Capitolo 5. Specifiche di progetto del Pixel Shader v.1.4

5.1. Linee generali del progetto

La presente parte della tesina è relativa alla progettazione in linguaggio VHDL del *Pixel Shader*, processore matematico del quale si è fornita una introduzione nel paragrafo 1.9.3.

Prima di addentrarci nella discussione concernente la realizzazione del processore è bene esporre le linee generali del progetto, in particolar modo le specifiche, gli obiettivi e le limitazioni.

Le specifiche sul funzionamento generale del processore sono state estratte dalla documentazione che le principali ditte costruttrici di schede grafiche mettono a disposizione dei programmatori: nei siti web di ATI, NVIDIA e anche nel sito Microsoft si trovano spiegazioni dettagliate del funzionamento di un pixel shader; tali informazioni sono state da noi usate come specifiche generali di progetto del Pixel Shader, versione 1.4.

La documentazione di cui sopra non contiene tuttavia informazioni sull'architettura interna del processore; quest'ultima deve essere così progettata sulla base delle funzionalità note del processore, seguendo il criterio della razionalità e cercando di aumentare, ove possibile, la velocità di esecuzione delle diverse operazioni.

E' bene fin da subito mettere in evidenza un aspetto importante. Il pixel shader è per sua natura un processore molto complesso, sia per il fatto di dover compiere operazioni matematiche in virgola mobile, sia per la necessità di eseguire un grandissimo numero di istruzioni nell'unità di tempo. A fronte di tali problematiche, nell'ambito della presente tesina sono state introdotte alcune semplificazioni.

Innanzitutto le ALU floating-point sono state progettate a livello comportamentale. D'altronde la possibilità di descrivere i blocchi funzionali di un sistema digitale usando diversi livelli di astrazione è una caratteristica fondamentale del linguaggio VHDL, che noi abbiamo usato a nostro vantaggio.

In secondo luogo, per maggior semplicità, la struttura del processore è stata concepita per eseguire le operazioni di *caricamento*, *decodifica* ed *esecuzione* delle istruzioni ciascuna in un momento diverso, ovvero non si è provveduto a parallelizzare le tre fasi, come invece avviene comunemente nei processori, allo scopo di aumentarne le prestazioni.

Occorre fare una precisazione relativa al periodo del clock di sistema e più in generale ai ritardi associati ai vari blocchi costituenti. Sotto tale punto di vista, il componente senza dubbio più critico è l'ALU floating-point, descritto nel nostro progetto a livello behavioural. La scorrelazione dall'hardware che deriva da una così elevata astrazione non consente di stimare i ritardi relativi alle porte logiche che implementeranno fisicamente l'ALU e quindi neanche il minimo periodo di clock, la cui scelta è indissolubilmente legata a tali ritardi.

La scelta del clock è stata allora basata su una stima grossolana dei ritardi degli altri blocchi componenti, più semplici e dunque descrivibili in modo da essere maggiormente legati all'hardware. I tempi di esecuzione delle singole istruzioni dell'ALU sono stati fissati in maniera piuttosto arbitraria, ma in ogni