- **Panoramica del sistema di calcolo a 32 bit basato su schema multi-ALU e rete di registri PIPO paralleli sincronizzati**

Ci è stato chiesto di creare una simulazione di un sistema di calcolo a 32 bit con l'utilizzo di ALU 74181(a 4 bit) in codifica C con aggiunta di registri 74198(a 8 bit) per la memorizzazione dei valori forniti in input e di quelli dati in output post calcolo dell'ALU.
Per raggiungere i 32 bit essendo ALU a 4 e registri a 8 viene utilizzata la funzionalità Parallel Input Parallel Output(PIPO) che pone più quantità delle stesse unità in relazione parallela tra loro, ossia ogni unità esegue la funzione in maniera autonoma dall'altra, nel caso dell'ALU l'unica cosa che le collega tra loro è il carry necessario per il riporto nel calcolo aritmetico.

Tutto il sistema deve simulare il funzionamento logico escludendo la parte elettronica, ciò quindi comprende anche la trasformazione di input decimali in binario per il loro calcolo e poi di ritrasmutarli in decimale per fornirli in output.

L'intero sistema deve inoltre simulare la sincronizzazione con la frequenza di clock ad onda quadra.
in aggiunta alla comunicazione con l'utente doveva anche avvenire una decisione di fornire dati tramite terminale o file sorgente.

- **Analisi tecnica (minima) del sistema logico-aritmetico di base per il calcolo booleano (unità ALU cod. 74181 a 4 bit)**

Le ALU funziona su 4 bit per ogni ingressi quindi devono essere 8 per raggiungere un massimo di 32 bit e funzionano secondo la base PIPO essendo collegati solo tramite carry, il modello utilizzato restituisce 4 bit in output.

Il loro funzionamento agisce sugli input in base alle operazioni logiche o aritmetiche utilizzate selezionate da prescritte combinazioni di 6 selettori, 4 che decidono il calcolo specifico, 1 che decide se fare operazioni logiche o

aritmetiche, 1 che decide se le operazioni aritmetiche devono avere carry o no.

Le varie combinazioni di selettori sono già pre settate assieme allo schema logico di calcolo.

- **Analisi tecnica (minima) dei registri di interfaccia I/O di base in configurazione PIPO (unità registro 74198 a 8 bit)**

I registri possiedono 8 entrate e rispettive uscite, per ogni ingresso vi è posizionata una piccola cella di memoria chiamata flip-flop di tipologia sr-latch incaricata della memorizzazione del singolo bit.

I registri devono essere posizionati a seguito degli ingressi al sistema in cui l'utente fornisce i 2 dati di ingresso e dove devono essere 4 per variabile con un totale di 8 per raggiungere i 64 bit totali, e a seguito delle ALU per poter immagazzinare i valori in uscita dopo il calcolo e tenerli memorizzati mentre le ALU successive completano il loro, infatti esse sono semplici celle di memoria.

All'utente inoltre è possibile ,per ogni registro, un'interazione che gli renda possibile le seguenti azioni:

-clear: che resetta e porta tutti i valori a 0

-overwrite: che permette di re-inserire i singoli valori immagazzinati nei flip-flop

-salvataggio e chiusura dell'interazione

- **Analisi tecnica (minima) del generatore del segnale di sincronismo (clock) ad onda quadra**

La frequenza di clock è generalmente un semplice segnale digitale, quindi di soli 2 valori (alto =1 e basso = 0), che determina la velocità dei calcoli in quanto ogni volta che l'onda si trova nel punto alto avviene il ciclo che genera il risultato del calcolo (ciclo macchina di un pc) quindi più la frequenza è alta più sarà la quantità di calcoli eseguibili in un determinato tempo.

Infatti deve sincronizzare ogni operazione che avviene all'interno del sistema e deve far avvenire i vari cambiamenti di stato.

Per generare questa frequenza è necessario richiamare quella interna del dispositivo su cui viene eseguita la simulazione.

Ogni PC ha una frequenza di base che però può essere alzata o abbassata a seconda delle scelte dell'utente portando così un overclocking del Computer.

- **Descrizione dell'algoritmo di simulazione dell'unità ALU di base (74181) e delle relative architetture procedurali sviluppate (costrutti sequenziali, selettivi, iterativi ed eventuali annidamenti)**

Il codice detto in maniera veloce è semplicemente la copiatura dello schema logico fornito nei datasheet.

Nota: L'altra opzione sarebbe stata la ricreazione della ALU tramite la tavola di verità che infatti ho sviluppato ma l'ho trovata ma non sono stato in grado di applicarla nel sistema anche se avrei preferito perché rimuoveva una buona quantità di errori.

l'algoritmo sviluppato preleva i valori commutati dall'input decimale a binario essendo salvati nei registri, e procede richiedendo i vari selettori globali per tutte le 8 ALU, in base ai selettori scelti esegue le varie operazioni che sono già pre-settate nel sistema logico.

- **Descrizione dell'algoritmo di simulazione delle unità di registro PIPO di base (74198) e delle relative architetture procedurali sviluppate**

Vengono prelevati i valori da memorizzare e viene regolato, in base alle scelte prese durante l'interazione con l'utente, il passaggio successivo che i registri devono compiere, le scelte sui valori sono:

-cancellarli

-modificarli

-trasmetterli in output

I registri inoltre sono completamente divisi l'uno dall'altro e lavorano ognuno a se in maniera parallela.

- **Descrizione degli algoritmi di selezione e gestione dello streaming input/output e delle relative architetture procedurali sviluppate**

Per questa sezione è stato creato un sistema funzionante solo su terminale, contrariamente da quanto richiesto, nel quale il valore viene inserito in input tramite decimale per poi convertirlo in valori binari per la lavorazione, oppure direttamente inserirlo già in binario.

Per l'output invece il codice binario viene commutato in decimale e stampato sul terminale.

La consegna di creare parallelamente un sistema per la comunicazione tramite file testo non è stata creata per svariate ragioni tra cui la mancanza di voglia e capacità cognitiva nell'individuo a cui è stato affidato il compito e la poca competenza generale riguardante il collegamento di file.

- **Descrizione dell'algoritmo principale (kernel) e delle relative architetture procedurali sviluppate**

L'algoritmo richiede che tipologia di input vogliono essere inseriti dall'utente,vengono inseriti dei valori dall'utente in base alla scelta (quindi binario o decimale) e anche i 6 selettori per i calcoli delle ALU, se decimale i valori vengono inoltre tramutati in binario.

Le cifre dei 2 valori in stringa vengono poi divise in singole variabili intere e inserite nei vari registri.

I registri sono poi collegati alle varie entrate delle ALU che dopo aver eseguito il calcolo a seconda dei valori dei selettori inseriti e gli output prodotti vengono inseriti in altri 4 registri.

Infine i valori vengono ricavati, riuniti in un singolo valore stringa e convertiti in binario per essere letti dall'utente.

Il funzionamento è dettato completamente da valori di riferimento e senza sincronia data dal clock (a causa di mancanza di utilizzo di esso).

- **Descrizione (sintetica) della trama sintattica delle procedure del codice sorgente del simulatore e della sua organizzazione modulare (kernel code + librerie funzioni / moduli sorgente satellite)**

Utilizza uno schema modulare per la scrittura del codice e l'utilizzo di molte funzioni per il riutilizzo soprattutto di quelle complesse come i registri o l'ALU, tutto quanto viene riunito nel programma principale tramite l'adattamento dei vari codici.

Usa come tipologie di variabili Stringhe per le sezioni binarie, Long Long Int per quelle decimali e Int in entrambi i casi.

Occasionalmente vi è utilizzo di cicli ma generalmente il codice è relativamente grezzo.

- **Descrizione dei costrutti sintattici delle procedure del kernel code sorgente e delle funzioni in esso invocabili**

Il codice principale è diviso in 3 principali sezioni ossia:
-input che comprende la richiesta all'utente di come inserire i valori di a e b(in che modo e di che tipologia) e i valori stessi assieme ai selettori necessari poi nel calcolo.

Continua con la loro successiva commutazione in valori binari, grazie alla funzione BIN, qualora decimali.

questi 2 valori (a e b) vengono poi divisi grazie alla funzione DIV nei rispettivi 4 registri per variabile, funzione REG.

-calcolo che possiede le 8 ALU parallele tramite il richiamo della medesima funzione collegata tramite riporto (carry) e i rispettivi 4 registri dei valori in output anch'essi paralleli e semplicemente collegati alle ALU tramite variabili.

-output che comprende l'unificazione dei valori in uscita dai registri in un singolo valore a stringa con un codice interno al sorgente e la sua successiva commutazione in decimale per la visualizzazione con l'utente.

Nota: per ogni registro sarà necessaria un'interazione in tempo reale per il controllo dei valori salvati.

- **Descrizione dei costrutti sintattici delle procedure delle singole funzioni accessorie invocabili dal kernel code (come libreria di funzioni o moduli sorgente satellite)**

I principali costrutti delle funzioni sono le porte logiche anche a più ingressi, la ALU e i registri.

-La ALU è composta in ordine da dichiarazione di valori, lo schema logico diviso in 3 sezioni per semplificare la lettura e infine restituisce i prodotti sotto forma di variabili fisse.

-I registri sono invece composti dall'inserimento dei valori in un ciclo composto da 8 flip-flop rompibile solo se deciso dall'utente e la successiva richiesta/interazione con l'utente per decidere cosa fare con i dati salvati, le opzioni sono: clear/reset, overwrite e trasporto dei bit alla sezione successiva.

Altre funzioni secondarie sviluppate riguardano il collegamento del codice principale e sono:

-DEC/BIN + invertitore che sono dedicati alla conversione dei valori in binario a decimale e viceversa.

-DIV che separa ogni singola cifra da un numero rendendola una variabile a sé, poi le divide nei vari registri.

-Porte logiche Classiche e a più ingressi utilizzate per la creazione di registri e ALU.

- **Descrizione delle scelte di ottimizzazione del codice sorgente relative a memoria e tempo-macchina riservati al calcolo**

Le tecniche utilizzate per l'ottimizzazione del codice sorgente sono state unicamente la commutazione in funzioni delle varie sezioni qualora fosse possibile e l'utilizzo di variabili pointer.

Facendo così in modo, per esempio nell'utilizzo delle 8 ALU, basta semplicemente chiamare 8 volte la funzione ogni volta con valori diversi per ottenere i risultati e collegarle tramite il valore Cn+4 = Cn (sistema di carry).

Però generalmente è un codice poco ottimizzato.

- **Esito del testing esecutivo sulle opzioni operative dell'unità ALU 74181 e descrizione dei criteri di scelta dei dataset di input adottati**

Testing eseguito sulla singola funzione e con più funzioni parallele con esito apparentemente positivo (mantenimento di valori di 1 e 0), riscontrato però un problema nella sezione

finale probabilmente causata da qualche errore di logica durante la creazione del codice che causa la generazione di un risultato sbagliato.

Gli input scelti per il testing sono stati valori completamente casuali e decisi sul momento.

- **Risultati dei benchmark test su dataset di input-campione predefiniti (forniti successivamente ai team di sviluppo)**

Risultati ricavati non confrontati con quelli forniti.