Una delle fasi del processo di progettazione è quella di scegliere quali parti comprare sul mercato già pronte e quali invece progettare autonomamente. In quel caso bisogna scegliere anche il mezzo su cui implementare il design. Nel caso di circuiti che richiedono funzionalità particolari e nel caso di elevati volumi di produzione si può creare un circuito integrato personalizzato detto ASIC (Application Specific Integrated Circuit). Le ASICs portano il vantaggio di poter scegliere autonomamente la tecnologia raggiungendo un’efficienza elevata. D’altra parte richiedono un corposo investimento per iniziare la produzione. Una via di mezzo tra l’uso di parti presenti sul mercato e quello delle ASICs è dato dalle FPGAs (Field-Programmable Gate Arrays) o più in generale dai dispositivi logici programmabili PLDs (Programmable Logic Devices). *[Digital design p.29]*

I PLDs in generale sono dispositivi logici regolari e riconfigurabili. A differenza degli altri dispositivi logici, al momento della produzione non hanno una funzione logica predefinita.  
Questi dispositivi si possono dividere in tre categorie di complessità crescente: simple programmable logic devices SPLDs, complex programmable logic devices CPLDs ed FPGAs.  
I SPLDs non differiscono molto da una ROM. In quel caso, in ogni riga sono codificate le uscite della funzione logica e le righe sono selezionate dalla giustapposizione degli ingressi della funzione. Dei SPLD, sono noti i PLAs (Programmable Logic Arrays) in cui la funzione logica è pressoché implementata per mezzo dei suoi mintermini ed ha una struttura come quella in Figura X.

Figura X – Esempio di PLA *[Digital design p.193]*

Le FPGAs invece presentano uno schema a blocchi regolare. Al suo interno sono presenti dei blocchi logici configurabili CLBs (Configurable Logic Blocks). Quest’ultimi implementano le funzioni logiche per mezzo di look-up tables, LUTs, che sono generalmente delle memorie, ed altre funzioni aggiuntive come sommatori, multiplexer o stadi flip-flops, FFs. Oltre a questi blocchi ce ne sono altri con funzioni specifiche, a cui Xilinx si riferisce come special features, ad esempio memorie, gestione della temporizzazione o interfacce con l’esterno. Tutti questi blocchi sono disposti a matricee collegati a dei bus bidirezionali per mezzo di connessioni programmabili, come si vede in Figura XX. La struttura regolare e i collegamenti selezionabili delle FPGAs permettono di implementare un ampio numeri di design andando a collegare e configurare questi blocchi elementare. Inoltre sono facilmente programmabili e riconfigurabili più volte in poco tempo.

Figura XX – Schema dei CLBs e dei canali d’interconnessione *[ug384.pdf p.37]*

ll CAD dopo aver svolto la sintesi del design si occupa di tradurre (translate) i componenti acquisiti con quelli di cui l’FPGA è costituita. Dopodiché si occupa di associarli (map) con quelli effettivamente presenti sulla FPGA ed infine di piazzarli e collegarli (place & route) creando il design richiesto. In questi passaggi si può andare in contro ad errori che possono condizionare la scelta dell’FPGA. Per esempio, si può scegliere un’FPGA con delle RAM già presenti per risparmiare CLBs od un’altra con un numero maggiore di CLBs poiché il design non riesce a essere associato correttamente.

Per svolgere il progetto ho usato la scheda di sviluppo AX309 prodotta da ALINX. Questa scheda presenta un ampio numero di periferiche già collegate. Tra queste, quelle di maggiore importanza, poiché sono state usate nel progetto, sono: il display da sei cifre a 7 segmenti, controllato per mezzo di un unico bus per i segmenti ed un bus per gli anodi; un’interfaccia USB-UART per comunicare con la scheda attraverso la micro USB di alimentazione; 4 LED e una porta JTAG. La porta JTAG serve per la programmazione dell’FPGA attraverso un programmatore compatibile ma anche per il debug attraverso le comunicazioni di un’ILA, se istanziata.

Figura XXX – Scheda di sviluppo AX309

La scheda monta un FPGA Xilinx XC6LSX9 con clock a 50MHz. Quest’FPGA memorizza la configurazione su una memoria volatile. Per cui sulla scheda è presente una memoria FLASH, programmabile via JTAG, per immagazzinare la configurazione e ricaricarla ad ogni riavvio.  
L’FPGA presenta uno schema regolare dei CLBs che sono organizzati in colonne, come mostrato in Figura XXXX. Ogni CLB è diviso internamente in due slices che non comunicano tra loro ma solo con i collegamenti al bus centrale. Queste possono essere di tre tipi: SLICEX, basilari che contengono quattro 6-Inputs LUTs e 8 FFs con clock ed enable comuni; SLICEL, che in aggiunta alle precedenti hanno la possibilità di gestire i riporti provenienti da altre slices; SLICEM, che presentano elementi di memoria per creare RAM distribuite e shift registers. Le slice nei CLBs seguono la seguente distribuzione: una slice di ogni CLBs è sempre una SLICEX mentre l’altra è alternata tra le altre due. Quest’ultime sono messe in colonna per permettere la propagazione del riporto. *[ug384.pdf]*

Figura XXXX – Disposizione delle slices all’interno di un CLB*[ug384.pdf p.8]*

Grazie agli elementi di memoria nelle SLICEMs, si possono creare delle RAM distribuite su più CLBs, anche dual-port, che hanno il vantaggio di essere facilmente utilizzabili e con lunghezze di parola arbitraria. La limitazione è nel numero poiché si possono implementare al massimo 90kb. *[ug384.pdf]* Per avere a disposizione più memoria, l’FPGA contiene già delle RAM, dette Block RAMs (BRAMs), che permettono un indirizzamento attraverso due port. L’FPGA che ho usato mette a disposizione 32 BRAMs da 18kb usabili come 2x9kb per un totale di 576kb.*[ug383.pdf]* L’uso di RAM dual-port permette di collegare due regimi a clock diversi all’interno del componente o, come nel mio caso, per permette l’uso delle memoria sia da parte dello Z80 che del controllore.  
Le BRAMs e le RAM distribuite sono utilizzabili nel progetto per mezzo di appositi IPs messi a disposizione dall’ambiente di sviluppo. Lo stesso ambiente mette a disposizione degli IPs che implementano delle FIFOs (First-In First-Out). Quest’ultime sono delle code che permettono di immagazzinare dati da un regime più veloce e renderli disponibili a uno più lento. Queste pile si basano su memorie RAM e l’ambiente permette di scegliere tra distribuite e BRAMs.

Nel progetto ho usato due BRAMs per implementare la ROM e la RAM collegate allo Z80, entrambe da 2kB per un totale di 32kb. Ho scelto di usare delle BRAMs per lasciare liberi i CLBs per altri scopi. Anche le FIFOs che ho usato le ho tutte implementate su BRAMs per lo stesso motivo.