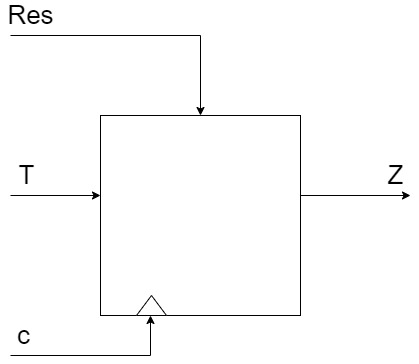
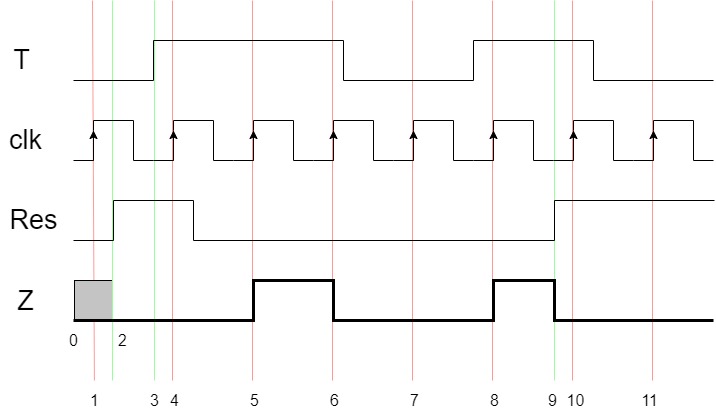
# Progetto di Un Flip Flop di tipo T sincronizzato sul fronte di salita del clock e dotato di Reset Asincrono attivo alto

Individuare i segnali coinvolti:

Visto dall’esterno il sistema si presenta dotato di 3 ingressi e di un’uscita:  
- un segnale a livelli *T* che quando alto abilita la commutazione del FlipFlop  
- un segnale *clk* i cui fronti di salita scandiscono gli istanti nei quali il FF viene a commutare  
- un segnale a livelli R che quando attivo riporta (senza attendere il sincronismo) il FF ad uno stato stabile con uscita bassa

.

Descrizione sintetica   
Comprendere pienamente il funzionamento del dispositivo, meglio se attraverso un esempio che evidenzi un ipotetico segnale di stimoli (composto dai segnali T, clk, Res) e la risposta attesa:



1. All’istante 0, ovvero appena acceso il circuito, il dispositivo si mette in uno stato casuale ignoto all’utilizzatore (in genere contrassegnato con ‘X’).
2. All’istante 1: viene rilevato un fronte di clock, ma essendo il segnale T basso il dispositivo non altera l’uscita (che comunque rimane sconosciuta).
3. All’istante 2: si alza il segnale di reset (asincrono) ed il circuito senza aspettare alcuna abilitazione da parte del segnale di sincronismo porta l’uscita allo stato basso nel più breve tempo possibile
4. All’istante 3: si alza il segnale T ma l’uscita non commuta fintanto che non rileva la presenza di un evento di sincronismo
5. All’istante 4: il segnale T è allo stato alto, siamo in presenza di un istante di sincronismo (fronte del clock), ma il segnale di Reset è ancora alto e pertanto il sistema continua a mantenere l’uscita allo stato basso
6. All’istante 5: Il segnale T è alto, il reset si è disattivato, siamo in presenza ad un istante di sincronismo, quindi l’uscita commuta e si porta a livello alto.
7. All’istante 6: (idem) T è alto, siamo in presenza ad un istante di sincronismo e l’uscita commuta portandosi a livello basso
8. All’istante 7: siamo in presenza di un sincronismo, ma il segnale T è basso e pertanto Non vi è commutazione dell’uscita
9. All’istante 8: T è alto, siamo in presenza di un fronte di clock, quindi vi è una commutazione dell’uscita
10. All’istante 9: Si alza il segnale di Reset e senza attendere alcuna abilitazione il dispositivo porta nel più breve tempo possibile il segnale d’uscita a livello basso
11. Pur essendo in presenza di un sincronismo il Reset è alto e l’uscita viene “forzata” al livello basso
12. Pur essendo in presenza di un sincronismo il Reset è alto e l’uscita viene “forzata” al livello basso

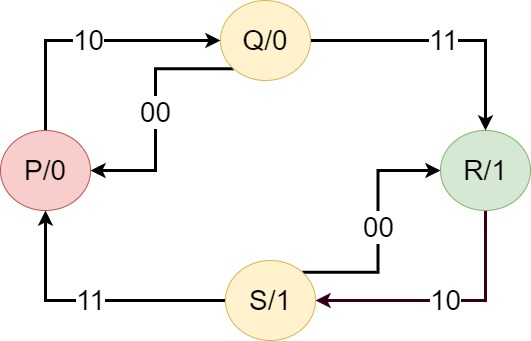
Evidenziare i meccanismi di funzionamento

Iniziamo a studiare il funzionamento “senza” considerare il segnale di reset che potrà essere agevolmente integrato alla fine.

La commutazione sia da 0 a 1 che da 1 a 0 avviene sempre con una particolare condizione, ovvero quando il segnale T è alto e arriva un FRONTE del clock. Questo si può evidenziare con una particolare sequenza degli ingressi: ovvero definiti “T” è il segnale di abilitazione e “c” è il segnale di clock la sequenza che fa commutare l’uscita è:

**TC prende il valore 10 seguito immediatamente dal valore 11**

Quindi, evidenziati due stati nei quali l‘uscita del FF è mantenuta a livello logico rispettivamente alto o basso (P e R) il passaggio dall’uno all’altro avviene attraverso una coppia di stati intermedi (Q ed S) che mantenendo l’uscita allo stesso valore dello stato di partenza memorizzino se sia stata rilevata la prima condizione della sequenza di commutazione (ovvero sugli ingressi è stata rilevata la combinazione 10). Se poi a questa segue la combinazione 11 la commutazione si completa, se in alternativa si presentasse qualche altra combinazione (e la sola possibile è 00) il sistema ritorna nello stato iniziale.



Descrizione completa attraverso Tavola di Huffman

Si può ora agevolmente descrivere il circuito attraverso una tavola di Huffmann che evidenzi le transizioni da eseguire per ciuscuno stato e per tutte le possibili combinazioni degli ingressi:

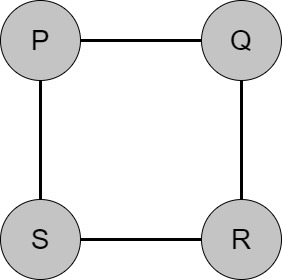
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Stato\Tc | 00 | 01 | 11 | 10 | Z |
| P | P | P | P | Q | 0 |
| Q | P | - | R | Q | 0 |
| R | R | R | R | S | 1 |
| S | R | - | P | S | 1 |

In Verde sono evidenziati gli stati stabili. Si noti in particolare

1. Dagli stati P ed R si esce solo quando viene rilevato sugli ingressi la prima combinazione della sequenza di commutazione (10)
2. Dagli stati Q ed S si prosegue completando la commutazione dell’uscita solo se la sequenza si completa con la seconda combinazione (11). Se la sequenza si interrompesse (00) si ritorna allo stato di partenza
3. Negli stati Q es S non può presentarsi la combinazione di ingresso 01 poiché questa richiederebbe la commutazione simultanea dei due ingressi.

Scelta di un’opportuna codifica degli Stati

Si può notare agevolmente attraverso il grafo delle transizioni che per evitare corse critiche basta che in fase di codifica si garantisca l’adiacenza delle seguenti coppie di stati: PQ, QR, RS, SP.



Quindi una possibile codifica degli stati che utilizzi solo due bit potrebbe essere ad esempio

P: 00  
Q: 01  
R: 11  
S: 10

Tavola di Flusso e realizzazione

Utilizzando la codifica scelta la tavola di flusso del sistema diventa:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| y1y2\Tc | 00 | 01 | 11 | 10 | Z |
| 00 | 00 | 00 | 00 | 01 | 0 |
| 01 | 00 | - | 11 | 01 | 0 |
| 11 | 11 | 11 | 11 | 10 | 1 |
| 10 | 11 | - | 00 | 10 | 1 |

Il che ci porta ad individuare le seguenti funzioni

Per la variabile y1:

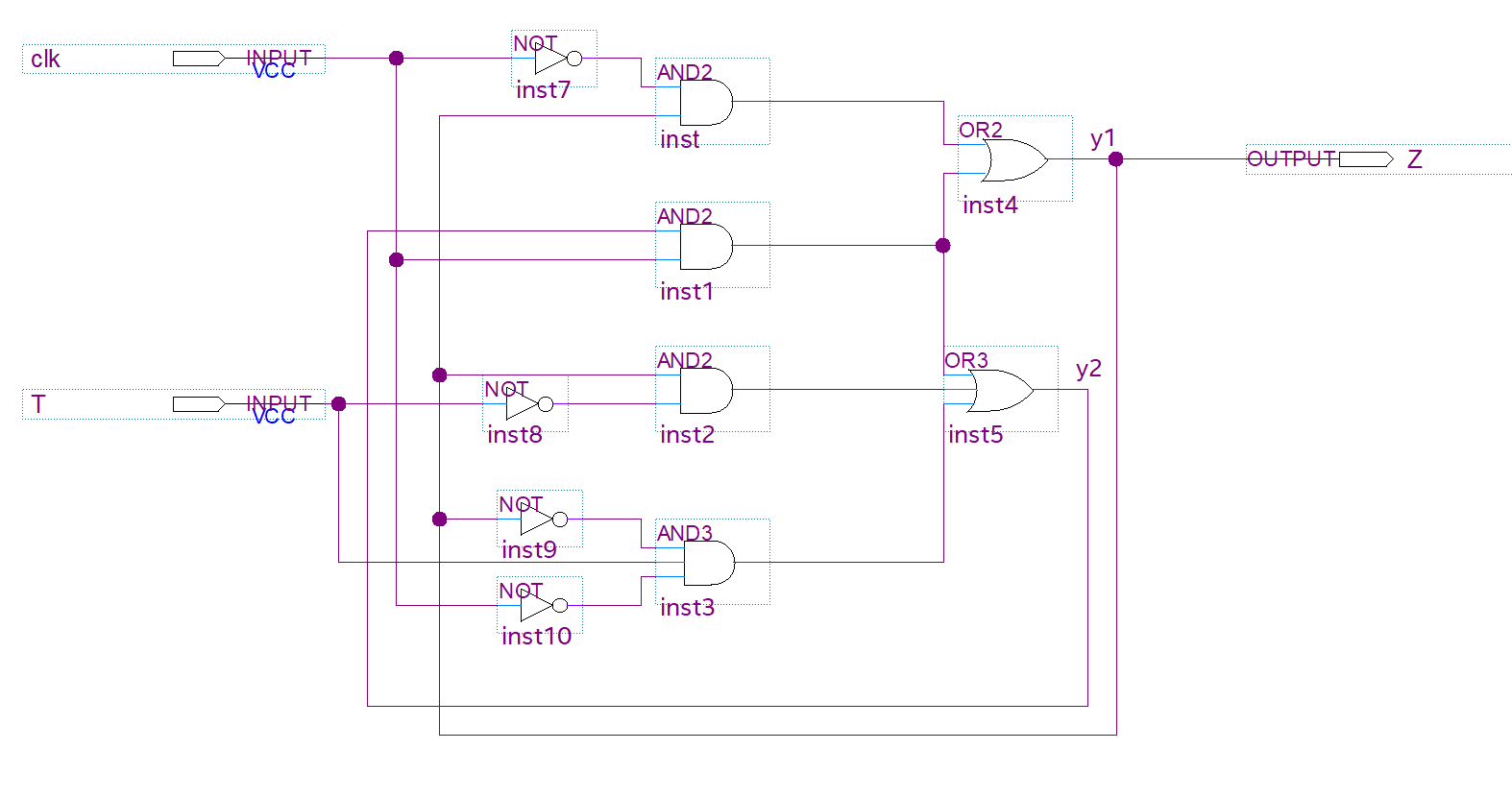
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| y1y2\Tc | 00 | 01 | 11 | 10 |  |
| 00 | 0 | 0 | 0 | 0 |  |
| 01 | 0 | - | 1 | 0 |  |
| 11 | 1 | 1 | 1 | 1 |  |
| 10 | 1 | - | 0 | 1 |  |

Per la variabile y2:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| y1y2\Tc | 00 | 01 | 11 | 10 |  |
| 00 | 0 | 0 | 0 | 1 |  |
| 01 | 0 | - | 1 | 1 |  |
| 11 | 1 | 1 | 1 | 0 |  |
| 10 | 1 | - | 0 | 0 |  |

Per l’uscita Z:

|  |  |
| --- | --- |
| y1y2 | Z |
| 00 | 0 |
| 01 | 0 |
| 11 | 1 |
| 10 | 1 |

Uno schema del circuito che potrebbe realizzare dette funzioni è il seguente: 

Considerazione

Un circuito siffatto, anche se funzionante da un punto di vista pratico, non potrebbe essere simulato con soddisfazione. Ciò è legato al fatto che sebbene perfettamente funzionante non vi è alcun meccanismo in grado di forzare il sistema in uno stato predefinito e noto.  
Ad esempio supponendo che entrambi gli ingressi siano a 0 non vi è modo di discriminare se il circuito si trova nello stato P (con uscita 0) o nello stato R (con uscita 1). Il simulatore indica questa ambiguità con il valore X (unknown). Se poi viene applicata la sequenza 10 + 11 il circuito cambia di stato, ma ancora una volta non si può sapere se da P sia andato in R attraverso Q oppure da R sia andato in P attraverso S.  
Per una corretta simulazione è pertanto fondamentale avere un meccanismo che forzi il sistema in uno stato noto attraverso un opportuno segnale (Reset). Per inciso anche per una corretta integrazione del circuito fisico con altri dispositivi è bene vi sia un meccanismo atto a mettere il dispositivo in una configurazione nota e certa.

Introduzione del segnale di reset

Introdurre alla fine il segnale di reset può sembrare controintuitivo, ma come vedremo anche se lo avessimo introdotto dall’inizio della procedura il risultato sarebbe rimasto il medesimo:  
  
Rivediamo i passi che avremmo dovuto compiere se avessimo progettato da subito un circuito dotato di reset asincrono

Tavola di Huffman:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Stato\RTc | 000 | 001 | 011 | 010 | 100 | 101 | 111 | 110 | Z |
| P | P | P | P | Q | P | P | P | P | 0 |
| Q | P | - | R | Q | - | - | - | P | 0 |
| R | R | R | R | S | P | P | P | - | 1 |
| S | R | - | P | S | - | - | - | P | 1 |

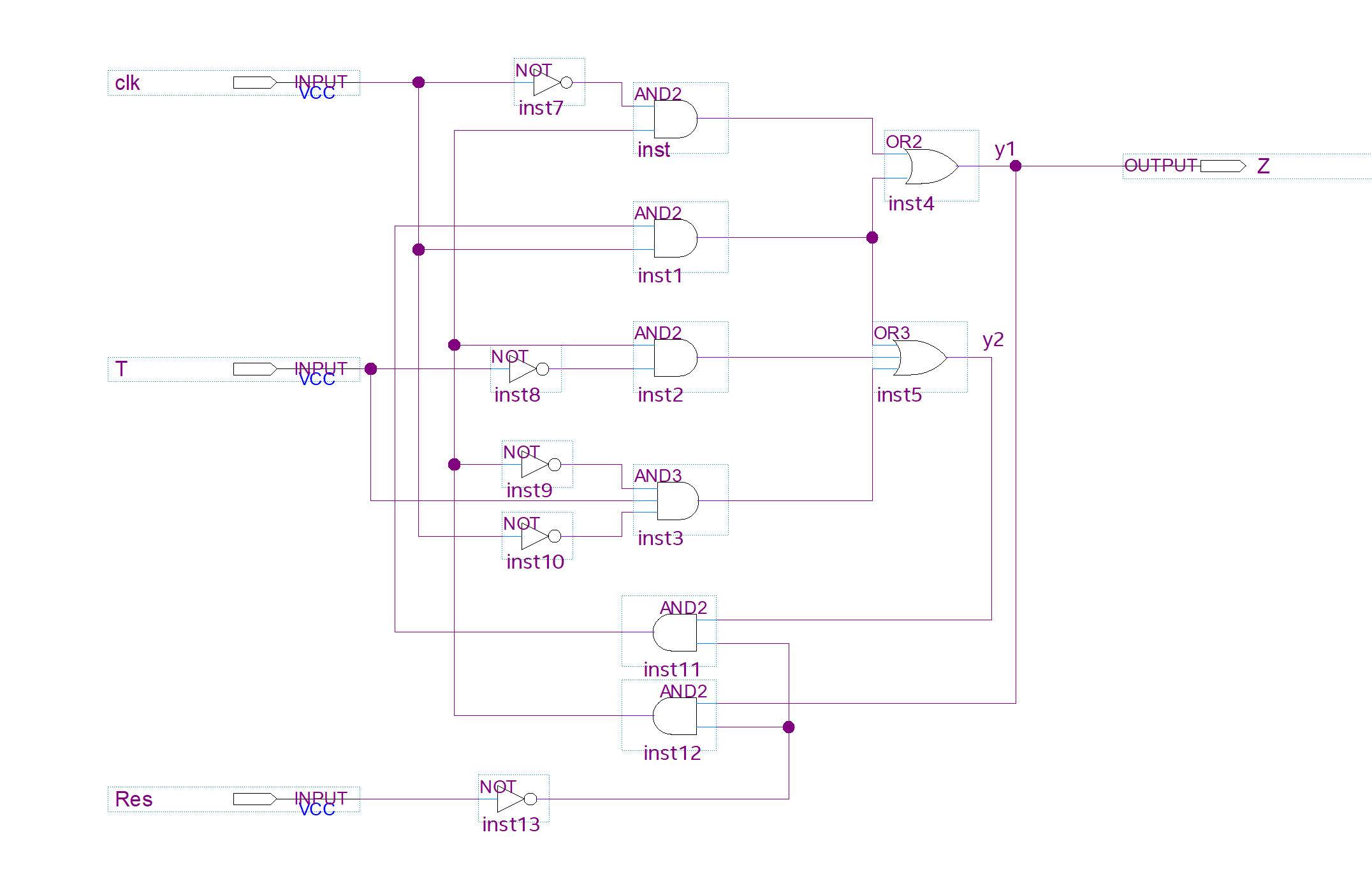
In essa è stata aggiunta la parte a destra che forza il dispositivo, ogni qualvolta si alzi il segnale R allo stato P nel quale permane fintanto che il segnale R non si sia abbassato. La parte sinistra è rimasta inalterata.

Adottando poi la codifica degli stati utilizzata sopra troveremmo la seguente tavola di flusso:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| y1y2\RTc | 000 | 001 | 011 | 010 | 100 | 101 | 111 | 110 | Z |
| 00 | 00 | 00 | 00 | 01 | 00 | 00 | 00 | 00 | 0 |
| 01 | 00 | - | 11 | 01 | - | - | -- | 00 | 0 |
| 11 | 11 | 11 | 11 | 10 | 00 | 00 | 00 | - | 1 |
| 10 | 11 | - | 00 | 10 | - | - | 00 | 00 | 1 |

Che in pratica è identica a prima per la parte sinistra mentre è riempita solo da valori uguali a zero nella parte destra. Questo fa sì che dopo la semplificazione con Karnaugh si trovino equazioni molto simili a quanto trovato precedentemente, ma dove ciascun implicante è moltiplicato per not(R)

Il che circuitalmente equivale ad intercettare i due loop di reazione relativi ad Y1 ed Y2 ed aggiungervi una logica atta a forzare entrambi al valore nullo.



Simulazione

Il circuito sopra rappresentato può venir descritto in linguaggio Verilog HDL:

module FFT\_ResAs(

clk,

T,

Res,

Z

);

input wire clk;

input wire T;

input wire Res;

output wire Z;

wire y1;

wire y2;

wire WIRE\_0;

wire WIRE\_15;

wire WIRE\_2;

wire WIRE\_16;

wire WIRE\_6;

wire WIRE\_7;

wire WIRE\_8;

wire WIRE\_17;

wire WIRE\_10;

wire WIRE\_11;

wire WIRE\_12;

assign WIRE\_10 = WIRE\_0 & WIRE\_15;

assign WIRE\_17 = WIRE\_2 & clk;

assign WIRE\_8 = ~clk;

assign WIRE\_2 = y2 & WIRE\_16;

assign WIRE\_15 = y1 & WIRE\_16;

assign WIRE\_16 = ~Res;

assign WIRE\_11 = WIRE\_15 & WIRE\_6;

assign WIRE\_12 = WIRE\_7 & T & WIRE\_8;

assign y1 = WIRE\_17 | WIRE\_10;

assign y2 = WIRE\_11 | WIRE\_12 | WIRE\_17;

assign WIRE\_0 = ~clk;

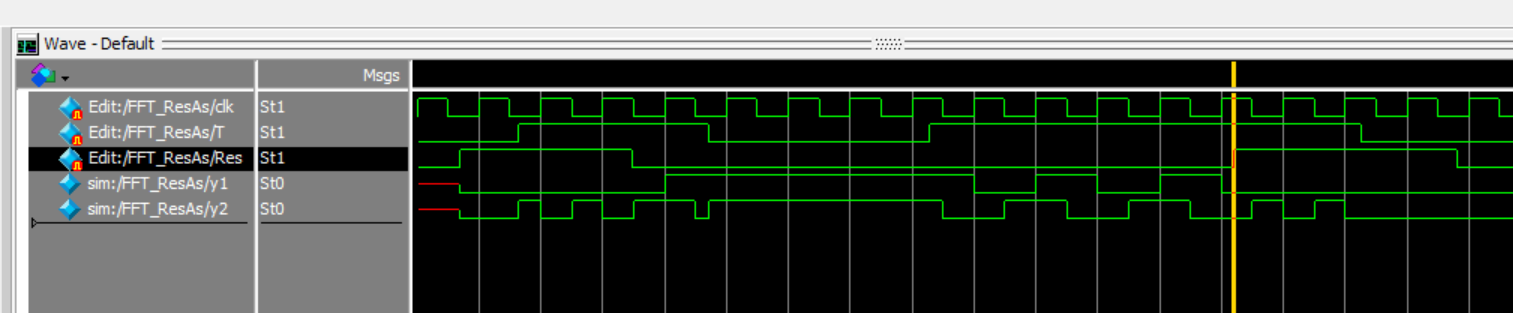
assign WIRE\_6 = ~T;

assign WIRE\_7 = ~WIRE\_15;

assign Z = y1;

endmodule

Che opportunamente simulato produce il seguente risultato:



Che sembra funzionare correttamente, vi è peraltro da notare che il circuito descritto è essenzialmente ideale e tutte le operazioni vengono svolte in un tempo praticamente nullo. Volendo fare una simulazione più aderente alla realtà si potrebbero introdurre degli ipotetici tempi di ritardo sulle varie porte logiche:

module FFT\_ResAs(

clk,

T,

Res,

Z

);

input wire clk;

input wire T;

input wire Res;

output wire Z;

wire y1;

wire y2;

wire WIRE\_0;

wire WIRE\_15;

wire WIRE\_2;

wire WIRE\_16;

wire WIRE\_6;

wire WIRE\_7;

wire WIRE\_8;

wire WIRE\_17;

wire WIRE\_10;

wire WIRE\_11;

wire WIRE\_12;

assign #1 WIRE\_10 = WIRE\_0 & WIRE\_15;

assign #1 WIRE\_17 = WIRE\_2 & clk;

assign #1 WIRE\_8 = ~clk;

assign #1 WIRE\_2 = y2 & WIRE\_16;

assign #1 WIRE\_15 = y1 & WIRE\_16;

assign #1 WIRE\_16 = ~Res;

assign #1 WIRE\_11 = WIRE\_15 & WIRE\_6;

assign #1 WIRE\_12 = WIRE\_7 & T & WIRE\_8;

assign #1 y1 = WIRE\_17 | WIRE\_10;

assign #1 y2 = WIRE\_11 | WIRE\_12 | WIRE\_17;

assign #1 WIRE\_0 = ~clk;

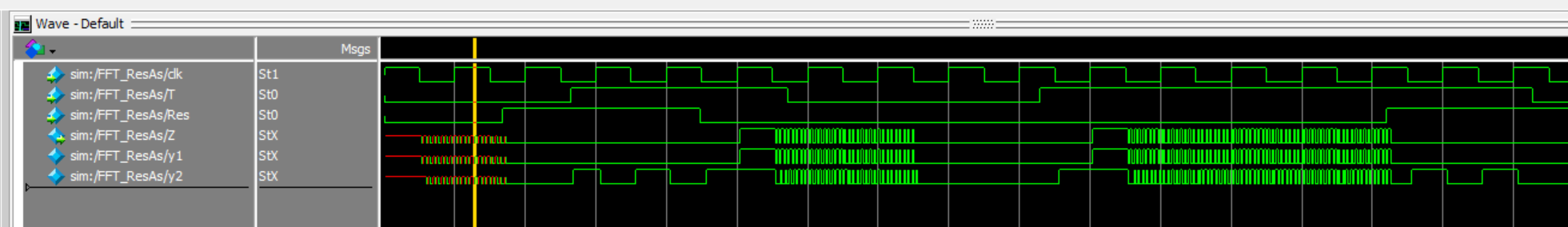
assign #1 WIRE\_6 = ~T;

assign #1 WIRE\_7 = ~WIRE\_15;

assign Z = y1;

endmodule

Circuito che quando simulato porta a risultati ben diversi da quelli attesi:



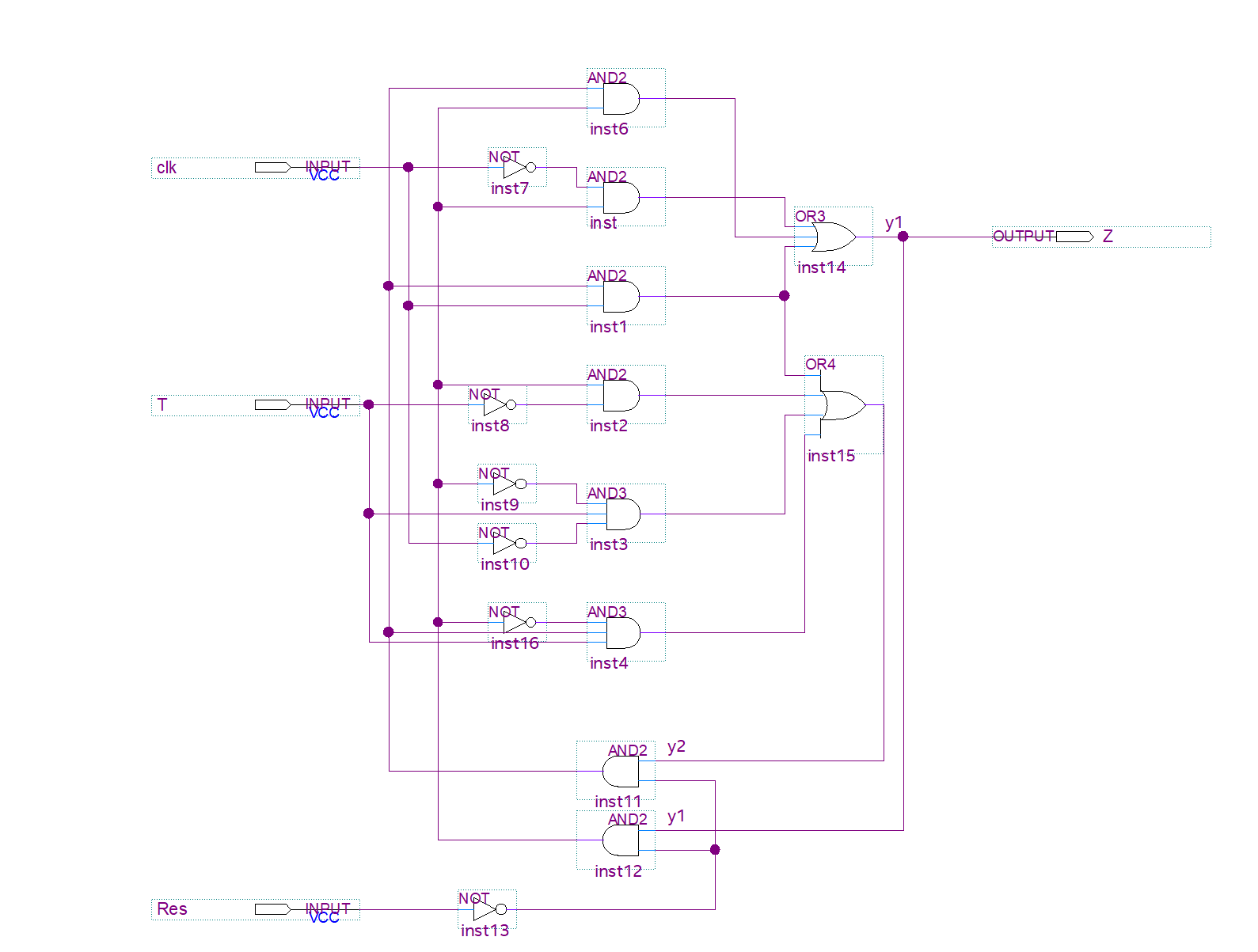
Ciò è legato alla presenza di “alee statiche” delle quali non avevamo tenuto conto in fase di progetto.  
Ripartendo pertanto dalla tavola di flusso e dalla realizzazione delle variabili di stato si vede che la soluzione più idonea sarebbe l’aggiunta di ulteriori due implicanti:

Per la variabile y1:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| y1y2\Tc | 00 | 01 | 11 | 10 |  |
| 00 | 0 | 0 | 0 | 0 |  |
| 01 | 0 | - | 1 | 0 |  |
| 11 | 1 | 1 | 1 | 1 |  |
| 10 | 1 | - | 0 | 1 |  |

Per la variabile y2:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| y1y2\Tc | 00 | 01 | 11 | 10 |  |
| 00 | 0 | 0 | 0 | 1 |  |
| 01 | 0 | - | 1 | 1 |  |
| 11 | 1 | 1 | 1 | 0 |  |
| 10 | 1 | - | 0 | 0 |  |

Il circuito potrebbe quindi diventare ad esempio:  


Circuito che opportunamente descritto in VerilogHDL con l’aggiunta degli ipotetici ritardi sulla logica

module FFT\_ResASv2(

clk,

T,

Res,

Z

);

input wire clk;

input wire T;

input wire Res;

output wire Z;

wire y1;

wire y2;

wire WIRE\_0;

wire WIRE\_22;

wire WIRE\_23;

wire WIRE\_24;

wire WIRE\_5;

wire WIRE\_25;

wire WIRE\_7;

wire WIRE\_9;

wire WIRE\_10;

wire WIRE\_11;

wire WIRE\_14;

wire WIRE\_15;

wire WIRE\_16;

wire WIRE\_17;

assign #1 WIRE\_7 = WIRE\_0 & WIRE\_22;

assign #1 WIRE\_25 = WIRE\_23 & clk;

assign #1 WIRE\_16 = ~clk;

assign #1 WIRE\_23 = y2 & WIRE\_24;

assign #1 WIRE\_22 = y1 & WIRE\_24;

assign #1 WIRE\_24 = ~Res;

assign #1 y1 = WIRE\_5 | WIRE\_25 | WIRE\_7;

assign #1 y2 = WIRE\_25 | WIRE\_9 | WIRE\_10 | WIRE\_11;

assign #1 WIRE\_17 = ~WIRE\_22;

assign #1 WIRE\_11 = WIRE\_22 & WIRE\_14;

assign #1 WIRE\_9 = WIRE\_15 & T & WIRE\_16;

assign #1 WIRE\_10 = WIRE\_17 & WIRE\_23 & T;

assign #1 WIRE\_5 = WIRE\_23 & WIRE\_22;

assign #1 WIRE\_0 = ~clk;

assign #1 WIRE\_14 = ~T;

assign #1 WIRE\_15 = ~WIRE\_22;

assign Z = y1;

endmodule

Porta a dei risultati soddisfacenti:

