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.

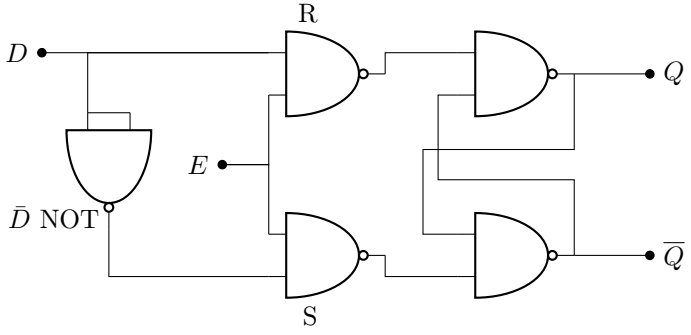

Figura 1: Schema logico del circuito D-Latch (con Enable) realizzato

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 $\varphi = \pm 90°$.

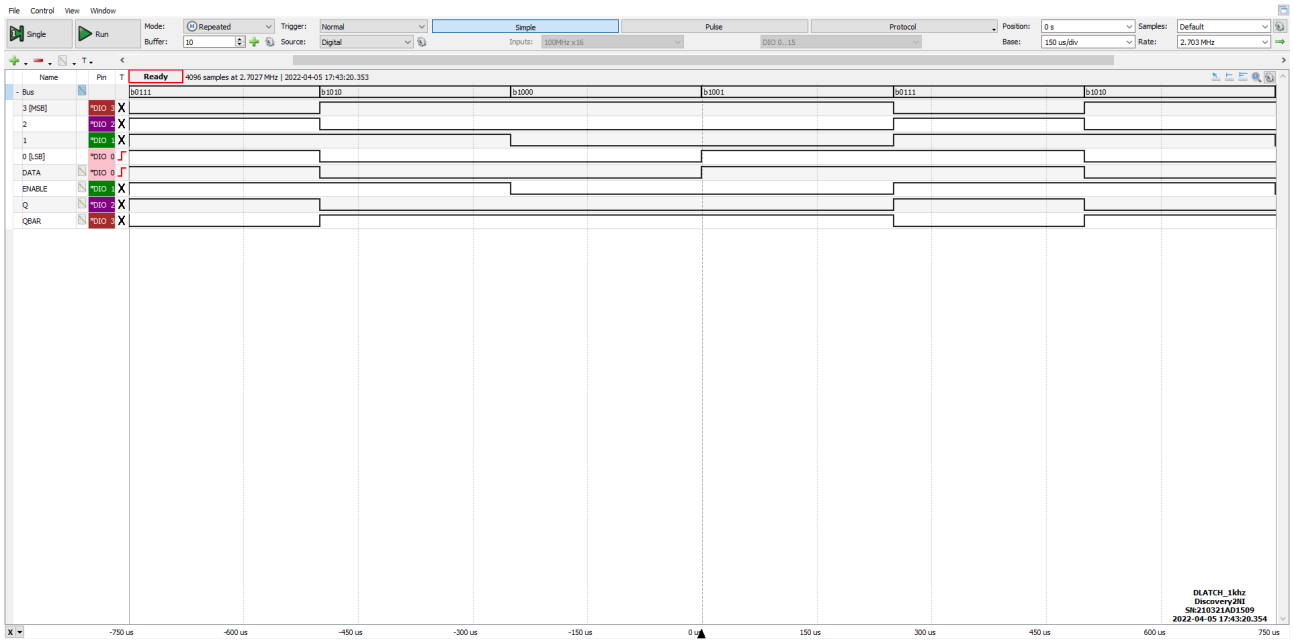

Figura 2: Acquisizione di un ciclo completo (frequenza 1 kHz) con Logic Analyzer dei segnali in ingresso (D = DIO 0, E = DIO 1) e in uscita (Q = DIO 2, $\overline{Q}$ = DIO 3) dal D-Latch.

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.

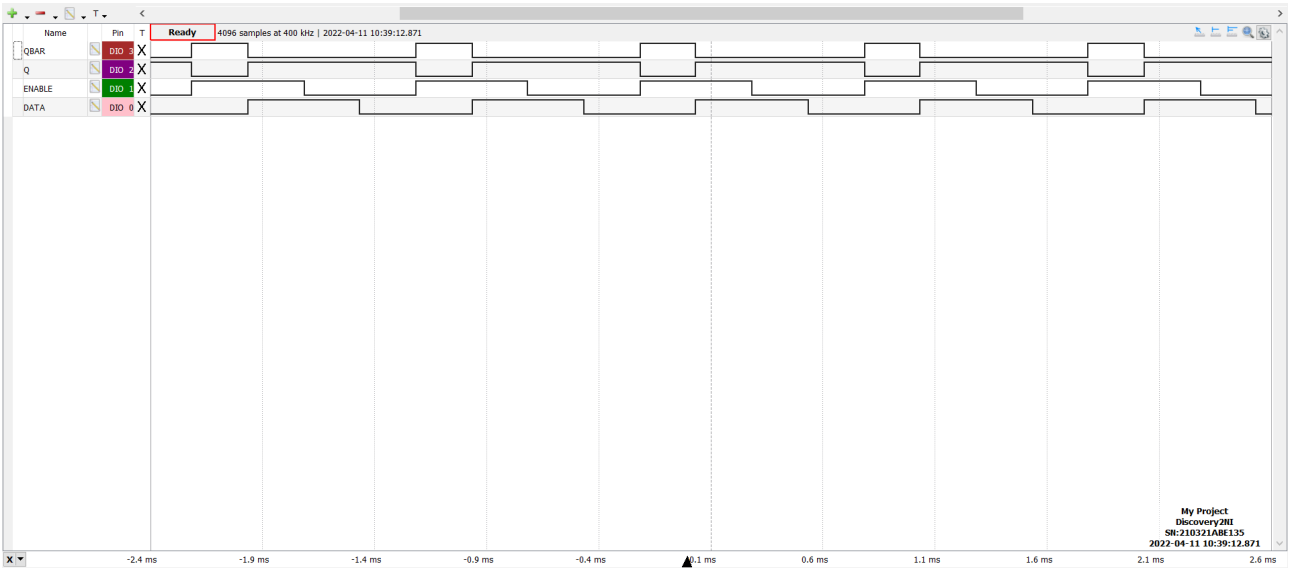

Figura 3: Acquisizione con Logic dell'andamento temporale dei segnali in ingresso uscita dal D-Latch per $\varphi = -90°$.

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

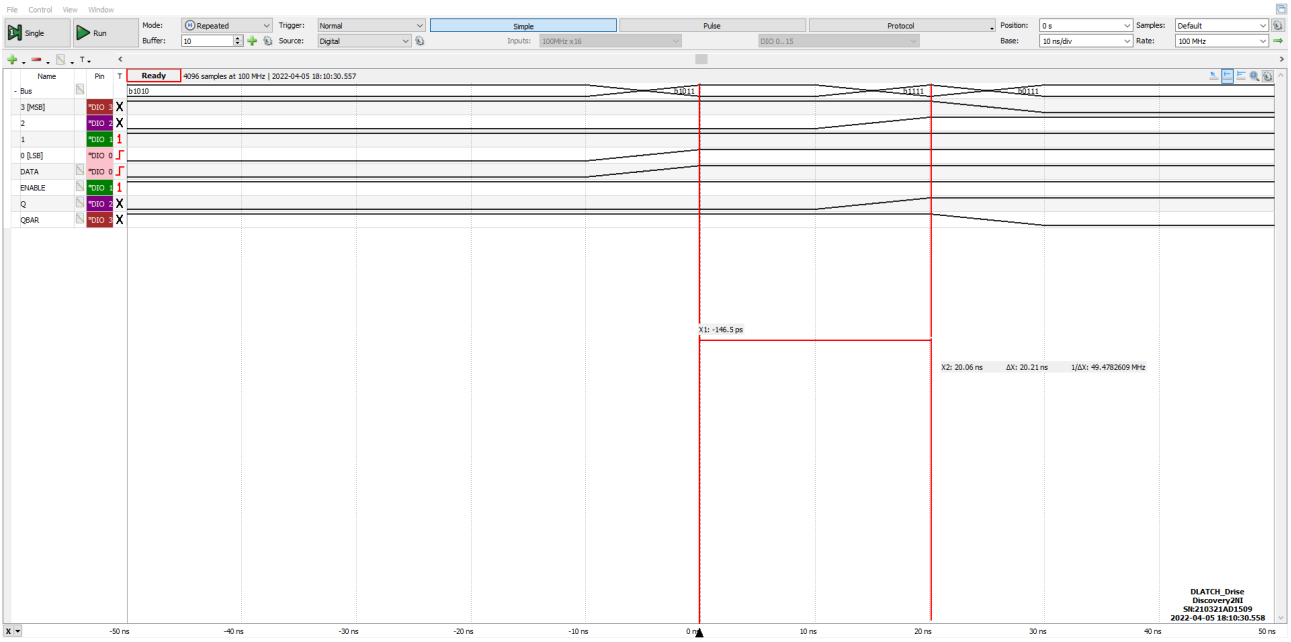

Figura 4: Acquisizione del Logic Analyzer durante la transizione 2 del D-Latch

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.

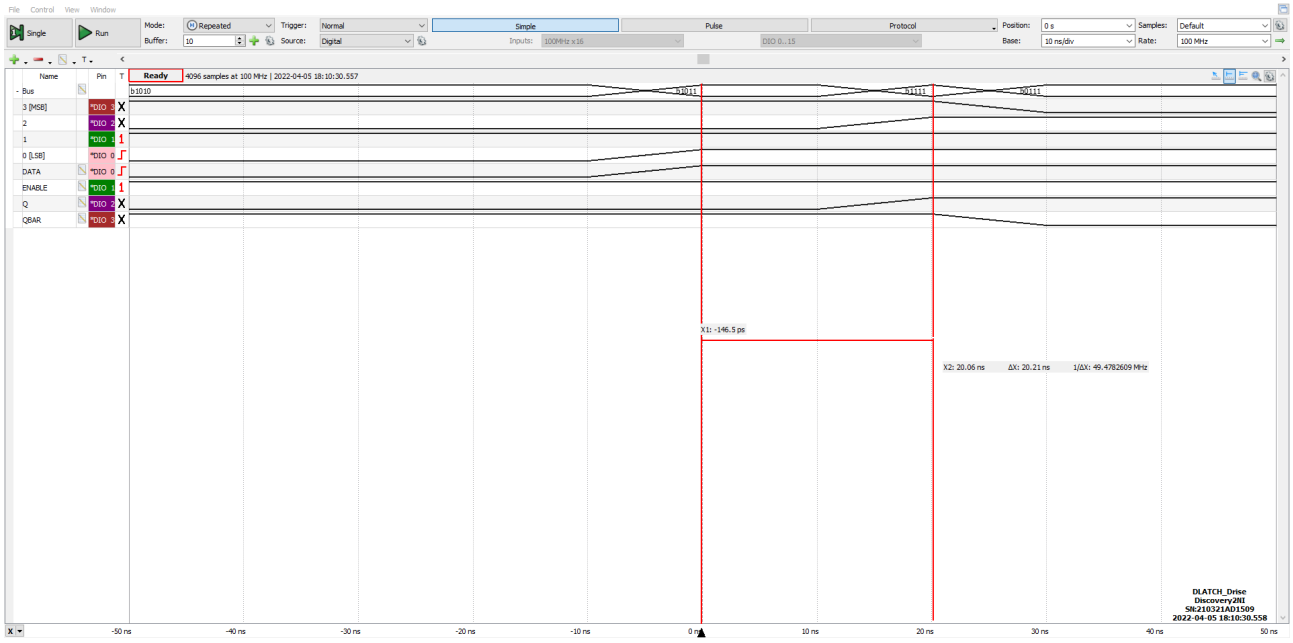

Figura 5: Acquisizione da oscilloscopio digitale della transizione 2 del D-Latch

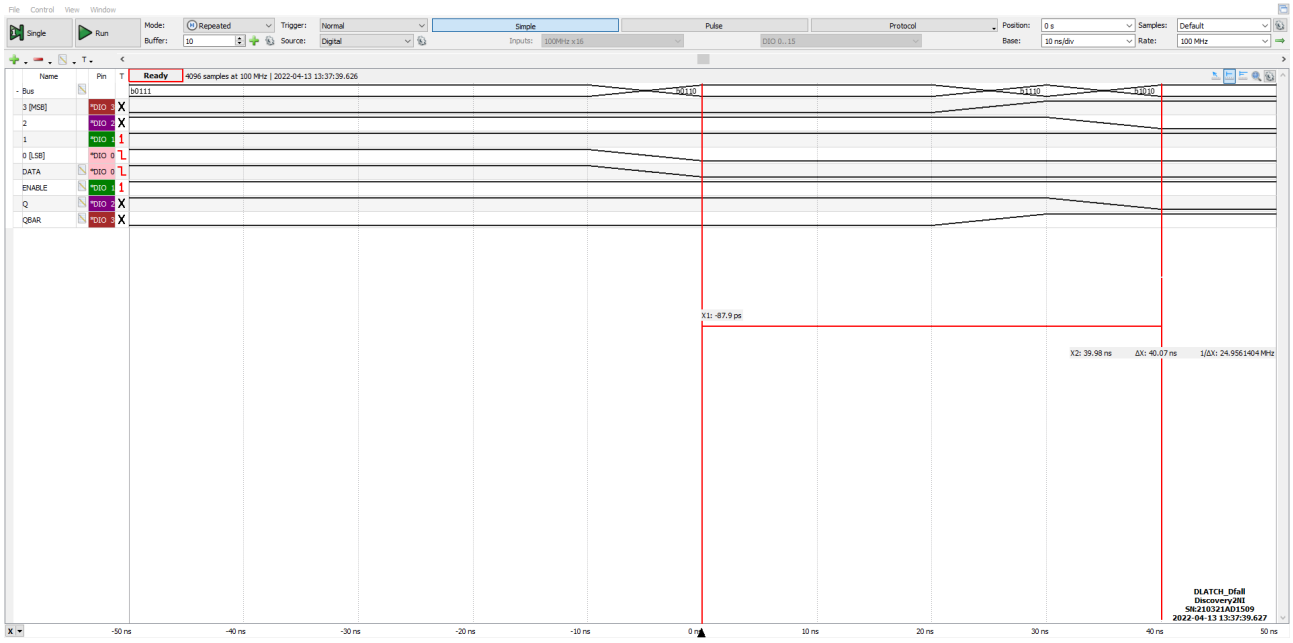

Figura 6: Acquisizione del Logic Analyzer durante la transizione 3 del D-Latch

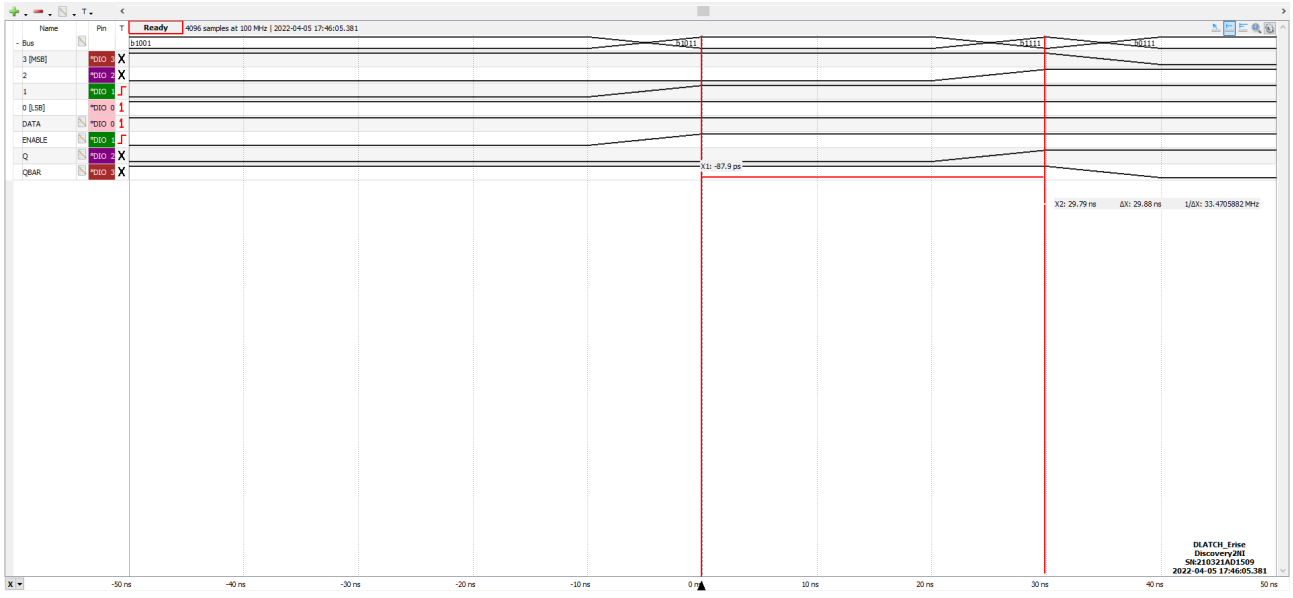

Figura 7: Acquisizione del Logic Analyzer durante la transizione 4 del D-Latch

	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.

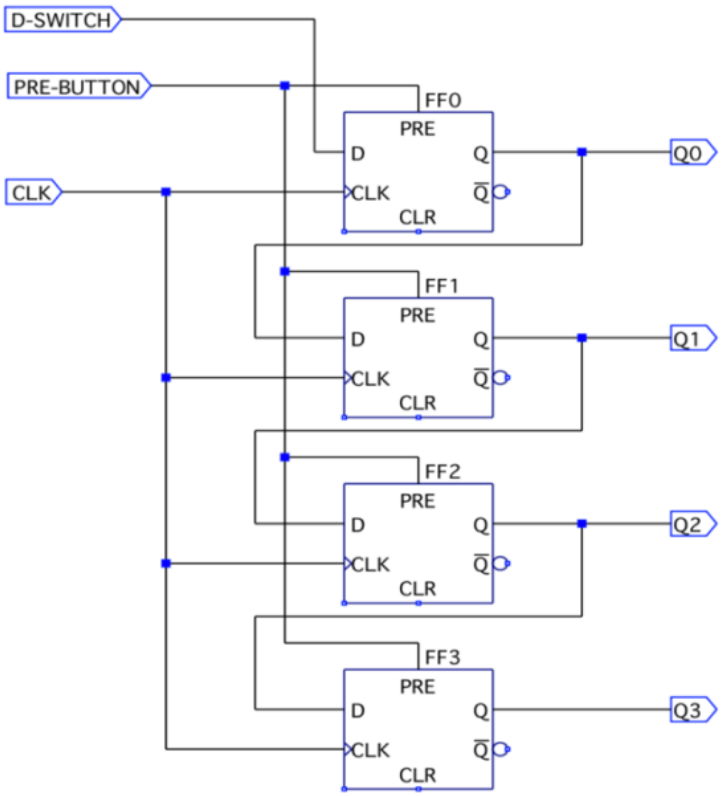

Figura 8

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

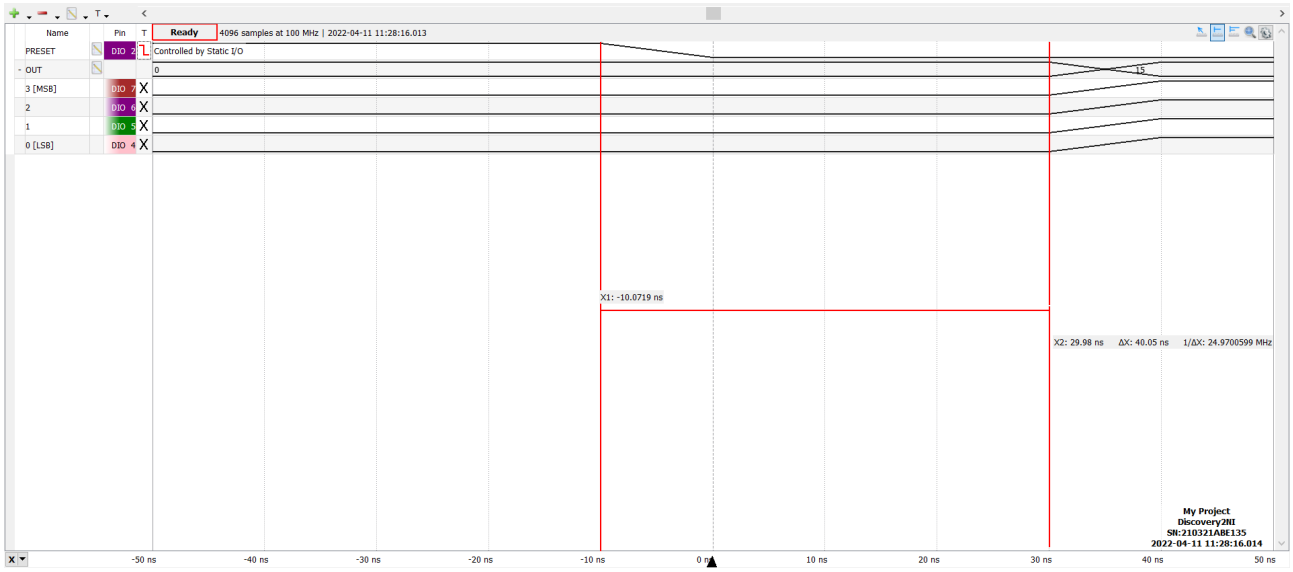

Figura 9: Acquisizione con Logic dei segnali in PRE-BUTTON le 4 uscite dal registro a scorrimento pilotato dall'ingresso di PRESET

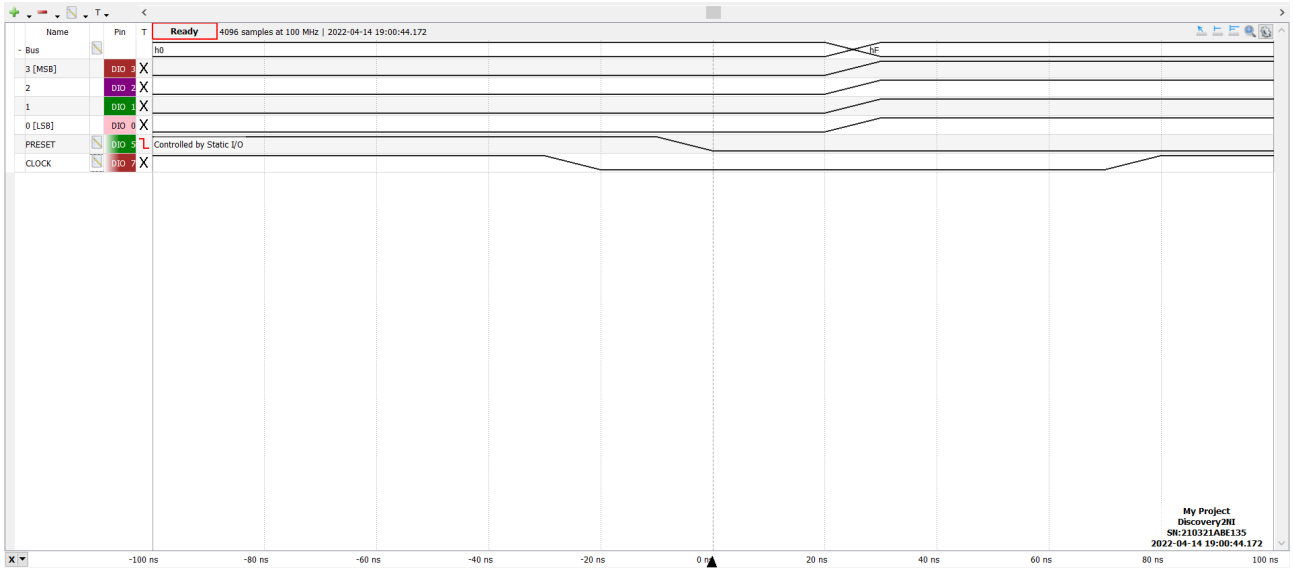

Figura 10: Acquisizione con Logic dei segnali in PRE-BUTTON, CLOCK e $Q_{0,\dots,3}$ in uscita dal registro a scorrimento studiato

$Q_i(t = t')$	$Q_i(t = t' + T)$
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.

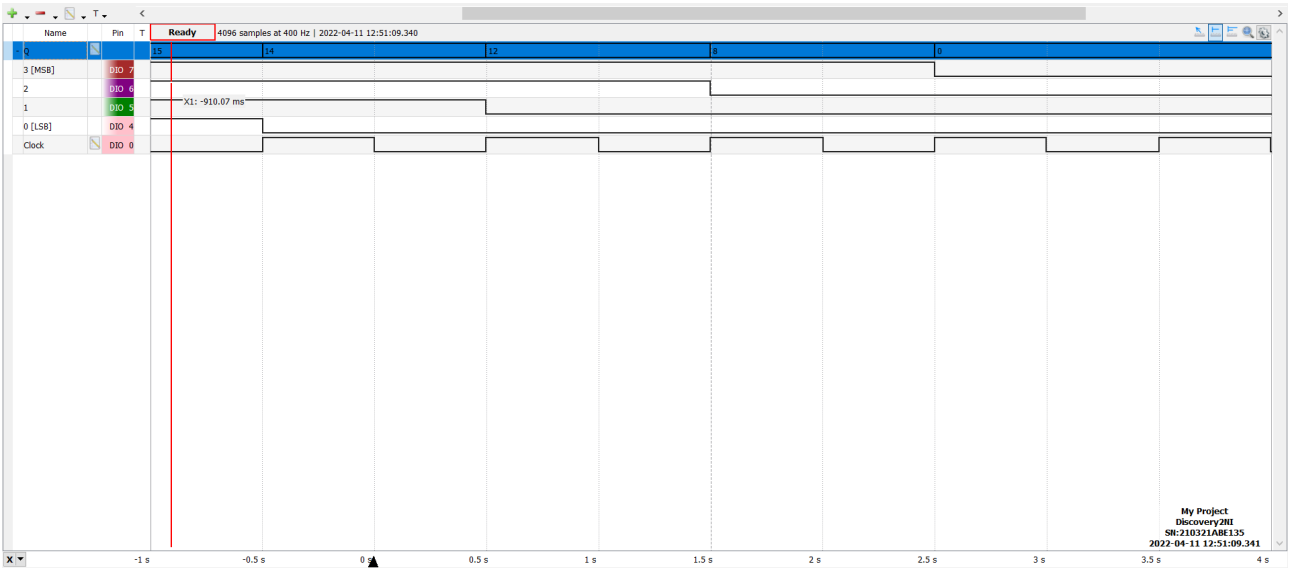

Figura 11: Acquisizione con Logic dei segnali in uscita da un registro a scorrimento di 4 bit come illustrato in sezione 3.c

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 $f_{\text{clk}} = 1$ kHz all'ingresso del circuito. Si riporta in fig. 12 l'acquisizione con Logic Analyzer dei segnali di clock e dei 4 output in funzione del tempo.

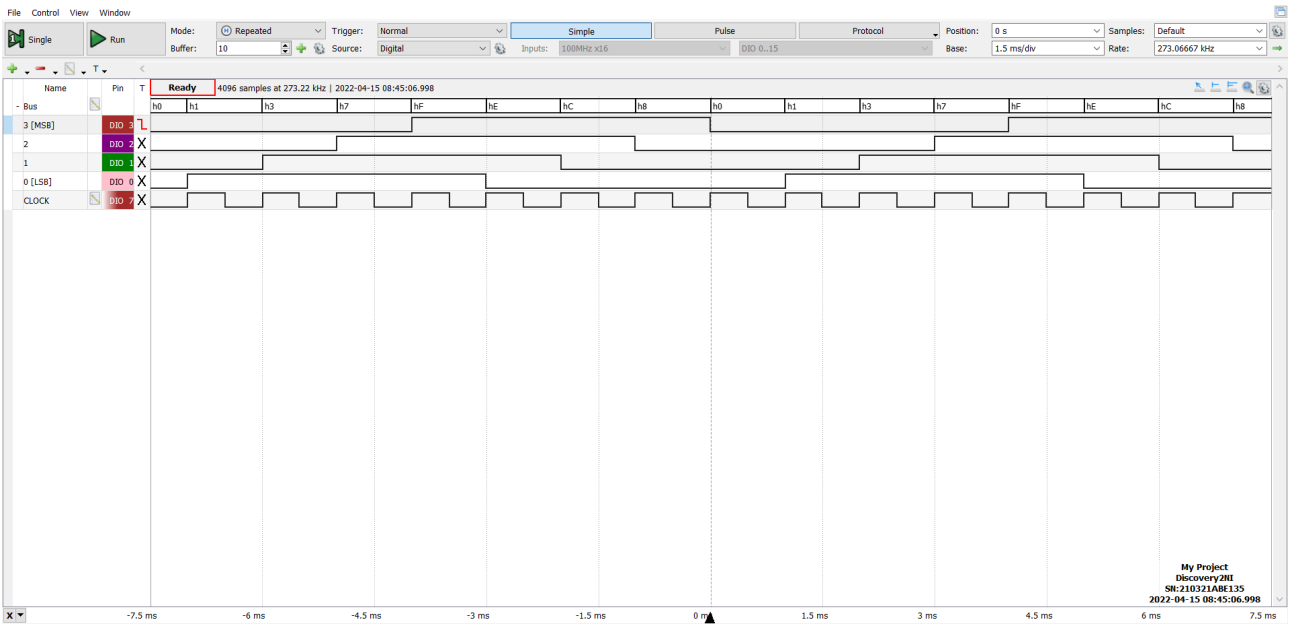

Figura 12: Acquisizione con Logic dei segnali in uscita dal registro a scorrimento con l'ultima uscita negata collegata all'ingresso del primo FF.

Da questa vediamo che in ogni uscita Q_i si ha un'onda quadra con frequenza $f = f_{\text{clk}}/8$, ciascuna traslata temporalmente di un periodo di clock rispetto alla Q_{i-1} precedente; per cui nel registro osserviamo la sequenza atomica 00001111 ripetersi ogni 8 cicli di clock.

Questo è dovuto al fatto che nel registro ora scorre in modo ciclico la concatenazione dello stato iniziale degli output e del suo negato (tramite il collegamento ad anello $\overline{Q_3} \to Q_0$). Supponendo infatti che nel circuito siano collegati ad anello n edge-triggered Flip-Flop, in una qualsiasi delle uscite del twisted-ring Johnson counter ci si aspetta di trovare un clock di frequenza divisa di un fattore $2n$.

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 studiato in precedenza e una porta XOR (dal chip integrato SN74LS86 Quad XOR Gate). La schematica del circuito che utilizzeremo è riportata in figura fig. 13.

4.b Analisi e verifica del funzionamento

Dopo aver montato il circuito si inizializzano tutti Flip-Flop a 1 con PRESET, si invia un segnale di clock a 10 kHz, sempre collegando gli output $Q_{0,\dots,3}$ ai canali del Logic Analyzer per verificarne il funzionamento.

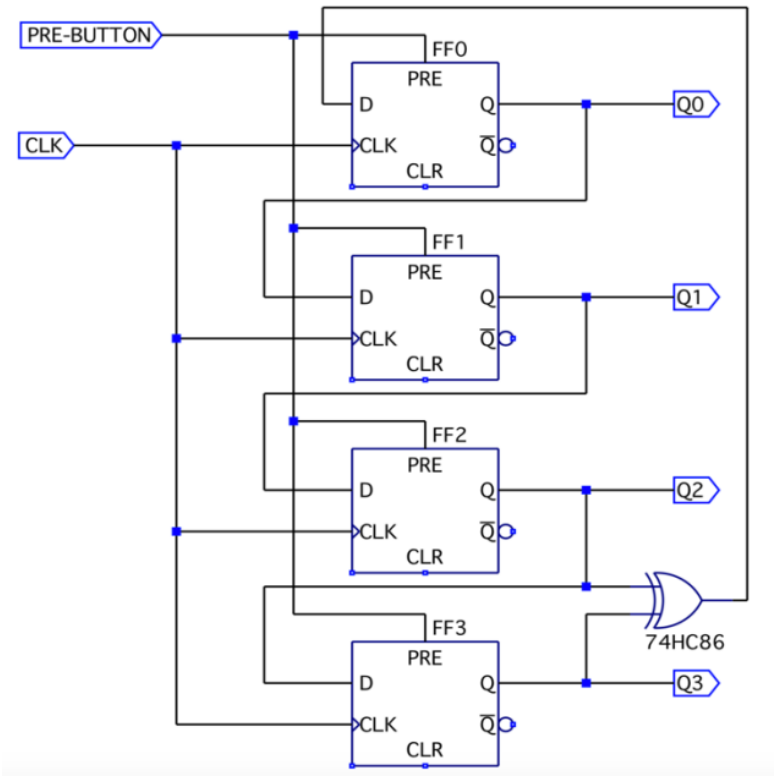

Figura 13

Poiché il registro costruito ha memoria di $n = 4$ bit, dalla teoria ci aspettiamo che la sequenza si ripeta dopo $2^4 = 16$ eventi al massimo, condizione che si ottiene utilizzando come TAP (i.e. segnali in ingresso alla porta XOR, la cui uscita è inviata all'ingresso $D = Q_0$ del primo FF) gli output Q_2 (o Q_0) e Q_3.

Dall'acquisizione riportata in fig. 14 si osserva la sequenza riportata in tabella 4 ripetersi ogni 16 periodi di clock in accordo con le aspettative (prendendo come riferimento una qualsiasi uscita dello shift-register, visto che la sequenza nelle altre uscite sarà la medesima, ma sfasata lungo l'asse temporale di qualche ciclo di clock).

4.c Studio delle sequenze generabili con diverse condizioni iniziali

Si provano quindi altre combinazioni di TAP, per verificare che la scelta di Q_2 e Q_3 produca una sequenza più lunga rispetto alle altre possibili configurazioni. Riportiamo in fig. 15, fig. 16 e fig. 17 i risultati ottenuti con Logic Analyzer dell'AD2.

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 (integrato SN74LS163 synchronous binary counter with synchronous clear/load) secondo lo schema riportato in fig. 18.

5.b Verifica del ciclo di funzionamento dei contatori

Con la funzione Patterns di Waveform si invia un segnale di clock di frequenza $f_{\text{clk}} = 10$ kHz al pin (CLK) del contatore e si acquisiscono i segnali in uscita $Q_{0,\dots,3}$ con la funzione Logic dello stesso.

Dalla fig. 19 si osserva che il bus ($Q_3Q_2Q_1Q_0$ in formato esadecimale) collegato ai segnali in uscita dal contatore a 4 bit incrementa in maniera sequenziale dallo stato 0000 (h0) fino allo stato 1111 (hF) in ordine crescente.

Da questa si verifica che il circuito esplora tutti gli stati possibili nell'ordine corretto per il buon funzionamento del contatore a 4 bit.

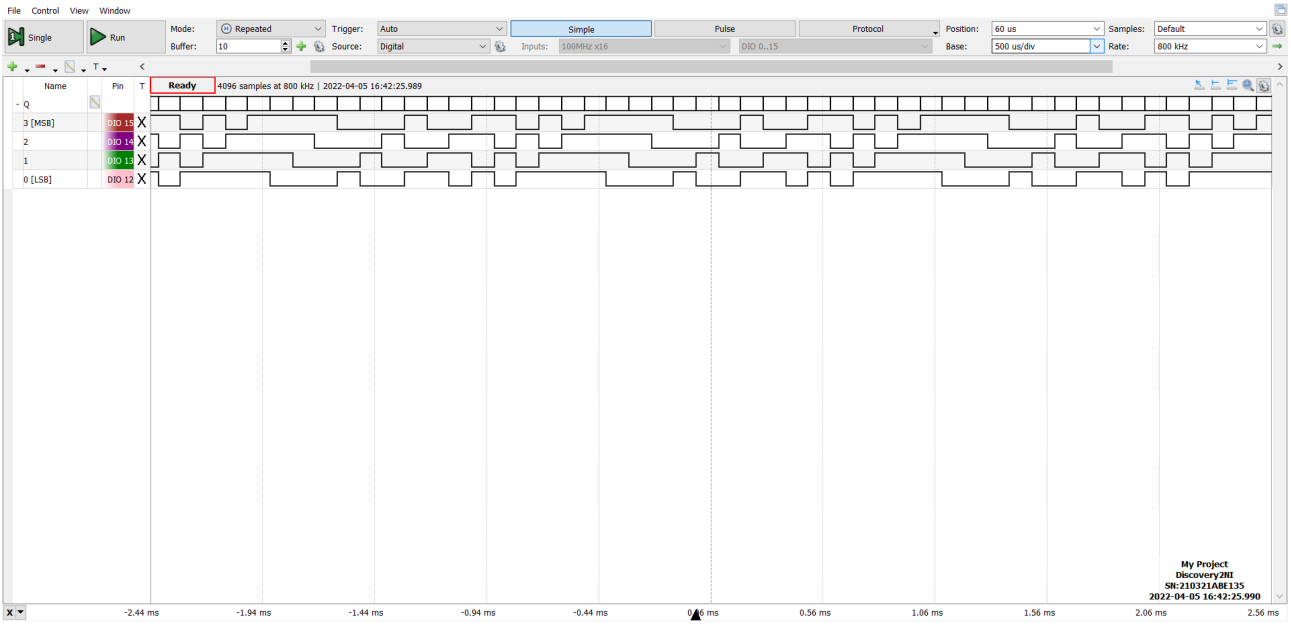

Figura 14: Acquisizione temporale con Logic del bus in uscita dal generatore di sequenze psudo-casuale descritto in fig. 13

Q_0	Q_1	Q_2	Q_3
0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	1
0	0	1	1
0	1	1	0
1	1	0	1
1	0	1	0
0	1	0	1
1	0	1	1
0	1	1	1
1	1	1	1
1	1	1	0
1	1	0	0
1	0	0	0
0	0	0	1

Tabella 4: Sequenza pseudo-casuale attesa collegando come TAP alla porta XOR i segnali Q_2 e Q_3 in uscita dal registro a scorrimento.

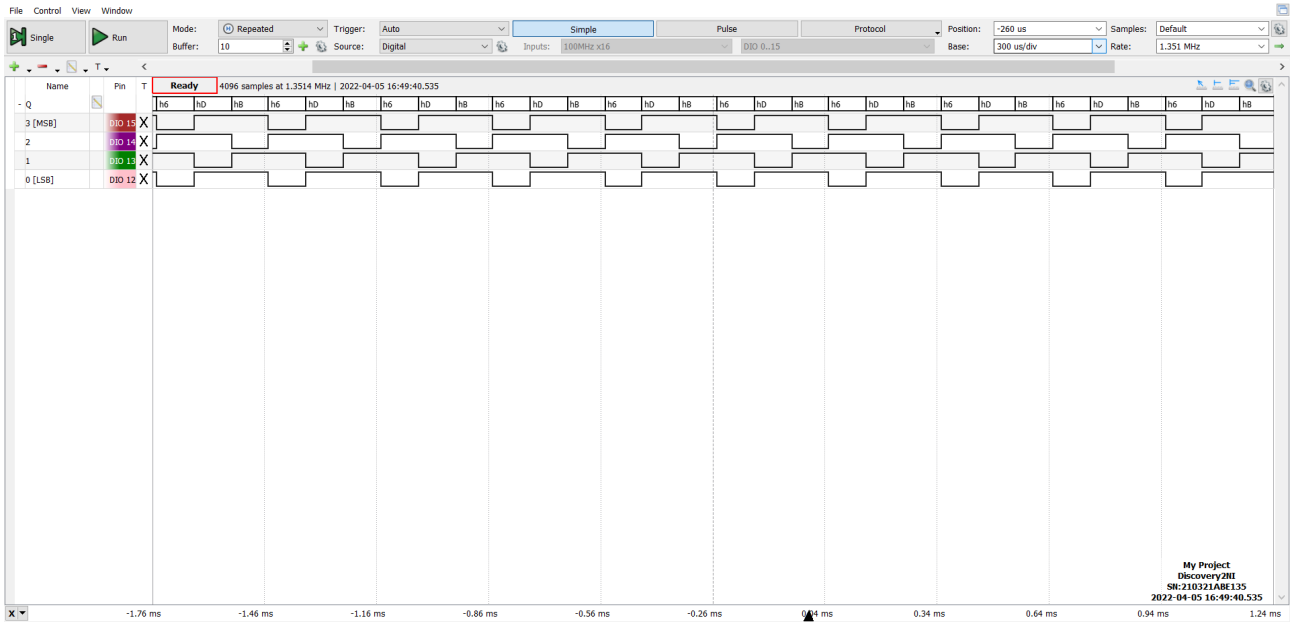

Figura 15: Acquisizione con Logic del bus in uscita dal generatore di sequenze pseudo-casuali con TAP sulle uscite Q_0 e Q_1; la sequenza si ripete ogni 4 cicli di clock.

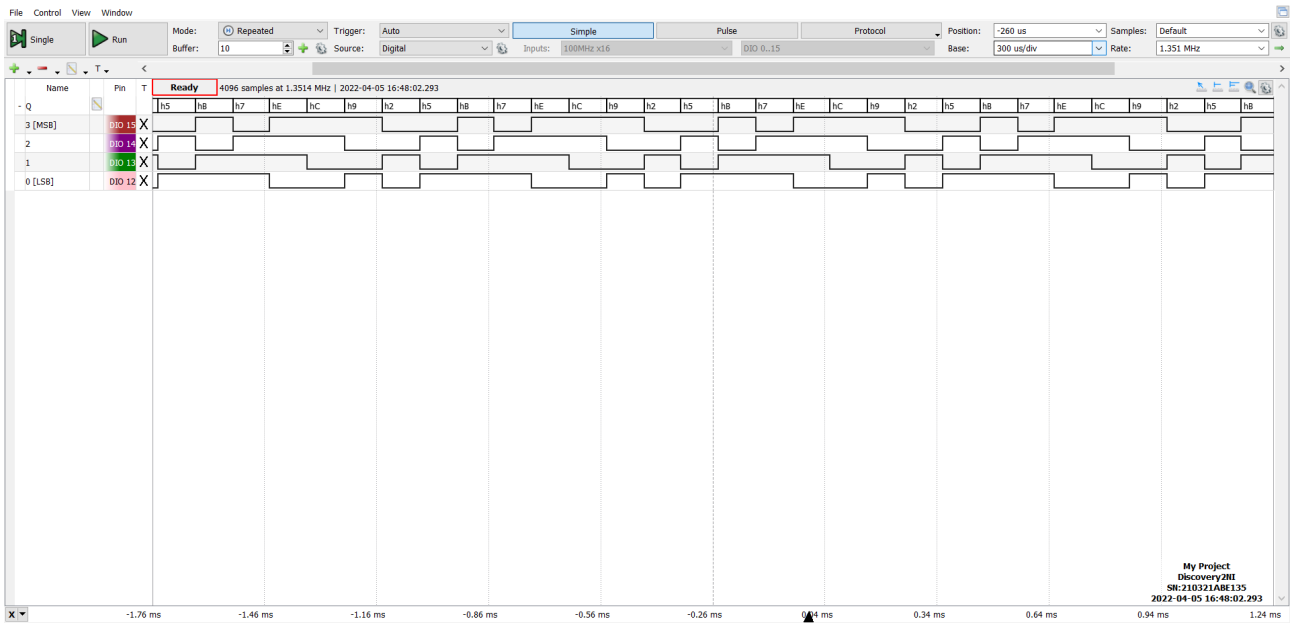

Figura 16: Acquisizione con Logic del bus in uscita dal generatore di sequenze pseudo-casuali con TAP sulle uscite Q_2 e Q_1; la sequenza si ripete ogni 8 cicli di clock.

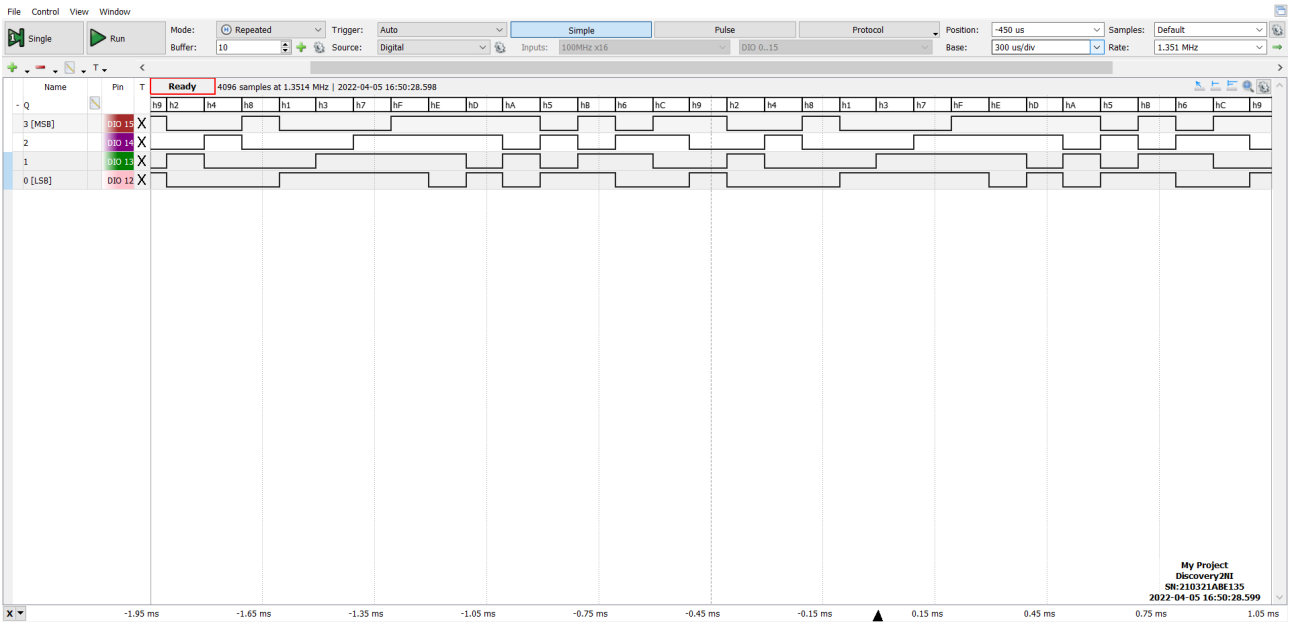

Figura 17: Acquisizione con Logic del bus in uscita dal generatore di sequenze pseudo-casuali con TAP sulle uscite LaQ_0 e Q_3, la sequenza è massimale (si ripete ogni $2^n = 16$ cicli di clock).

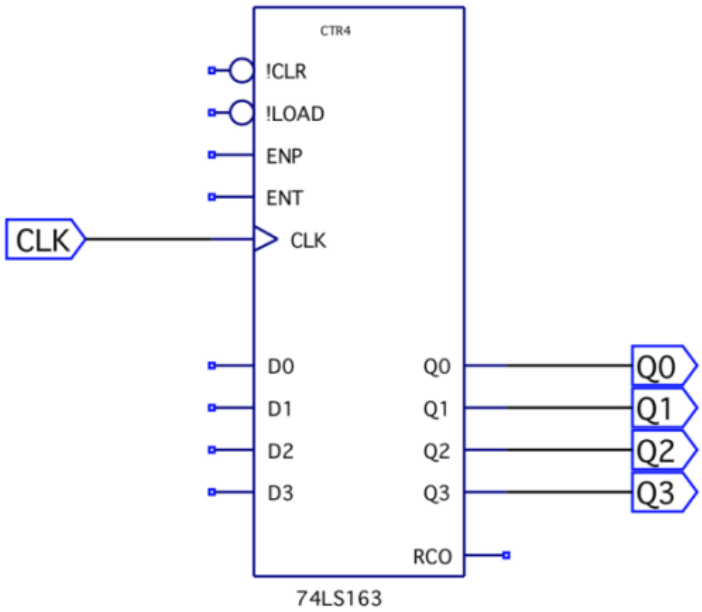

Figura 18

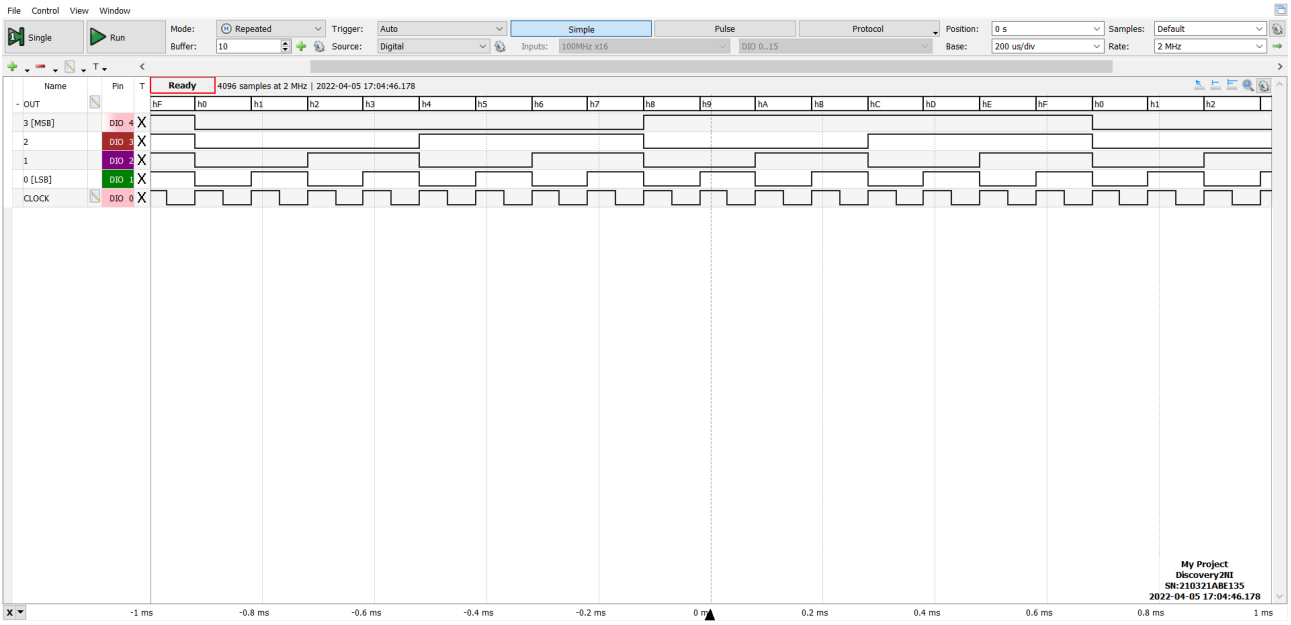

Figura 19: Acquisizione con Logic dell'andamento temporale dei segnali in uscita dal contatore SN74LS163, con frequenza di clock pari a 10 kHz.

5.c Verifica della divisione in frequenza

Dato che il contatore (composto da 4 positive-edge-triggered FF) incrementa di 1 ad ogni fronte di salita del clock, ci si aspetta che il segnale di ordine i abbia come frequenza $f_{\text{clk}}/2^{i+1}$, la metà di quella del bit precedente.

In termini del BUS di prima con parola di lunghezza 4 bit, ci si aspetta quindi che le frequenze dei segnali per ciascun output del contatore siano:

- $f_{\text{clk}}/2$ per il LSB Q_0
- $f_{\text{clk}}/4$ per Q_1
- $f_{\text{clk}}/8$ per Q_2
- $f_{\text{clk}}/16$ per il MSB Q_3

Sempre facendo riferimento alla fig. 19 risulta evidente come le uscite del contatore si comportino da divisori in frequenza; infatti i periodi che si trovano analizzando l'acquisizione sono rispettivamente 2 (LSB), 4, 8 e 16 (MSB) volte il periodo del clock $T_{\text{clk}} = 1/f_{\text{clk}}$.

5.d Transizione sincrona del contatore

Impostiamo ora la condizione di trigger in Logic affinché l'acquisizione abbia inizio quando il segnale in Q_3 scende da alto a basso, in modo da poter studiare gli eventuali ritardi delle singole uscite nella transizione $1111 \rightarrow 0000$ e verificare il comportamento sincrono del contatore.

Dalla fig. 20 possiamo vedere che la commutazione delle uscite è sincrona e avviene simultaneamente per ogni pin di output con un ritardo di $t_{PHL} = 30 \pm$ ns dal fronte di salita del clock, in linea con quanto previsto dalle specifiche dell'integrato.

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

Per realizzare un divisore in frequenza che generi un segnale di periodo pari a $T = 10T_{\text{clk}}$ facciamo uso del pin CLEAR (Active-Low) del contatore, il quale resetta il contatore allo stato 0000 quando riceve un segnale a livello logico basso.

Affinché il contatore raggiunga un totale di 10 stati prima di ripartire, dobbiamo imporre la condizione che arrivati allo stato 9 (b1001) il contatore riceva il comando di reset da CLEAR. Per farlo colleghiamo ai 2 ingressi di una porta NAND il LSB e il MSB e la sua uscita al pin di Clear, cosicché quando il contatore arriva a 9 (entrambi Q_0 e Q_3 sono alti) CLEAR riceverà un segnale di reset, per cui la frequenza del segnale in uscita da Q_3 corrisponderà ad un decimo di quella di clock. Si riporta in fig. 21 lo schema proposto per il circuito.

Come prima, si utilizza la funzione Logic per acquisire i segnali provenienti dal Bus dei 4 bit in uscita, il clock e il bit di Clear.

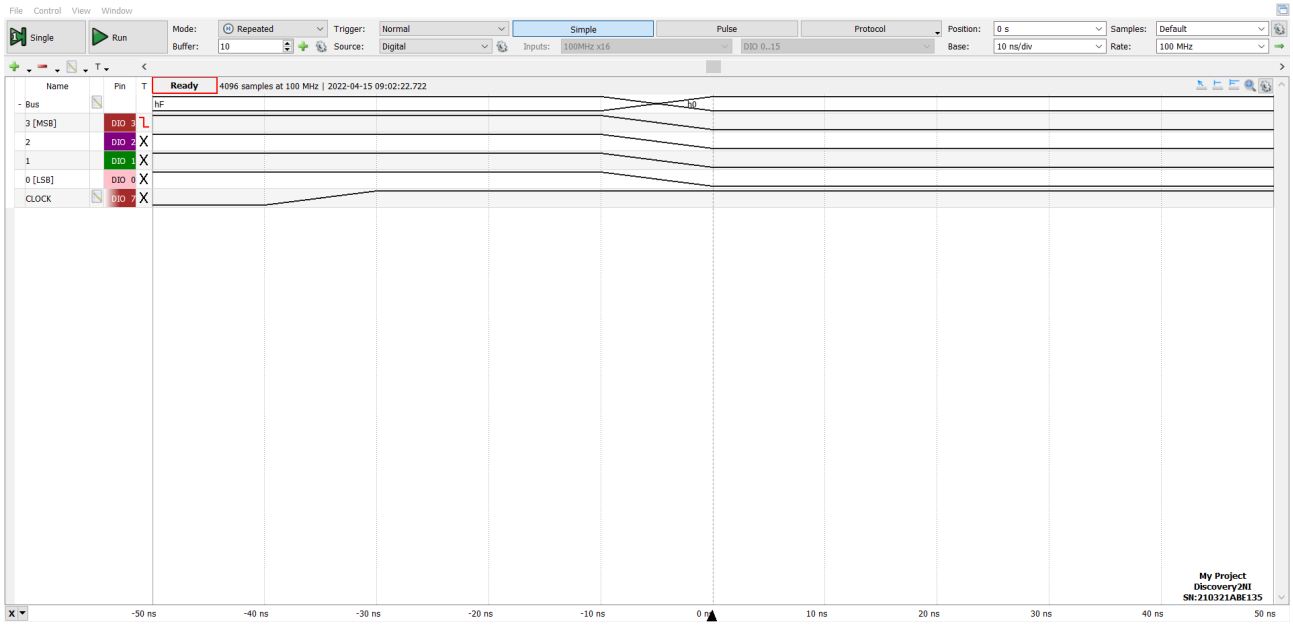

Figura 20: Acquisizione temporale con Logic del bus in uscita dal contatore durante la transizione 15->0; dall'immagine possiamo notare il comportamento sincrono della commutazione delle uscite del contatore

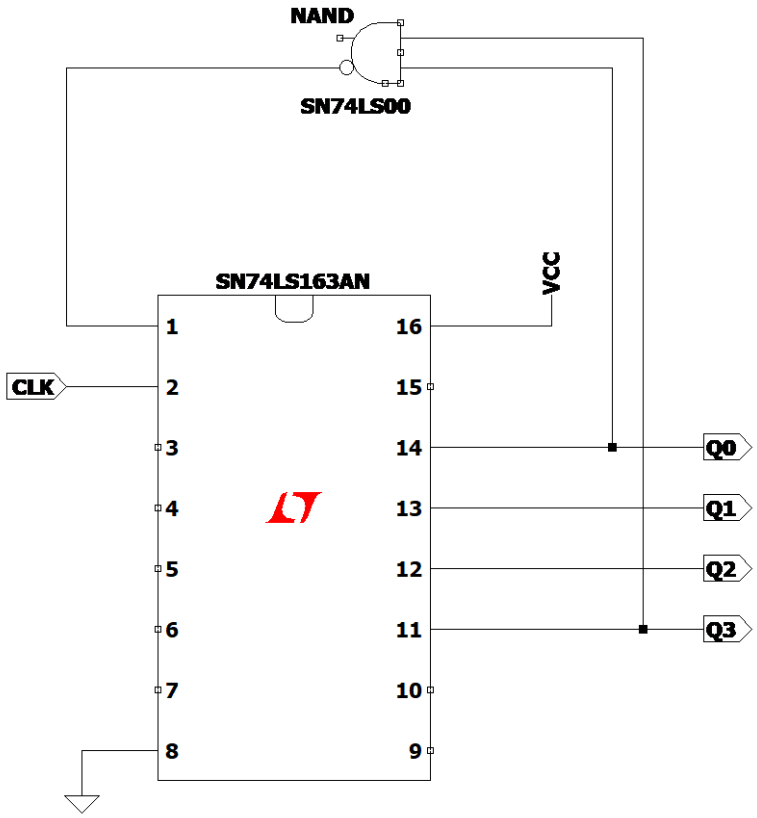

Figura 21: Schematica utilizzata per il divisore per 10 di frequenza

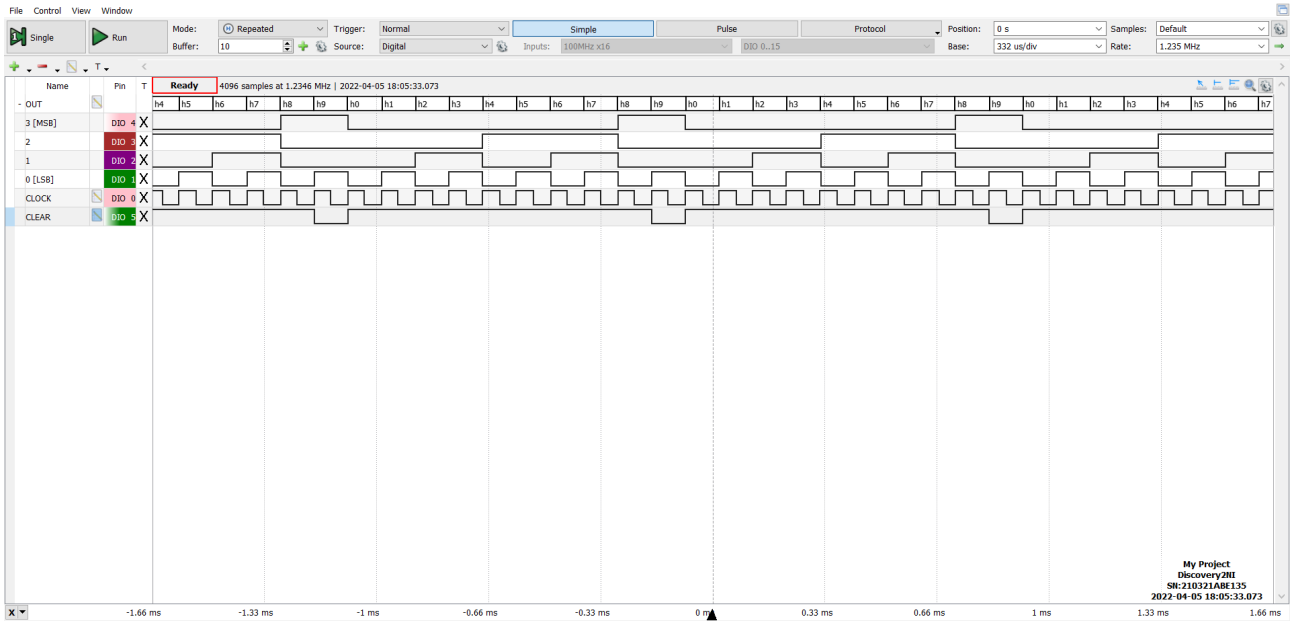

Figura 22: Acquisizione con Logic del bus in uscita dal circuito contatore BCD. Si può notare che il segnale presente in Q_3 ha periodo pari a 10 volte quello del clock

Come visibile dall'acquisizione riportata in fig. 22, troviamo che il segnale in Q_3 è un'onda quadra di frequenza $f = f_{\text{clk}}/10$ come volevamo e Duty-Cycle pari al 20 %.

5.f Divisore di frequenza programmabile con RCO

Infine si vuole costruire un divisore di frequenza programmabile: per questo scopo utilizzeremo il bit RCO (Ripple Carry Output), la cui funzione è di generare un segnale alto solo nel caso in cui il contatore abbia raggiunto lo stato massimo (1111) e il bit Load (Active-Low), che quando è basso sovrascrive lo stato $Q_3, \ldots, Q_0$ del contatore con quello presente nel bus di input $D_3, \ldots, D_0$.

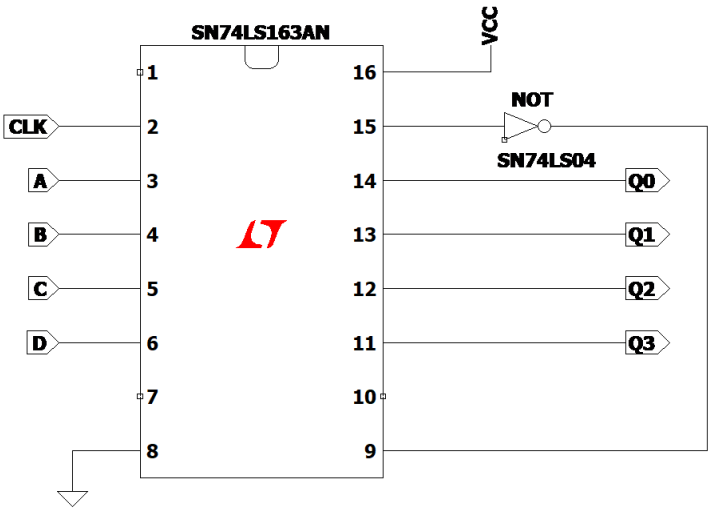

Figura 23: Schematica utilizzata per il divisore di frequenza programmabile

Vogliamo preliminarmente verificare il funzionamento del RCO; dopo 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. Dall'acquisizione riportata in sezione 5.f possiamo notare che il bit RCO funziona correttamente.

Quindi, per costruire il divisore programmabile schematizzato in fig. 23 si collega l'uscita dal pin RCO all'ingresso del pin LOAD dello stesso contatore tramite una porta NOT (integrato SN74LS04). Si utilizza

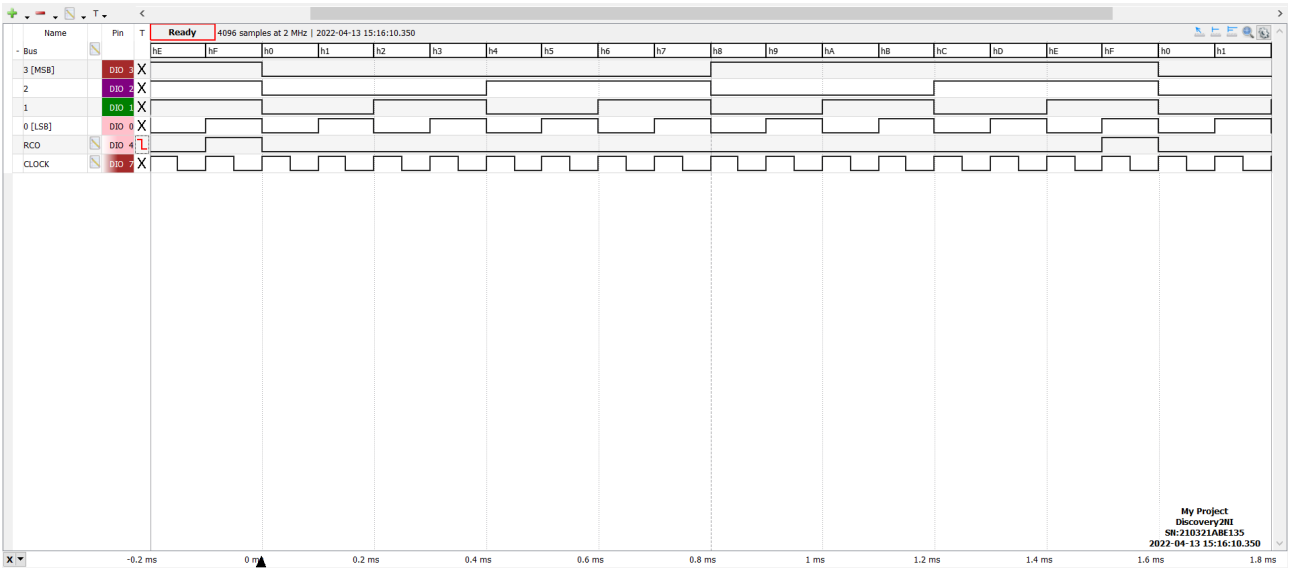

Figura 24: Acquisizione con Logic dei segnali in uscita dal contatore per verificare il funzionamento del bit RCO

dunque la funzione StaticIO per caricare nel Bus di ingresso lo stato 0101 (5 arbitrariamente scelto), mantenendo collegato al circuito il segnale di clock generato da Patterns. Dalla fig. 25 notiamo che quando il contatore riceve

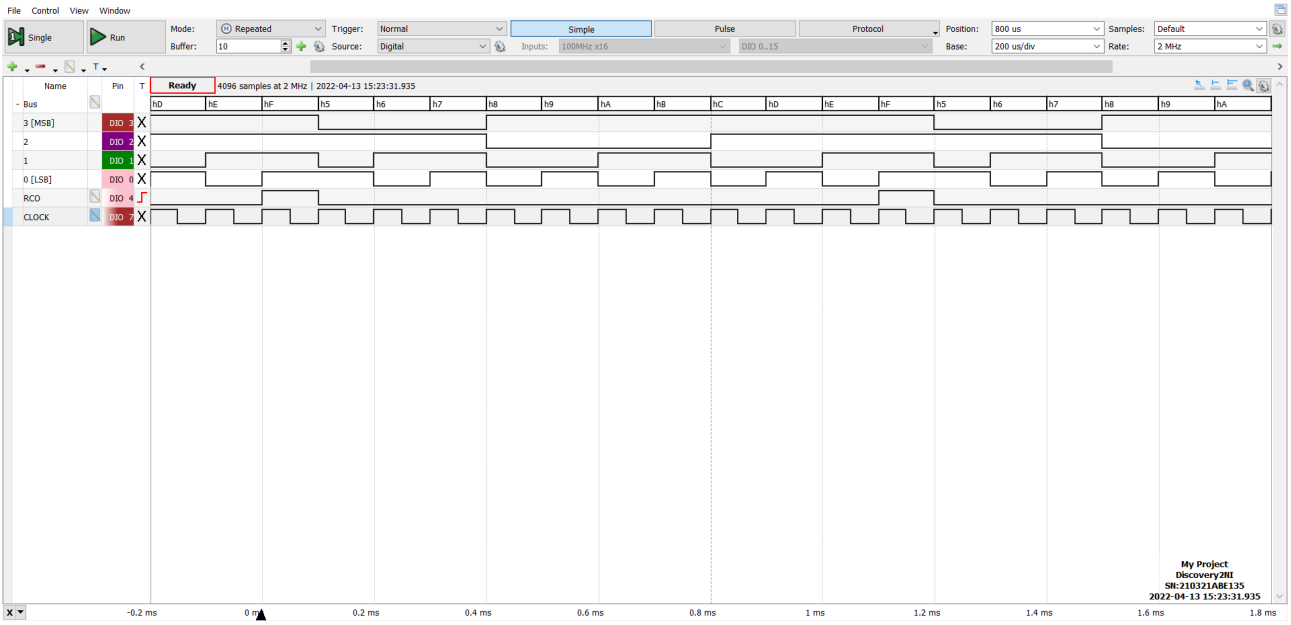

Figura 25: Acquisizione con Logic dei segnali che interessano il circuito divisore di frequenza programmabile, con sequenza di avvio 0101

il segnale di LOAD, quindi quando si registra 1 nel bit RCO, la sequenza iniziale non viene immediatamente caricata, ma sarà posticipata da un evento di clock (≈ 100 µs). 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 $0 \rightarrow 1$: in seguito a quanto visto nella sezione precedente ci aspettiamo che il contatore passi dallo stato 1111 allo stato iniziale.

Visualizziamo i segnali sulle linee di dato e RCO su scala dei tempi molto ristretta per misurare i ritardi tra di loro e verificare quanto affermato nella sezione 5.d. Dalla fig. 26 si può notare che è sempre presente un ritardo dall'inizio del nuovo fronte di salita del clock alla commutazione delle uscite (che come prima è sincrona/simultanea) e che vale come prima 30 ± 10 ns, per cui la funzione di LOAD risulta sincrona.

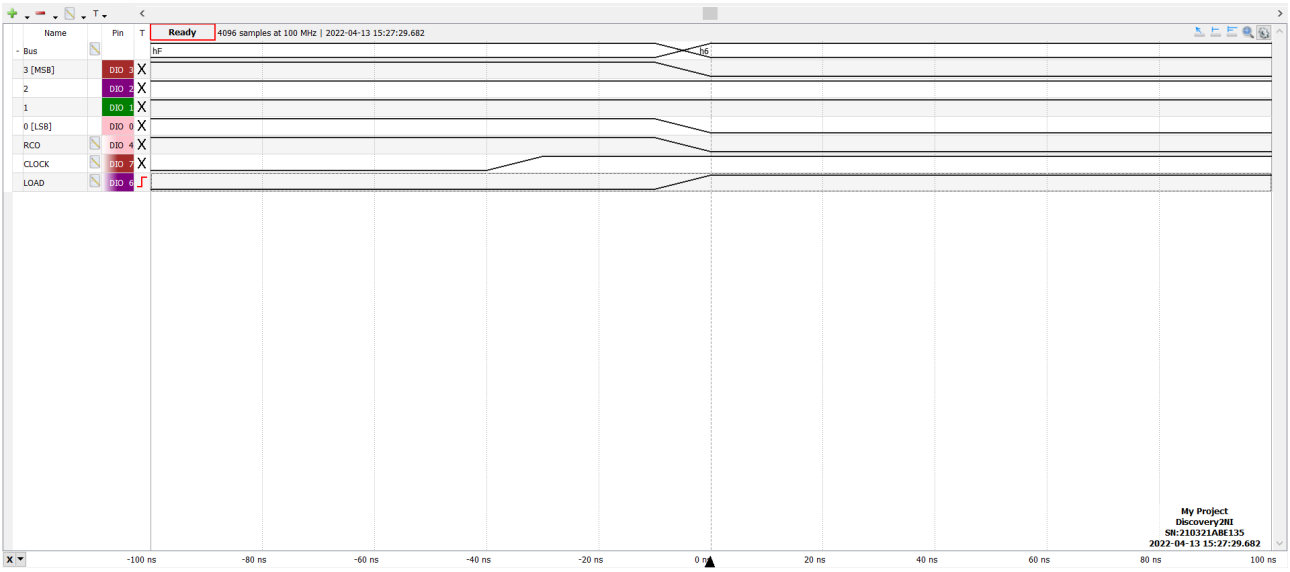

Figura 26: Acquisizione con logic dei segnali di interesse per il circuito divisore di frequenza programmabile per un fondo scala dei tempi molto ristretto, con sequenza di avvio 0110

5.h Analisi e verifica del comportamento del divisore RCO

In questa sezione prendiamo la frequenza del segnale in uscita dal RCO come frequenza quoziente/desiderata in uscita dal divisore.

Dato che RCO è attivo quando tutte le uscite Q_i sono alte, il LOAD (normalmente alto) viene attivato quando l'output corrisponde a 15. Al fronte di salita del clock successivo le uscite vengono impostate ai valori degli ingressi e il conteggio riparte. Dunque se si vuole realizzare un divisore di frequenza $1/n$, o un contatore a n stati, occorre impostare il bus di input in modo che il conteggio parta da $16 - n$, quindi si osservano in uscita i numeri da $(16 - n)_{16}$ fino a 15 ripetersi.

In altre parole 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; detto I il valore (in base decimale) caricato nello stato iniziale si ha dunque:

$$T_{\text{divisore,I}} = T_{\text{clk}} \times (16 - I) \tag{2}$$

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 iniziale a 0000 invece ci si aspetta che il comportamento del circuito si riduca allo stesso visto precedentemente 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 per inizializzare il contatore, per cui in particolare ci si aspetta di trovare:

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

Dalle acquisizioni del Logic Analyzer riportate in si vede che il comportamento del circuito è coerente con quanto ci aspettavamo.

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

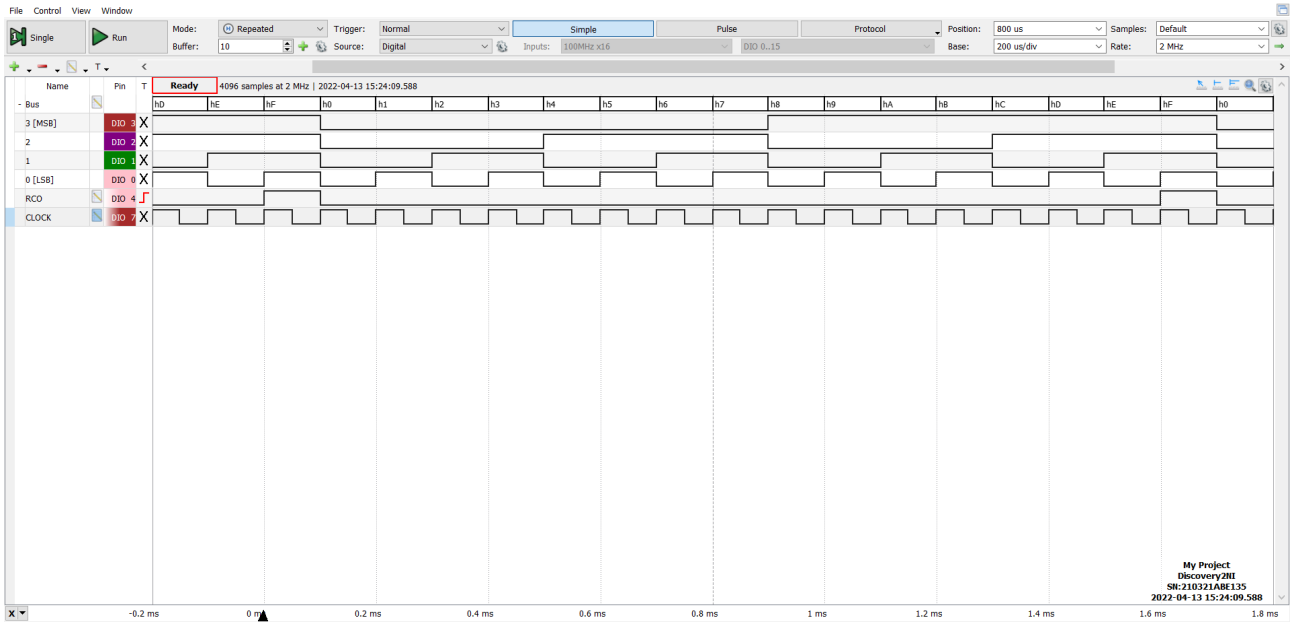

Figura 27: Acquisizione con Logic dei segnali di interesse per il circuito divisore di frequenza programmabile, con sequenza di avvio 0000, da cui si misura $T = 16T_{\text{clk}}$

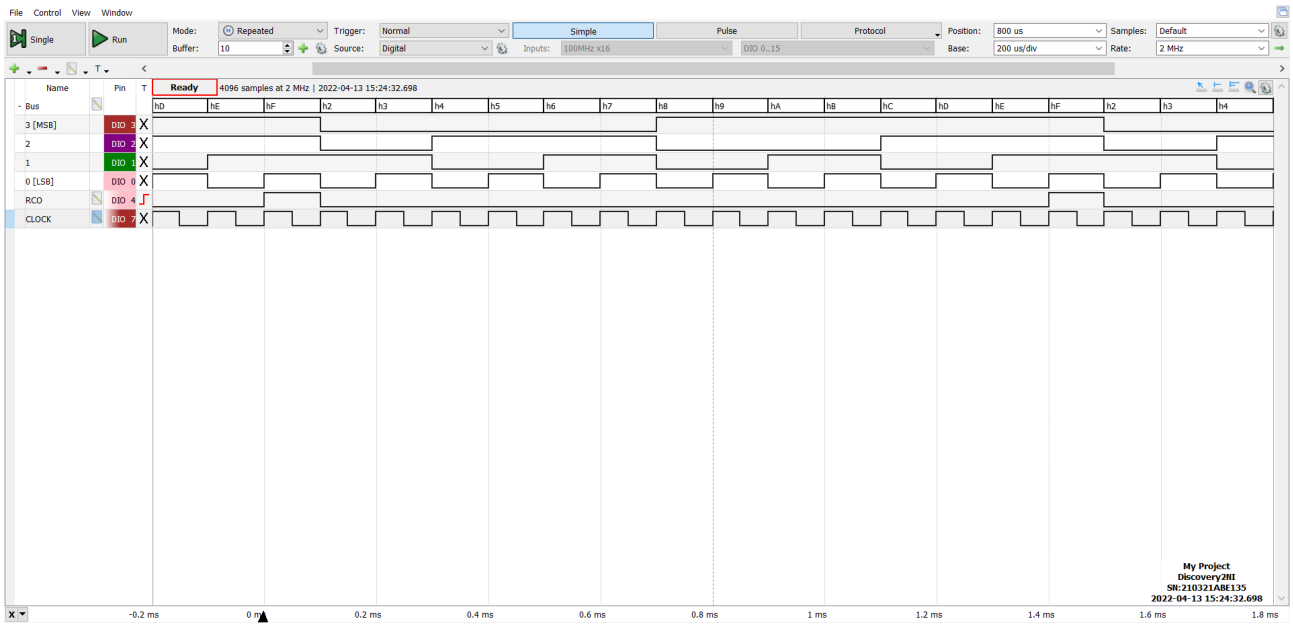

Figura 28: Acquisizione con Logic dei segnali di interesse per il circuito divisore di frequenza programmabile, con sequenza di avvio 0010, da cui si misura $T = 14T_{\text{clk}}$

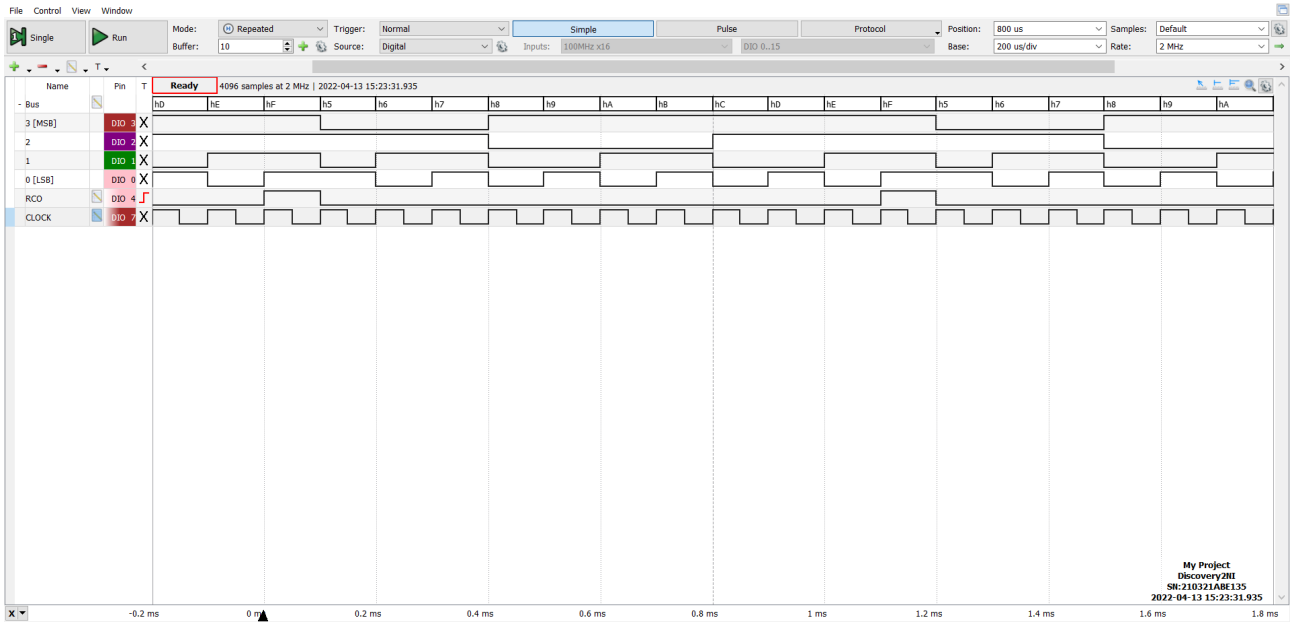

Figura 29: Acquisizione con Logic dei segnali di interesse per il circuito divisore di frequenza programmabile, con sequenza di avvio 0101, da cui si misura $T = 11T_{\text{clk}}$

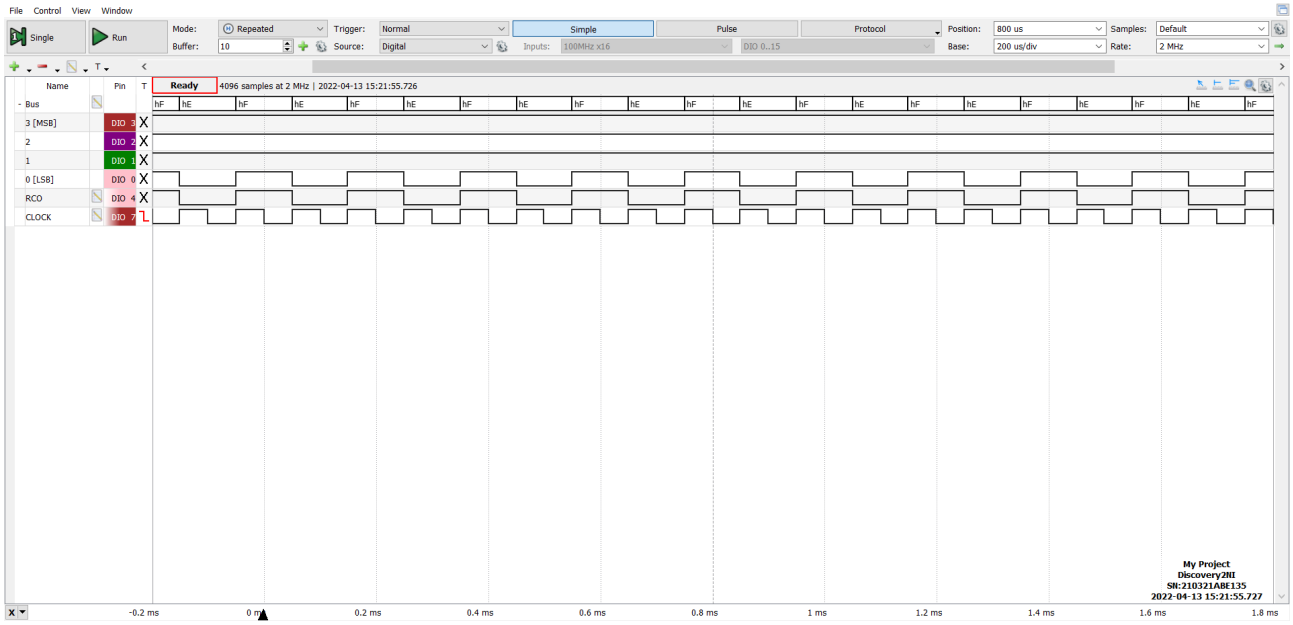

Figura 30: Acquisizione con Logic dei segnali di interesse per il circuito divisore di frequenza programmabile, con sequenza di avvio 1110, da cui si misura $T = 2T_{\text{clk}}$

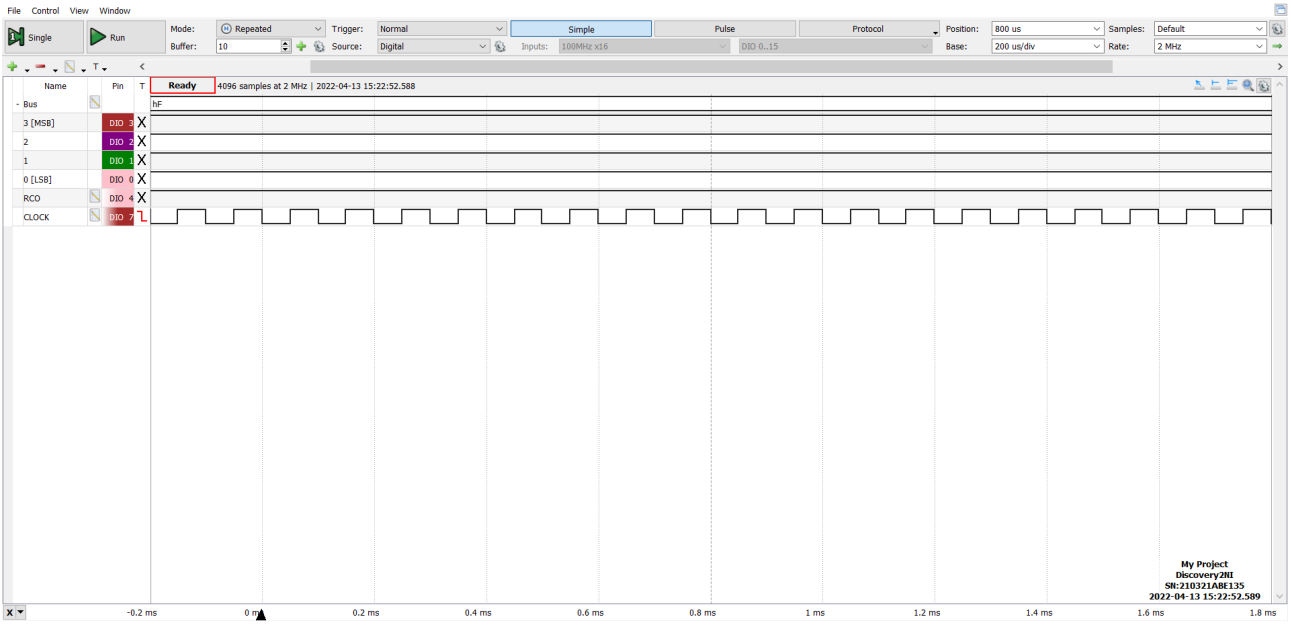

Figura 31: Acquisizione con Logic dei segnali di interesse per il circuito divisore di frequenza programmabile, con sequenza di avvio 1111, dal quale si vede che RCO è un segnale costante

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.

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.