Algoritmi e Calcolatori A.A. 2018/2019
Progetto di Algoritmi

Sistema di simulazione reti logiche

1. Descrizione del problema

Un sistema di simulazione di circuiti digitali riceve la descrizione di un circuito ed una lista di vettori di input al fine di generare, per ogni vettore, il risultato fornito in uscita dal circuito, simulando le porte logiche e le interconnessioni. Quando il circuito in ingresso contiene elementi di memoria (Flip-Flop) il circuito viene simulato anche in funzione di un numero di colpi di clock fornito dall'utente.
L'obiettivo del progetto è la realizzazione di un simulatore di circuiti digitali che sia in grado di fornire diverse funzionalità:

1. Simulazione di circuiti combinatori
2. Simulazione di circuiti sequenziali
3. Analisi e misure sui circuiti.

2. Organizzazione del Sistema Informativo

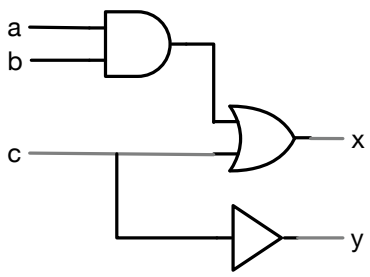
Il simulatore deve accettare due modalità di descrizione dei circuiti:

1. Circuiti semplici, descritti con soli componenti di base (porte logiche e flip-flop)
2. Circuiti composti, descritti tramite la composizione di più circuiti semplici connessi tra loro.

La descrizione di un circuito semplice si basa su una versione semplificata del VERILOG. Per ogni circuito semplice, vengono forniti: un'etichetta che lo identifichi univocamente, un ***insieme*** di segnali di ingresso e di uscita e la descrizione della logica combinatoria. Il formato del file che descrive un singolo circuito combinatorio è il seguente:

```
module NOME_CIRCUITO_COMB (
                input a, b
                input c
                output x, y
        );

        assign x = c OR (a AND B)
        assign y = NOT c
endmodule
```

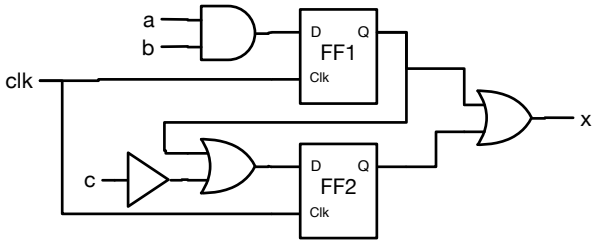


Dove la parola chiave **assign** è usata per impostare il risultato della connessione tra la logica combinatoria ed il singolo output. Nel caso di circuiti combinatori è vietato l'uso di feedback all'interno della definizione del circuito.

Quando invece il circuito contiene flip-flop (solo di tipo D) per implementare operazioni sequenziali è obbligatorio il segnale di clock (che è anche un input del circuito stesso). Il seguente esempio implementa un circuito sequenziale:

```
module NOME_CIRCUITO_SEQUENZIALE (
                        clk
                        input a, b
                        input c,
                        output x
                    );

        FF1 = a AND b
        FF2 = FF1 OR (NOT c)
        assign x = FF1 OR FF2
endmodule
```



In questo esempio, con la sigla `FF<numero>` si indentifica un singolo elemento di tipo flip-flop. Nel caso dei circuiti sequenziali, ciascun segnale accetta tre valori: `0`, `1` e `X`. Poiché il simulatore, per semplicità, non include i meccanismi di reset propri di un reale flip-flop, `X` è il valore che esprime l'output di un FF finché non viene inizializzato con un valore al colpo di clock opportuno.

I circuiti composti, di cui un esempio è fornito nella sezione successiva, devono essere definiti, senza la modalità libreria implementata, attraverso un unico file. In questo caso, ci si aspetta che il file contenga un circuito corretto se l'ultimo componente rappresenta quello composto e prima si trovano tutte le definizioni dei moduli usati. I file di input per le simulazioni, definiti in seguito, faranno rifermento agli input di quest'ultimo.

Specifiche Gruppi

Ai gruppi è richiesto di accettare anche input e output vettoriali, nel formato ereditato dal C: **`vettore[numero elementi]`**. In questo caso, l'input viene definito come nel seguente esempio:

```
input a[2]
```

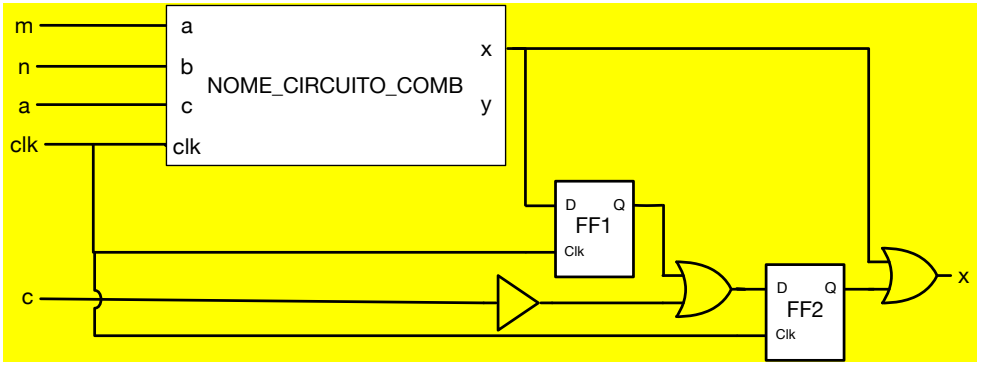
ed indica due segnali distinti indicizzabili come **`a[0]`** e **`a[1]`** all'interno della descrizione del circuito.
Inoltre, prevedere la possibilità di avere un flip-flop "completo", aggiungendo la seconda uscita negata come potenziale collegamento di uscita. Si scelga in fase di definizione delle specifiche come distinguere l'uso dell'uscita normale e di quella uscita negata.

Specifiche a partire dall'appello della sessione di Settembre 2019 per tutti i successivi

Sia ai gruppi che ai singoli è richiesto di predisporre una modalità aggiuntiva di funzionamento del simulatore, denominata "libreria". Il simulatore deve essere in grado di accettare diversi circuiti e memorizzarli al suo interno per poi essere in grado di accettare descrizioni di circuiti in modalità composita, come nell'esempio di seguito (riferito ai due circuiti precedentemente mostrati):

```
module NOME_CIRCUITO_COMPOSITO (
                        clk
                        input m, n
                        input a, c
                        output x
                );

        NOME_CIRCUITO_COMB instance (.a = m, .b = n, .c = a, .x = FF1)
        FF2 = FF1 OR (NOT c)
        assign x = FF1 OR FF2
endmodule
```



Dove la parola chiave **instance** specifica la presenza di un circuito combinatorio già definito in precedenza, mentre tra parentesi si assegnano i collegamenti tra ingressi ed uscite dei due circuiti. Il clock, per semplicità, viene assegnato automaticamente. Si noti che il carattere '.' davanti ad un input definito per il circuito combinatorio serve anche a disambiguare i nomi.
In questa modalità, tutte le verifiche di correttezza eventualmente indicate nel corso della definizione del progetto continuano a valere e, naturalmente, si aggiunge quella di congruenza con l'esistenza dei moduli utilizzati.
La modalità "libreria" deve essere gestita indicando un file di input che contenga, linea per linea, il percorso di tutti i file contenenti le descrizioni dei circuiti che si intendono utilizzare.

Specifiche a partire dall'appello della sessione di Gennaio/Febbraio 2020

Si richiede a gruppi e singoli di permettere all'utente di gestire la modalità libreria in modo da fornire anche funzionalità di modifica dell'insieme dei circuiti che formano la libreria. Questo significa poter aggiungere o rimuovere circuiti dalla libreria e, se necessario, salvare la nuova libreria in un file.

3. Esecuzione

Il programma è composto da due modalità di utilizzo:

1. Simulazione
2. Analisi

Il programma deve permettere all'utente di specificare, da riga di comando, il file di definizione del circuito. Dopo averlo letto e verificato la sua correttezza, un menù deve permettere all'utente di effettuare le seguenti operazioni:

1. **Simulazione**: un nuovo file di input viene chiesto all'utente. Il file deve contenere i vettori di input secondo il formato specificando, su ogni riga, la sequenza binaria dei valori di da assegnare ai segnali di ingresso seguendo l'ordine specificato nell'intestazione del circuito. *Ad esempio, per il circuito combinatorio visto in precedenza a pagina 1, una sequenza **010** indica che la simulazione deve applicare **0** all'ingresso **a**, **1** all'ingresso **b** e **0** all'ingresso **c**.* La simulazione deve tenere traccia, in modo ordinato, di tutti gli output prodotti dal circuito a fronte di ciascun vettore di ingresso e permettere all'utente di visualizzarli a schermo oppure di salvarli su file (in questo caso, permettere all'utente di specificare il nome del file di output della simulazione).
   Nel caso di circuiti sequenziali, la sequenza fornita per gli ingressi rappresenta il valore degli ingressi ogni singolo colpo di clock (un colpo ogni riga). Anche in questo caso, ci si aspetta che il simulatore sia in grado di tenere traccia dei vettori di output, legandoli al colpo di clock in cui sono stati generati.
2. **Analisi**: la funzionalità di analisi deve essere in grado di analizzare il circuito al fine di riportare alcune delle sue caratteristiche fondamentali:
   a. *Path Minimo*: indicare il percorso più corto esistente nel circuito. Il percorso è individuato da una coppia input-output.
   b. *Path Massimo*: similmente al path minimo, individua il percorso più lungo.
   c. *Coni Logici*: per ciascun output del circuito, devono essere indicati il set di input che contribuiscono a definirne il valore.

Specifiche Gruppi

d. *Analisi di potenza:* Sia consentito calcolare il consumo di potenza a partire dalla definizione, tramite file fornito dall'utente contenente il consumo di ogni elemento di base (porte logiche e FF) per il passaggio da 0 ad 1 e da 1 a 0. Tale consumo è da calcolare sull'intera esecuzione di una sequenza di pattern ed è dato dalla somma del contributo fornito da ciascun elemento (porte logiche o FF) a fronte dei dati in uscita. Il formato del file di input ricalca il seguente esempio:

```
# RIGA COMMENTO
FF;0.5;0.25
NOT;0.1;0.34
AND;0.01;0.94
```

in cui, per ogni riga, vengono indicati, separati dal carattere ';', l'etichetta del componente, il consumo in caso di transizione 0->1 e quello per la transizione opposta.

Specifiche a partire dall'appello della sessione di Settembre 2019 per tutti i successivi

Si preveda una ulteriore capacità di analisi denominata *Loop*: per un circuito sequenziale individuare il loop più lungo e quello più corto. Per loop si definisce un percorso che vada da un segnale di uscita verso l'ingresso di un componente posto prima nella catena logica.

4. Richieste e Descrizione dell'Elaborato

Tutte le funzionalità implementate devono fare riferimento ad una interfaccia testuale le cui linee guida di implementazione sono lasciate agli sviluppatori.

5. Istruzioni di consegna dell'Elaborato

I gruppi di lavoro possono essere composti da un minimo di 1 ad un massimo di 3 persone.
Per sottomettere l'elaborato, occorre creare un archivio zip con il nome

`Prog2019_<matricole>.zip`

In cui il campo matricole deve essere sostituito con l'elenco (ordinato) delle matricole dei partecipati, separate dal carattere '_'. Il file dovrà essere caricarlo attraverso il portale della didattica entro le date di scadenza pubblicate di volta in volta per ogni appello disponibile. La creazione del file .zip che non rispetti le specifiche di consegna, **influirà negativamente** sulla valutazione.

L'archivio zip **deve** essere organizzato secondo le seguenti cartelle:

- *src*: contiene tutti i file sorgenti (ivi inclusi anche eventuali file di progetto utili alla ri-compilazione);
- *input*: contiene tutti i file di input utili (devono essere diversi dagli esempi pubblicati);
- *doc*: contiene una relazione dettagliata (pdf) su:
  - struttura del programma
  - algoritmi e strutture dati utilizzati
  - analisi dettagliata della complessità di tutti gli algoritmi sviluppati.

Verrà programmata una sessione di sottomissione/discussione dei progetti in corrispondenza degli appelli d'esame (comunicata attraverso il portale della didattica). I gruppi che intendono consegnare l'elaborato dovranno sottometterlo entro una specifica data, stabilita per ogni appello, e procedere alla prenotazione dell'orale. La prenotazione avverrà tramite procedura pubblicata sul portale della didattica. L'orale sarà finalizzato alla verifica del corretto funzionamento del programma ed alla discussione del progetto, in ogni sua parte, con ciascun componente del gruppo.
La data di consegna massima è fissata per l'ultima sessione dell'A.A. 2018/2019. Dopo tale data si dovrà fare riferimento alle regole decise per il corso in essere. Non saranno fatte eccezioni di nessun genere.

6. Metodo di valutazione

Il voto finale sarà unico per tutto il gruppo e sarà valido per ogni componente del gruppo.

Per l'elaborato, i criteri di valutazione comprendono:

- Rispetto delle specifiche;
- Algoritmi e strutture dati utilizzati;
- Flessibilità delle soluzioni adottate;
- Progettazione (es: organizzazione in classi, meccanismi di ereditarietà, ecc.);
- Gestione degli errori;
- Completezza delle soluzioni adottate;
- Accuratezza nella generazione di un numero opportuno di casistiche di test (set di file di input per la verifica del programma);
- Aderenza al linguaggio di programmazione (es: uso della STL, const reference, ecc.).

Nel caso in cui il progetto non venga ritenuto sufficiente, al gruppo sarà richiesto di rivedere l'intero progetto e sottometterlo nuovamente in una sessione successiva. La non sufficienza del progetto genera un decremento del voto finale massimo raggiungibile: ogni rifiuto diminuisce il voto massimo raggiungibile di 2 punti (es: progetto non sufficiente una volta sola, voto massimo 28; due volte, voto massimo 26).