EsD2: Costruzione di D-Latch, contatori e shift-register

Gruppo 1.AC
Matteo Rossi, Bernardo Tomelleri

15 aprile 2022

1 Misura componenti dei circuiti

Riportiamo per completezza il valore della tensione continua di alimentazione per i circuiti integrati misurata con il multimetro

$$V_{CC} = 4.99 \pm 0.03\text{V}$$

e il valore di capacità del condensatore di disaccoppiamento che collega le linee di alimentazione a massa (sempre misurato con il multimetro)

$$C_d = 97 \pm 4\ \text{nF}$$

2 D-Latch con Enable

2.a Costruzione del circuito

Si è costruito un circuito D-Latch secondo lo schema mostrato in fig. 1 utilizzando le porte NAND di due integrati SN74LS00.

Per studiarne il comportamento generiamo nei due pin DIO 0 (DATA) e DIO 1 (ENABLE) dell'AD2 due segnali di clock di frequenza $f = 1$ kHz e sfasati tra loro di 90° agli ingressi D ed E del circuito.

2.b Analisi del funzionamento del circuito

Il circuito è composto da un Latch RS i cui ingressi sono collegati a due porte NAND, di cui un ingresso per ciascuna è collegato all'input E, mentre gli altri due ingressi sono collegati l'uno al segnale opposto dell'altro tramite una porta NOT (in figura la porta NAND più in alto tra le due (R) è collegata all'input D, mentre quella più in basso (S) a $\overline{D}$).

L'equazione fondamentale del circuito è quindi data dalla

$$Q(t + \Delta t) = \overline{\overline{(D \cdot E)}} + \overline{(\overline{D} \cdot E)} \cdot Q(t) = E \cdot D + \overline{E} \cdot Q(t) \tag{1}$$

da cui si può ricavare la corrispondente tabella di verità

Come si può vedere dalla tabella di verità (tabella 1) l'uscita Q funge da memoria a un bit se E è al livello logico basso (stato di HOLD), mentre assume il valore logico dell'input D quando il segnale di ENABLE è acceso. Questo rende il valore dell'uscita indipendente dalle caratteristiche temporali delle porte NAND e protegge il circuito dallo stato proibito di oscillazione/racing $R = S = 1$ da cui è affetto il semplice RS-Latch.

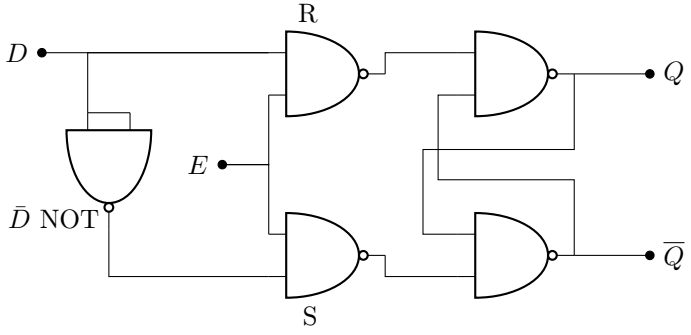

Figura 1: Schema logico del circuito D-Latch (con Enable) realizzato

E	D	$Q(t)$	$Q(t+\Delta t)$
0	X	0	0
0	X	1	1
1	0	X	0
1	1	X	1

Tabella 1: Tabella di verità del circuito D-Latch con Enable (con X si indica valore logico indefinito/don't care)

2.c Verifica della tabella di verità del Latch

Per conferma del corretto funzionamento del Latch possiamo confrontare le uscite ottenute da un'acquisizione con Logic Analyzer con i valori riportati in tabella 1 inviando all'ingresso del circuito con Patterns due segnali di clock, di modo che E sia lo stesso segnale in D, a cui si è però aggiunta una fase $\varphi = \pm 90°$.

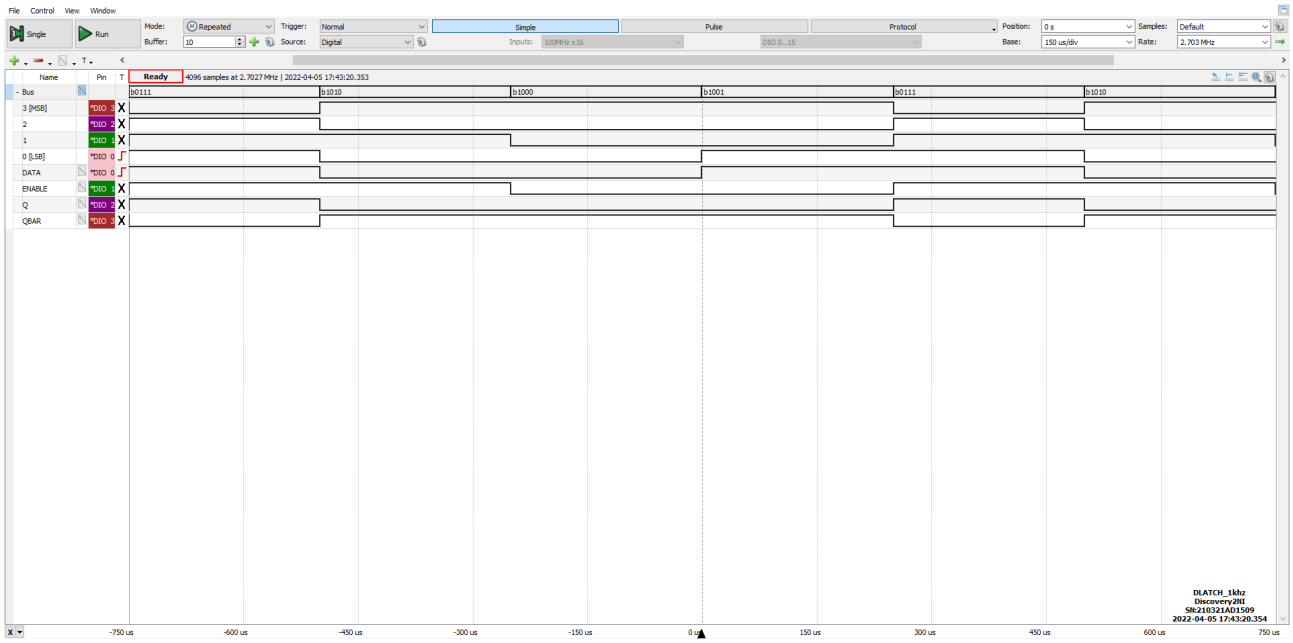

Figura 2: Acquisizione di un ciclo completo (frequenza 1 kHz) con Logic Analyzer dei segnali in ingresso (D = DIO 0, E = DIO 1) e in uscita (Q = DIO 2, $\overline{Q}$ = DIO 3) dal D-Latch.

Dalle fig. 2 e fig. 3 si osserva come durante lo stato basso di Enable il segnale in uscita rimanga costante rispetto a variazioni del segnale in D, mentre quando $E = 1 \implies Q(t+\Delta t) = D$ coerentemente con quanto previsto dalla tabella di verità e come principio di funzionamento della memoria a 1 bit.

2.d Misura dei tempi di propagazione nelle transizioni di stato

Si riescono a distinguere due diverse transizioni dei segnali in ingresso per ciascun valore di sfasamento tra i due segnali di clock in D ed E; per $\varphi = 90°$:

1. $D \coloneqq 0$, $E : 0 \to 1$
2. $D : 0 \to 1$, $E \coloneqq 1$.

   Mentre per $\varphi = -90° = 270$ °:

3. $D : 1 \to 0$, $E \coloneqq 1$
4. $D \coloneqq 1$, $E : 0 \to 1$.

Si sono misurati i ritardi tra la transizione dei segnali in ingresso e i corrispondenti cambiamenti di stato in uscita su scala dei tempi minima (10 ns) con i cursori dalle acquisizioni con Logic Analyzer e per riconferma anche con l'oscilloscopio da banco (a cui associamo come incertezza il contributo dato dalle specifiche del datasheet, tenendo conto dell'instabilità e dello spessore delle tracce sullo schermo). Riportiamo di seguito e nella fig. 4, fig. 5, fig. 6 e fig. 7 i risultati ottenuti.

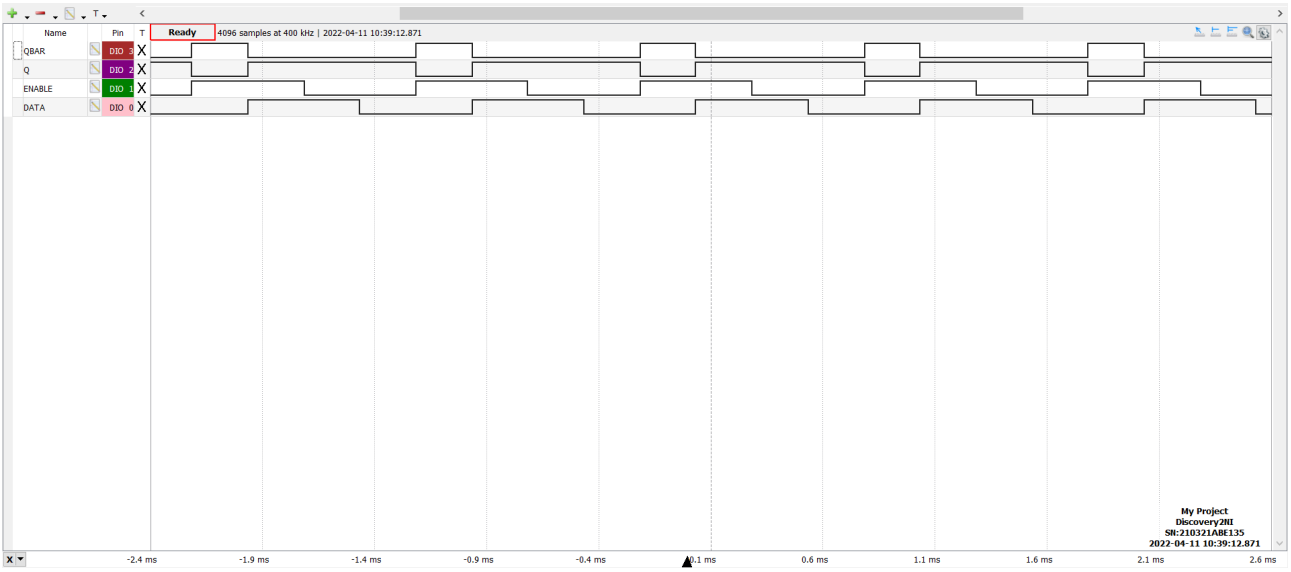

Figura 3: Acquisizione con Logic dell'andamento temporale dei segnali in ingresso uscita dal D-Latch per $\varphi = -90°$.

1. $t_{PHL} = 30 \pm 10$ ns
2. $t_{PLH} = 20 \pm 10$ ns
3. $t_{PHL} = 40 \pm 10$ ns
4. $t_{PLH} = 30 \pm 10$ ns

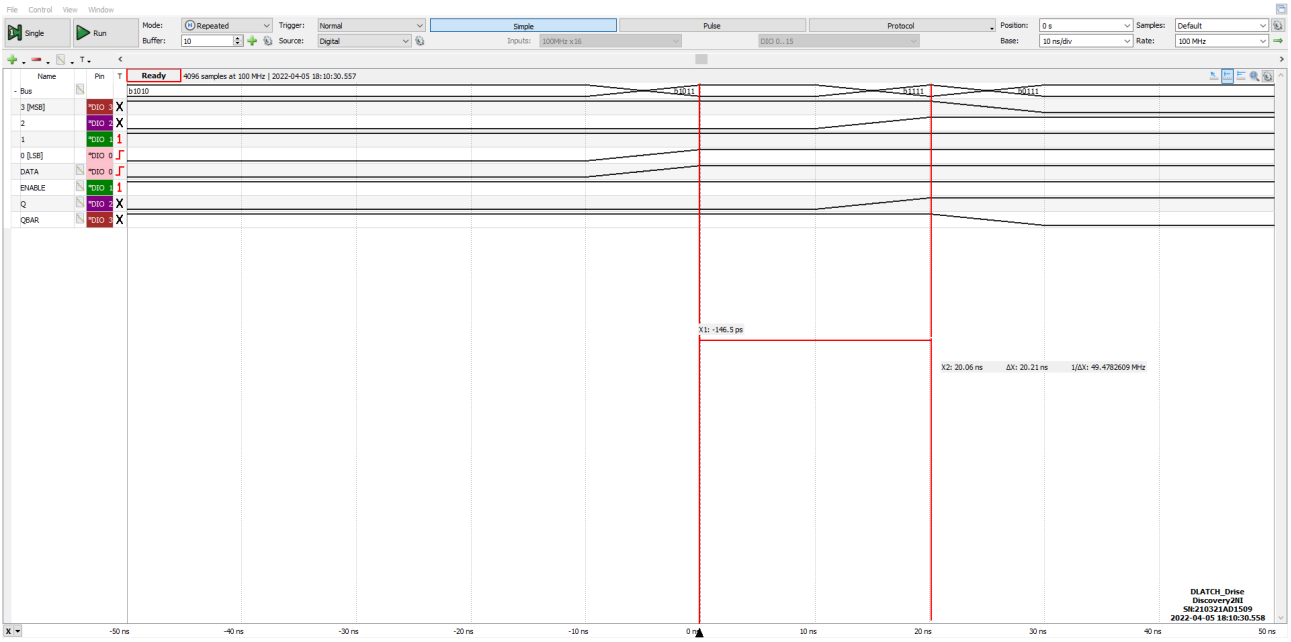

Figura 4: Acquisizione del Logic Analyzer durante la transizione 2 del D-Latch

Il massimo ritardo indotto si è trovato in corrispondenza della transizione 3 dell'input D da alto a basso (con i cursori dell'oscilloscopio $t_{PHL} = 35 \pm 2$ ns) mentre il minimo per la 2 (con oscilloscopio $t_{PLH} = 11 \pm 1$ ns).

Dalle specifiche del DS si trova che i tempi di propagazione tipici e massimi per una singola porta NAND sono: da cui vediamo che le nostre misure dei ritardi accumulati tra uscita e ingresso del circuito sono compatibili con i tempi necessari per il numero di porte NAND che devono commutare per il cambiamento di stato del D-Latch.

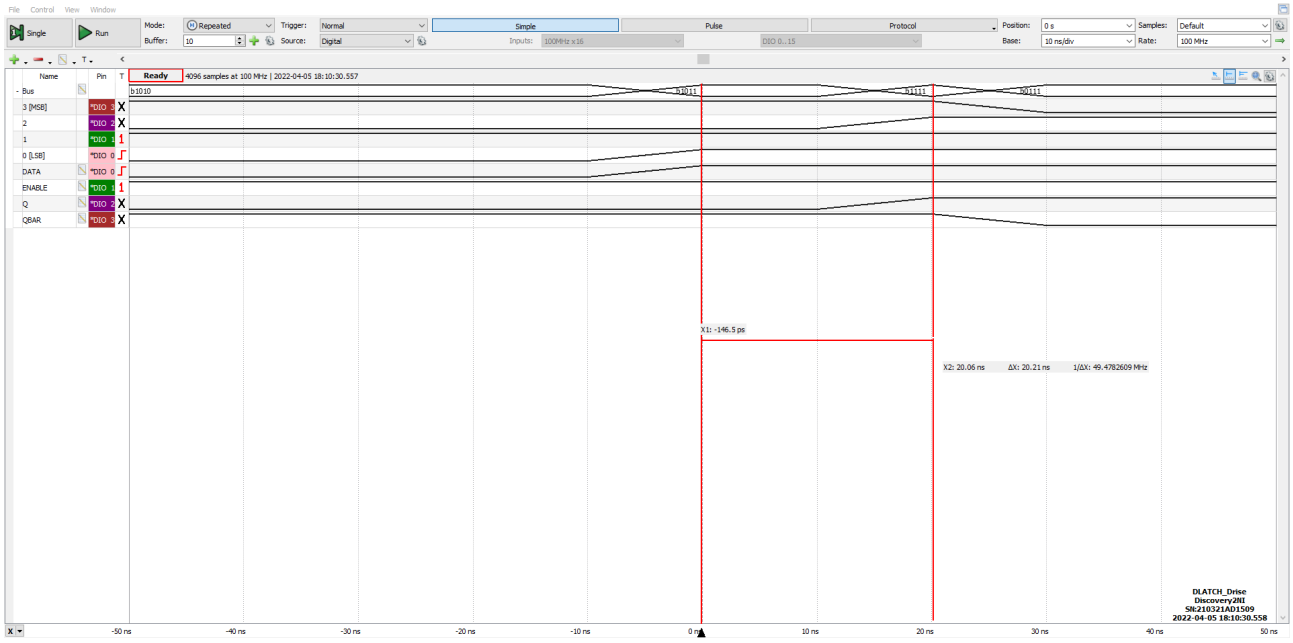

Figura 5: Acquisizione da oscilloscopio digitale della transizione 2 del D-Latch

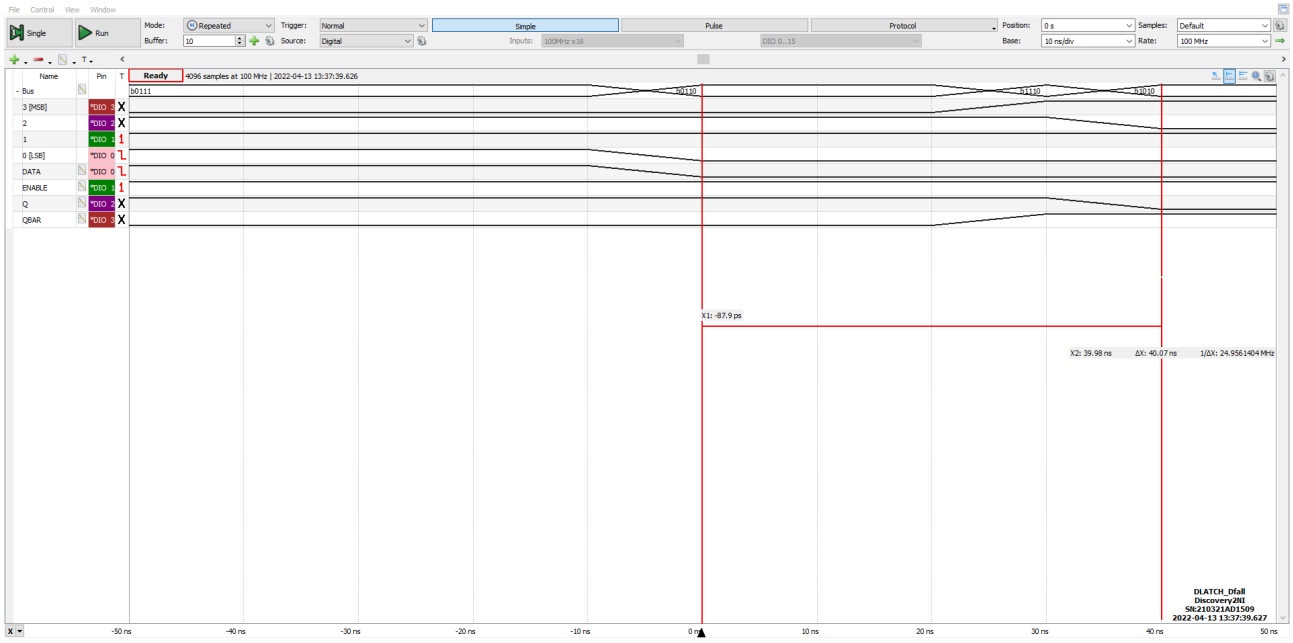

Figura 6: Acquisizione del Logic Analyzer durante la transizione 3 del D-Latch

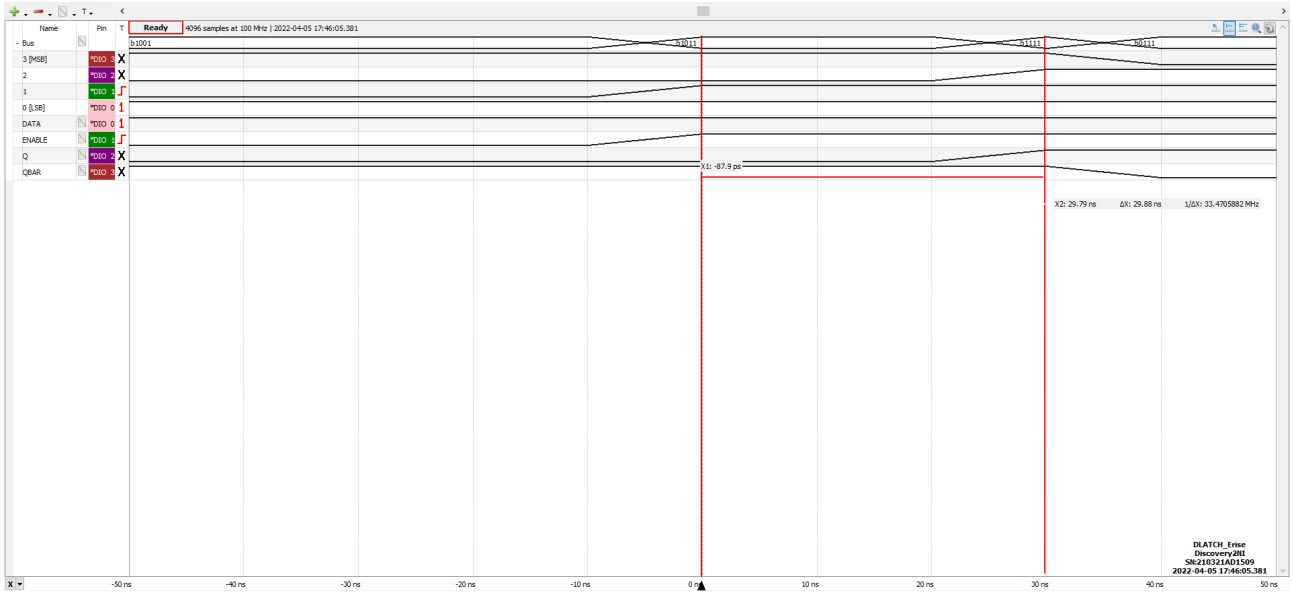

Figura 7: Acquisizione del Logic Analyzer durante la transizione 4 del D-Latch

	typ	max	[units]
t_{PLH}	11	22	ns
t_{PHL}	7	15	ns

3 Shift-register con edge-triggered D-Flip Flop

3.a Costruzione del circuito

Si vuole ora costruire uno Shift Register a 4 bit a partire da due integrati della serie SN74LS74 (Dual Positive-Edge-Triggered Flip-Flops), secondo lo schema riportato in fig. 8 e verificarne il funzionamento.

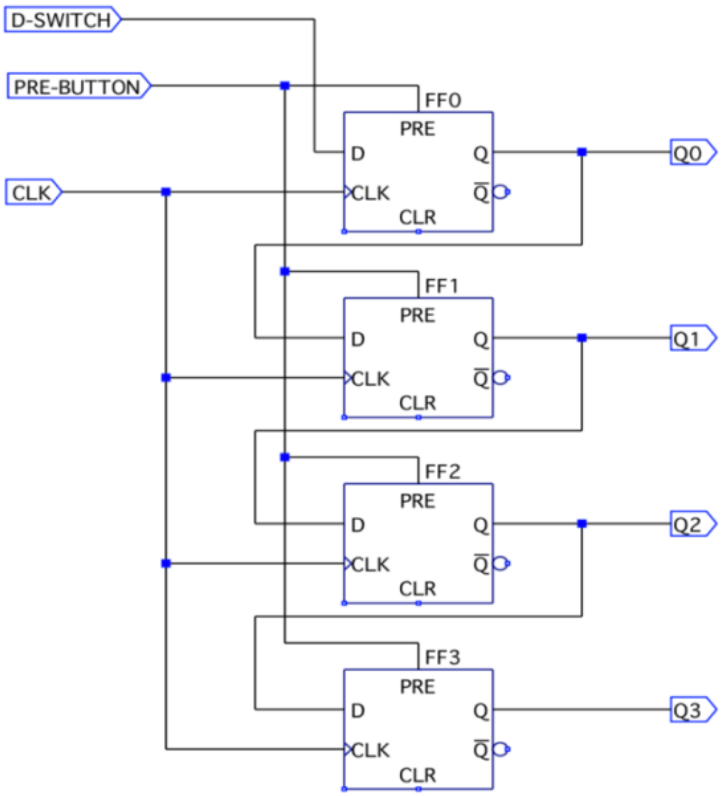

Figura 8

3.b Verifica della sincronia delle uscite rispetto a PRESET

Una volta controllato che le uscite Q_0, Q_1, Q_2 e Q_4 fossero nello stato 0000, abbiamo utilizzato la funzione StaticIO per pilotare il PRESET (ingresso PRE-BUTTON di fig. 8) tramite un pulsante inverso (di tipo pressed= 0 e released= 1). Dunque si è acquisito con Logic Analyzer l'andamento temporale dei 4 segnali in uscita con condizione di trigger coincidente alla pressione del pulsante.

Dalla fig. 9 vediamo che le commutazioni delle uscite avvengono allo stesso tempo, dopo un $\Delta t = 40$ns a partire dalla pressione del pulsante di preset le 4 uscite raggiungono lo stato 1111.

Nella fig. 10 si mette in evidenza come questo cambiamento di stato avvenga indipendentemente dal segnale di clock in ingresso ai FF, da cui si vede che l'ingresso di PRESET dei Flip-Flop è asincrono come atteso dalle specifiche tecniche.

3.c Verifica del funzionamento tramite clock

Possiamo costruire una tabella di verità per il registro in funzione del tempo (discreto) battuto dal periodo T del clock:

Nel caso in cui il pulsante di PRESET inizializzi tutte le uscite a 1, e il D-Switch sia impostato (sempre con StaticIO) al valore $Q'_0 = 0$, la tabella 2 si traduce nelle commutazioni di stato previste dalla tabella 3

Per verificare il corretto funzionamento del circuito si pilota il registro con un segnale di clock di frequenza $f = 1$ Hz e si fa ancora uso di Logic per acquisire gli andamenti nel tempo dei valori assunti dalle uscite. Dalla

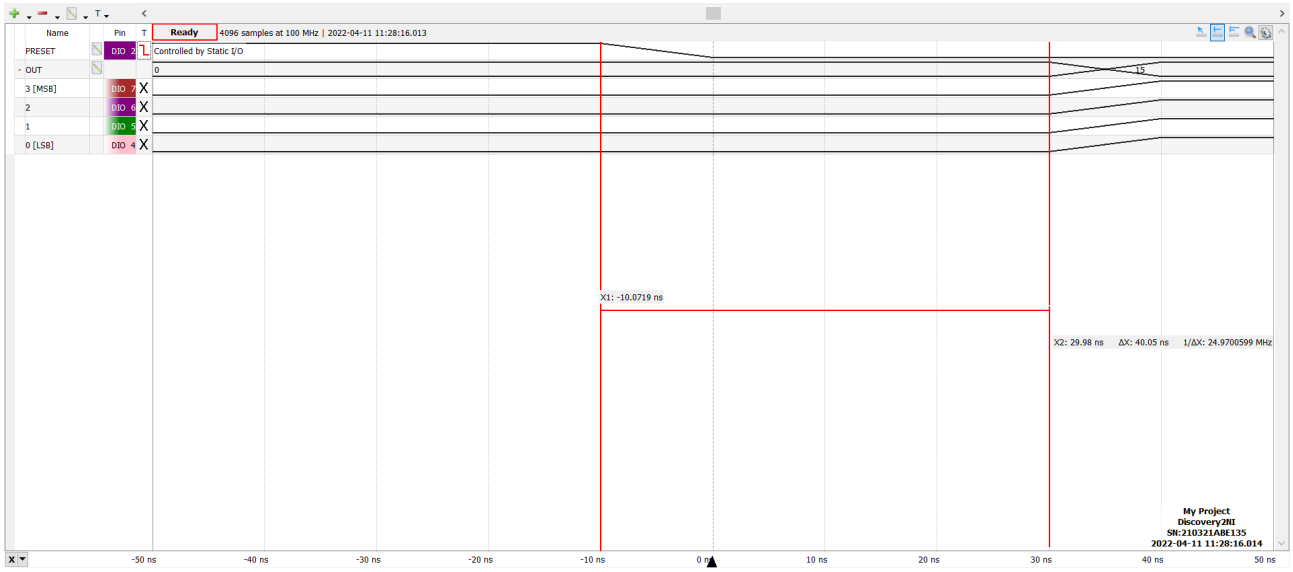

Figura 9: Acquisizione con Logic dei segnali in PRE-BUTTON le 4 uscite dal registro a scorrimento pilotato dall'ingresso di PRESET

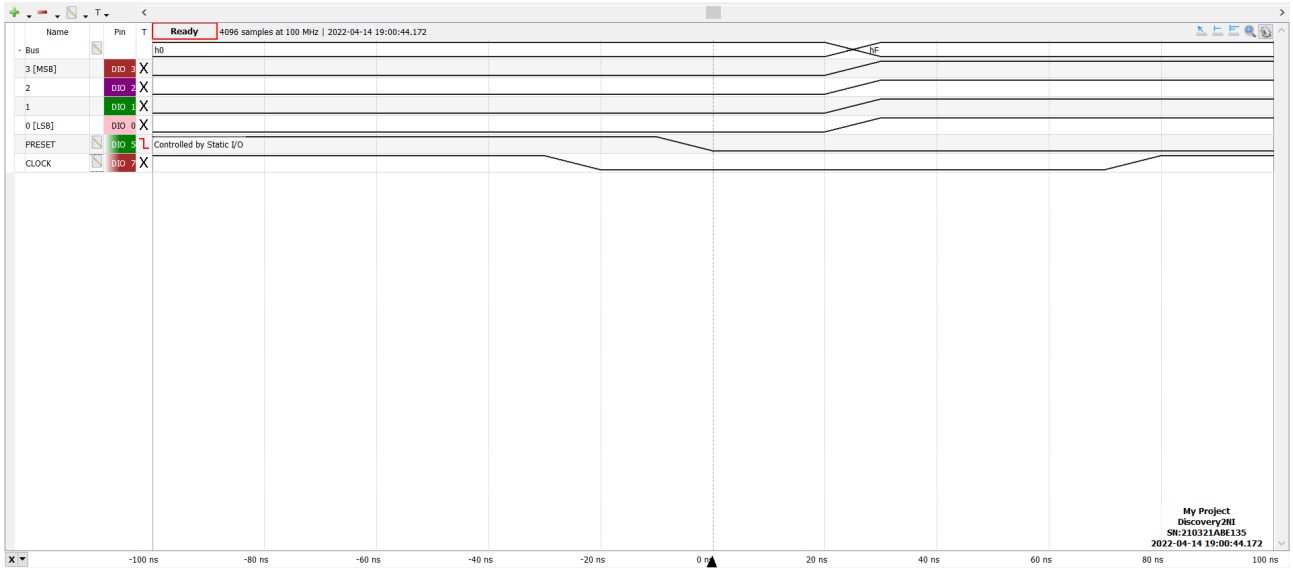

Figura 10: Acquisizione con Logic dei segnali in PRE-BUTTON, CLOCK e $Q_{0,\dots,3}$ in uscita dal registro a scorrimento studiato

$Q_i(t = t')$	$Q_i(t = t' + T)$
Q_0	Q'_0
Q_1	$Q_0(t = t')$
Q_2	$Q_1(t = t') = Q_0(t = t' - T)$
Q_3	$Q_2(t = t') = Q_0(t = t' - 2T)$

Tabella 2: Tabella "di verità" di un registro a scorrimento, in cui Q'_0 è il valore che viene inviato durante il fronte di salita t' del clock tramite il D-SWITCH

t	Q_0	Q_1	Q_2	Q_3
t'	1	1	1	1
$+T$	0	1	1	1
$+2T$	0	0	1	1
$+3T$	0	0	0	1
$+4T$	0	0	0	0

Tabella 3: Sequenza di transizioni che si osservano a partire da quando viene premuto il pulsante di PRESET (t') lasciando a livello basso il D-SWITCH del registro a scorrimento.

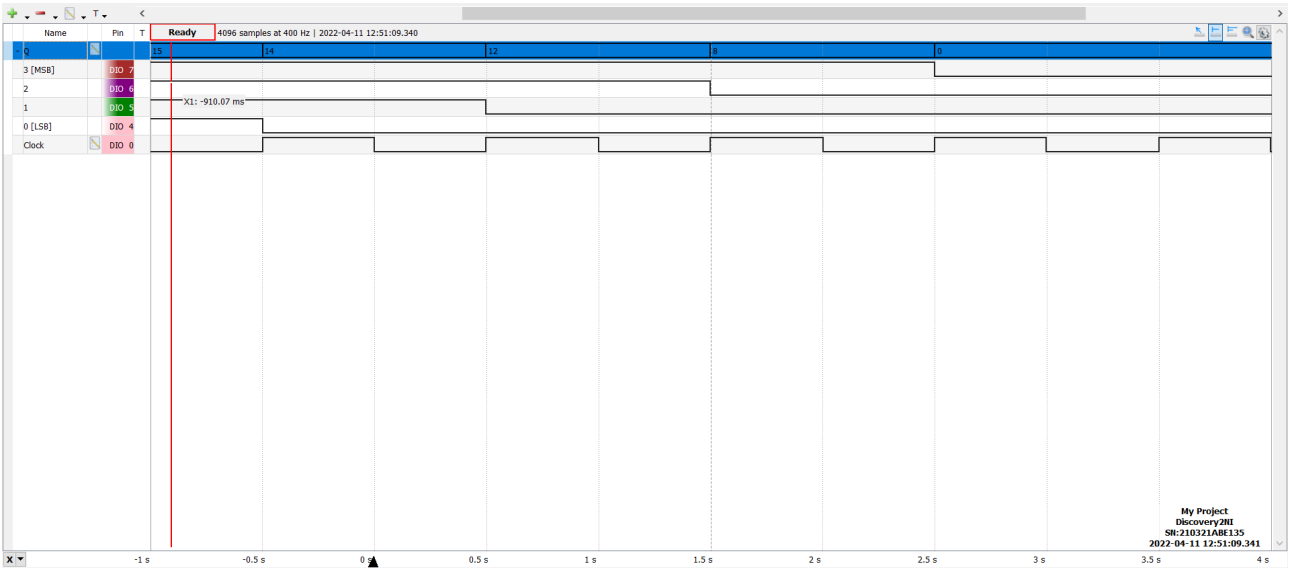

Figura 11: Acquisizione con Logic dei segnali in uscita da un registro a scorrimento di 4 bit come illustrato in sezione 3.c

fig. 11 si verifica come atteso che dopo 3 periodi di clock (3 secondi) a partire da quando l'uscita $Q_0 = 0$, le 4 uscite diventano (e si mantengono costanti nel tempo visto che il D-Switch rimane fisso a 0) tutte quante basse.

Da questo si intuisce che collegando l'uscita (NOT) Q_3 all'ingresso D-switch, possiamo generare una sequenza periodica/costruire un contatore con modulo.

3.d Prevalenza tra gli ingressi D-switch e PRESET

Utilizzando allo stesso tempo gli ingressi D-switch e PRESET del registro si vede che è l'ultimo a pilotare il segnale in uscita, dal momento che PRESET è asincrono (indipendente dalla salita del clock). Questo risulta consistente con la regola generale per cui le architetture asincrone prendono la precedenza sulle istruzioni sincrone.

3.e Twisted-ring Johnson counter

Dopo aver reimpostato il registro nello stato $Q_{i=0,\dots,3} = 0$, si collega l'uscita $\overline{Q_3}$ all'ingresso D del primo Flip-Flop e si invia un clock di frequenza pari $f = 1$ kHz all'ingresso del circuito.

Ci si aspetta che, prendendo un'uscita a caso, si osservi un segnale di clock di frequenza un quarto di quella di clock: questo effetto è dovuto a come la sequenza viene caricata nel registro a partire dall'uscita $\overline{Q_3}$. In generale supponendo che nel circuito ci siano in cascata n Flip-Flop, e che tutte le uscite siano inizializzate a 0, in una qualsiasi delle uscite otterrò un clock di frequenza pari a $\frac{f_{\text{clk}}}{n}$.

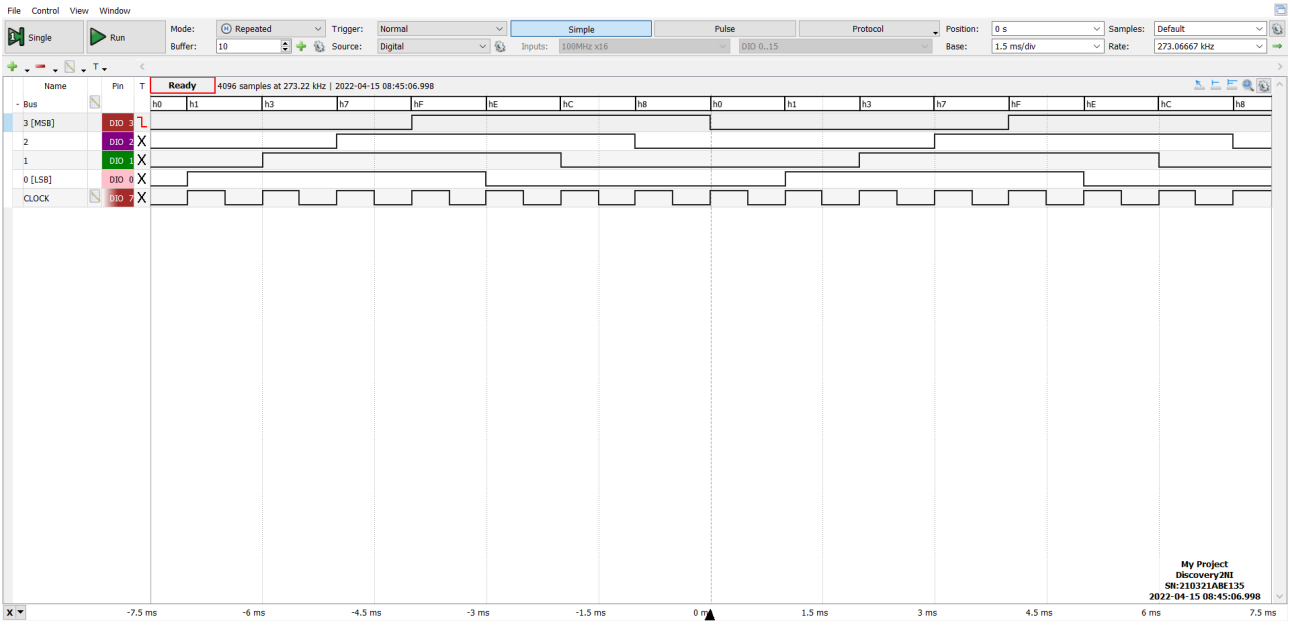

Figura 12: Acquisizione temporale con Logic dei segnali in uscita da un registro a scorrimento di 4 bit con l'ultima uscita negata collegata all'entrata del primo flip flop

4 Generatore di sequenze pseudo-casuali

4.a Costruzione del circuito

Si vuole ora costruire un generatore di sequenze pseudo-casuali a 4 bit utilizzando lo shift register costruito in precedenza e una porta XOR; la schematica del circuito che utilizzeremo è riportato in figura fig. 13.

4.b Analisi e verifica del funzionamento

Dopo aver montato il circuito si inizializzano tutti Flip-Flop a 1 e si invia un segnale di clock a 10 kHz per verificare il funzionamento: essendo il registro di lunghezza pari a 4 bit, dalla teoria ci aspettiamo che la sequenza si ripeta dopo $2^4 = 16$ eventi al massimo, condizione che si ottiene utilizzando come TAP (segnali in ingresso alla porta XOR, la cui uscita sarà inviata all'entrata D del primo Flip-Flop) le uscite Q_2 e Q_3.

Dalla fig. 14 si verificano le aspettative per cui la sequenza generata a una qualsiasi uscita si ripete ogni 16 periodi di clock (essendo uno shift register la sequenza nelle altre uscite sarà la medesima, solo che saranno sfasate lungo l'asse temporale le une con le altre).

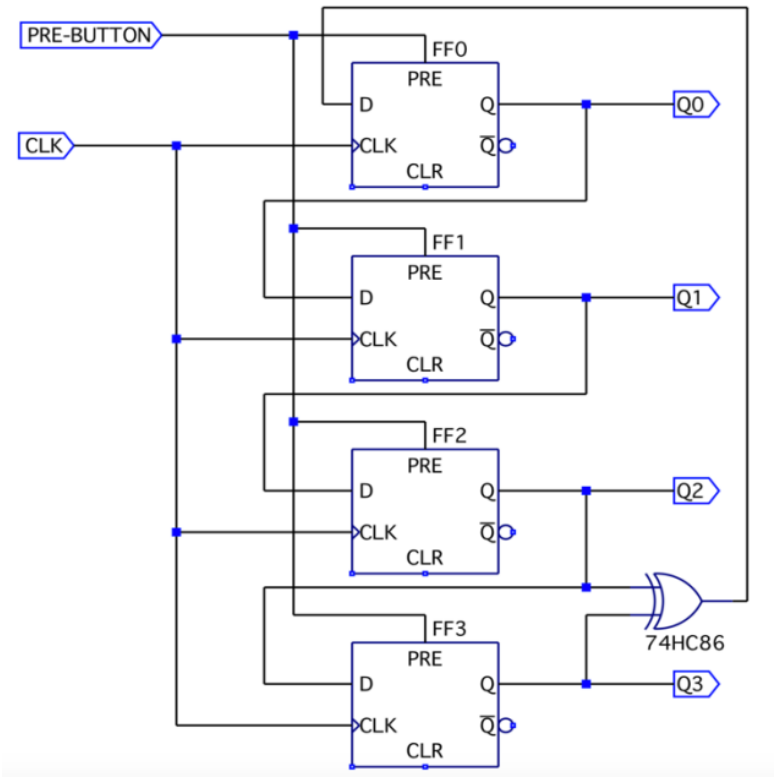

Figura 13

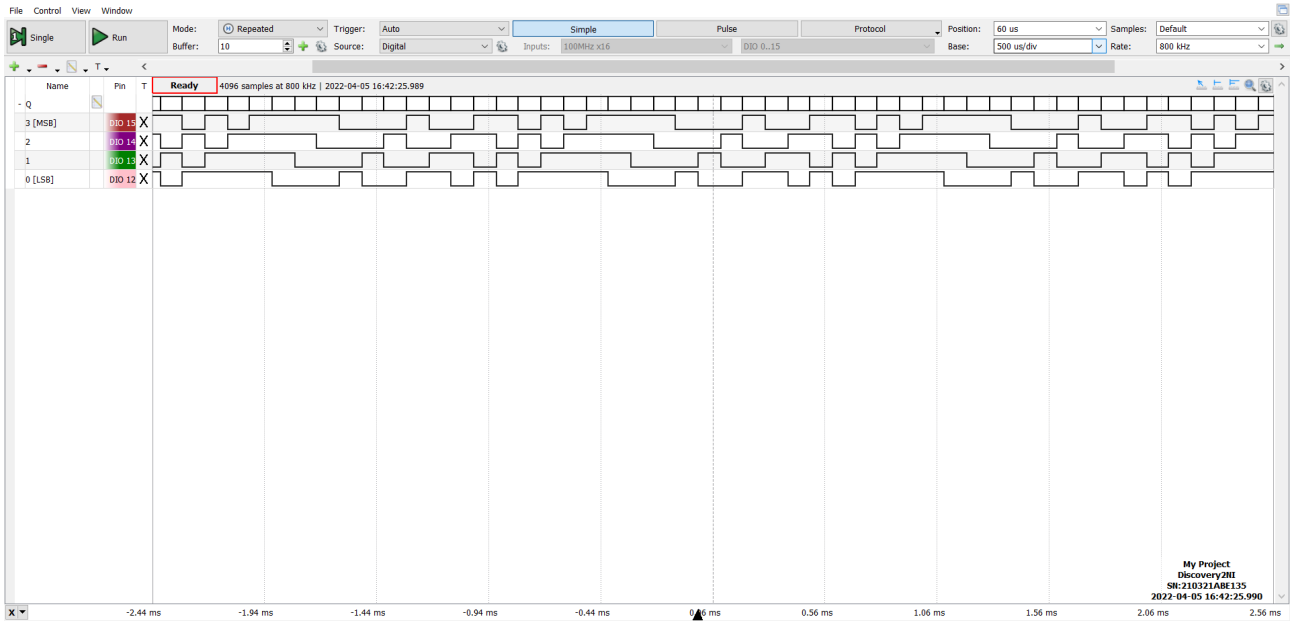

Figura 14: Acquisizione temporale con Logic del bus in uscita dal generatore di sequenze psudo-casuale descritto in fig. 13

4.c Studio delle sequenze generabili con diverse condizioni iniziali

Si provano quindi altre combinazioni di TAP, per verificare che la scelta di utilizzare l'uscita Q_2 e Q_3 produce una sequenza più lunga rispetto a qualunque altra configurazione

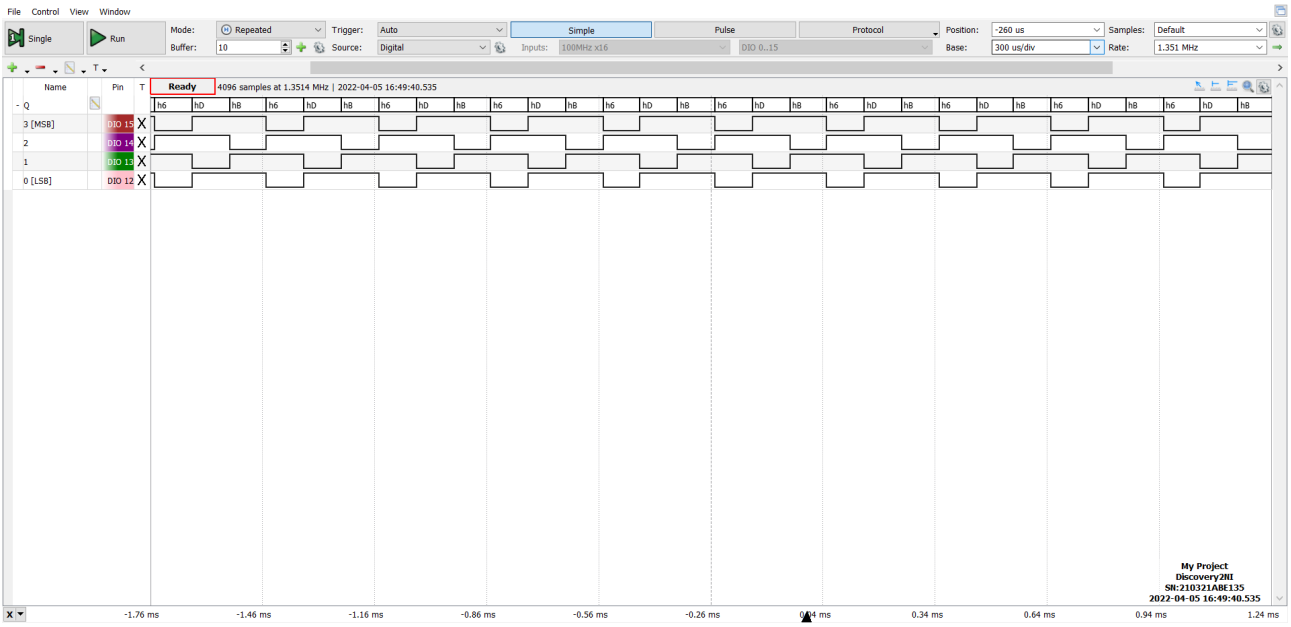

Figura 15: Acquisizione temporale con Logic del bus in uscita dal generatore di sequenze psudo-casuale con TAP sulle uscite Q_0 e Q_1, la sequenza si ripete ogni 4 eventi

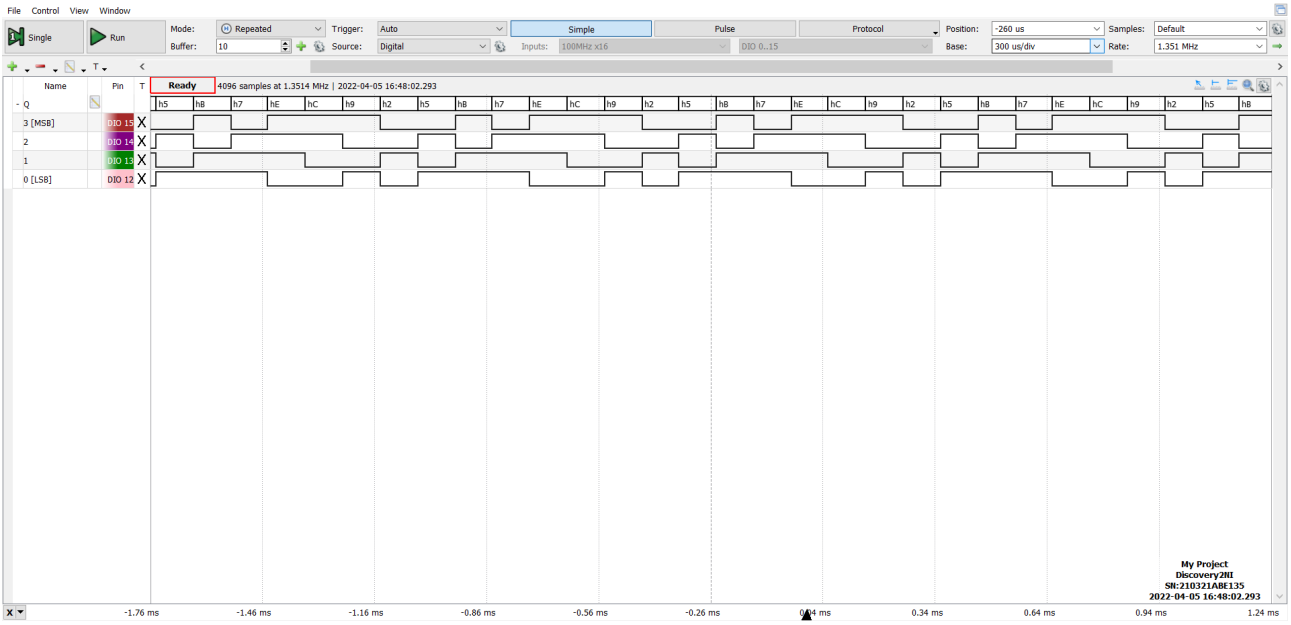

Figura 16: Acquisizione temporale con Logic del bus in uscita dal generatore di sequenze psudo-casuale con TAP sulle uscite Q_2 e Q_1, la sequenza si ripete ogni 8 eventi

5 Divisori di frequenza con contatori binari

5.a Costruzione del circuito

Si intende costruire un divisore di frequenza a partire da un contatore binario a 4 bit, utilizzando l'integrato SN74LS163, presente in fig. 18. Si vuole innanzitutto verificarne il funzionamento.

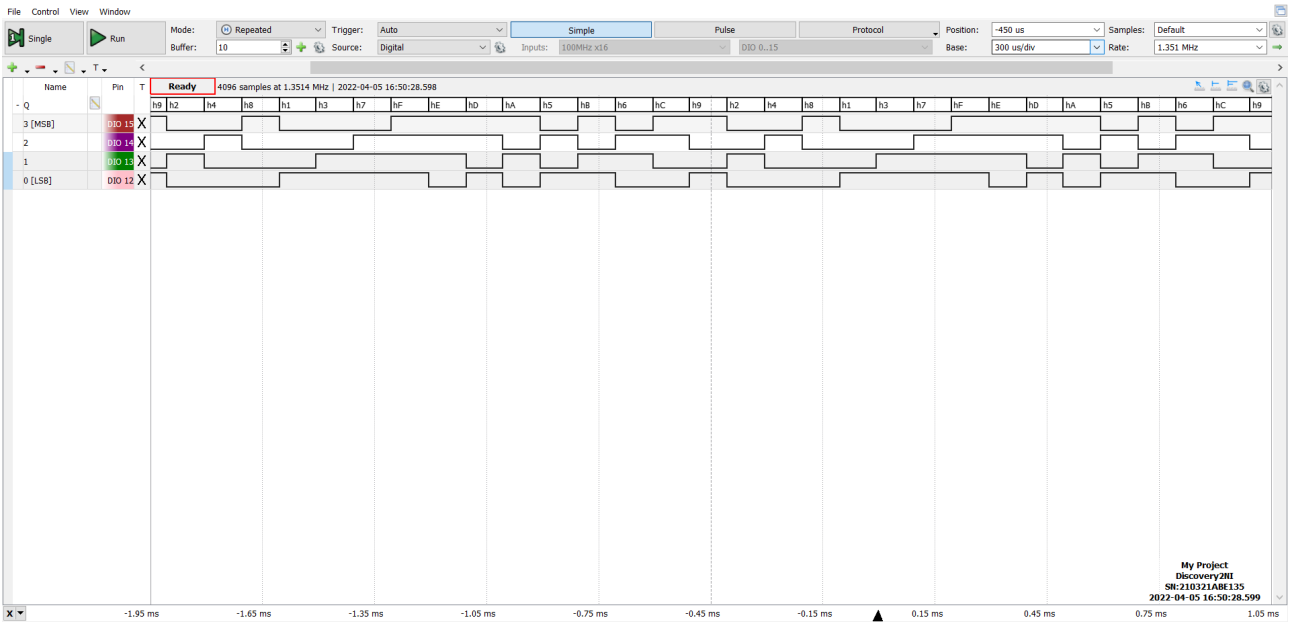

Figura 17: Acquisizione temporale con Logic del bus in uscita dal generatore di sequenze pseudo-casuale con TAP sulle uscite Q_0 e Q_3, la sequenza si ripete ogni 16 eventi

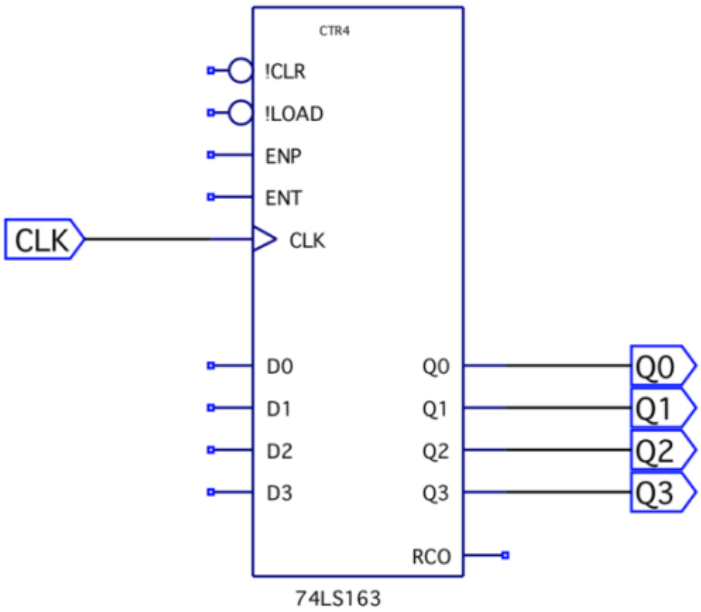

Figura 18

5.b Verifica del ciclo di funzionamento dei contatori

Dopo aver montato il circuito utilizziamo la funzione Pattern di Waveform per inviare un segnale di clock di frequenza pari a 10 kHz al pin di clock (CLK) del contatore, e utilizzeremo Logic per acquisire le uscite Q_0->Q_3. Essendo un contatore a 4 bit, ci si aspetta che il Bus conti dallo stato 0000 (0 in decimale) fino allo stato 1111 (15 in decimale) in ordine crescente. Per questo motivo utilizzeremo un formato di Bus esadecimale per verificare che il circuito generi tutti gli stati possibili (0->F). Dalla fig. 19 si può notare che il comportamento

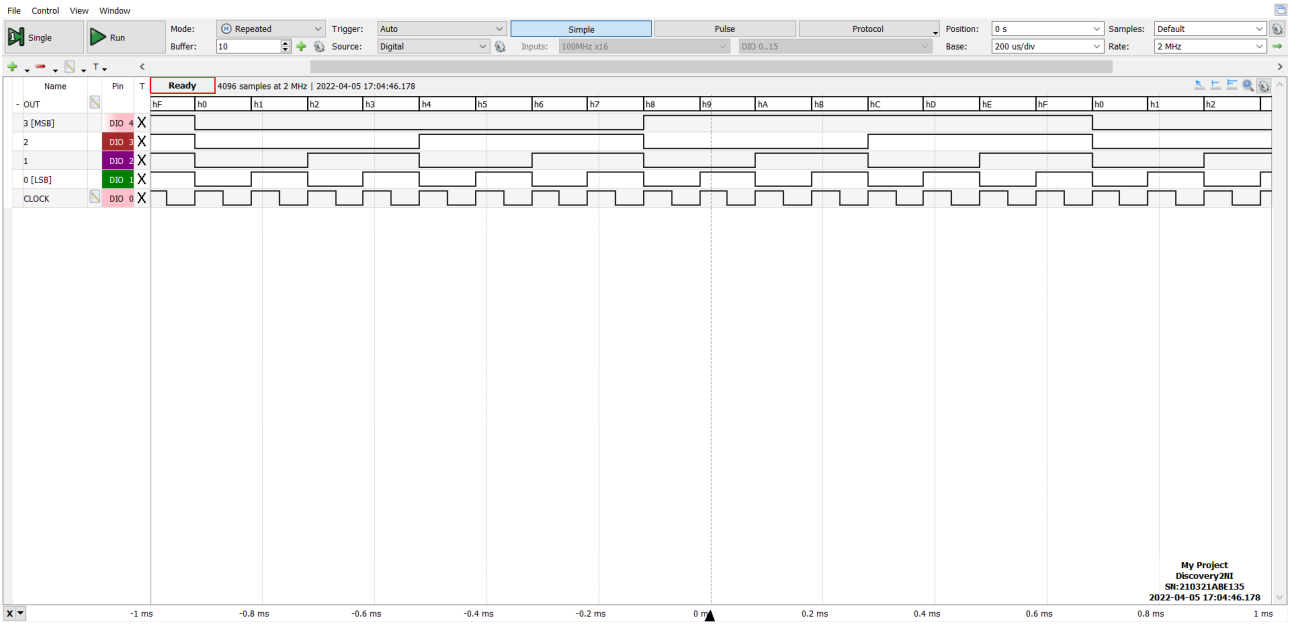

Figura 19: Acquisizione temporale con Logic del bus in uscita dal contatore, con frequenza di clock pari a 10 kHz

del circuito è come atteso, e che gli stati del bus in uscita comprendono tutti i valori in ordine crescente della numerazione esadecimale

5.c Verifica della divisione in frequenza

Dato che il contatore binario incrementa di 1 ad ogni evento di clock, il bit di ordine n avrà come frequenza la metà di quella del bit di ordine n-1. Avendo noi un BUS composto da 4 bit, ci si aspetta quindi che le frequenze ottenute siano $\frac{f_{clock}}{2}$ (per il LSB), $\frac{f_{clock}}{4}$, $\frac{f_{clock}}{8}$ e $\frac{f_{clock}}{16}$ (per il MSB). Sempre facendo riferimento all fig. 19 si può notare come le uscite del contatore si comportino da divisori in frequenza: infatti i periodi che si trovano analizzando l'acquisizione sono 2 (LSB), 4, 8 e 16 (MSB) volte il periodo del clock.

5.d Transizione sincrona del contatore

Utilizziamo la funzione di trigger di Logic e la impostiamo in modo tale che l'acquisizione cominci quando il segnale presente in Q_3 cambia da 1 a 0 in modo da poter studiare gli eventuali ritardi delle singole uscite nella transizione F->0 e verificare il comportamento sincrono del contatore. Dalla fig. 20 possiamo infatti vedere che la transizione è sincrona e avviene con un ritardo di 30ns da quando il clock compie il salto 0->1; il passaggio di stato 1->0 avviene contemporaneamente in ogni uscita.

5.e Divisore di frequenza 1/10; Contatore BCD

Si vuole quindi realizzare un divisore in frequenza, che generi un segnale di periodo pari a 10 volte quello di clock: per farlo utilizzeremo il pin CLEAR del contatore, il quale quando gli viene inviato un segnale Low resetta il contatore allo stato 0000. È quindi necessario che il contatore conti un totale di 10 stati e poi si resetti, per cui partendo dallo 0, dobbiamo imporre la condizione che arrivati allo stato 9 (1001) il contatore riceva il comando di reset tramite Clear (che ricordiamo essere Active Low). Per farlo utilizzeremo una porta NAND (utilizzando l'integrato SN74LS00) ai cui 2 ingressi invieremo il LSB e il MSB e invieremo l'uscita al pin di Clear. In questo modo, quando il contatore arriva a 9 il pin di Clear riceverà un segnale Low e il circuito si azzererà; pertanto la frequenza del segnale Clear sarà un decimo di quella di clock. Inoltre essendo il segnale di Clear sullo stato Low solamente quando il bus in uscita registra lo stato 1001, mi aspetterò un Duty Cycle pari al 90 % Si costruisce

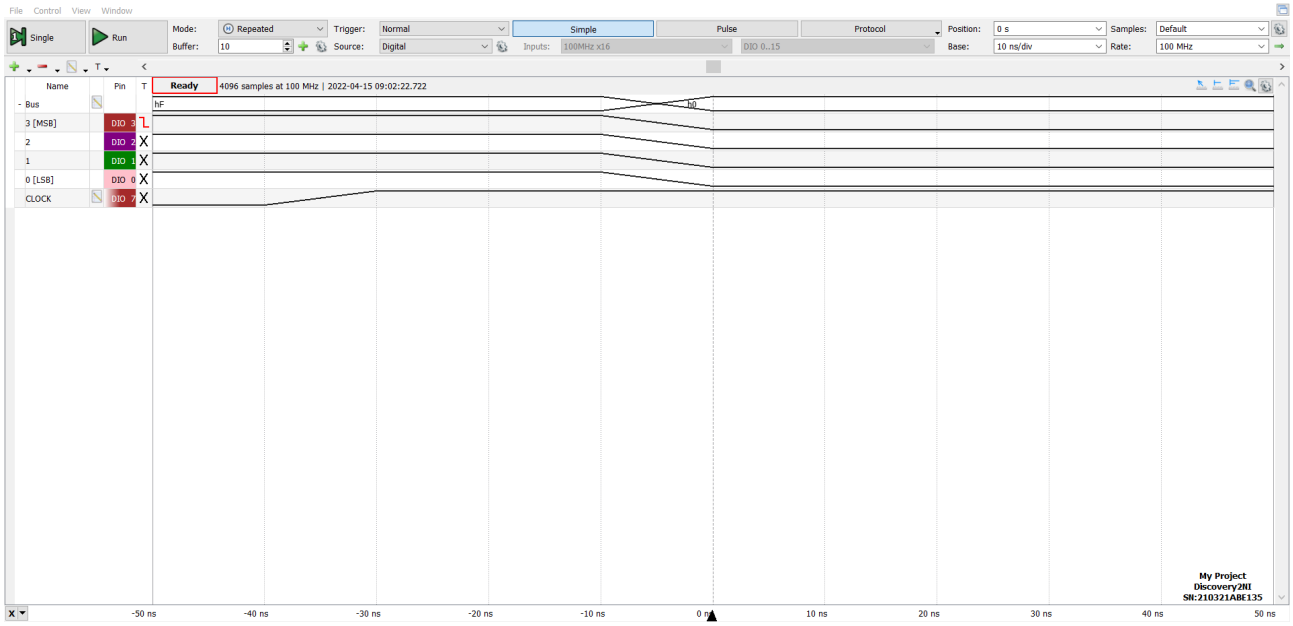

Figura 20: Acquisizione temporale con Logic del bus in uscita dal contatore durante la transizione 15->0; dall'immagine possiamo notare il comportamento sincrono della commutazione delle uscite del contatore

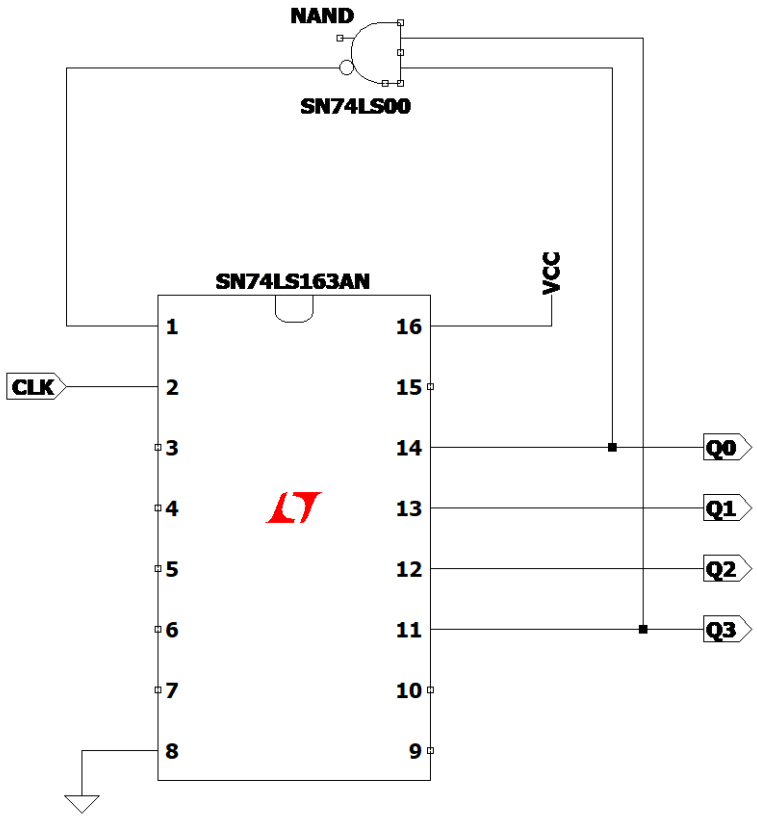

Figura 21: Schematica utilizzata per il divisore per 10 di frequenza

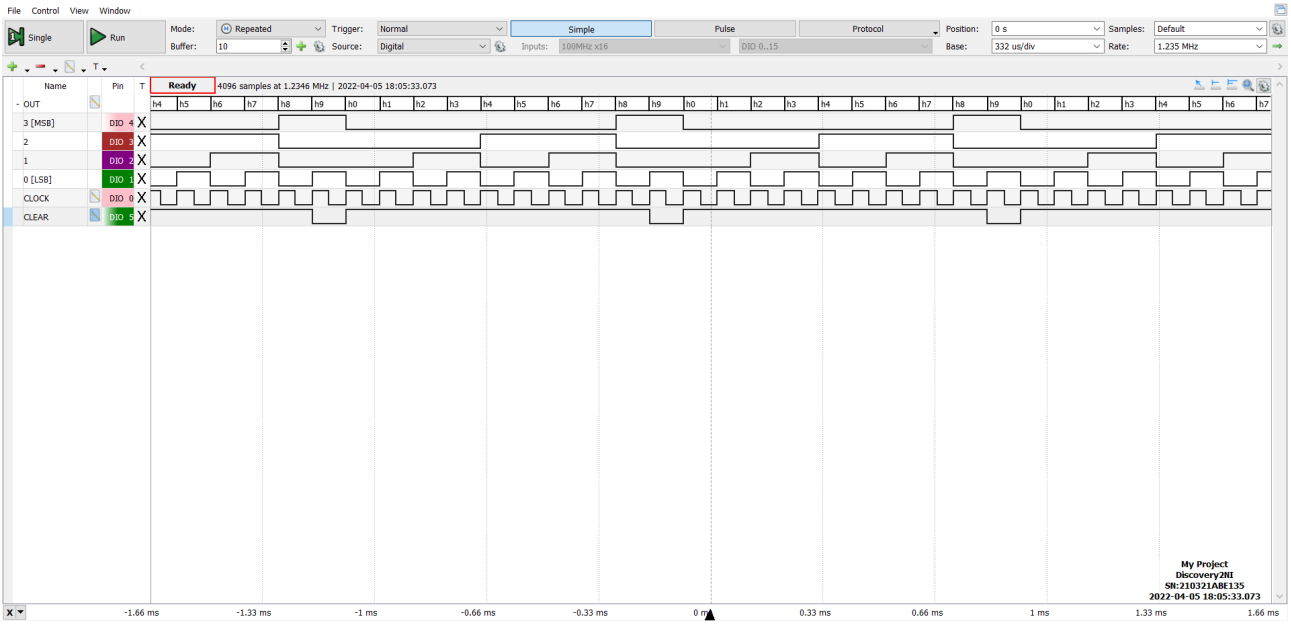

Figura 22: Acquisizione temporale con Logic del bus in uscita dal circuito che conta fino a 10; si può notare che il segnale in entrata nel pin Clear ha effettivamente periodo pari a 10 volte quello del clock

quindi il circuito presente in fig. 21. Come prima, si utilizza la funzione Logic per acquisire il Bus dei 4 bit in uscita, il clock e il bit di Clear. Osservando l'acquisizione presente in fig. 22, possiamo concludere come da aspettative che il segnale presente su clear ha una frequenza pari a $\frac{f_{clock}}{10}$ e Duty Cycle pari al 90 %.

5.f Divisore di frequenza programmabile con RCO

Per concludere, si vuole costruire un divisore in frequenza programmabile: per questo scopo utilizzeremo il bit RCO (Ripple Carry Output), la cui funzione è di generare un segnale H solo nel caso in cui il contatore abbia raggiunto lo stato massimo (1111) e il bit Load, che invece quando vale 0 sovrascrive lo stato del contatore in quello presente all'interno del bus di input. Vogliamo intanto verificare il funzionamento del RCO; dopo

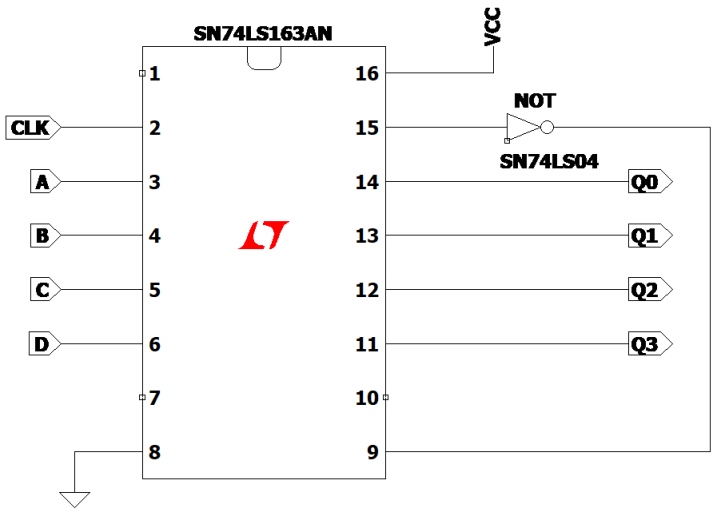

Figura 23: Schematica utilizzata per il divisore di frequenza programmabile

aver montato nuovamento il circuito presente in fig. 18 inviamo un segnale di clock di frequenza 10 kHz e utilizziamo Logic per acquisire i segnali del bus dei 4 bit di output e il segnale di RCO in funzione del tempo. Dalla sezione 5.f possiamo notare che il bit RCO funziona come da aspettative. Si collega ora l'uscita dal RCO all'ingresso di una porta NOT dell'integrato SN74LS04; l'uscita di tale porta sarà inviata al pin LOAD del contatore, come schematizzato in sezione 5.f. Si utilizza la funzione StaticIO per inserire nel Bus di ingresso

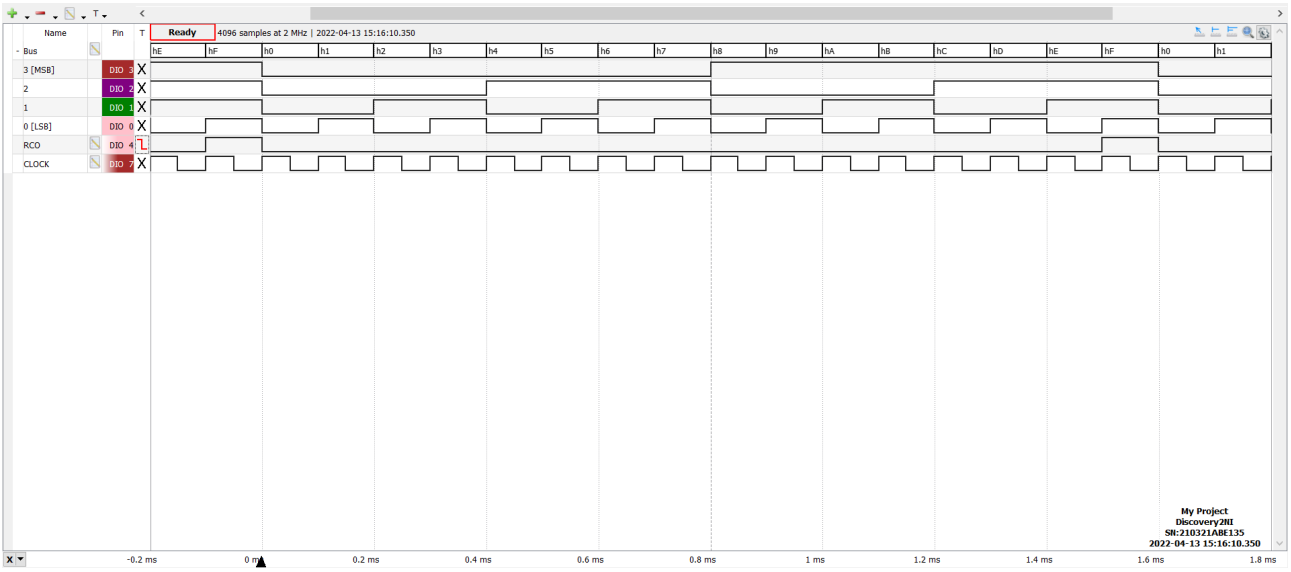

Figura 24: Acquisizione con Logic dei segnali del contatore per verificare il funzionamento del bit RCO

(A,B,C,D) lo stato 0101 e accendiamo il segnale di clock generato tramite Pattern a frequenza 10 kHz. Dalla

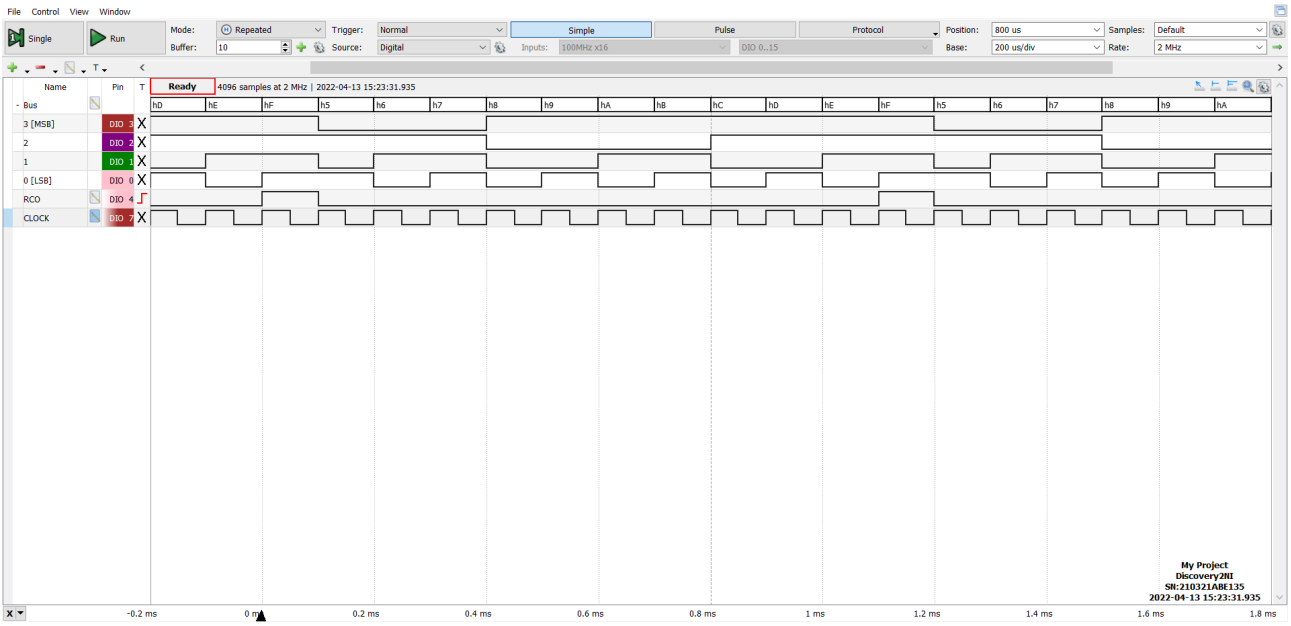

Figura 25: Acquisizione con Logic dei segnali all'interno del circuito divisore di frequenza programmabile, con sequenza di avvio 0101

sezione 5.f ci accorgiamo che quando il contatore riceve il segnale di LOAD, quindi quando si registra 1 nel bit RCO, la sequenza iniziale non viene immediatamente caricata, ma sarà rimandata di un evento di clock. Di conseguenza, qualunque sia la sequenza iniziale del nostro circuito lo stato finale 1111 verrà sempre "contato".

5.g Misura dei tempi caratteristici del divisore RCO

Si cambia ora la sequenza di inizializzazione a 0110, e utilizzando la funzione Logic impostiamo un trigger quando il bit LOAD compie la transizione 0->1: in seguito a quanto visto nella sezione precedente ci aspettiamo che il contatore passi dallo stato 1111 allo stato iniziale. Utilizzeremo una scala dei tempi molto ristretta per verificare che quanto affermato nella sezione 5.d sia ancora vero. Dalla sezione 5.g si può notare che è sempre presente un ritardo dall'inizio del nuovo Tick di clock alla commutazione delle uscite (che come prima è sincrona) che vale indicativamente 30ns.

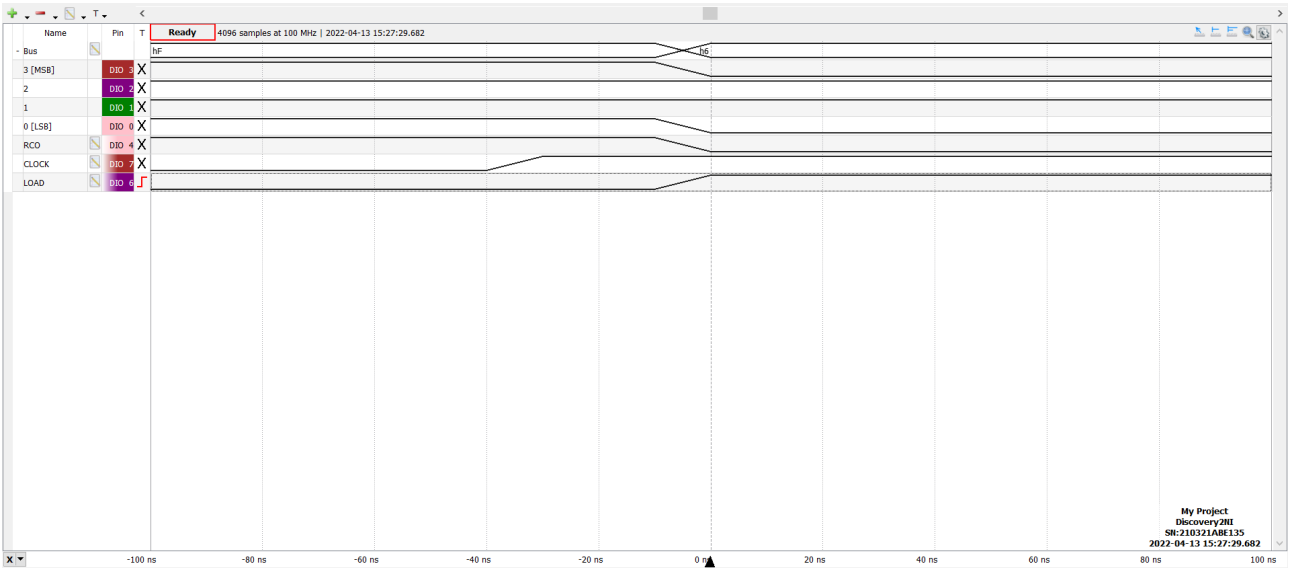

Figura 26: Acquisizione con logic dei segnali all'interno del circuito divisore di frequenza programmabile per un fondo scala dei tempi molto ristretto, con sequenza di avvio 0110

5.h Analisi e verifica del comportamento del divisore RCO

Si passa infine a verificare la corrispondenza tra sequenza iniziale e la frequenza ottenuta. In questa sezione utilizzeremo come frequenza quoziente quella del segnale in uscita dal RCO. Analizzando il comportamento del circuito facendo uso di quanto visto nelle sezioni precedenti ci si aspetta che il periodo del segnale RCO sia pari a quello di clock moltiplicato per il numero di salti di stato necessari per farlo arrivare a 0000; chiamando A il valore in base 10 che corrisponde allo stato iniziale si ha dunque:

$$T_{divisore,A} = T_{clock} \times (16 - A) \tag{2}$$

Visto però il funzionamento dei pin LOAD e RCO, se utilizzassimo come sequenza iniziale 1111, il circuito produrrebbe un segnale costante: infatti 1111 è anche lo stato a cui avviene il caricamento della sequenza, che però risulta essere sempre la medesima, inducendo quindi uno stato incommutabile. Programmando la sequenza con 0000 invece mi aspetto che il comportamento sia analogo a quello presente in sezione 5.b essendo 0000 lo stato iniziale a cui si resetta autonomamente il counter in assenza di segnali di LOAD. Si provano quindi diverse sequenze iniziali per il circuito per cui ci si aspetta: Dalle acquisizioni fatte con logic possiamo concludere

$Seq.iniziale$	$T_{RCO}[T_{clock}]$
0000	16
0010	14
0101	11
1110	2
1111	0

che i periodi sono quelli aspettati.

6 Sintetizzatore musicale

Conclusioni e commenti finali

Si è riusciti a verificare il corretto funzionamento di circuiti logici sequenziali di crescente complessità e svariate applicazioni (e.g. crittografia, sistemi di controllo e misura) costruiti con porte NAND, XOR, D-Latch e contatori sincroni. In particolare sono stati realizzati e studiati un D-Latch, uno shift-register con positive edge-triggered D-FF, un generatore di sequenze pseudocasuali e alcuni tipi di divisore di frequenza con contatori binari. Inoltre si è riusciti ad apprezzare l'effetto dei tempi di propagazione delle porte sul loro comportamento, seppur in maniera limitata dalla bassa risoluzione temporale dell'AD2.

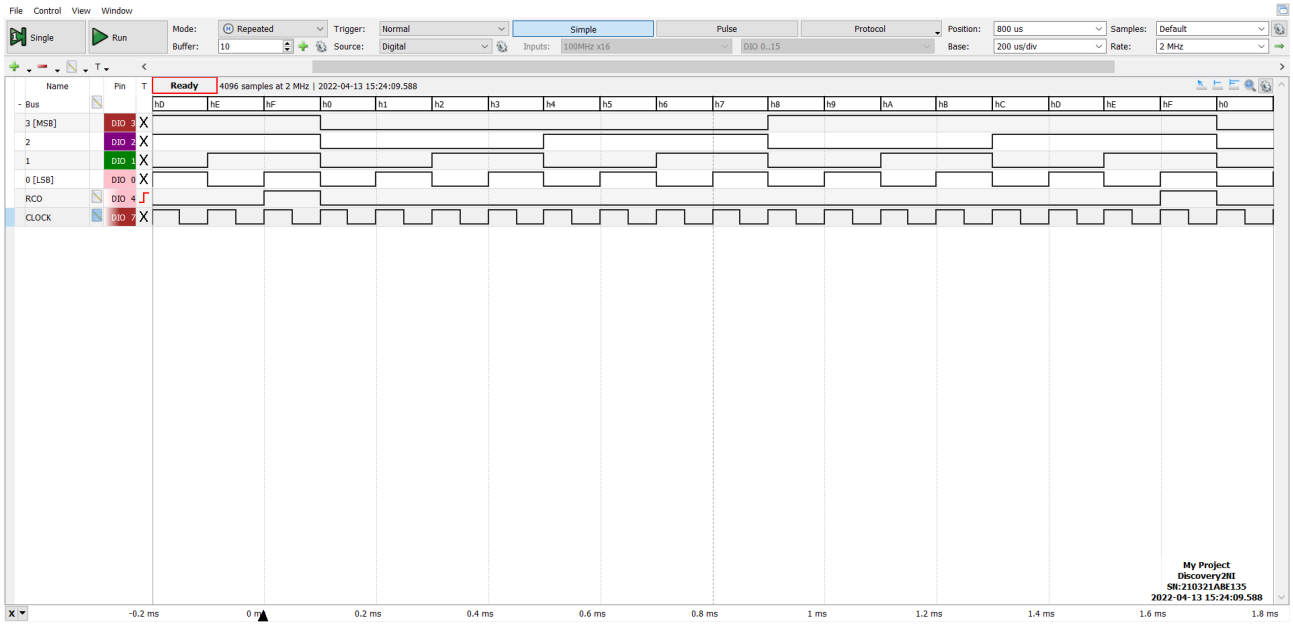

Figura 27: Acquisizione con Logic dei segnali all'interno del circuito divisore di frequenza programmabile, con sequenza di avvio 0000, da cui si misura $T = 16T_{clock}$

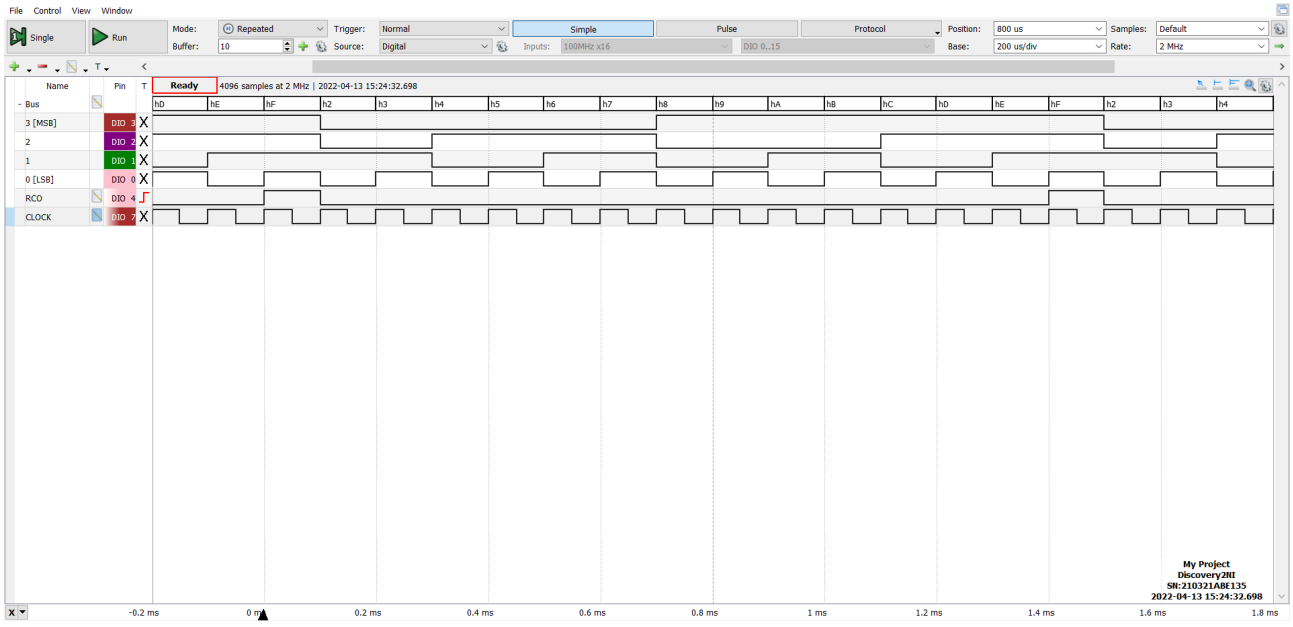

Figura 28: Acquisizione con Logic dei segnali all'interno del circuito divisore di frequenza programmabile, con sequenza di avvio 0010, da cui si misura $T = 14T_{clock}$

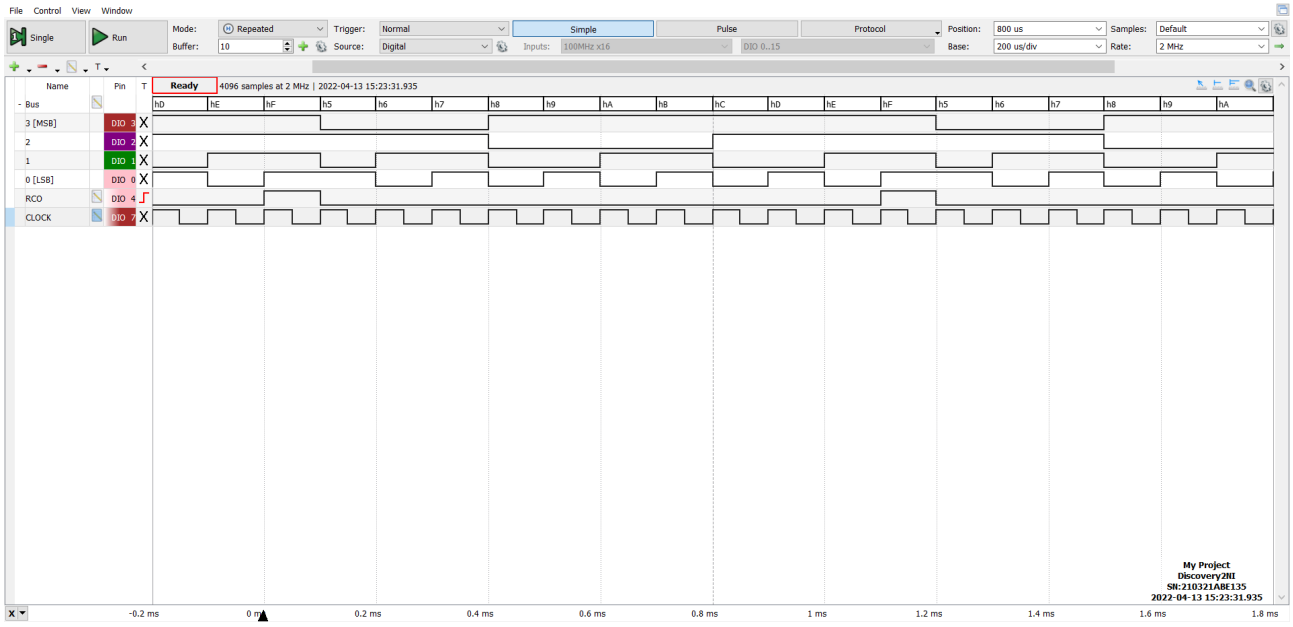

Figura 29: Acquisizione con Logic dei segnali all'interno del circuito divisore di frequenza programmabile, con sequenza di avvio 0101, da cui si misura $T = 11T_{clock}$

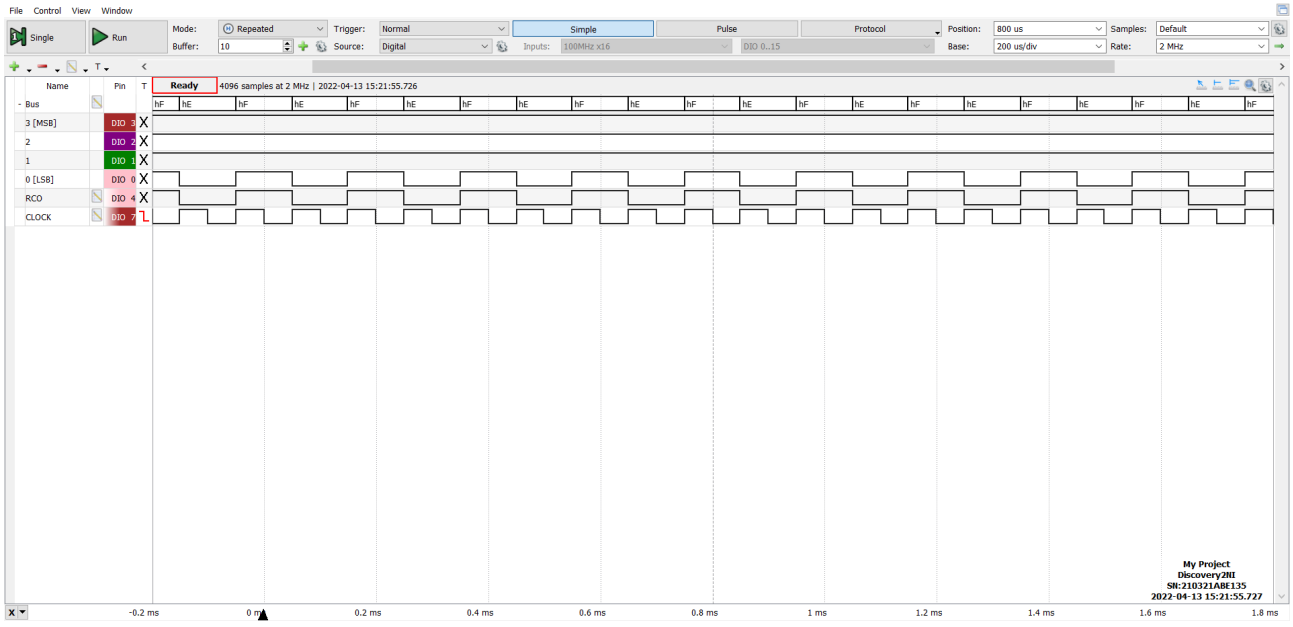

Figura 30: Acquisizione con Logic dei segnali all'interno del circuito divisore di frequenza programmabile, con sequenza di avvio 1110, da cui si misura $T = 2T_{clock}$

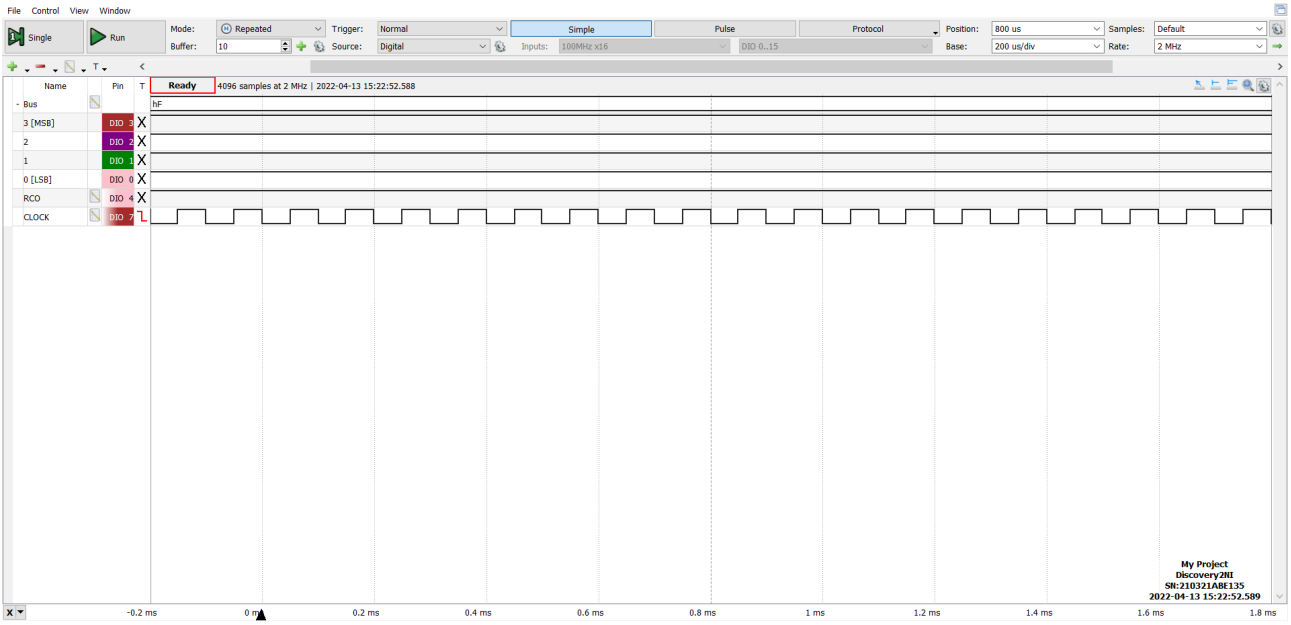

Figura 31: Acquisizione con Logic dei segnali all'interno del circuito divisore di frequenza programmabile, con sequenza di avvio 1111, dal quale si vede che RCO è un segnale costante

Dichiarazione

I firmatari di questa relazione dichiarano che il contenuto della relazione è originale, con misure effettuate dai membri del gruppo, e che tutti i firmatari hanno contribuito alla elaborazione della relazione stessa.