Capitolo 3. Realizzare una ALU Floating Point

3.1. Funzioni di conversione

Uno dei punti di forza del VHDL è la possibilità di lavorare con la metodologia top-down, realizzando in un primo momento le entità con descrizioni comportamentali e scendendo nei dettagli solamente dopo che il design è stato testato e verificato. Seguendo questa metodologia è possibile progettare una FPU (Floating Point Unit), la cui realizzazione a livello circuitale è sicuramente di complessità notevole, utilizzando una descrizione comportamentale che si appoggia sulle librerie numeriche floating point del compilatore. Una volta che il design del processore è stato completato, è sufficiente sostituire la descrizione behavioural dell'ALU con una descrizione comportamentale o comunque sintetizzabile dal compilatore.

Per poter lavorare correttamente con ALU ed FPU a livello comportamentale è comunque necessario realizzare alcune funzioni di conversione, che consentano di trasformare la rappresentazione binaria del numero nella corrispettiva quantità reale o intera e viceversa. I segnali infatti sono definiti mediante **bit_vector** e non è possibile operare direttamente su di essi con gli operatori o le funzioni matematiche standard presenti nella libreria del ModelSim. Il VHDL permette infatti di definire variabili di tipo **integer** o **real**, ma non è possibile assegnare loro direttamente un valore contenuto in un segnale o in un'altra variabile di tipo bit_vector. La prima cosa da fare è allora realizzare le funzioni di conversione tra i tipi interi e reali e la loro rappresentazione binaria.

3.1.1 Conversioni di quantità integer

La conversione di quantità intere non presenta alcun problema: le funzioni che svolgono questo compito, oltre che molto semplici, si possono trovare in qualsiasi libro che tratti il VHDL. Per completezza è di seguito riportata la realizzazione presente nel VHDL Cookbook.

```
function bits_to_int (bits : in bit_vector) return integer is
     variable temp : bit_vector(bits'range);
     variable result : integer := 0;
  begin
     if bits(bits'left) = '1' then -- negative number
        temp := not bits;
     else
        temp := bits;
     end if;

     for index in bits'range loop -- sign bit of temp = '0'
        result := result * 2 + bit'pos(temp(index));
     end loop;
     if bits(bits'left) = '1' then
        result := (-result) - 1;
     end if;
  return result;
end bits_to_int;
```

Questa funzione effettua la conversione tra una sequenza di cifre binarie