Modellazione e Sintesi di un Moltiplicatore Floating-point Single Precision

Enrico Sgarbanti - VR446095

***Sommario*—Il progetto ha 3 diversi obiettivi:**

- **Sviluppare in VHDL, Verilog e SystemC un componente che esegua due moltiplicazioni in virgola mobile a precisione singola**
- **Sintetizzare i componenti scritti in VHDL e Verilog**
- **Eseguire High-Level-Synthesis del moltiplicatore a precisione singola**

I. Introduzione

Il sistema è composto da un modulo top-level chiamato "double_multiplier" il quale esegue la moltiplicazione di due operandi dati in input nello stesso ciclo di clock in cui ready viene posto a 1, e i due opereandi passati al ciclo di clock successivo

Nell'introduzione viene descritto in maniera astratta quello che poi viene dettagliato nel seguito del report. Una buona scaletta per l'introduzione può essere la seguente:

- Descrizione ad alto livello delle principali caratteristiche del sistema che si vuole modellare.
- Descrizione delle motivazioni principali per l'utilizzo delle tecnologie descritte nel corso. Qual è il problema che si vuole risolvere?
- Descrizione dei passi utilizzati per arrivare all'implementazione finale. Descrivere la motivazione di ciascun passo. La descrizione dei passi dovrebbe formare la descrizione del flusso di lavoro svolto per completare l'assignment.
- Rapidissima descrizione dei risultati principali.

L'introduzione non dovrebbe andare oltre la metà della seconda colonna (nel caso a due colonne), o la prima pagina (nel caso a colonna singola): bisogna cercare di essere concisi (e chiari). Alla fine, l'introduzione è solo "chiacchiere": deve semplicemente rendere chiari quali sono gli obiettivi del lavoro (*e nel caso del corso, deve far capire a me che avete gli obiettivi chiari in testa*). Consiglio: l'introduzione (e spesso l'abstract) è l'ultima parte che viene completata.

II. Background

- descrivere cos'è un floating-point - descrivere la moltiplicazione - arrotondamento numero binario

Il background dovrebbe contenere una descrizione, abbastanza breve, dei concetti principali che vengono utilizzati nel lavoro. Ad esempio, può contenere una breve descrizione delle caratteristiche principali degli HDL , e delle sue estensioni. Una colonna dovrebbe bastare.

In questa sessione saranno citati anche lavori dello stato dell'arte. Ad esempio, si può usare [**?**] per citare SystemC. **I riferimenti bibliografici vanno inseriti ogni volta in cui si va a citare qualcosa contenuto in un documento.**

Questa sessione non deve ripetere quanto detto a lezione, ma dare un overview dei concetti principali utilizzati durante lo svolgimento dell'elaborato.

III. Metodologia applicata

Per la realizzazione di questo progetto sono necessari 3 moduli:

- modulo del moliplicatore.
- modulo di alto livello che gestisce la moltiplicazione delle due coppie di operandi.
- modulo di testbench.

A. Modellazione delle EFSM

I segnali utilizzati sono:

- **rst:** (1 bit) per riportare il sistema allo stato iniziale .
- **ready:** (1 bit) per permettere al sistema di uscire dallo stato iniziale .
- **done:** (1 bit) per indicare che il valore attuale di "res" è il risultato della moltiplicazione.
- **norm_again:** (1 bit) per indicare se fare un'altra normalizzazione del numero intermedio.
- **res_type:** (2 bit) per indicare se il risultato vale 0, NAN o ∞ e quindi andare direttamente allo stato finale oppure se è un numero (il caso denormalizzato viene gestito come se fosse normalizzato) e quindi proseguire nell'elaborazione.

Sono inoltre necessari 14 stati:

- **ST_START:** in cui si pone *done* e *norm_again* uguali a 0 e si ottengono le informazioni di segno, esponente e mantissa dei due operandi. In esso si rimane finchè *ready* vale 0 altrimensi si passa a *ST_EVAL1*.
- **ST_EVAL1:** e *ST_EVAL2* in cui viene valutato il tipo dei due operandi fra *T_ZER, T_INF, T_NAN e T_NUM*
- **ST_EVAL3:** in cui si ricava il tipo del risultato in base al tipo degli operandi.
- **ST_CHECK1:** dove si passa a *ST_FINISH* se *res_type* è diverso da *T_NUM*.
- **ST_ELAB:** in cui si sommano i due esponenti e si sottrae 127 perchè entrambi, per lo standard, sono incrementati di 127 ($(esp + 127) = (esp1 + 127) + (esp127) - 127$). Per compiere la somma è necessario usare una variabile "esp_tmp" 10 bit altrimenti si andrebbe in overflow per valori invece leciti che ritornerebbero validi dopo la sottrazione di 127. Inoltre viene eseguita anche la

moltiplicazione delle due mantisse, che essendo a 23 bit più un bit che vale sempre 1, necessita di una variabile "mant_tmp" di 48 bit
- **ST_UNDERF:** in cui si verifica se l'esponente del risultato è in uno stato di underflow, ovvero guardando se il bit *esp_tmp[9]*, che nel complemento a 2 indica il segno, è 1. Infatti i valori disponibili per l'esponente vanno da 0 a 255.
- **ST_CHECK2** dove si passa a *ST_FINISH* se *res_type* è diverso da *T_NUM*, perchè diventato *T_ZER* per l'underflow.
- **ST_NORM1:** in cui si compie la normalizzazione della mantissa che deve sempre essere della forma *1.valori*. Essendo la virgola posta tra il 46esimo bit e il 45esimo, si verificano due casi: Se il 47esimo bit vale 1 bisogna incrementare l'esponente, altrimenti il valore è già corretto, ma viene effettuato uno shift a sinistra per trattare allo stesso modo i due casi durante l'arrotondamento.
- **ST_ROUND:** in cui si effettua l'eventuale arrotondamento dovuto al fatto che il valore della mantissa è attualmente a 48 bit, ma bisogna portarlo a 24 bit. L'arrotondamento è fatto per eccesso, quindi si incrementerà *mant_tmp[47:24]* solo se *mant_tmp[23:00]* sarà $\geq$ a "01..1". L'arrotondamento effettivò verrà fatto nello stato *ST_NORM2*, qui ci si limita a porre *norm_again* uguale a 1 per poterci andare.
- **ST_CHECK3:** dove si passa a *ST_NORM2* se *norm_again* è uguale a 1 altrimenti a *ST_OVERF* per il controllo di eventuali overflow
- **ST_NORM2:** in cui si effettua il vero arrotondamento della mantassina. Bisogna tenere conto del caso in cui sia della forma "1..1" e che quindi con l'incremento vada a "0..0" e venga incrementato l'esponente.
- **ST_OVERF:** in cui si verifica se l'esponente del risultato è in uno stato di overflow, ovvero guardando se il bit *esp_tmp[8]* vale 1 ovvero se corrisponde ad un valore maggiore di 255.
- **ST_FINISH:** in cui si pone *done* uguale 1, si ricava *res[31]*, ovvero il segno del risultato facendo lo XOR fra i segni degli operandi, e in base al valore di *res_type* si ottiene il resto.

In questa sezione viene descritto tutto il procedimento, ed è dunque la sezione più importante del report. Va descritto passo passo quello che avete fatto, facendo capire "esattamente" cosa è stato fatto.

Questa sezione può essere divisa in sottosezioni. Le informazioni riportate nella lista seguente dovrebbero essere identificabili nel testo del report (**anche in ordine diverso**):

- Definizione dell-architettura del programma
- Descrizione del componente di HW digitale.
- Processo di "sintesi" verso RTL. Definizione dei sottocomponenti del componente HW, e della sua struttura. Definizione dell'interfaccia RTL, definizione della Macchina a Stati Finiti Estesa (EFSM) del componente e dei sottocomponenti. Realizzazione del componente HW utilizzando i processi in diversi stili a livello RTL. È inoltre possibile discutere la scelta dei tipi di dato.
- Descrizione dei meccanismi di comunicazione tra le diverse parti del sistema.

In questa sezione deve essere riportato (brevemente) anche l'organizzazione dell'implementazione consegnata assieme al report.

IV. Risultati

Qua vanno "messi i numeri". Questa sezione dovrebbe contenere i risultati della simulazione. La simulazione mostra che il sistema funziona correttamente? Com'è stato provato? Che tipo di testbench sono stati utilizzati? Com'è stato scomposto il sistema per verificarne la correttezza?

Per quanto riguarda le performance:

- cosa si può dire in merito alla latenza?
- Qual è la frequenza massima del design?
- Qual è l'area occupata dal design?

Questa sezione può contenere anche riflessioni personali sui risultati ottenuti. Importante: tutte le affermazioni devono essere supportate da numeri[1].

V. Conclusioni

Le conclusioni dovrebbero riassumere in poche righe tutto ciò che è stato fatto. Un paio di righe descrivono i risultati osservati, in modo da introdurre poi la conclusione "vera e propria". Nel caso del corso, la "lezione da portare a casa" sarà quello che si è imparato svolgendo l'elaborato.

Appendice

Se non avete abbastanza spazio, potete inserire le figure delle EFSM in una pagina extra, appendice. Un esempio di come potete fare solo le Figure 1, 2, 3.

[1] Richard Feynman on Scientific Method (1964) - https://www.youtube.com/watch?v=OL6-x0modwY

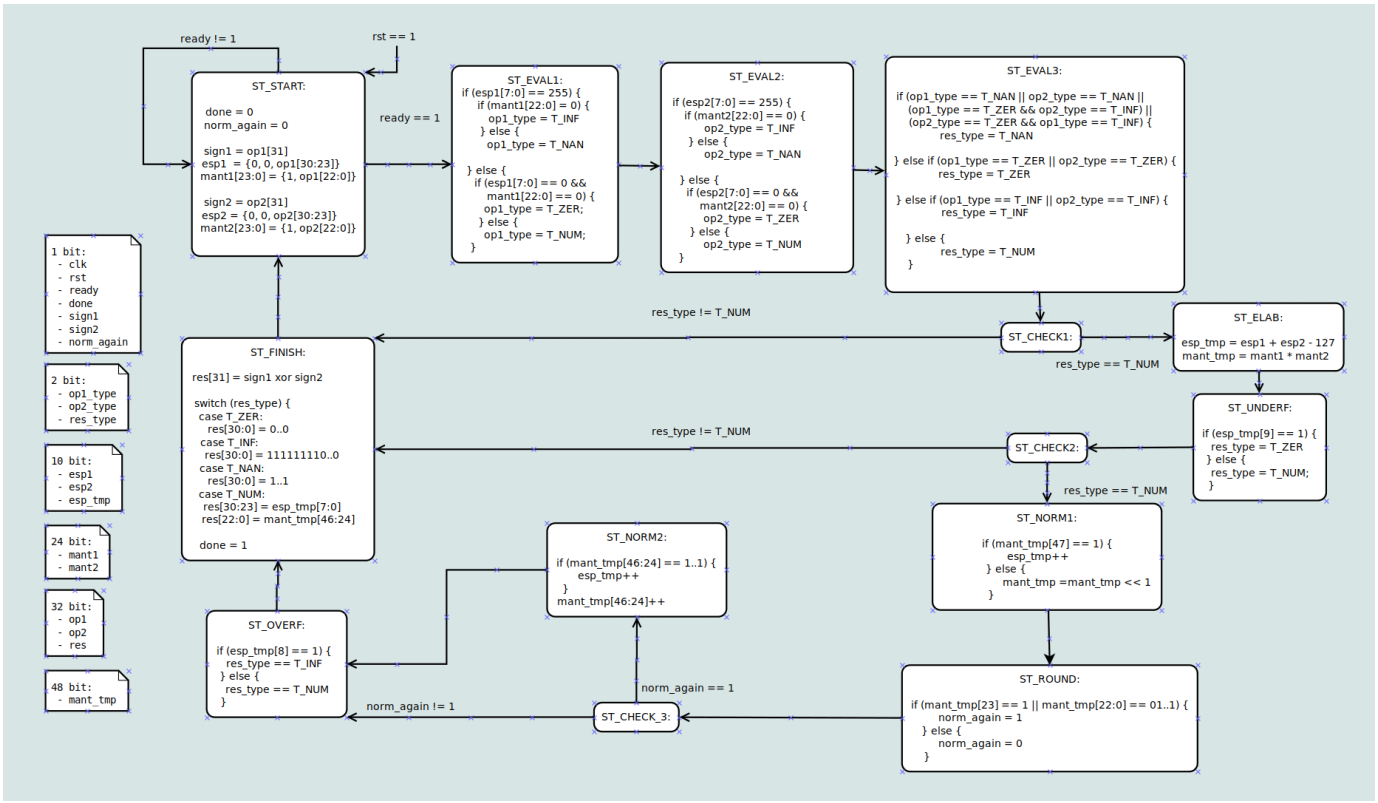

Figura 1. Figura in formato grande.

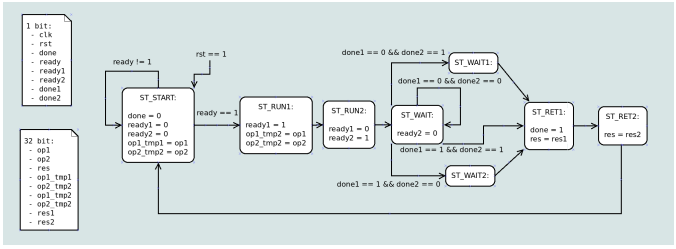

Figura 2. Figura in formato piccolo, 1.

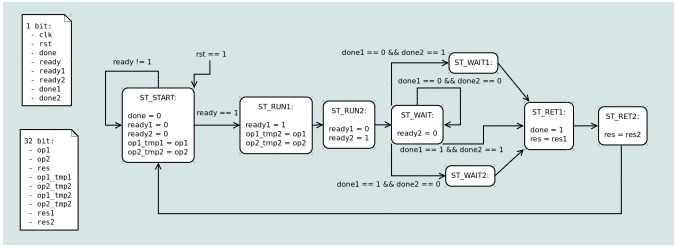

Figura 3. Figura in formato piccolo, 2.