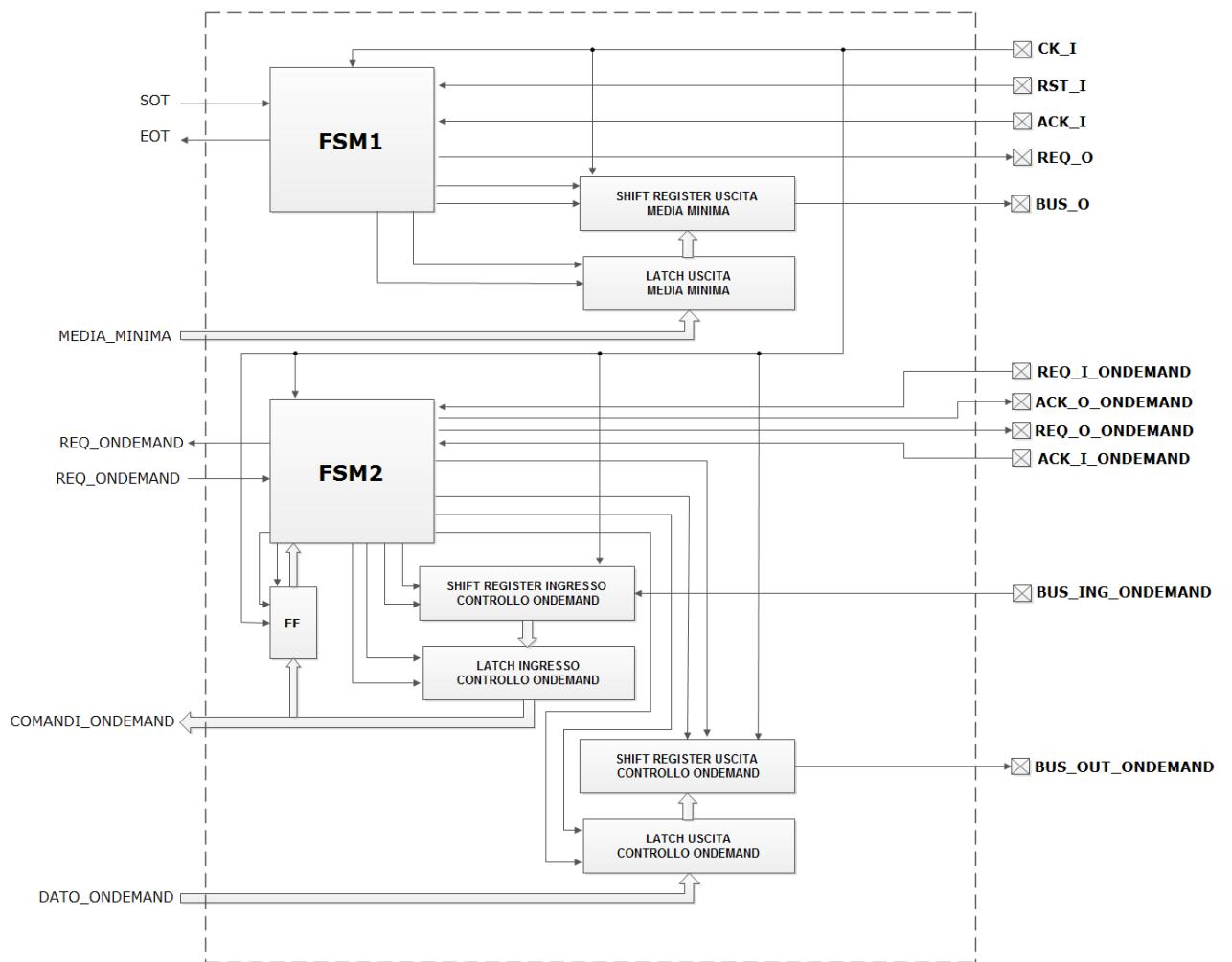


Figura 2.1: Architettura dell’interfaccia lato sensore.

calcolo. Il *latch* è stato inserito per evitare che la commutazione dei segnali all'interno dello *shift register* in fase di ricezione si propaghi all'interno dell'interfaccia inutilmente. Quindi, questo *latch* serve per diminuire la potenza dinamica richiesta dall'interfaccia.

La situazione di invio dei dati è molto simile. In tal caso i bit che compongono il dato vengono caricati parallelamente nello *shift register*, passando prima per un *latch* che ha lo scopo di implementare la tecnica *guarded evaluation*, ossia evita che eventuali transizioni dei bit del dato non si propaghino all'interno dell'interfaccia se questo dato non deve essere inviato in quel momento. Successivamente, quando la trasmissione viene avviata, lo *shift register* viene abilitato e invia sul bus di uscita un bit per colpo di clock, shiftando il suo contenuto in ogni colpo di clock sino al termine della trasmissione.

Le tipologie di Shift Register che sono state utilizzate nel progetto sono dunque il Parallel-Input Serial-Output e lo Serial-Input Parallel-Output Shift Register.

2.1.2 FSM 1

La macchina a stati che abbiamo chiamato *FSM1* è quella incaricata di coordinare la trasmissione del dato chiamato che trasporta l'informazione sulla media minima. Come già detto precedentemente, questo dato deve essere inviato costantemente, al fine di garantire la stabilità all'unità di controllo. Il funzionamento di questa macchina è descritto dal suo diagramma degli stati, mostrato in (Fig.2.2).

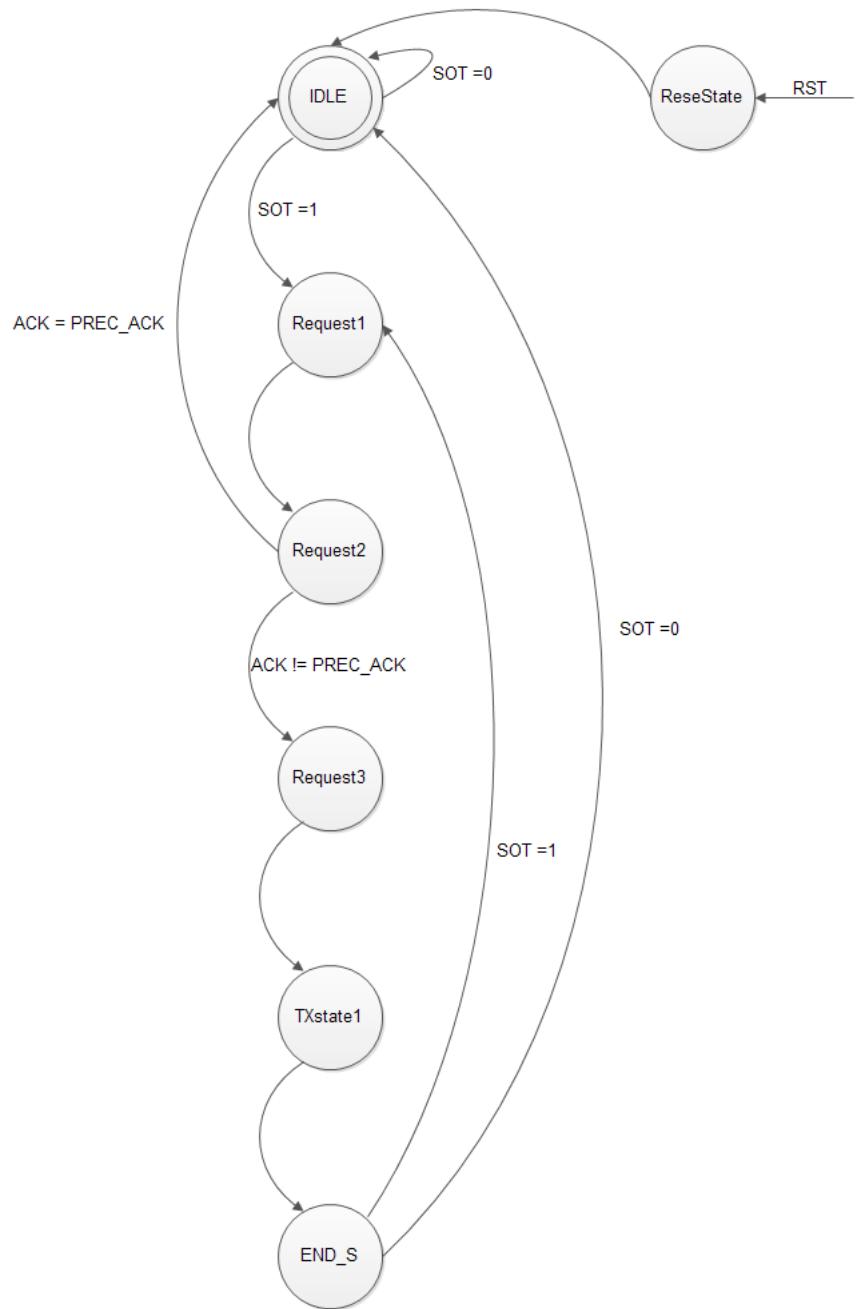


Figura 2.2: Macchina a stati della trasmissione media minima, lato sensore.

Da questo diagramma degli stati si può vedere che il segnale di trigger, ossia quello che determina lo spostamento dallo stato di riposo è *SOT* (Start Of Transmission).

Nello stato *Request1* la macchina comanda il campionamento nello *shift register* del dato da inviare. Nei successivi 3 stati il dato viene inviato per via seriale, bit per bit. Nell'ultimo stato si indica la fine della trasmissione all'unità di calcolo asserendo per un colpo di clock il segnale di *EOT* (End Of Transmission). Sempre dalla (Fig.2.2) si riesce a vedere la funzione del segnale di *acknowledge* spiegata in (Sec.1.3.2) e come sia possibile effettuare più trasmissioni ripetute saltando lo stato di *IDLE*.

Per via della forte sequenzialità che lega gli stati della macchina, una tecnica *low-power* che può essere facilmente utilizzata è quella della **codifica degli stati**. Essa prevede che gli stati con maggior probabilità di transizione abbiano una codifica tale da minimizzare la distanza di *Hamming*. Questo con lo scopo di diminuire la switching activity della parte che si occupa della memorizzazione dello stato.

La (Tab.2.1) mostra dunque la codifica fatta al fine di minimizzare la potenza consumata da questa macchina a stati.

STATO	CODIFICA
RESET	101
IDLE	100
Request1	000
Request2	001
Request3	011
TXsate1	111
END_S	110

Tabella 2.1: Codifica degli stati di FSM 1 al lato sensore

In questo modo, qualora si dovesse trasmettere un dato per compiere un ciclo completo, indipendentemente dal fatto che questo ciclo sia quello breve (più trasmissioni in sequenza) o meno, i bit che commutano sono 6.

2.1.3 FSM 2

La seconda macchina a stati risulta un po' più complessa della prima ed è colei che coordina la ricezione del comando e la trasmissione della misura richiesta.

Il funzionamento di questa macchina è descritto dal suo diagramma degli stati, mostrato in (Fig.2.3).

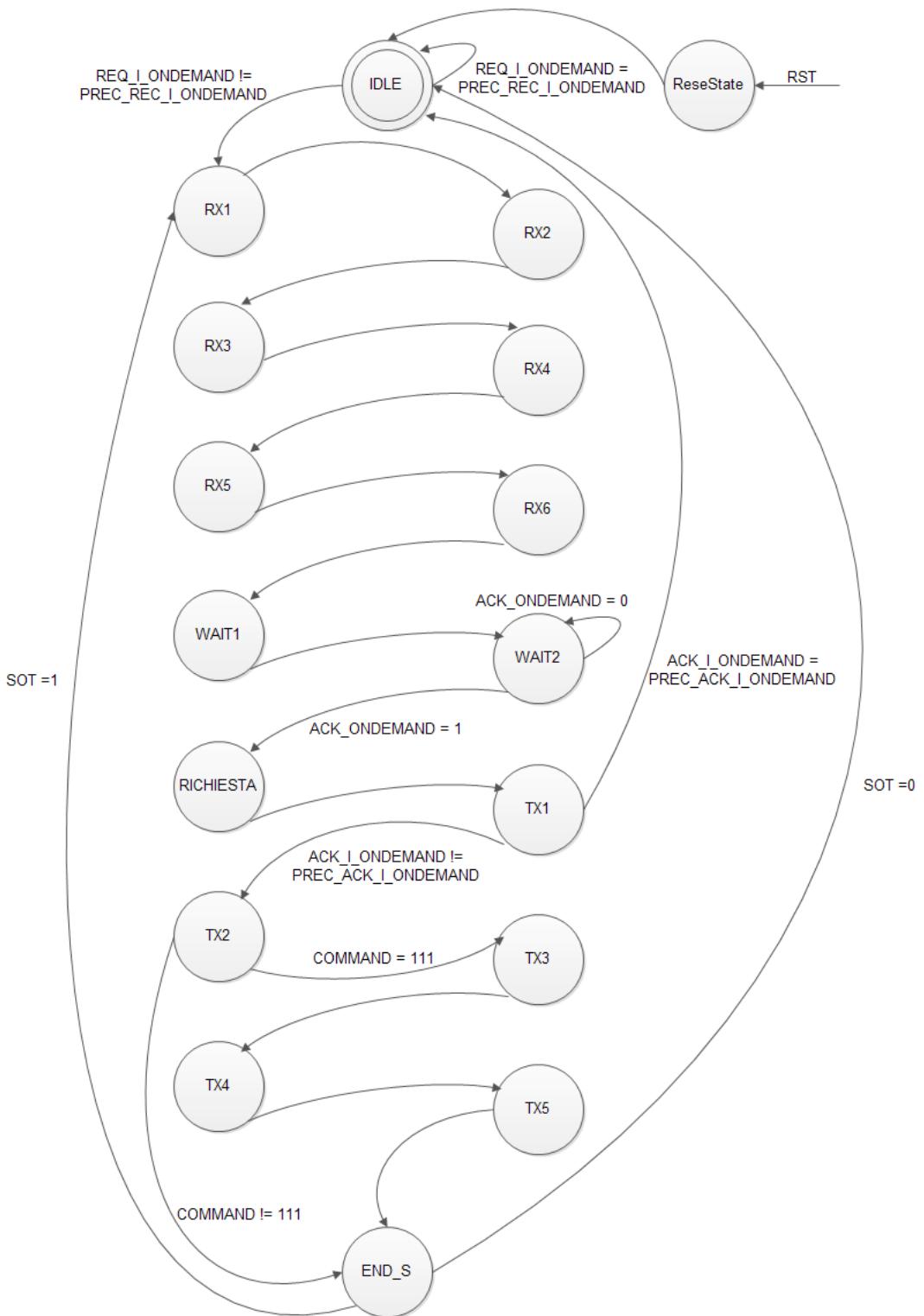


Figura 2.3: Macchina a stati della trasmissione ondemand, lato sensore.

Facendo riferimento a quanto già detto in (Sez.1.3.4) risulta facile capire il funzionamento di questa *FSM*. Tutto inizia con l'asserzione della richiesta di ricezione dati inviata dall'interfaccia al lato controllo, il segnale *REQ_L_ONDEMAND* sveglia la macchina a stati rendendola in grado di ricevere i sei bit che compongono il segnale di comando. Successivamente, dopo aver fornito all'unità di calcolo il comando, la *FSM* si porta in uno stato di *wait* dove aspetta il tempo necessario impiegato dall'unità di calcolo per eseguire la richiesta. Successivamente si procede con la trasmissione dei dati che può essere a 6 o 3 bit a seconda del comando precedentemente inviato e memorizzato all'interno del *flip flop* visibile in (Fig.2.1).

Anche in questo caso la natura sequeziale della nostra macchina a stati rende efficace una corretta codifica degli stati.

La (Tab.2.2) mostra dunque la codifica fatta al fine di minimizzare la potenza associata alla memorizzazione dello stato.

STATO	CODIFICA
RESET	00000
IDLE	00001
RX1	00011
RX2	01011
RX3	01111
RX4	11111
RX5	11101
RX6	11001
WAIT1	11011
WAIT2	10011
RICHIESTA	10001
TX1	10101
TX2	10100
TX3	11100
TX4	11000
TX5	10000
END_S	10010

Tabella 2.2: Codifica degli stati di FSM 2 al lato sensore

In questo modo, nel caso di una trasmissione a 6 bit, le commutazioni dovute alla memorizzazione dello stato compiute in un intero ciclo sono 17, mentre nel caso di trasmissione a 3 bit sono 15.

2.2 Interfaccia lato controllo

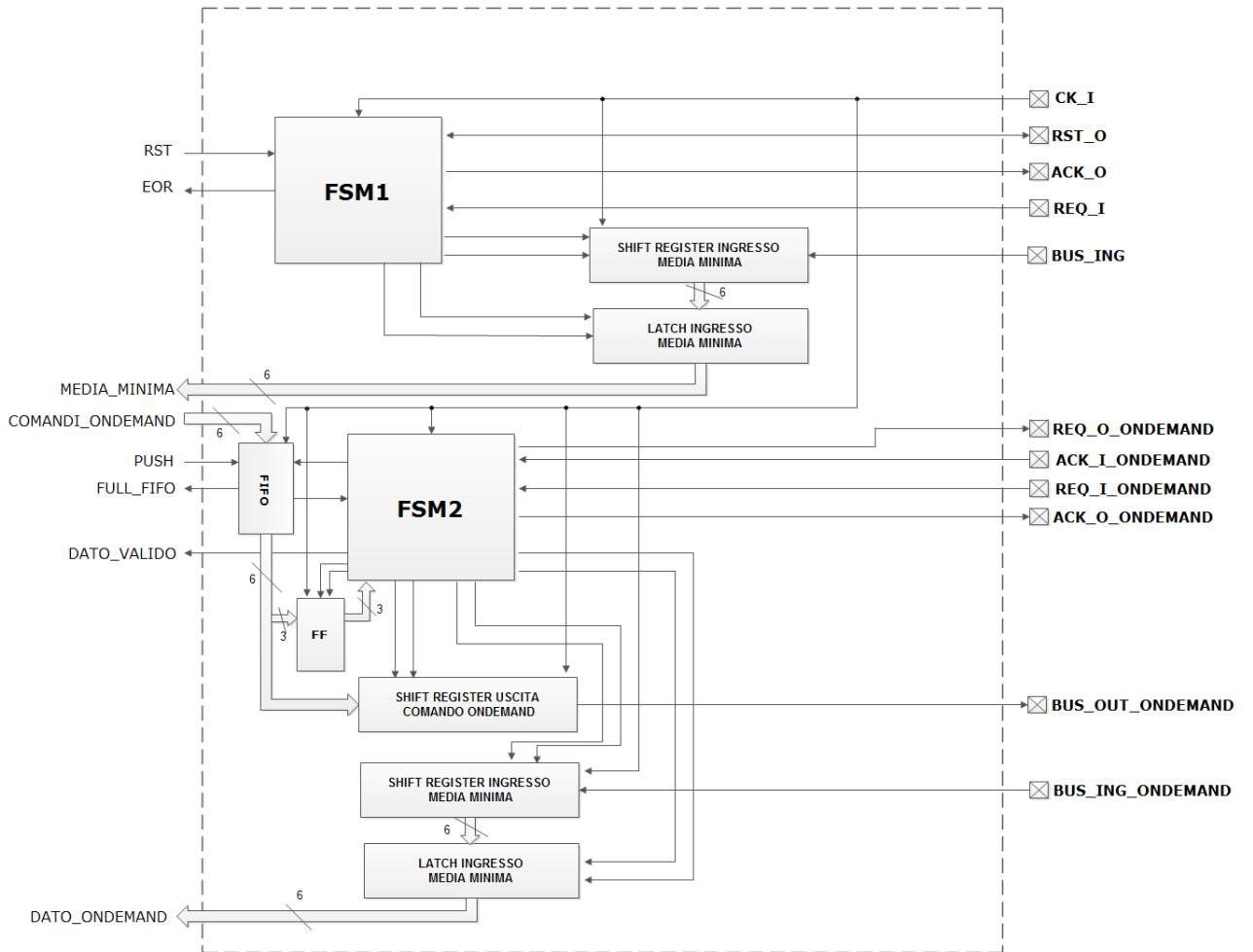


Figura 2.4: Architettura dell’interfaccia lato controllo.

L’architettura che la descrive l’interfaccia al lato controllo è illustrata nello schema a blocchi di (Fig.2.4), si osserva subito che molti componenti sono gli stessi descritti precedentemente e quindi non verranno ridecritti.

2.2.1 Memoria FIFO

L’inserimento della memoria *FIFO* è stato necessario per poter gestire la parte di trasmissione dati su richiesta (*on demand*). Infatti, come si sarà capito dalle spiegazioni precedenti, è necessario servire una richiesta per volta e, onde evitare errori dovuti ad un errato sincronismo delle richieste si è

fatto uso di una *FIFO* per la gestione della coda delle richieste. Siccome la creazione di una coda di dati da trasmettere è in qualche modo un evento che in questo tipo di progetto non dovrebbe verificarsi, è stato deciso di fare la *FIFO* profonda quattro parole, dimensione non eccessiva che non dovrebbe influire troppo né sui consumi, né sulla dimensione del dispositivo. Una scelta progettuale è stata realizzare la *FIFO* di tipo circolare. Questo significa che quando la memoria è piena (e si ha dunque che l'uscita *FULL_FIFO* è asserita), un successivo ciclo di scrittura andrà a sovrascrivere l'elemento più vecchio contenuto all'interno della *FIFO* stessa.

2.2.2 FSM 1

In maniera assolutamente duale a quanto descritto in precedenza, la macchina a stati *FSM1* è incaricata di coordinare la ricezione della media minima. Il diagramma degli stati di questa *FSM* è illustrato in (Fig.2.5).

Il diagramma degli stati evidenzia come in questo sia il segnale di *REQ_I*, la causa del risveglio della macchina a stati. Nello stato *Ric1* la *FSM* si occupa di asserire il segnale di *acknowledge* e di abilitare lo *shift register* d'ingresso, in modo tale da riuscire a campionare il primo bit in arrivo dal bus. I successivi due stati servono per la ricezione degli ultimi due bit, lo stato *RXstate2* invece serve per abilitare il latch di ingresso e rendere disponibile il dato contenente l'informazione sulla media minima all'ingresso dell'unità di controllo. Infine lo stato *END_S* serve per asserire il segnale di *EOR* (End Of Receive) e riportare il latch di ingresso in modalità memorizzazione.

Si nota inoltre che anche in questo caso è possibile effettuare letture ripetute bypassando lo stato di *IDLE*.

Con le stesse premesse fatte per le due macchine a stati precedenti illustriamo attraverso la (Tab.2.3) la codifica degli stati.

STATO	CODIFICA
RESET	101
IDLE	100
Ric1	000
Ric2	001
Rxstate1	011
Rxstate2	111
END_S	110

Tabella 2.3: Codifica degli stati di FSM 1 al lato controllo

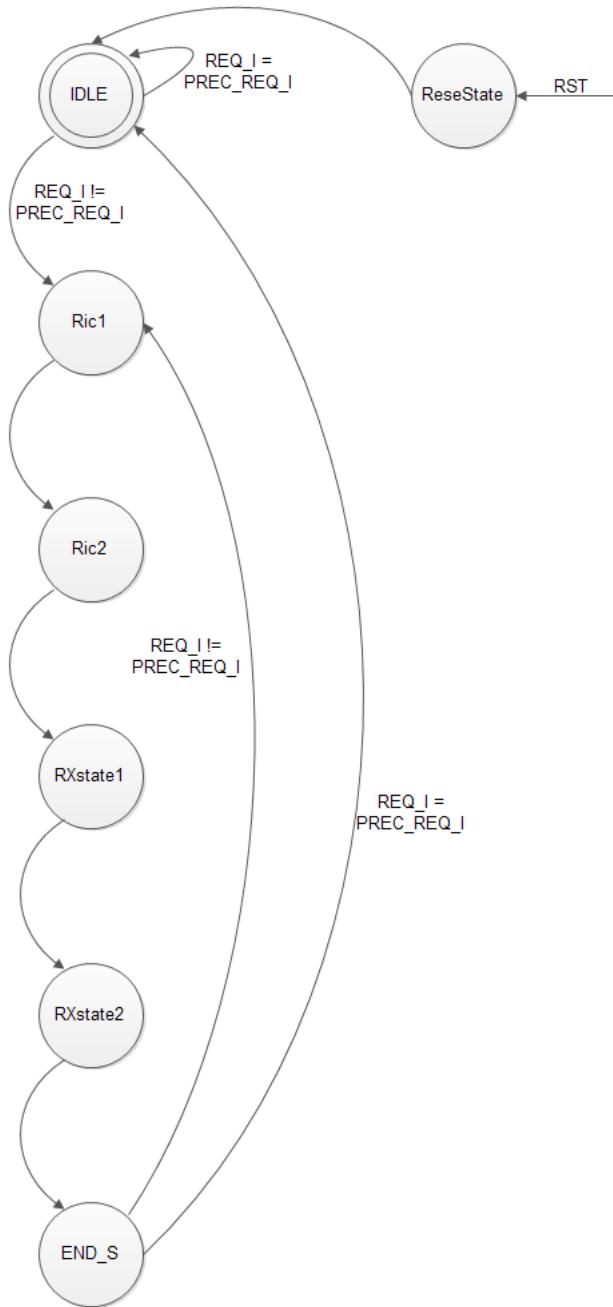


Figura 2.5: Macchina a stati della trasmissione media minima, lato controllo.

Così, come è accaduto per la sua gemella, sia in caso di ricezione singola, sia in caso di ricezioni ripetute, le commutazioni legate alla memorizzazione dello stato saranno 6.

2.2.3 FSM 2

Infine, la (Fig.2.6) mostra il diagramma degli stati del *FSM* che comanda tutte le operazioni necessarie per la comunicazione ondemand vista dal lato controllo.

Si può notare come la macchina stia in stato di riposo per tutto il tempo in cui la *FIFO* in ingresso è vuota (segnale *EMPTY* asserito), una volta che almeno un comando è stato inserito nella coda di ingresso allora inizia la trasmissione passando per lo stato di *RICHIESTA* dove oltre ad avanzare all'altro lato dell'interfaccia la richiesta di avviare una trasmissione agendo sull'apposita linea di *request*, esegue una *pop* del dato dalla memoria *FIFO* e lo inserisce all'interno dello *shift register* d'uscita. I restanti stati di trasmissione, attesa e ricezione sono assolutamente identici a quelli visti sinora.

Nella (Tab.2.4) viene mostrata la codifica degli stati inerente a questa *FSM*.

STATO	CODIFICA
RESET	00000
IDLE	00001
RICHIESTA	00011
TX1	00010
TX2	00110
TX3	00111
TX4	00101
TX5	01101
TX6	01001
WAIT1	01011
WAIT2	01111
RX1	11111
RX2	10111
RX3	10011
RX4	11011
RX5	11001
RX6	10001
END_S	10000

Tabella 2.4: Codifica degli stati di FSM 2 al lato sensore

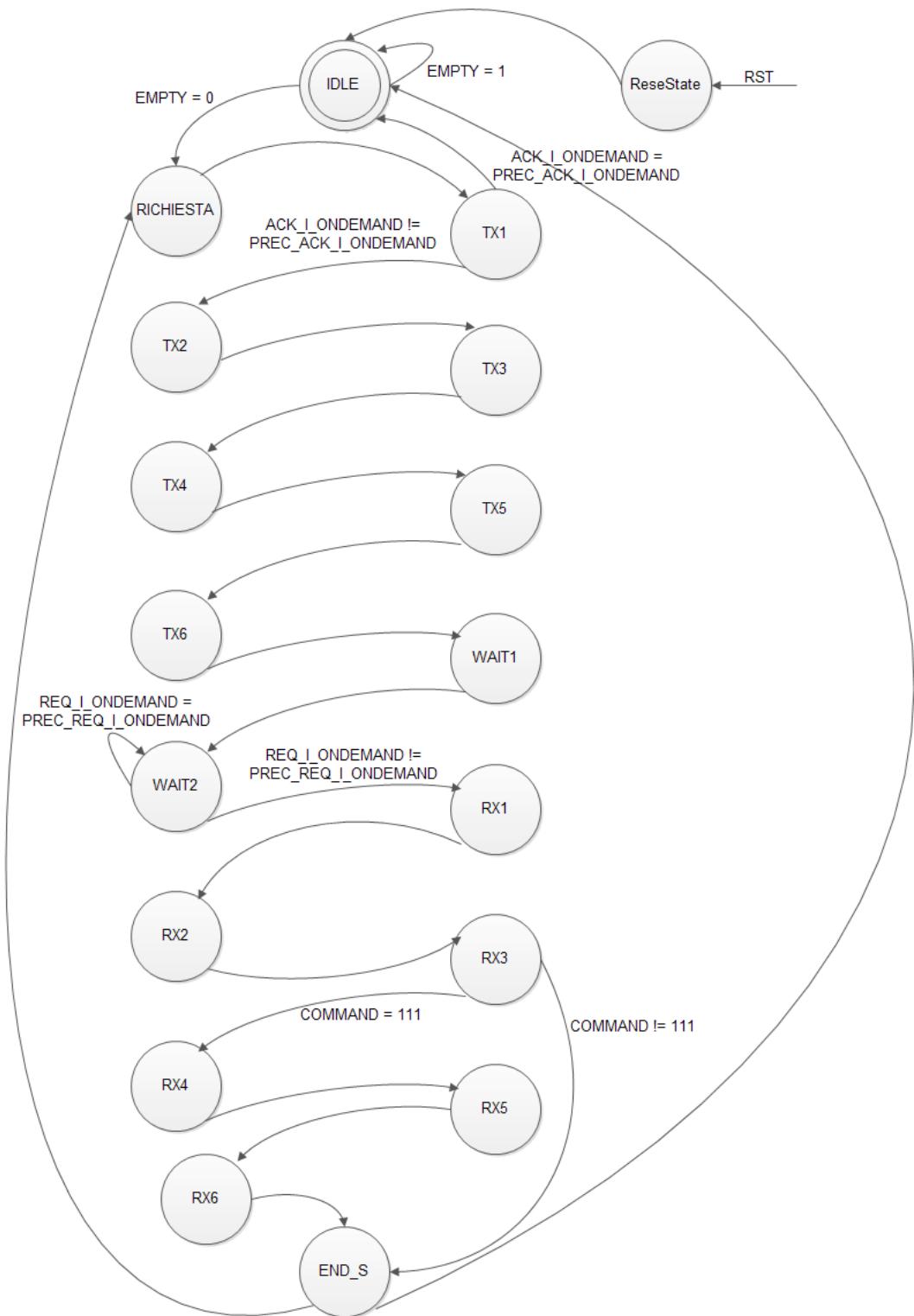


Figura 2.6: Macchina a stati della trasmissione ondemand, lato controllo.

Capitolo 3

Codifica dei segnali

Per quanto riguarda la codifica dei segnali sono state fatte alcune ipotesi accordate anche con gli altri progetti.

Innanzitutto, vista la natura seriale dell'interfaccia, le tecniche di codifica dei bus viste durante il corso di *Sistemi Elettronici a Basso Consumo* non sono applicabili. E quindi, nell'ottica di minimizzare i consumi nessuna decisione è stata qui presa.

La codifica riguarda solo i comandi che l'unità di controllo invia all'unità di calcolo attraverso il canale di trasmissione che abbiamo chiamato ondemand. Questo perchè le misure naturalmente non dipenderanno da noi ma dal sensore.

All'interno della *word* di comando devono essere contenute 2 informazioni distinte:

1. **La richiesta**
2. **Il sensore al quale si riferisce la richiesta**

Per quanto riguarda il primo punto, abbiamo sei tipi di richiesta differenti e, quindi sono necessari tre bit di codifica. La codifica delle richieste è illustrata in (Tab:[3.1](#)).

RICHIESTE	CODIFICA
Media lunga su 16 valori	000
Media lunga su 32 valori	001
Media lunga su 64 valori	010
Valore massimo	100
Valore minimo	110
Valore RMS	111

Tabella 3.1: Codifica delle richieste

Per quanto riguarda il secondo punto invece si ipotizza di fornire ai vari sensori un indirizzo fisso in fase di progetto. Inoltre si è ritenuto che 7 potesse essere un buon numero, veritiero, di sensori. Quindi in questo caso si fa riferimento alla (Tab:3.2).

SENSORE	INDIRIZZO
I SENSORE	000
II SENSORE	001
III SENSORE	010
IV SENSORE	011
V SENSORE	100
VI SENSORE	101
VII SENSORE	110

Tabella 3.2: Codifica degli indirizzi dei sensori

Quindi, queste due parole da tre bit ciascuna, vengono trasferite assieme in un'unica *word* da sei bit seguendo l'ordine mostrato in (Fig.3.1).



Figura 3.1: Comando, divisione interna della parola.

Ora possiamo finalmente spiegare perchè si hanno casi in cui la misura trasmessa in risposta alla ricezione di un comando è pari a 3 e altri invece in cui è pari a 6. Le misure effettuate dal sensore hanno lunghezza pari a 3 bit, quindi i valori di massimo, minimo e media hanno lo stesso parallelismo della media, mentre il valore RMS, per definizione raddoppia la lunghezza della parola e poiché non si vogliono perdite di precisione, non si effettua alcun troncamento o arrotondamento ma si mantiene dunque tale valore a 6 bit. Quindi, riprendendo come riferimento la (Fig.2.1) e la (Fig.2.4) si può notare come in entrambe le architetture sia presente un *flip flop* che campiona i 3 bit più significativi, ossia la richiesta, così facendo dopo aver trasmesso (nel primo caso) e ricevuto (nel secondo) i primi tre bit, si va a vedere se il contenuto del *flip flop* dice che si sta trasmettendo un valore RMS o meno. Se si sta inviando tale valore, allora si procede con la trasmissione (ricezione dei tre bit meno significativi), altrimenti si interrompe la comunicazione. Un'ultima considerazione nell'ottica del risparmio nei consumi è la seguente: dopo aver studiato empiricamente il sistema, si traccia una lista delle coppie —textitrichieste-indirizzi inviati più frequentemente e, a tali coppie si associa una codifica dove si hanno il numero massimo di bit consecutivi uguali, in modo tale da far commutare il meno possibile sia i registri che realizzano lo *shift register* sia i bus che collegano i due lati dell'interfaccia.

Capitolo 4

Simulazioni

Una volta definita l'architettura interna e, una volta che è stato definito il protocollo di comunicazione, descrivendolo mediante i *timing diagram*, il passo seguente è stata la simulazione del codice *VHDL* scritto. Al fine di verificare se le forme d'onda descritte nei capitoli precedenti fossero le stesse implementate dalla nostra architettura si è utilizzato il software di simulazione *Modelsim*.

Nello svolgimento del progetto, ogni singolo blocco che compare nelle architetture di (Fig.2.4) e (Fig.2.1) è stato testato singolarmente prima essere connesso con gli altri a formare l'interfaccia. Per evitare di rendere questa relazione troppo lunga e pesante tuttavia riportiamo di seguito solo le forme d'onda dei due lati dell'interfaccia.

Iniziamo con il lato controllo, la (Fig.4.1) mostra la simulazione della parte di interfaccia di controllo dedicata alla ricezione del valore legato alla media minima. Nella figura si può notare anche come questa sia la parte di interfaccia adibita alla propagazione del segnale di reset verso il lato sensore. La (Fig.4.2) mostra la simulazione della parte di interfaccia di controllo dedicata alla comunicazione ondemand. Passiamo poi al lato sensore, la (Fig.4.3) mostra la simulazione della parte di interfaccia lato sensore dedicata alla trasmissione della media minima. La (Fig.4.2) mostra la simulazione della parte dedicata alla comunicazione ondemand dell'interfaccia lato sensore .

Come si può facilmente vedere le forme d'onda simulate sono coerenti con i *timing diagram* descritti precedentemente, quindi i nostri listati descrivono l'hardware desiderato e ampiamente esposto nel corso di questa relazione.

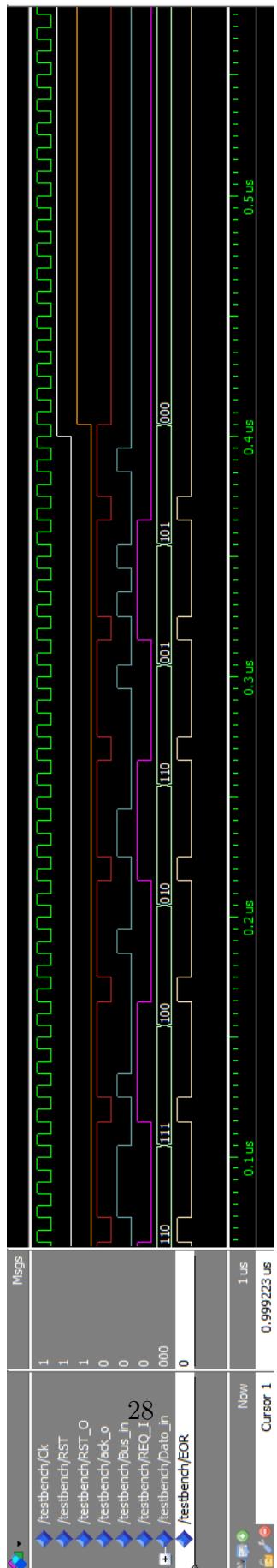


Figura 4.1: Simulazione della ricezione della media minima, lato controllo

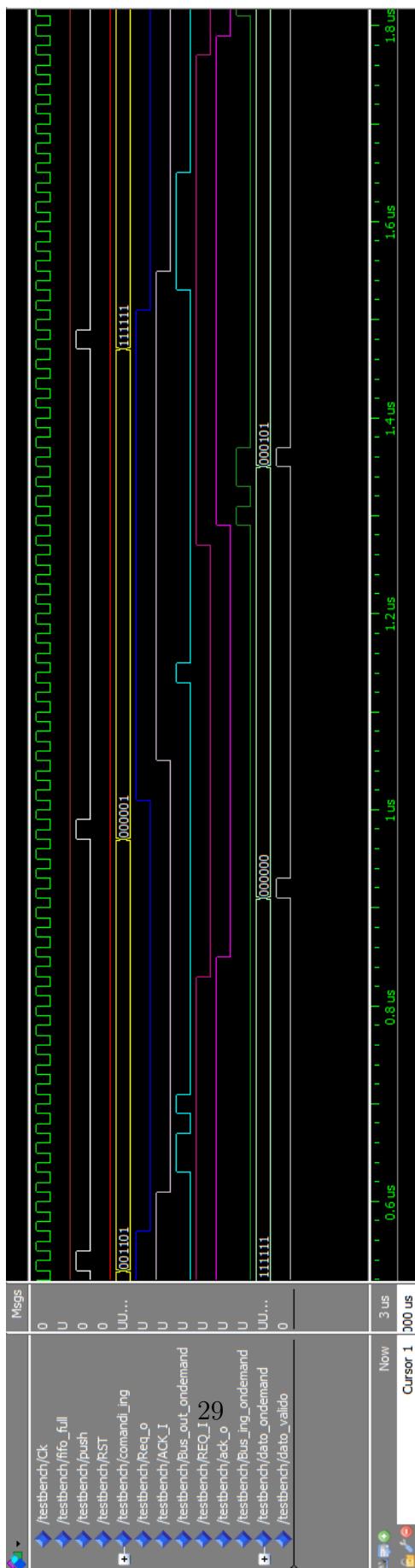


Figura 4.2: Simulazione della comunicazione ondemand, lato controllo

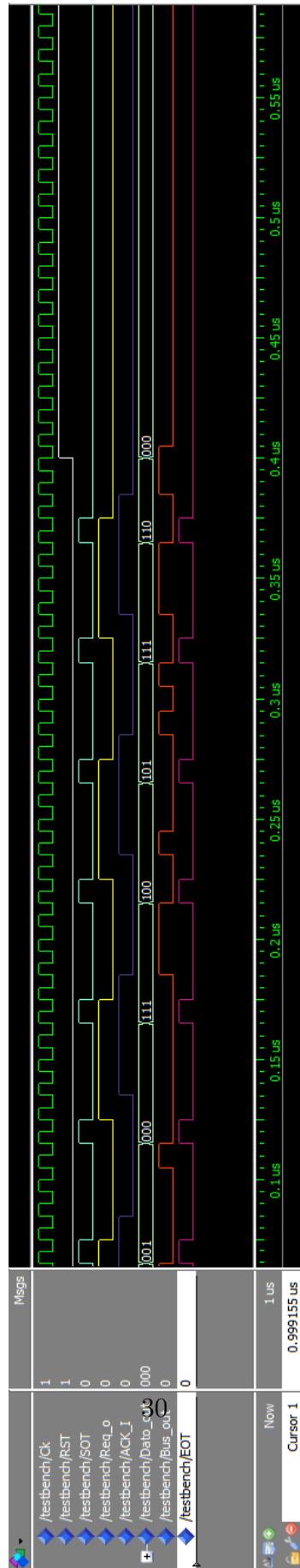


Figura 4.3: Simulazione della trasmissione della media minima, lato sensore

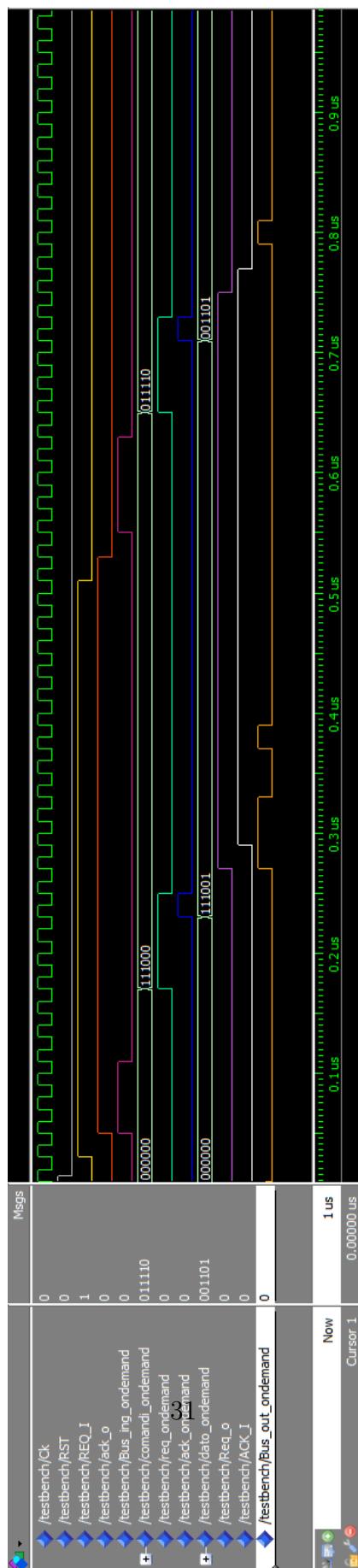


Figura 4.4: Simulazione della comunicazione ondemand, lato sensore

Capitolo 5

Clock Gating

5.1 Cenni Teorici

Il clock gating è una delle tecniche più utilizzate per la riduzione della potenza dissipata, dovuta al segnale di clock. Il segnale di clock può richiedere sino al 40% della potenza totale del circuito. Tuttavia in molti casi è possibile ottenere un notevole risparmio con l'applicazione del clock gating, in quanto molti blocchi o unità in alcuni casi non sono attivi, e per tale motivo possono essere disabilitati. Il concetto su cui si basa il clock gating è appunto quello di bloccare il segnale di clock ai blocchi che non devono lavorare per un certo periodo, impedendo dunque che avvengano inutili commutazioni dei registri e delle altre unità non utilizzate.

Il clock gating può essere applicato a diversi livelli: possiamo applicarlo a livello locale, coinvolgendo diversi registri, banchi di registri, oppure a livello globale, bloccando il clock a interi blocchi di logica. Chiaramente più si agisce a livelli alti, maggiori saranno i risparmi che si ottengono in fatto di consumi.

Approcci al clock gating Sono possibili differenti approcci per l'applicazione della tecnica del clock gating.

- Il sintetizzatore può determinare in quali parti può essere applicato il clock gating, e quindi procedere con la sua implementazione. Questo approccio richiede l'esecuzione di alcune apposite istruzioni nel sintetizzatore utilizzato.
- Il progettista può indicare esplicitamente, a livello RTL, in quali celle dev'essere applicato il clock gating.

Nell'applicazione tramite sintetizzatore, senza indicazioni da parte del progettista nel codice RTL, si può verificare il caso in cui non vengano rilevate alcuni registri o blocchi ai quali il clock gating può essere applicato. Il progettista può dunque procedere con una modifica del codice VHDL in modo da forzare il sintetizzatore all'applicazione del clock gating ai blocchi desiderati.

5.2 Applicazione del clock gating alla nostra interfaccia

Per mantenere più generalità possibile nel progetto si è deciso di optare per l'applicazione del clock gating attraverso l'utilizzo di Synopsys. In questo modo non è necessario scrivere il codice VHDL e implementare direttamente la tecnica attraverso l'inserimento del latch e della porta AND, ma sarà sufficiente fornire a Synopsys una descrizione tale del sistema in modo che questo riesca a rilevare le possibili applicazioni del clock gating. Questo ci permette di realizzare una tecnica di ottimizzazione di tipo technology indipendend.

Si è deciso di effettuare prima la sintesi senza l'applicazione del clock gating, con la generazione degli appositi power reports. Solo in un secondo momento è stata effettuata una seconda sintesi e la rispettiva generazione dei power reports per valutare i risparmi ottenuti.

Per avere un report della potenza accurato sarebbe necessario avere delle informazioni statistiche dettagliate sulle attività dei vari segnali. In questo caso noi non disponiamo delle informazioni sufficienti, e ci siamo limitati a impostare le attività dei vari segnali a valori plausibili. Inizialmente i segnali sono tutti impostati a una switching activity di 0.5. Questa è un'ipotesi irrealistica, in quanto segnali come i reset e gli enable hanno delle statistiche sicuramente molto lontane da questo valore.

Si è dunque proceduto con la sintesi e con il settaggio delle switching activities dei segnali con il seguente comando, eseguito per ogni segnale a cui abbiamo modificato la statistica:

```
set switching activity -static probability 0 -clock ck {name_signal}
```

5.2.1 Applicazione del clock gating al lato controllo

Si è dunque proceduto con la sintesi attraverso synopsys senza l'applicazione del clock gating, utilizzando i comandi:

```

analyze -format vhdl Controllo.vhdl
elaborate Lato_controllo -library work -architecture A
uniquify
compile -exact_map
create_clock -name ck -period 20 {CK_I}
report_power -include_input_nets

```

A questo punto è stato possibile osservare lo schematic sintetizzato e procedere con la generazione del report della potenza attraverso il comando:

```
report_power -net -include_input_nets
```

Il power report generato è il seguente:

```

Global Operating Voltage = 1
Power-specific unit information :
  Voltage Units = 1V
  Capacitance Units = 1.000000pf
  Time Units = 1ns
  Dynamic Power Units = 1mW      (derived from V,C,T units)
  Leakage Power Units = 1pW

Cell Internal Power   =  58.5912 uW   (86%)
Net Switching Power   =   9.7880 uW   (14%)
-----
Total Dynamic Power    =  68.3792 uW   (100%)

Cell Leakage Power     =   3.4503 uW

```

E' importante anche effettuare un confronto dal punto di vista dell'area. Generiamo dunque il report corrispondente con i comandi:

```
report_area -nosplit
```

Il report ottenuto è il seguente:

```
*****
Report : area
Number of ports:          25
Number of nets:           26
Number of cells:          2
Number of references:     2
```

```

Combinational area:      948.326422
Noncombinational area: 1469.764795
Net Interconnect area: undefined (No wire load specified)

Total cell area:        2418.091309
Total area:             undefined

```

Successivamente è stata rieseguita la sintesi, stavolta inserendo dei nuovi comandi per Synopsys che ci hanno permesso di realizzare il clock gating sullo shift register. I comandi eseguiti sono i seguenti:

```

set clock gating style
analyze -format vhdl Controllo.vhd
elaborate Lato_controllo -library work -gate clock
propagate constraints -gate clock
uniquify
compile -exact map
create clock -name ck -period 20 {ck}

```

A questo punto abbiamo potuto osservare lo schematico sintetizzato, e notare la presenza della circuiteria aggiuntiva che permette l'implementazione del clock gating.

Ripetendo i comandi per la generazione dei report corrispondenti al consumo di potenza e all'area otteniamo:

```
*****
Report : power

Global Operating Voltage = 1
Power-specific unit information :
  Voltage Units = 1V
  Capacitance Units = 1.000000pf
  Time Units = 1ns
  Dynamic Power Units = 1mW    (derived from V,C,T units)
  Leakage Power Units = 1pW

  Cell Internal Power   = 31.0362 uW   (82%)
  Net Switching Power   = 6.8861 uW   (18%)
  -----
  Total Dynamic Power   = 37.9223 uW   (100%)
```

Cell Leakage Power = 3.2354 uW

Report : area
 Design : Lato_controllo_N3_M6
 Version: Z-2007.03-SP1
 Date : Sat Aug 3 03:15:09 2013

Number of ports:	25
Number of nets:	26
Number of cells:	2
Number of references:	2
Combinational area:	712.656018
Noncombinational area:	1555.142395
Net Interconnect area:	undefined (No wire load specified)
Total cell area:	2267.798340
Total area:	undefined

Notiamo quindi che l'applicazione del clock gating ci permette di avere un notevole risparmio di potenza. Si passa da $68.3792 \mu W$ di potenza dinamica consumata prima dell'applicazione del clock gating, a $37.9223 \mu W$. Per quanto riguarda invece la potenza di leakage non abbiamo miglioramenti, come ci si attendeva dall'applicazione di questa tecnica. I blocchi che implementano il clock gating sono riportati nelle figure seguenti. Come si può osservare l'implementazione viene fatta con l'inserimento di un latch e una porta AND.

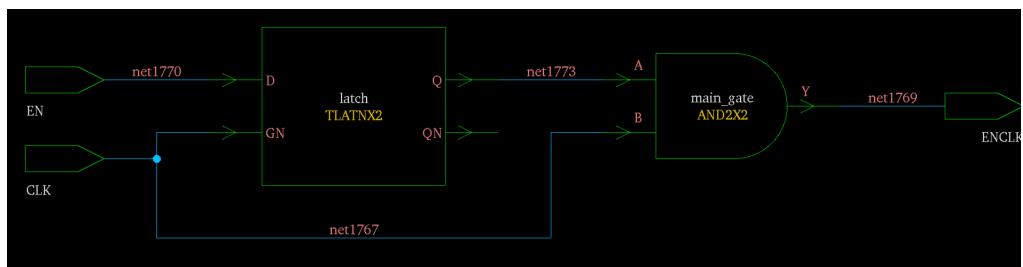


Figura 5.1: Implementazione del clock gating attraverso latch e porta AND

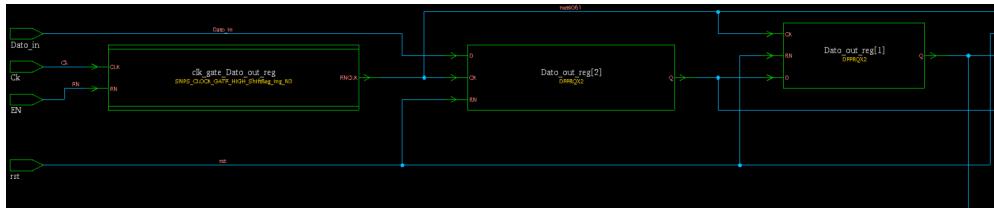


Figura 5.2: Inserimento dei blocchi che realizzano il clock gating.

5.2.2 Applicazione del clock gating al lato sensore

Lo stesso procedimento seguito nella sezione precedente viene ripetuto anche per il lato sensore.

Anche in questo caso andiamo a valutare il consumo di potenza del circuito nei due casi, con e senza clock gating.

Quindi, partendo dal circuito prima che gli venga applicato la tecnica del clock gating, si generano i power report seguendo il procedimento precedentemente descritto:

```
*****
Report : power
    -analysis_effort low
    -include_input_nets
Design : Lato_sensore_N3_M6
Version: Z-2007.03-SP1
Date   : Sat Aug  3 03:23:45 2013
*****

Cell Internal Power  =  19.7708 uW  (84%)
Net Switching Power =  3.8264 uW  (16%)
-----
Total Dynamic Power  =  23.5973 uW  (100%)

Cell Leakage Power     =  1.7519 uW
```

```
*****
Report : area
Design : Lato_sensore_N3_M6
Version: Z-2007.03-SP1
Date   : Sat Aug  3 03:24:25 2013
```

```
*****
```

Number of ports:	30
Number of nets:	30
Number of cells:	2
Number of references:	2
Combinational area:	584.236815
Noncombinational area:	622.339199
Net Interconnect area:	undefined (No wire load specified)
Total cell area:	1206.576050
Total area:	undefined

Ripetendo la sintesi con il clock gating, e realizzando i relativi report si ottiene:

```
*****
```

Report : power
 -analysis_effort low
 -include_input_nets
Design : Lato_sensore_N3_M6
Version: Z-2007.03-SP1
Date : Sat Aug 3 03:29:33 2013

Cell Internal Power	=	12.1223 uW	(78%)
Net Switching Power	=	3.3494 uW	(22%)

Total Dynamic Power	=	15.4717 uW	(100%)
Cell Leakage Power	=	1.7121 uW	

```
*****
```

Report : area
Design : Lato_sensore_N3_M6
Version: Z-2007.03-SP1
Date : Sat Aug 3 03:30:24 2013

Number of ports:	30
Number of nets:	30
Number of cells:	2
Number of references:	2
Combinational area:	511.560015
Noncombinational area:	661.852798
Net Interconnect area:	undefined (No wire load specified)
Total cell area:	1173.412842
Total area:	undefined

Anche in questo caso viene messa in evidenza una riduzione della potenza dinamica, che è passata dal valore di $23.5973\mu W$ a $15.4717\mu W$. Anche in questo caso la potenza di leakage è rimasta pressochè invariata.

Osservando inoltre i report relativi all'area, le ottimizzazioni attraverso synopsys hanno portato anche a una riduzione di quest'ultima in entrambi i casi, seppur con valori abbastanza ridotti. In ogni caso questo è un fattore totalmente positivo, in quanto attraverso l'applicazione del clock gating non si mira a una riduzione della potenza di leakage e dell'area. Infatti nella maggioranza dei casi questa tecnica porta a un aumento di entrambe per via dell'inserimento dei latch e delle porte AND che sono necessarie per la sua realizzazione.

Capitolo 6

Conclusioni

Concludendo possiamo dire di essere riusciti a realizzare un'interfaccia che rispettasse i vincoli di progetto imposti dai progetti sviluppati parallelamente e siamo anche riusciti a valutare le ottimizzazioni *low-power* realizzabili su di essa.

Concludiamo dicendo che visto l'utilizzo che se ne fa dell'interfaccia, a livello teorico la probabilitá che chi riceva non fornisca il segnale di *acknowledge* é estremamente bassa. Se questa supposizione teorica venisse verificata anche tramite prove pratiche allora si puó dire che i tre segnali di *acknowledge* sono dei segnali ridondanti e quindi eliminabili riducendo cosí i pin dell'interfaccia da 11 a 8.

Un'ultima nota la dedichiamo alla comunicazione tra interfaccia e unitá di calcolo, in particolare per quanto riguarda la parte della comunicazione on-demand. Una volta che l'interfaccia asserisce il segnale di *request* si pone nello stato di attesa dell'*acknowledge*. Quest'ultimo per diverse ragioni legate al funzionamento dell'unità di controllo, puó non asserirsi mai e questo fatto blocca la macchina a stati. Per evitare questo malfunzionamento si puó pensare di introdurre un *watchdog timer* che eviti questi stalli resettando la parte di interfaccia dedicata alla trasmissione ondemand.