**Introduzione**  
*Scopo della tesi e quadro generale: uso del VHDL, implementazione della CPU Z80 e del sistema che fa da contorno.*

**1 - VHDL e FPGA**

1.1 - VHDL: Linguaggio di descrizione hardware  
*Cos’è un HDL, a cosa serve, che peculiarità ha il VHDL. Cosa vuol dire fare il design e implementare un circuito per mezzo di un HDL. Cosa vuol dire “descrizione dal punto di vista comportamentale”. Cos’è un IP.*

1.2 - FPGA e la scheda di sviluppo usata  
*Cos’è un FPGA e perché permette di implementare un ampio numero di sistemi logici. Info sulla scheda usata (ALINX 309) e sulla FPGA Xilinx Spartan 6. Cos’è una slice, cosa sono le special features usate (BRAMS, Clock buffers, …).*

**2 – Lo Z80**

2.1 - Generalità e storia della CPU Z80  
*Cos’è. Perché è ben noto cioè dov’è stato usato. La presenza di molte periferiche dedicate e dell’architettura adattata. Come Faggin l’ha fatto. Chi è Faggin.*

2.2 - Architettura e Organizzazione dello Z80  
*CISC. Pinout. Dal datasheet cosa si vede: registri, interrupt, set di istruzioni, ecc. cioè l’*architettura*. Il fatto che non si conosca l’*organizzazione *da documentazione ma ci siano state persone che hanno fatto il reverse engineering del chip attraverso lo studio al microscopio. Informazioni note dal reverse engineering: ALU a 4 bit, registri W e Z, presenza PLA per la decodifica delle istruzioni.*

2.3 – Implementazione della CPU Z80 su FPGA  
*Effettiva organizzazione della CPU per mezzo di una macchina a microcodice, dei bus principali, dei registri indirizzabili e l’ALU. Scelta della realizzazione per mezzo di una FSM e di un decoder statico.* Schema dell’organizzazione usata e dei bus*. Aggiunta pin per controllo uscite e clock fpga per potersi avvicinare al comportamento a latch.*

2.3.1 - Implementazione del ciclo principale  
*Descrizione della divisione in cicli macchina e in cicli T. Divisione delle fasi e come vengono selezionate. Descrizione eventi di Interrupt e Busreq.*

2.3.2 - Implementazione delle sotto macchine  
*Descrizione delle temporizzazioni e delle forme d’onda delle fasi più importanti: Fetch, Memory R/W, IO R/W, Halt,….*

2.3.2.1 - Instruction Opcode Fetch

2.3.2.2 - Memory Read or Write Cycles

2.3.2.3 - Input or Output Cycles

2.3.2.4 – Interrupt  
*Gestione degli eventi di interrupt da parte dello Z80: nINT e nNMI. Modalità di gestione degli nINT. FFs per la memoria e il controllo via SW.*

2.3.2.5 - Bus request

2.3.2.6 - Reset cycle

2.3.3 - ALU e incrementer  
*Implementazione e scelte dal punto di vista comportamentale.*

2.3.4 - Registri

2.4 - Problemi noti  
*Non è sempre rispettata la divisione dei cicli macchina come nel datasheet ma non porta a grossi problemi rilevabili.*

**3 –** **Memorie, interfacce verso l’esterno e controllore***Generalità, scopo e organizzazione: per programmare velocemente le memorie, fare il debug in tempo reale e gestire il clock, permettere output verso il display 7 segmenti.*

3.1 - Bus e interfacce a registri

3.2 – Memorie  
*Descrizione e scopo delle FIFO usate e delle BRAM.*

3.3 - Display a 7 Segmenti e driver  
*Scopo, architettura e funzionamento.*

3.4 - Sistema di controllo via UART  
*Scopo, architettura e funzionamento.*

3.4.1 – UART  
*Descrizione sistema di comunicazione. Descrizione interfaccia da FPGA a PC.*

3.4.2 - Controllore e interfaccia verso il computer  
*Scopo, architettura, organizzazione e schema comandi via UART.*

3.4.3 - Generatore di clock variabile  
*Architettura e organizzazione.*

3.4.4 - Z80 snapper e Z80 reporter  
*Scopo, architettura e organizzazione.*

**4 – Conclusione**

*Dove sono arrivato, cosa si può migliorare e come creare un’applicazione rappresentativa.*

**Bibliografia**