**Progetto di reti Logiche**

Divisore intero con resto

Ergasti Alex

# Introduzione

Si vuole realizzare un circuito che esegue la divisione tramite il metodo della lunga divisione che abbia ingressi e uscite a 32 bit in codifica binaria naturale.

Il circuito deve, forniti in ingresso due valori a(dividendo) e b(divisore), fornire in uscita due valori q (quoziente) e r (resto) tale che: **a=b\*q+r**

### Ipotesi

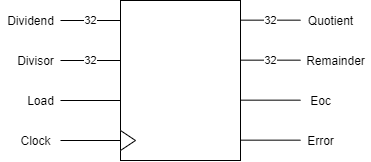
I valori forniti a e b sono numeri naturali positivi (>=0), è quindi necessario gestire il caso in cui il divisore è uguale a zero tramite opportuni segnali

### Limitazioni

Il metodo della lunga divisione è un’operazione lenta, richiederà quindi almeno 32 colpi di clock per essere eseguita dopo un primo colpo di clock di setup.

# Specifica

## Interfaccia del sistema



### Segnali di ingresso:

* **Dividend**: Indica il dividendo, segnale a 32 bit in codifica binaria naturale\*
* **Divisor**: Indica il divisore, segnale a 32 bit in codifica binaria naturale\*
* **Load**: Segnale di 1 bit attivo alto sincrono, permette al componente di eseguire una fase di “setup” e di configurarsi per operare la divisione
* **Clock**: Clock del sistema

### Segnali di uscita:

* **Quotient**: Indica il quoziente, segnale a 32 bit in codifica binaria naturale\*
* **Remainder**: Indica il resto della divisione, segnale a 32 bit in codifica binaria naturale\*
* **Eoc** (End Of Computation): Segnale di 1 bit attivo alto, se ha valore logico 0 allora il circuito non ha ancora prodotto l’uscita su **Quotient** e **Remainder**, se ha valore logico 1 allora il circuito ha prodotto e fornito su **Quotient** e **Remainder** il risultato.
* **Error**: Segnale di 1 bit, se ha valore logico 0 allora non sono presenti errori e il risultato fornito è considerato valido, se ha valore logico 1 allora gli ingressi non sono validi e l’uscita ha un valore non precisato.

### Utilizzo del componente:

Per utilizzare il componente è necessario posizionare sui segnali di ingresso **dividend e divisor** i segnali su cui si intende operare la divisione e alzare a valore logico alto il segnale di **load**.

Il primo colpo di clock ricevuto dopo aver correttamente configurato gli ingressi permetterà al circuito di configurarsi e portare a valore logico 0 il segnale **Eoc**, a questo punto è obbligatorio abbassare il segnale di **load** a valore logico 0, poiché lasciandolo a valore logico 1 il circuito rimarrebbe continuamente in fase di “setup”, non è invece importante il valore mantenuto sugli ingressi **dividend e divisor** dopo la fase di “setup”

Il circuito fornirà dopo 32 colpi di clock dal colpo di setup il risultato sui segnali di uscita **Quotient** e **Remainder** e alzerà il segnale **Eoc** per segnalare il completamento dell’operazione.

Finchè il segnale **Eoc** non assume valore logico 1 il significato di **Quotient** e **Remainder** non è da considerarsi significativo.

In caso di errore l’uscita **Error** assumerà valore logico 1, sarà 0 invece in tutti gli altri casi

## Architettura del sistema

L’architettura è formata da:

* 2x Parallel to Serial Shifter (32 bit)
* 1x Registro (32 bit)
* 1x Sommatore
* 1x C1 Complement
* 1x Controllore

## Parallel to Serial Shifter

Shifter a 32bit sincrono.

### Segnali di ingresso:

* **LD/**: Se valore logico 1 abilità il caricamento parallelo, se valore logico 0 abilità lo shifting
* **PI (Parellel Input):** segnale a 32 bit utilizzato se **LD/** = 1L
* **SI** **(Serial Input):** segnale ad 1 bit utilizzato se **LD/** = 0L, il bit in ingresso è relativo al LSB
* **Reset** pone il valore di tutti i bit all’interno del registro a 0, ha la precedenza su ogni altro segnale
* **EN** se = 1L lo shifter reagisce ai colpi di clock, se = 0L il registro non reagisce ai colpi di clock

### Segnali di uscita:

* **PO (Parallelo Output):** segnale a 32 bit di uscita parallela del contenuto del registro
* **SO (Serial Output):** segnale ad 1 bit che prende il valore del MSB del valore memorizzato all’interno del registro

### Utilizzo:

Viene utilizzato per due componenti, il NUM\_ParSerialShifter e il REM\_ParSerialShifter

#### NUM\_ParSerialShifter\_32:

Inizialmente contiene il valore del dividendo (segnale **dividend**), shifta verso sinistra (nello schema) facendo uscire da **SO** il MSB e tramite **SI** entra il valore del quoziente, alla fine dei 32 colpi di clock si troverà su questo shifter il valore finale del quoziente (segnale **Quotient**).

#### REM\_ParSerialShifter\_32:

Inizialmente contiene il valore 0, al suo interno viene salvato il risultato parziale del resto e dopo 32 colpi di clock conterrà il resto dell’operazione di divisione.

In base al valore del **COUT** del sommatore viene fatto shiftare oppure prende in ingresso parallelo il risultato della sottrazione.

## C1 Complement

Esegue il complemento ad 1 del segnale in ingresso e lo fornisce sull’uscita

### Segnali di ingresso:

* **In**: Segnale a 32bit

### Segnali di uscita:

* **Out**: Segnale a 32bit pari al valore in ingresso complementato a 1

## Register

Registro a 32 bit sincrono

### Segnali di ingresso:

* **In**: Segnale a 32bit pari al valore da memorizzare all’interno del registro
* **En:** se = 1L il registro reagisce ai colpi di clock, se = 0L il registro non reagisce ai colpi di clock

### Segnali di uscita:

* **Out**: Segnale a 32bit pari al valore memorizzato nel registro

### Utilizzo:

#### DEN\_Register\_32:

Registro a 32 bit che contiene il valore del divisore già complementato

## Adder

Sommatore a 32 bit.

### Segnali di ingresso:

* **A** e **B:** Segnali a 32 bit da sommare
* **CIN**: Riporto in ingresso, sempre a valore 1L nello schema

### Segnali di uscita:

* **Res**: Segnale a 32 bit, risultato della somma a 32 bit dei due addendi
* **COUT:** Segnale ad 1 bit contenente il riporto della somma

### Utilizzo:

Viene utilizzato per eseguire la somma tra il remainder e il dividend, ma, poiché il dividend è in complemento ad 1 e il valore di CIN è sempre posto ad 1 esegue di fatto la sottrazione.

## Controller

Genera i segnali di controllo dell’intera architettura

### Segnali di ingresso:

* **Divisor**: Segnale a 32 bit in codifica C1 contenente il divisore, influisce sul valore di **Error**
* **Load:** Segnale a 1 bit che resetta i segnali di uscita **Eoc** ed **En** rispettivamente a 0L e 1L
* **Carry:** Segnale ad 1 bit, influisce sul valore di uscita di **LD/**

### Segnali di uscita:

* **LD/**: Segnale ad 1 bit per selezionare se il registro “REM\_ParSerialShifter\_32” debba eseguire un’operazione di load oppure di shift, il valore di questo segnale dipende dall’ingresso **Carry**
* **En:** Segnale ad 1 bit che vale 0L se **Eoc** vale 1L, vale 1L durante tutta la fase di computazionee se **Load** vale 1L
* **Eoc:** Segnale ad 1 bit che assume valore logico 1 se l’operazione di divisione è conclusa, 0 altrimenti
* **Error:** Segnale ad 1 bit che assume valore logico 1 se l’operazione di divisione contiene un valore di dividendo (segnale **Dividend**) invalido, 0 altrimenti

# Verifica

Per la verifica del componente viene utilizzato un test bench che stimola il componente con vari segnali di ingresso.

Il clock scelto per il test bench è pari a 55ns, valore deciso tramite analisi post-route (statica e dinamica) dei vari componenti.

Il registro (RegShifter\_PS) presenta infatti un tempo CLK\_edge to PAD che varia tra i 10ns e i 14ns, invece l’RCA presenta un tempo di PAD to PAD da C0 a COUT di circa 38ns, sommando i tempi si ottiene un tempo di clock di 52ns, quindi è stato scelto un valore pari a 55ns per avere un margine di 3ns sul clock.

La somma dei ritardi del registro e dell’RCA determina quindi il valore del clock, poiché ogni altra serie di componenti ha ritardo minore di 55ns.

## Test-bench



## Casi d’uso

Il test bench permette di verificare tutte le condizioni di uso del componente, esegue infatti tutte le combinazioni di divisioni (grande/piccolo, piccolo/piccolo, grande/grande, piccolo/grande), verifica il corretto funzionamento del segnale EOC, che si alza correttamente dopo 32 colpi di clock dal colpo di setup, verifica il corretto funzionamento del segnale ERROR, che si alza quando il dividend ha valore 0 e verifica inoltre che sia possibili interrompere una divisione e caricarne un’altra alzando il segnale LOAD per rieseguire la fase di SETUP.