22.05.2021

Implementare timer cu ajutorul VHDL

Realizat de: Popa Cristian Mihai

Profesor îndrumător: ing. Pop Diana-Irena

Cuprins

1. Specificația proiectului

Scopul lucrării este realizarea unui dispozitiv cu funcția de temporizator/cronometru cu ajutorul unei plăcuțe FPGA. Dispozitivul realizat poate afișa pe 4 afișoare BCD cu 7 segmente timpul, măsurat în minute și secunde. El prezintă două moduri de funcționare: poate funcționa ca și cronometru, măsurând timpul crescător, sau poate funcționa ca un temporizator, măsurând timpul descrescător, iar când se atinge valoarea 00:00, un semnal sonor va atenționa utilizatorul. Vom marca trecea unei secunde cu ajutorul unui semnal cu frecvența de 1 Hz, deci cu o perioadă T = 1 sec.

2. Proiectarea timer-ului

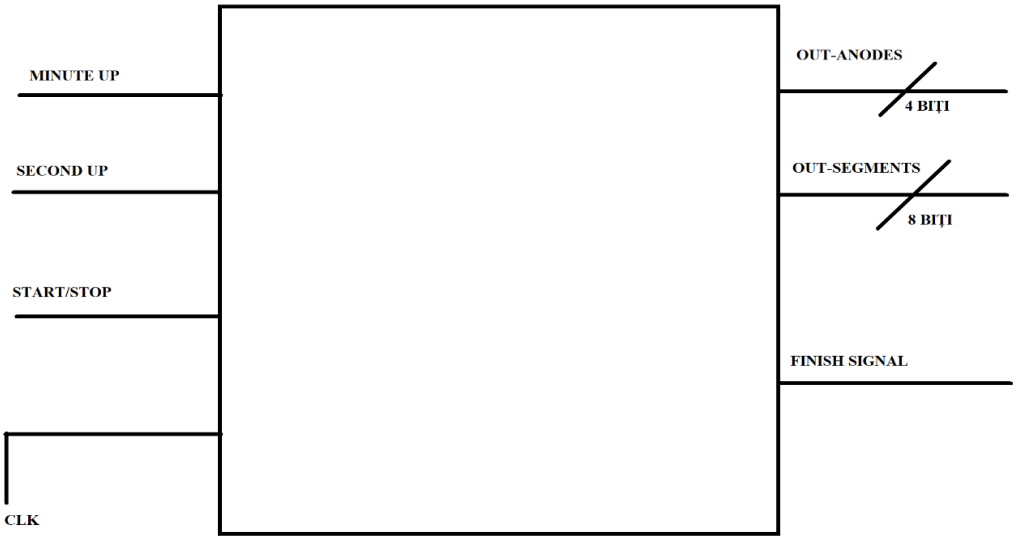
2.1. Schema Bloc

2.1.1. Resurse

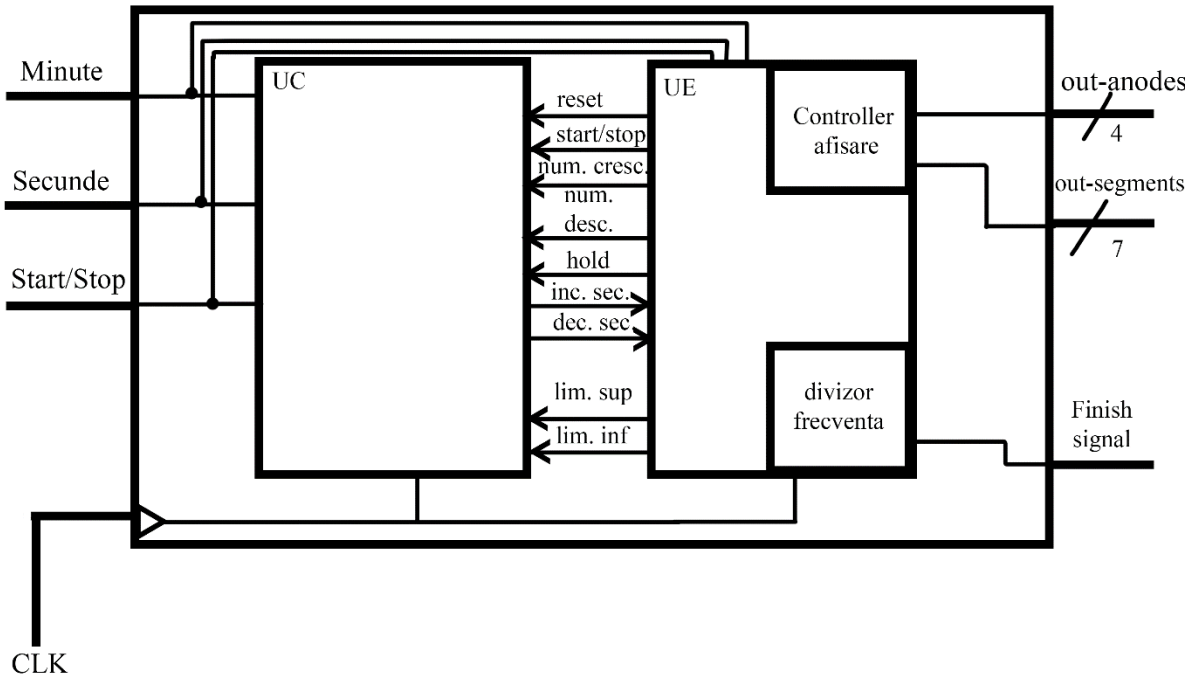
În vederea realizări unui timer ce poate număra crescător/descrescător în formatul mm:ss, vom folosi:

- 3 butoane (Buton incrementare minute, buton incrementare secunde, buton start/stop)
- 4 afișoare BCD cu 7 segmente
- 1 timer, care are o frecvență de 1 Hz(=> perioadă de o secundă, T=1)
- Un dispozitiv care produce un semnal sonor la expirarea timpului

2.1.2. Black box



2.2. Descompunerea în UC și UE



2.2.1. Lista completă a resurselor

În vederea realizării proiectului, am folosit următoarele resurse:

- 3 numărătoare modulo p, care numără in bucla 0-9 (de la 0000 la 1001)
- 1 numărător care numără în bucla 0-5 (de la 0000 la 0101)
- 4 decodificatoare BCD to 7-segement (se primește un cod pe 4 biți, iar output-ul va fi pe 7 biți corespunzători segmentelor unui afișor)
- Divizor de frecvență
- Debouncer
- Modul de control al anozilor
- Bistabil T
- Bistabil JK
- Bistabil D (pentru memorarea stărilor)
- Multiplexoare
- Demultiplexor

2.3. Descrierea detaliată a resurselor

Multiplexoarele, demultiplexoarele, bistabilele T, D și JK sunt folosite în mod normal, fără nicio schimbare în comportamentul lor de bază.

- ***NUMĂRĂTOARE***

Pentru realizarea numărătorului modulo p care numără în bucla 0-9, s-au folosit 4 bistabile JK, folosind algoritmul de realizare a numărătoarelor reversibile. De ce 4? Avem nevoie să reprezentăm 10 stări diferite, de la 0 la 9. Deci p = 10. După regulă știm că 2^n >= p, deci vom alege n = 4, care reprezintă numărul de bistabile.

Tabelul de tranziții(obținut din tabelul de excitație a bitabilului JK) folosit pentru sintetizarea logicii necesare fiecărui bistabil(ulterior):

DIR	Q_3 (t)	Q_2 (t)	Q_1 (t)	Q_0 (t)	Q_3 (t+1)	Q_2 (t+1)	Q_1 (t+1)	Q_0 (t+1)	J_3	K_3	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	1	0	0	1	1	X	0	X	0	X	1	X
1	0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
0	0	0	0	1	0	0	0	0	0	X	0	X	1	X	X	1
1	0	0	0	1	0	0	1	0	...	...	...	...	...	...	...	...

Tabelul cuprinde foarte multe cazuri care trebuie sa fie tratate. Pentru fiecare stare(o cifră de la 0 la 9) trebuie tratate două cazuri, în funcție de direcție, 0 dacă

direcția se numără descrescător, 1 dacă se numără crescător. De asemenea, trebuie luat în considerare starea trecută/viitoare pentru fiecare cifră.

Pentru a deduce logica combinațională necesară fiecărui bistabil, din tabelul de tranziții vom lua pentru fiecare pereche J_n și K_n, starea curentă $Q_3Q_2Q_1$ și valoarea J_n din dreptul stării curente. Analog și pentru K_n.

De exemplu, dorim să rezolvăm pentru J_3 și K_3 .Din tabelul de tranziții obținem:

J_3

$Q_{210}(t)$ / DIR $Q_3(t)$	000	001	011	010	110	111	101	100
00	1	0	0	0	0	0	0	0
01	X	X	X	X	X	X	X	X
11	X	X	X	X	X	X	X	X
10	0	0	0	0	0	1	0	0

Din tabel, aplicând metoda de rezolvare a digramei Karnaugh, obținem pentru J_3:

$J_3 = DIR' * Q_2' * Q_1' * Q_0' + DIR * Q_2 * Q_1 * Q_0$

Pentru K_3 se procedează la fel:

$Q_{210}(t)$ / DIR $Q_3(t)$	000	001	011	010	110	111	101	100
00	X	X	X	X	X	X	X	X
01	1	0	X	X	X	X	X	X
11	0	1	X	X	X	X	X	X
10	X	X	X	X	X	X	X	X

Astfel pentru K_3 avem:

$K_3 = DIR' * Q_0' + DIR * Q_0$

Pentru restul bistabilelor se procedează la fel ca mai sus, astfel fiind în total 8 diagrame Karnaugh de rezolvat. De menționat faptul că din cauza numărului de stări puține pe care le-am ales și numărul mare de stări posibile din diagrama Karnaugh, în toate tabelele datele relevante se vor afla în zona verde, zona roșie reprezentând din start stări în care nu vom ajunge niciodată.

$Q_{210}(t)$ / DIR $Q_3(t)$	000	001	011	010	110	111	101	100
00	-	-	-	-	-	-	-	-
01	-	-	X	X	X	X	X	X
11	-	-	X	X	X	X	X	X
10	-	-	-	-	-	-	-	-

Restul de ecuații pentru celelalte bistabile sunt:

- Bistabil 2:

  $J_2 = DIR' * Q_3 * Q_0' + DIR * Q_1 * Q_0$

  $K_2 = DIR' * Q_1' * Q_0' + DIR * Q_1 * Q_0$

- Bistabil 1:

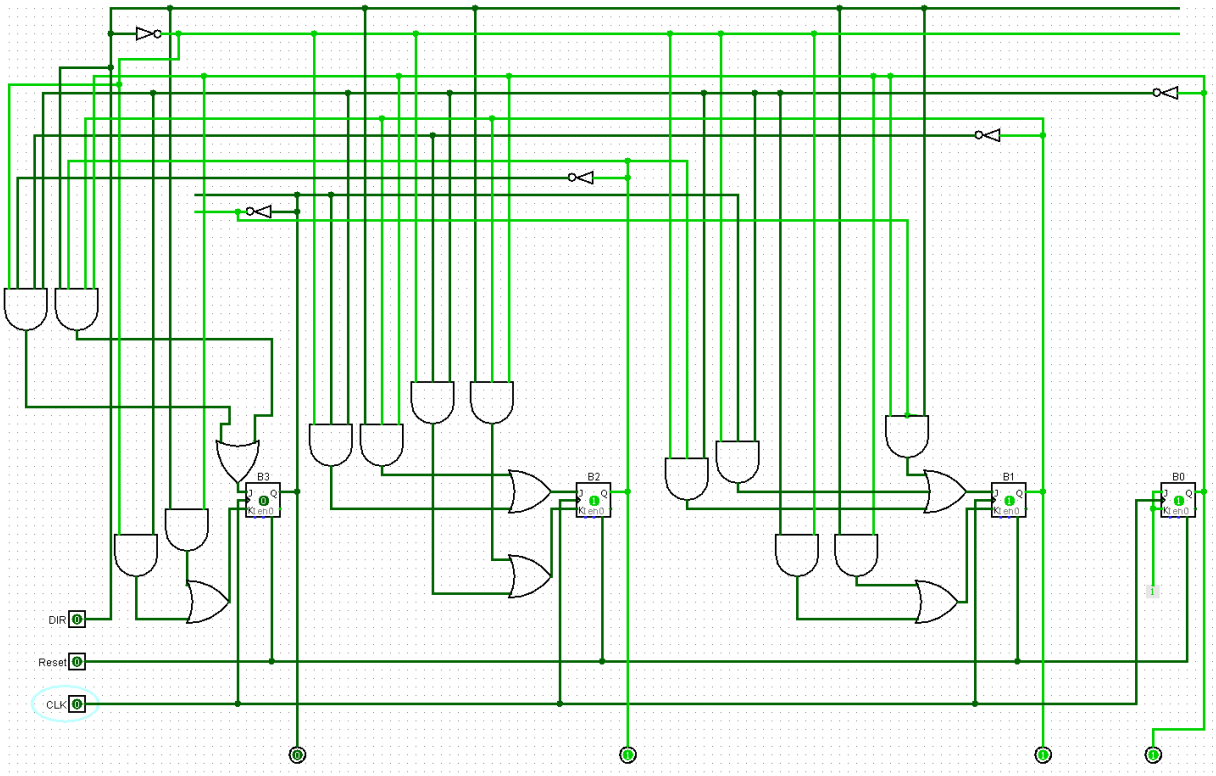
$J_1 = DIR' * Q_2 * Q_0' + DIR' * Q_3 * Q_0' + DIR * Q_3' * Q_0$

$K_1 = DIR' * Q_0' + DIR * Q_0$

- Bistabil 0:

$J_0 = 1$

$K_1 = 1$

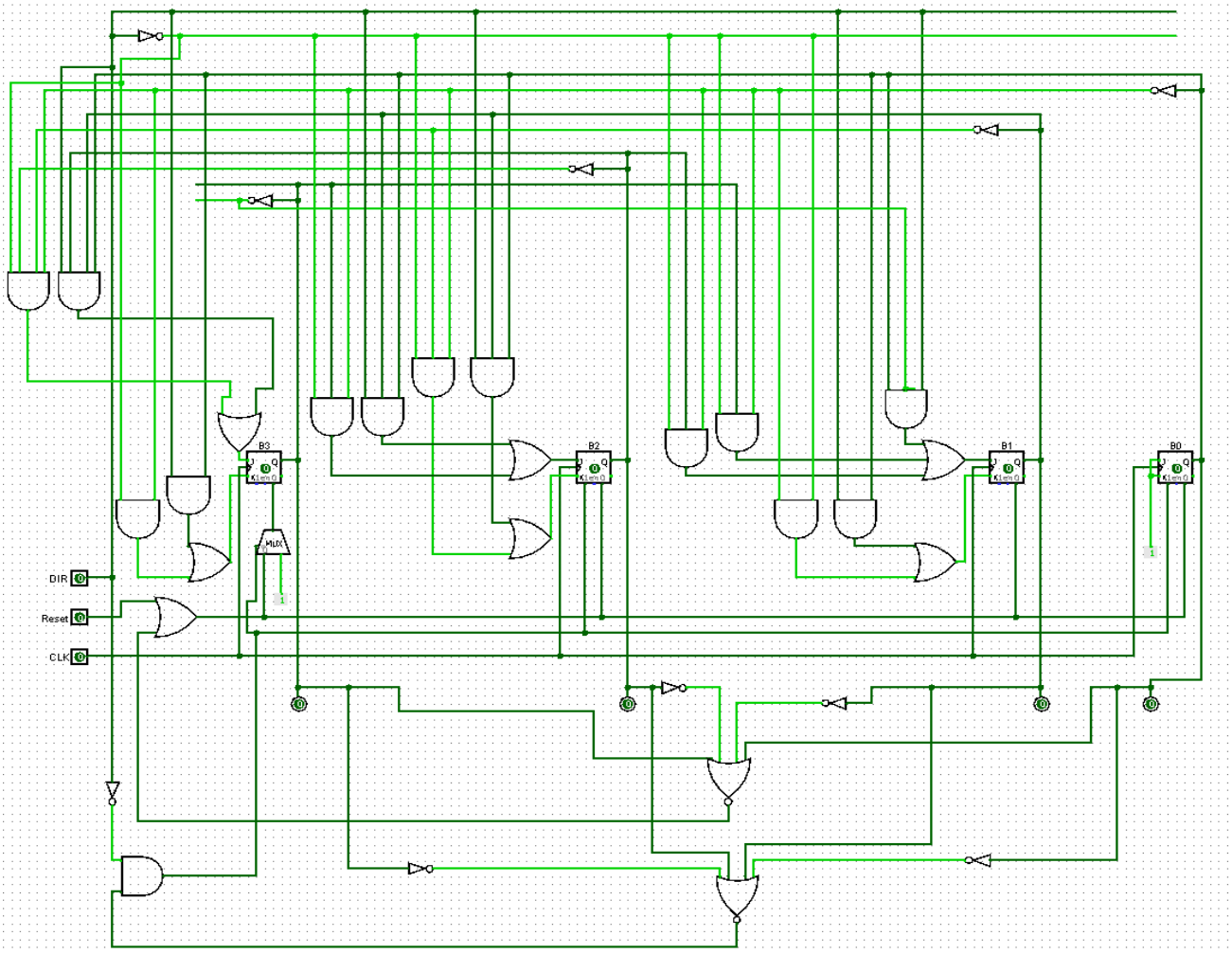


- schema logică a numărătorului 0-9 -

Numărătorul care numără în bucla 0-5 este realizat pe baza numărătorului anterior, dar ca să putem număra în bucla dorită, folosim logică combinațională adițională pentru a forța trecerea dintre o stare și la alta la capătul intervalului de numărare. De exemplu, ca sa nu trecem din starea 5 în 6(pt. numărătoare crescătoare), vom verifica când am ajuns în starea 6 și vom reseta forțat circuitul. Dacă numărăm descrescător, ca să trecem din stare 0 în 5 vom seta forțat bistabilele JK, astfel vom ajunge în starea dorită.

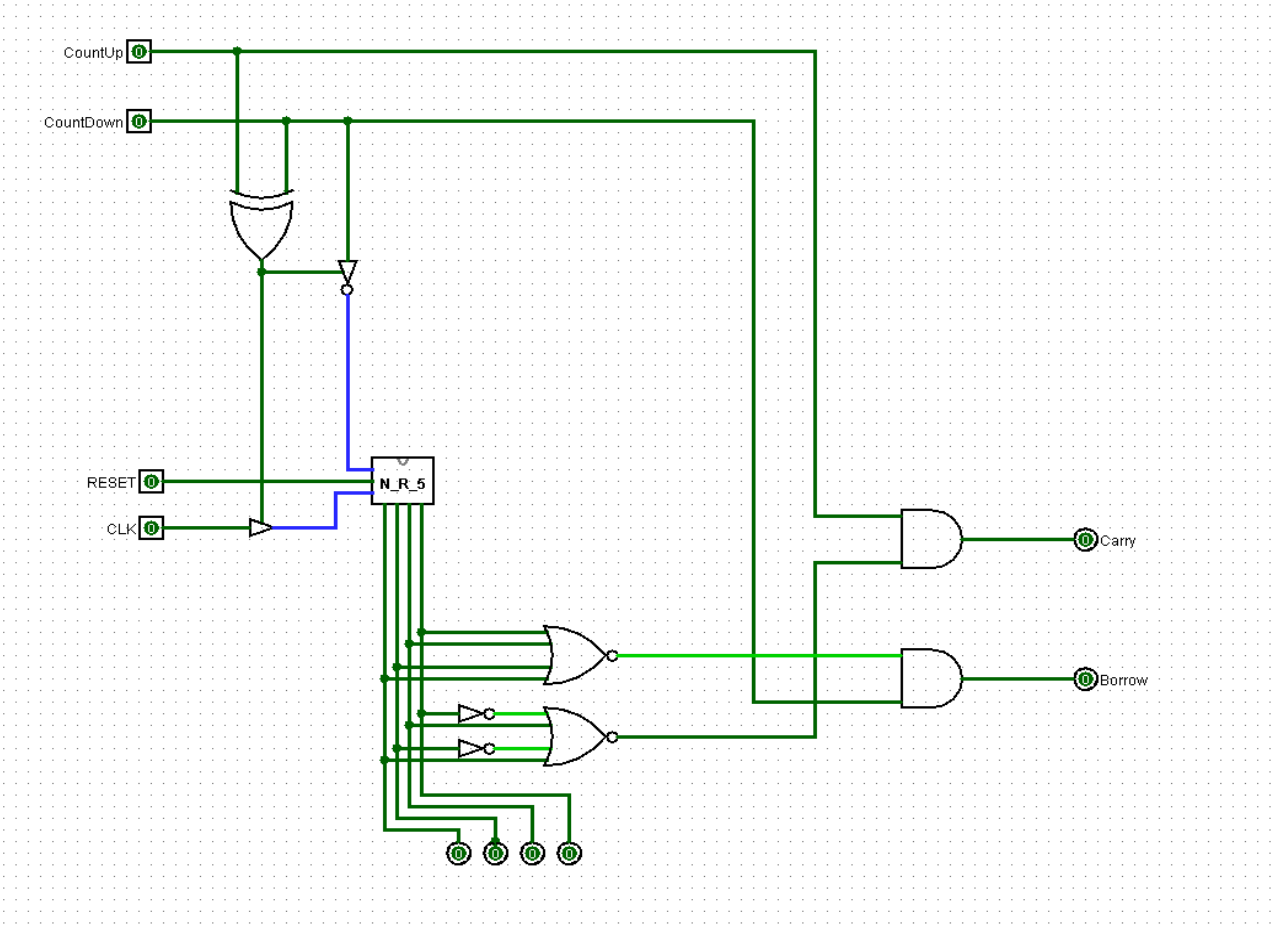
Pe scurt, facând aceste artificii, nu a mai fost nevoie de alt tabel de tranziții și după de sintetizat 8 diagrame Karnaugh.

Astfel, schema logică a numărătorului reversibil 0-9 este:



-se poate observa logica combinațională adițională în partea inferioară a circuitului-

Numărătoarele reversibile 0-9 și 0-5 au fost integrate ulterior într-un numărător mai avansat care are ca intrări adiționale CountUp/CountDown (pentru numărare crescătoare/descrecătoare, 1 bit) și ieșiri două semnale Carry/Borrow(pentru a semnaliza sensul numărări, 1 bit) și încă o ieșire pe 4 biți pentru cifre. Pentru ambele numărătoare există o variantă mai îmbunătățită. Astfel, schema logică pentru un numărător 0-5 avansat este:



-unde N_R_5 este numărătorul reversibil 0-5 prezentat mai sus-

Analog se procedează și pentru numărătorul avansat reversibil 0-9.

- ***DECODIFICATOR BCD to 7 SEG***

Ieșirile pe 4 biți din numărătoarele avansate sunt cele care intră în decodificatorul nostru, care primește un cod pe 4 biți și generează alt cod pe 7 biți corespunzător pentru fiecare segment care va fi iluminat în funcție de cifra dorită.

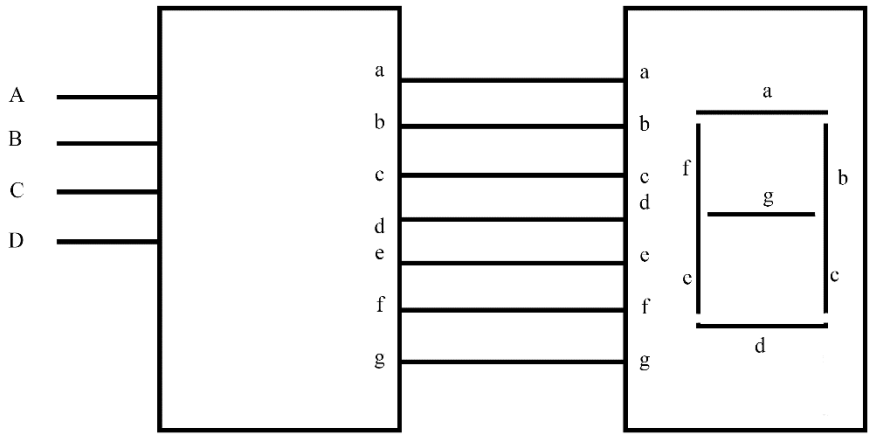
Într-un afișor 7 segment, fiecărui segment îi corespunde o literă de la a la g. Ca un segment să fie activ, vom aplica un semnal cu valoarea logică de ‘1’.

De ce 7 segmente pentru a reprezenta o cifră? Este numărul minim de segmente pentru a fi posibil reprezentarea cifrelor din intervalul 0-9. Evident, există afișoare și mai complexe cu mai multe segmente, unele fiind capabile sa afișeze și litere.

Tabelul de adevăr pentru decodificatorul nostru este:

cifra	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	1	1	1	1
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

Schema logică pentru cum ar funcționa decodificatorul și afișorul 7 bcd ar fi:



Din câte putem observa, pentru fiecare literă în parte avem atribuită o literă. Astfel, ecuația pentru fiecare literă se poate deduce din tabelul de adevăr de mai sus. De exemplu, pentru litera a, avem:

a:

AB \ CD	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

Din tabel obținem că:

a = A + C + B*D + B’ * D’

Analog se procedează pentru restul literelor b, c, d, e, f, g, rezolvând în total 7 diagrame Karanaugh. În final vom avea ecuațiile:

a = A + C + B*D + B’ * D’

b = B’ + C’ * D’ + C*D

c = B + C’ + D

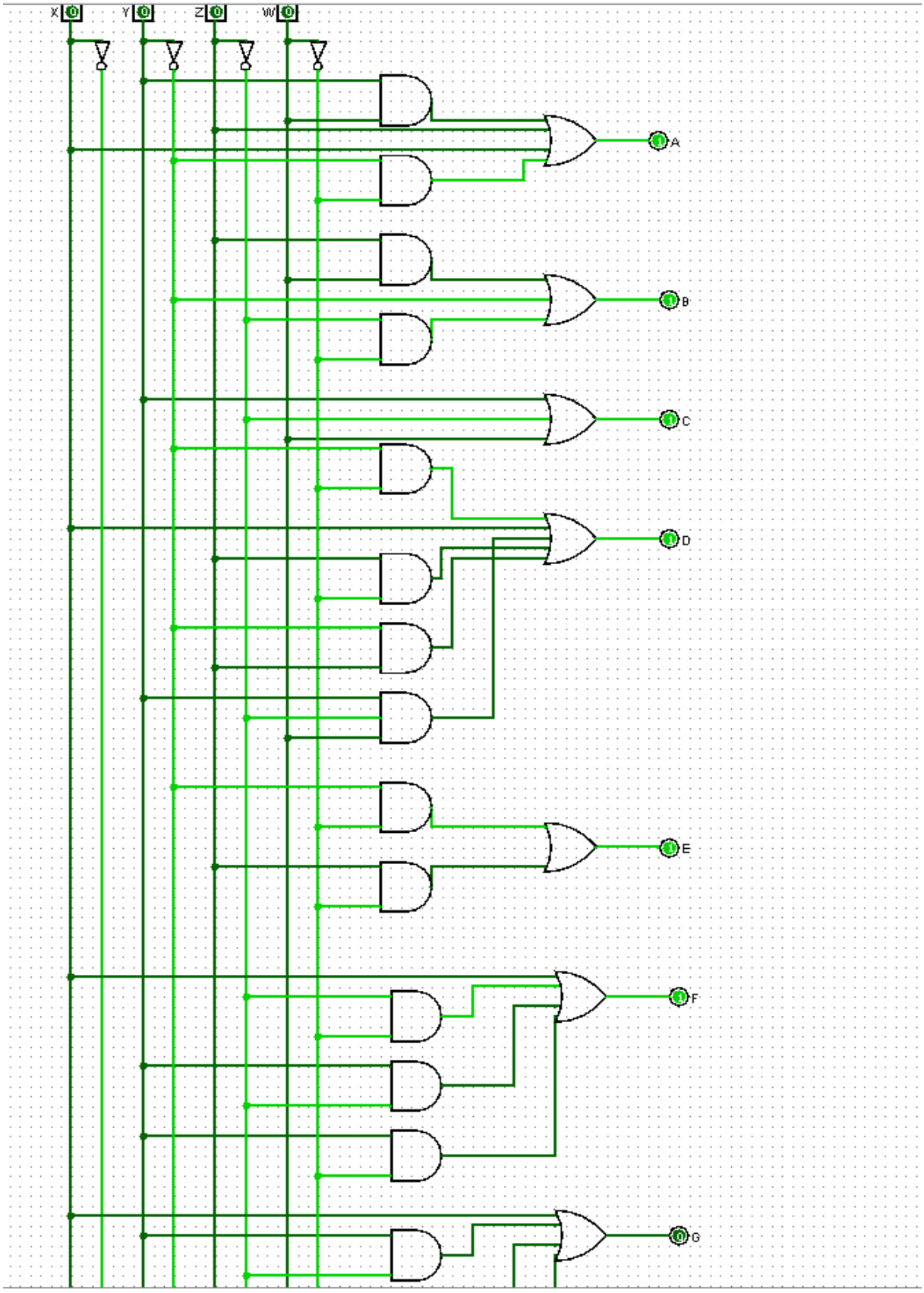
d = B’ * D’ + C * D’ + B * C’ * D + B’ * C +A

e = B’ * D’ + C * D’

f = A + C’ * D’ + B * C’ + B * D’

g = A + B * C’ + B’ * C + C * D’

Schema logică pentru decodificatorul nostru este:



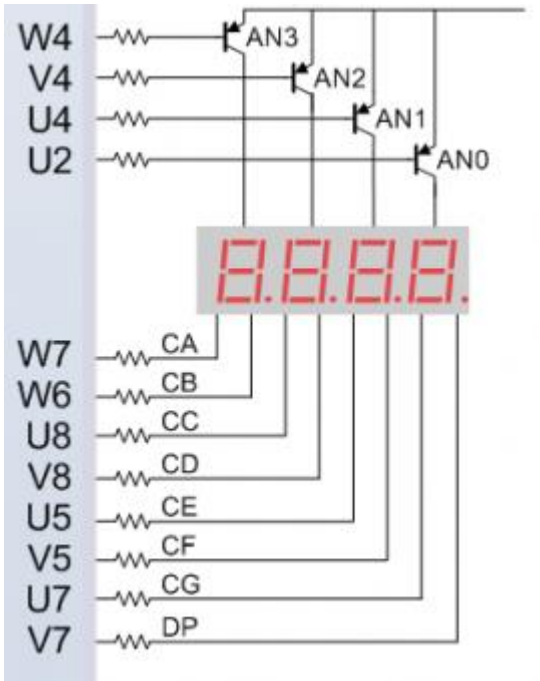
- ***Divizor de frecvență*** *(vivado)*

În practică, divizorul de frecvență se realizează cu ajutorul mai multor numărătoare conectate unul după altul.

Pe placa de dezvoltare FPGA Basys-3 regăsim un semnal de tact cu o frecvență de 100Mhz, mult prea mare pentru ceea ce avem noi nevoie să facem. Pentru numărare, avem nevoie de o frecvență de 1 Hz pentru a produce o perioadă de o secundă. Astfel, pentru a simula numărătoarele înlănțuite, vom alege un vector de 26 de biți ca să numere până la 100.000.000. Vectorul se va incrementa cu 1 la fiecare semnal de tact generat de ceasul intern al plăcuței. Când ajunge la 100.000.000, putem marca trecerea unei secunde.

- ***Modul de control al anozilor*** *(vivado)*

Trebuie luat în considerare faptul că nu putem afișa 4 cifre deodată, trebuie să afișăm pe rând pe fiecare afișor în parte, deoarece toți cei 4 afișori sunt conectați la același semnal pe 7 biți necesar afișării cifrelor. Mai jos avem o schemă în care se vede foarte clar cum sunt configurate cele 4 afișoarele pe 7 segmente:

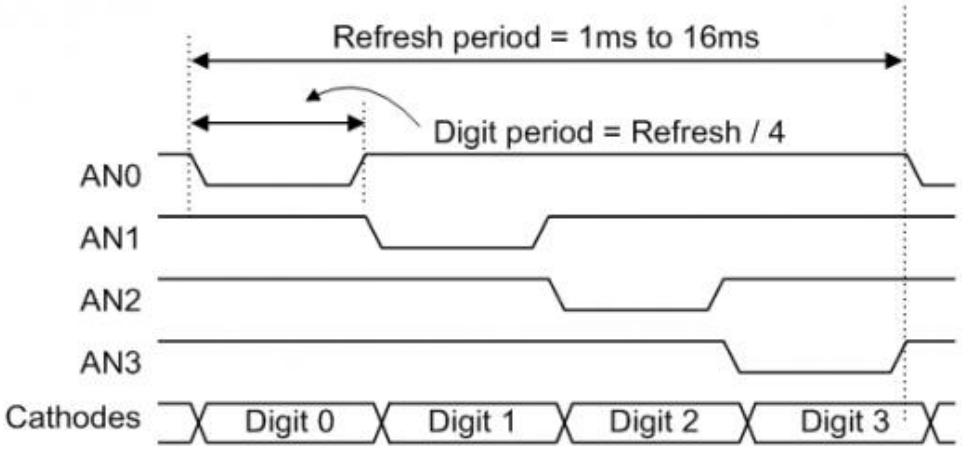


Vom controla cele 4 afișoare cu ajutorul anozilor de mai sus: W4, V4, U4, U2. De menționat faptul ca anozii funcționează în logică negativă.

Ca să afișăm pe 4 afișoare în același timp avem nevoie sa facem refresh la afișoare destul des. Practic, să creăm o iluzie optică, de care utilizatorul nu se poate prinde.

Avem nevoie de o perioadă de refresh între 1ns și 16ns, sau 1khz și 100hz.

Conform documentației Basys3, avem următorul graf:



Din graf observăm că perioada de refresh se împarte la 4, pentru a putea activa fiecare anod pe rând în parte.

Astfel am folosit un vector de 20 de biți pentru a crea o rată de refresh optimă și pentru a putea activa anozii pe rând. Numărăm în bucla 00...000 -> 00111..111, iar cei mai 2 semnificativi biți îi folosim pentru a genera un cod pe 4 biți de activare pentru fiecare anod în parte(ex: anod 0 = “0111”).

Când numărătoare ajunge în 00111..111, mai avem 4 combinații posibile:

00 111..111 -> anod 0

01 111..111 -> anod 1

10 111..111 -> anod 2

11 111..111 -> anod 3

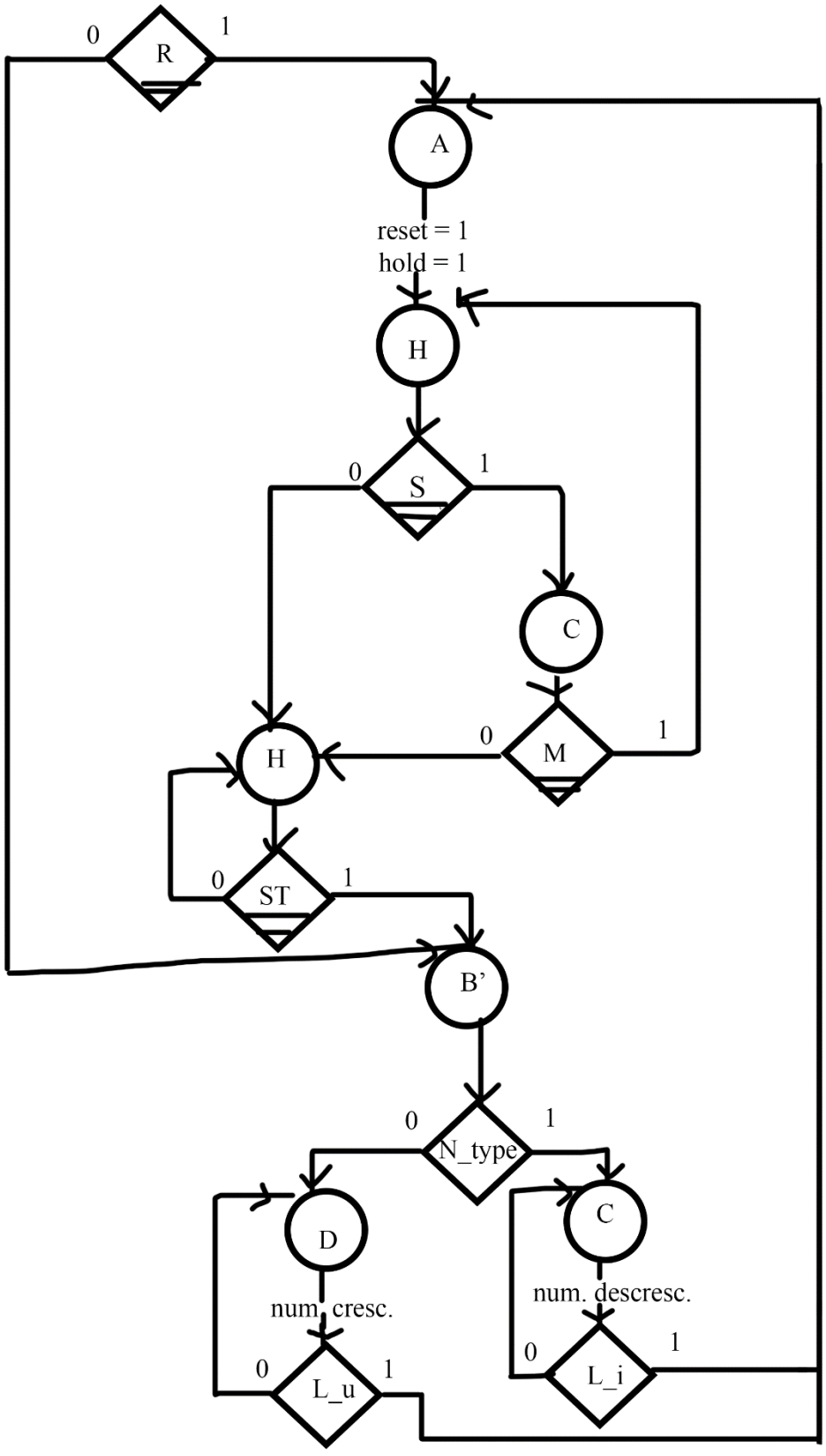
După se resetează numărul la 00..00 și o luăm de la capăt.

- ***Debouncer***

Se folosesc pentru a elimina orice sunet static la apăsarea unui buton.

În mod tradițional, un debouncer se realizează folosind bistabile/registrii sau chiar cu ajutorul numărătoarelor. În proiect a fost implementat cu ajutorul a 3 bistabile conectate înre ele, care se actualizează la o perioadă de sub 1 secundă.

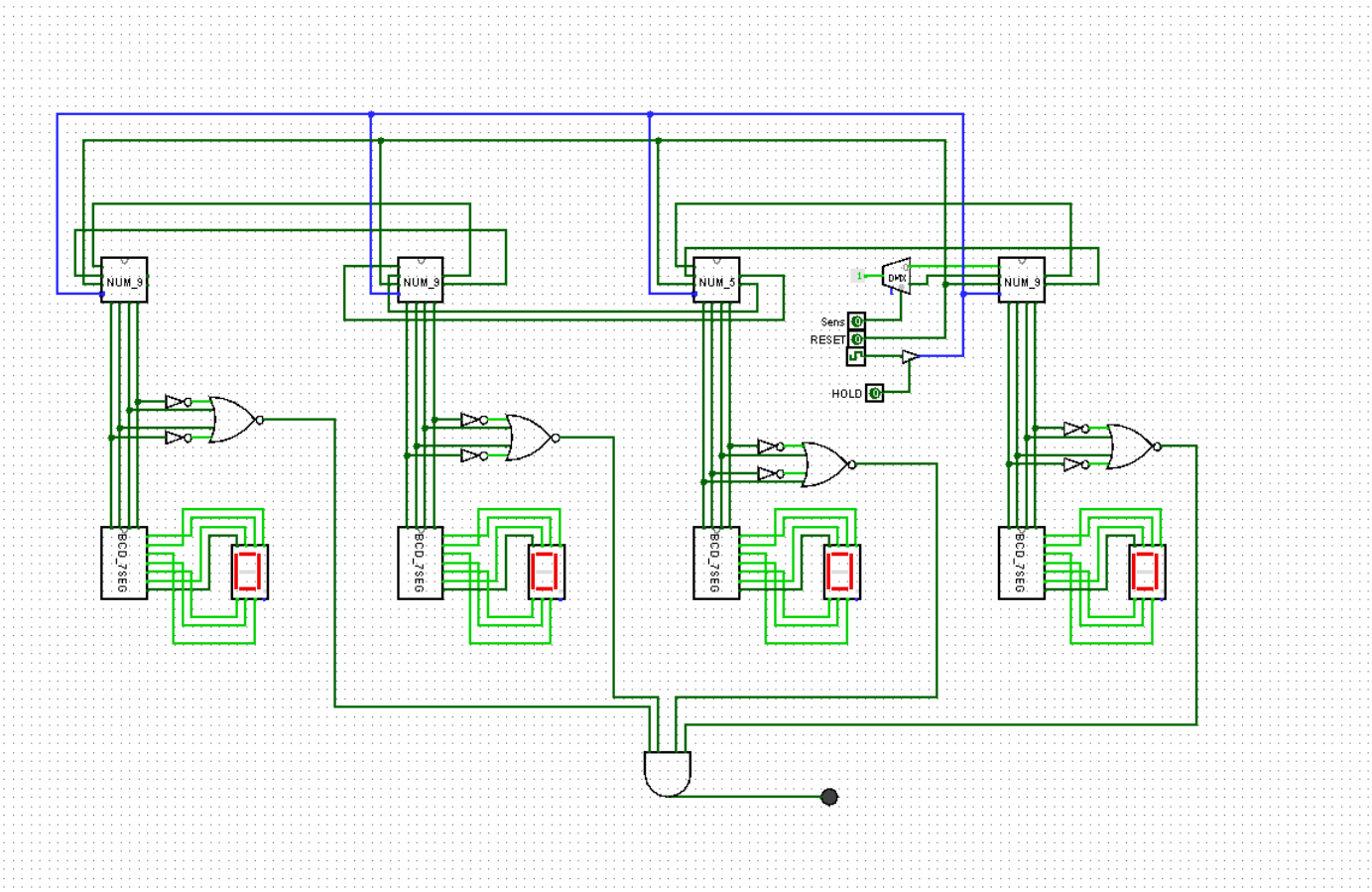
2.4. Organigrama UC



Unde avem:
R = reset
S = butonul pt. secunde
M = butonul pt. minute
ST = butonul pentru start/stop
N_type = tipul de numărare la momentul dat
L_u = s-a atins limita superioară
L_i = s-a atins limita inferioară

Stări:
A – Reset, inițializare
H – hold
C – se face hold și se incrementează doar când se apasă butoanele de minute și secunde
B – stare care semnifică ca în momentul de față se numără
C – numărarea este descrescătoare
D – numărătoarea este crescătoare

2.5. Schema de detaliu a UE:



3. Justificarea soluției alese

Am ales o metodă intuitivă de rezolvare a proiectului cu timer-ul care are două funcții. Din start am decis să folosesc un numărător pentru fiecare cifră astfel să-mi fie ușor să incrementez/decrementez fiecare cifră care urma să facă parte din afișare. Numărătoarele modulo p reversibile sunt foarte bun pentru acest lucru, deoarece, fiind implementate cu bistabile JK, ne permite să facem mici artificii, cum a fost și în cazul numărătorului reversibil 0-5.

Restul componentelor erau esențiale(decodificator, controller de afișor, divizor de frecvență etc.) și nu prea prezentau altă alternativă.

Tot proiectul gravitează în jurul modului cum alegem sa facem numărarea. O altă soluție interesantă ar fi fost să alegem un vector de 16 biți, să-l împărțim în bucăți de câte 4 biți corespunzătoare pentru fiecare cifră din afișare.

4. Instrucţiuni de utilizare şi întreţinere

Timer-ul prezintă mai multe regimuri de funcționare:

a) Numărare crescătoare până la limita de 99:59
b) Numărare descrescătoare până la limita de 00:00, iar utilizatorul va fi avertizat
c) Start/Stop
d) Reset (automat cât și manual)

Practic, timer-ul nostru este de fapt un cronometru și un temporizator într-un singur device. Vrem să cronometrăm cât timp durează sa facem un exercițiu la mate? Apăsăm butonul Start/Stop, timer-ul începe să numere crescător, iar dacă mai apăsăm încă o dată, de oprește. Vrem sa facem ceva cât timp am lăsat la cuptor ceva mâncare? Incrementăm minutele și secundele în funcție de preferință, iar la expirarea timpului vom fi notificați. De asemenea, în orice moment putem reseta timer-ul. Este dispozitivul perfect de uz casnic pentru a cronometra activitățile de acasă și nu numai.

Atenție! Dispozitivul nu este certificat din punct de vedere al rezistenței la apă și la praf, se recomandă o folosire normală a timer-ului, ținându-l cât mai departe de lichide și praf ce poate avaria circuitele. De asemenea nu e prea rezistent la șocuri mari, se recomandă manevrarea timer-ului cu grijă mare.

De menționat și foarte important este ca timer-ul funcționează doar dacă este conectat la o “priză”. Nu merge pe baterii

5. Posibilităţi de dezvoltare ulterioare

Timer-ul poate fi îmbunătățit astfel:

a) funcția de afișare a secundelor (cu ajutorul ledurilor) iar pe afișoare vom afișa doar minutele și orele
b) circuite de debounce îmbunătățite
c) funcția de memorare a unor valori de bază de temporizator, iar la apăsarea unui buton să înceapă automat să numere descrescător(ex: preset să seteze ora automat la 25:00 și să numere descrescător)
d) Atenționare vizuală, nu doar auditivă la expirarea timpului
e) Integrarea lui într-un circuit mai compact

6. SURSE

https://en.wikipedia.org/wiki/Seven-segment_display

https://reference.digilentinc.com/programmable-logic/basys-3/reference-manual

https://www.geeksforgeeks.org