19AIE101-ELEMENTS OF COMPUTING SYSTEMS

BTECH CSE(AI) 2020-2024

AMRITA VISWA VIDHYAPEETHAM, COIMBATORE

DATE: - 25/02/2021

1. CB.EN. U4AIE20075 -M VISWESWARAN
2. CB.EN. U4AIE20074 -VISHNU RADHAKRISHNAN
3. CB.EN. U4AIE20072 -THUSHIT KUMAR R
4. CB.EN. U4AIE20064 - R SAIVARSHA
5. CB.EN. U4AIE20040 – MENTA SAI AKSHAY

ACKNOWLEDGEMENT: We would like to express our special gratitude to our teacher Dr. Jyothish lal.G for his able guidance and support in completing our project.

AIM: To Design and Implement A digital clock using D-FFs which can show time up to one minute.

INTRODUCTION: To simulate and obtain the output of a clock in Hack platform and circuitverse.

ABSTRACT: We will be using DFF's, counter ,7 segment display. The counter is designed using DFF and the input pins as the expression obtained out of the KMaps.

In this project we will be designing a circuit capable of displaying the number of seconds (from 0 to 59) , minutes (from 0 to 59).

We Know That, 60 SECONDS = 1 MINUTES.

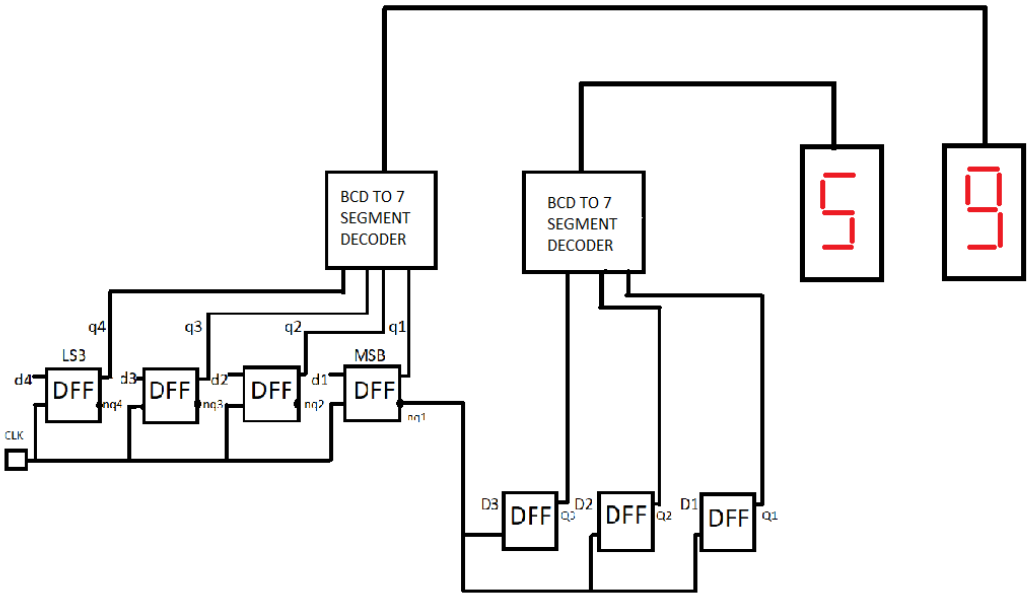
60 MINUTES = 1 HOURS.

To design a second's counter that displays up to 59 sec we use a combination of mod-10, mod-6 counters that will act as a mod-60 counter.

This can be achieved by connecting the negation of the output of the MSB of mod-10 counter to the clock input of mod-6 counter.

The same design of the mod-10, mod-6 counter is used to calculate the ones, tens digit of minute respectively that displays up to 59 min.

DESIGN:



- The minute mod-10 counter is clocked by the negation of the MSB of the mod-6 counter of the seconds. The mod-6 counter of the minute is clocked by the negation of the MSB of minute mod-10 counter.
- The mod-10 counter represents the ones place of the seconds/minute whereas the mod-6 counter represents the tens place.
- Seconds and minutes expressed in binary coded decimal (BCD) where each of the digits (units and tens digit) of the quantities are expressed in binary form.
- The output of each of the counters is fed into a **BCD to 7 segment display decoder,** the output of which is utilised to display the seconds and minutes.
- The clock is designed from 14 synchronous d flip flops. D flip flops are used to change the values of the output of each bit of the BCD code.

The following is the naming convention used in our design

Seconds Counter

MOD-10

DFF1: in: D1, out: sec10[3], out: Q1 [MSB]

DFF2: in: D2, out: sec10[2], out: Q2

DFF3: in: D3, out: sec10[1], out: Q3

DFF4: in: D4, out: sec10[0], out: Q4 [LSB]

MOD-6

DFF5: in: d1, out: sec1[2], out: q1[MSB]

DFF6: in: d2, out: sec1[1], out: q2

DFF7: in: d3, out: sec1[0], out: q3[LSB]

MINUTE

MOD-10

DFF8: in: mD1, out: min10[3], out: mQ1 [MSB]

DFF9: in: mD2, out: min10[2], out: mQ2

DFF10: in: mD3, out: min10[1], out: mQ3

DFF11: in: mD4, out: min10[0], out: mQ4 [LSB]

MOD-6

DFF12: in: md1, out: min1[2], out: mq1[MSB]

DFF13: in: md2, out: min1[1], out: mq2

DFF14: in: md3, out: min1[0], out: mq3[LSB]

The tens digit of the seconds and minutes should be changed when the units digit goes from 9 to 0 in this case the most significant bit of the unit digit goes from 1 to 0 (1001 to 0000) hence we connect the negation of this bit to the clock of the mod-6 counter which gives the tens digit of seconds and minutes, where it is triggered when the most significant bit goes from 1 to 0 when this clock is triggered the DFF starts to work and hence gives the bit value of the numbers which has been represented in BCD form according to the equations obtained from the k-maps, this output is then fed into four 7 segment displays in order to get the four digits two each of seconds and two each of minutes.

We also know that the unit's digit of minute should be changed when the tens digit of the second goes from 5 to 0(101 to 000) hence we connect the negation of the most significant bit of the ten digit of the seconds to the clock of the units digit of minutes hence this clock is also triggered during the correct time and the output is calculated.

TRUTH TABLE AND K-Maps:

MOD-6 TRUTH TABLE FOR 10TH DIGIT OF SECONDS

q1	q2	q3	q1*	q2*	q3*	d1	d2	d3
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	0	0	0	0	0	0

MOD-10 TRUTH TABLE FOR ONES DIGIT OF SECONDS

Q1	Q2	Q3	Q4	Q1*	Q2*	Q3*	Q4*	D1	D2	D3	D4
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	0
0	0	1	0	0	0	1	1	0	0	1	1
0	0	1	1	0	1	0	0	0	1	0	0
0	1	0	0	0	1	0	1	0	1	0	1
0	1	0	1	0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	1	0	1	1	1
0	1	1	1	1	0	0	0	1	0	0	0
1	0	0	0	1	0	0	1	1	0	0	1
1	0	0	1	0	0	0	0	0	0	0	0

MOD-6 TRUTH TABLE FOR 10TH DIGIT OF MINUTE

mq1	mq2	mq3	mq1*	mq2*	mq3*	md1	md2	md3
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	0	0	0	0	0	0

MOD-10 TRUTH TABLE FOR ONES DIGIT OF MINUTE

mQ1	mQ2	mQ3	mQ4	mQ1*	mQ2*	mQ3*	mQ4*	mD1	mD2	mD3	mD4
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	0
0	0	1	0	0	0	1	1	0	0	1	1
0	0	1	1	0	1	0	0	0	1	0	0
0	1	0	0	0	1	0	1	0	1	0	1
0	1	0	1	0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	1	0	1	1	1
0	1	1	1	1	0	0	0	1	0	0	0
1	0	0	0	1	0	0	1	1	0	0	1
1	0	0	1	0	0	0	0	0	0	0	0

TRUTH TABLE FOR BCD TO 7 SEGMENT

D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

K-Maps FOR MOD-6

		Q2' Q3'	Q2' Q3	Q2 Q3	Q2 Q3'
	D1	0 0	0 1	1 1	1 0
Q1'	0	0	0	1	0
Q1	1	1	0	0	0

GROUPS

[3]	Q1'Q2Q3
[4]	Q1Q2'Q3'

D1=Q1'Q2Q3 + Q1Q2'Q3'

		Q2' Q3'	Q2' Q3	Q2 Q3	Q2 Q3'
	D2	0 0	0 1	1 1	1 0
Q1'	0	0	1	0	1
Q1	1	0	0	0	0

GROUPS

[1]	Q1'Q2'Q3
[2]	Q1'Q2'Q3'

D2=Q1'Q2'Q3 + Q1'Q2Q3'

		Q2' Q3'	Q2' Q3	Q2 Q3	Q2 Q3'
	D2	0 0	0 1	1 1	1 0
Q1'	0	0	1	0	1
Q1	1	0	0	0	0

GROUPS

[1]	Q1'Q2'Q3
[2]	Q1'Q2'Q3'

D2=Q1'Q2'Q3 + Q1'Q2Q3'

		Q2' Q3'	Q2' Q3	Q2 Q3	Q2 Q3'
	D3	0 0	0 1	1 1	1 0
Q1'	0	1	0	0	1
Q1	1	1	0	0	0

GROUPS

[0,2]	Q1'Q3'
[0,4]	Q2'Q3'

D3=Q1'Q3' + Q2'Q3'

KMaps for MOD-10

		Q3' Q4'	Q3' Q4	Q3 Q4	Q3 Q4'
	d1	0 0	0 1	1 1	1 0
Q1' Q2'	0 0	0	0	0	0
Q1' Q2	0 1	0	0	1	0
Q1 Q2	1 1	0	0	0	0
Q1 Q2'	1 0	1	0	0	0

GROUPS

[7]	Q1'Q2Q3Q4
[8]	Q1Q2'Q3'Q4'

d1=Q1'Q2Q3Q4 + Q1Q2'Q3'Q4'

		Q3' Q4'	Q3' Q4	Q3 Q4	Q3 Q4'
	d2	0 0	0 1	1 1	1 0
Q1' Q2'	0 0	0	0	1	0
Q1' Q2	0 1	1	1	0	1
Q1 Q2	1 1	0	0	0	0
Q1 Q2'	1 0	0	0	0	0

GROUPS

[4,5]	Q1'Q2Q3
[3]	Q1'Q2'Q3Q4
[4,6]	Q1'Q2Q4'

d2=Q1'Q2Q3' + Q1'Q2Q4' + Q1'Q2'Q3Q4

		Q3' Q4'	Q3' Q4	Q3 Q4	Q3 Q4'
	d3	0 0	0 1	1 1	1 0
Q1' Q2'	0 0	0	1	0	1
Q1' Q2	0 1	0	1	0	1
Q1 Q2	1 1	0	0	0	0
Q1 Q2'	1 0	0	0	0	0

GROUPS

[1,5]	Q1'Q3'Q4
[2,6]	Q1'Q3Q4'

d3=Q1'Q3'Q4 + Q1'Q3Q4'

		Q3' Q4'	Q3' Q4	Q3 Q4	Q3 Q4'
	d4	0 0	0 1	1 1	1 0
Q1' Q2'	0 0	1	0	0	1
Q1' Q2	0 1	1	0	0	1
Q1 Q2	1 1	0	0	0	0
Q1 Q2'	1 0	1	0	0	0

GROUPS

[0,2,4,6]	Q1'Q4'
[8,0]	Q2'Q3'Q4'

d4=Q1'Q4' + Q2'Q3'Q4'

K-Maps FOR BCD TO 7 SEGMENT

		B'A'	B'A	BA	BA'
	a	0 0	0 1	1 1	1 0
D'C'	0 0	1	0	1	1
D'C	0 1	0	1	1	1
DC	1 1	0	0	0	0
DC'	1 0	1	1	0	0

Groups

(2,3,6,7)	**D'B**
(0,2)	**D'C'A'**
(5,7)	**D'CA**
(8,9)	**DC'B'**

a= D'B + D'C'A' + D'CA + DC'B'

		B'A'	B'A	BA	BA'
	b	0 0	0 1	1 1	1 0
D'B'	0 0	1	1	1	1
D'C	0 1	1	0	1	0
DC	1 1	0	0	0	0
DC'	1 0	1	1	0	0

Groups

(0,1,2,3)	**D'C'**
(0,1,8,9)	**C'B'**
(0,4)	**D'B'A'**
(3,7)	**D'BA**

b = D'C' + C'B' + D'B'A' + D'BA

		B'A'	B'A	BA	BA'
	c	0 0	0 1	1 1	1 0
D'B'	0 0	1	1	1	0
D'C	0 1	1	1	1	1
DC	1 1	0	0	0	0
DC'	1 0	1	1	0	0

Groups

(0,1,8,9)	**C'B'**
(1,3,5,7)	**D'A**
(4,5,6,7)	**D'C**

c= C'B' + D'A + D'C

		B'A'	B'A	BA	BA'
	d	0 0	0 1	1 1	1 0
D'C'	0 0	1	0	1	1
D'C	0 1	0	1	0	1
DC	1 1	0	0	0	0
DC'	1 0	1	1	0	0

Groups

(0,2)	**D'C'A'**
(2,3)	**D'CB**
(2,6)	**D'BA'**
(8,9)	**DC'B'**
[5]	**D'CB'A**

d= D'C'A' + D'C'B + D'BA' + DC'B' + D'CB'A

		B'A'	B'A	BA	BA'
	e	0 0	0 1	1 1	1 0
D'C'	0 0	1	0	0	1
D'C	0 1	0	0	0	1
DC	1 1	0	0	0	0
DC'	1 0	1	0	0	0

Groups

(0,8)	**C'B'A'**
(2,6)	**D'BA'**

e = C'B'A' + D'BA'

		B'A'	B'A	BA	BA'
	f	0 0	0 1	1 1	1 0
D'C'	0 0	1	0	0	0
D'C	0 1	1	1	0	1
DC	1 1	0	0	0	0
DC'	1 0	1	1	0	0

Groups

(0,4)	**D'B'A'**
(4,5)	**D'CB'**
(4,6)	**D'CA'**
(8,9)	**DC'B'**

f= D'B'A' + D'CB' + D'CA' + DC'B'

		B'A'	B'A	BA	BA'
	g	0 0	0 1	1 1	1 0
D'C'	0 0	0	0	1	1
D'C	0 1	1	1	0	1
DC	1 1	0	0	0	0
DC'	1 0	1	1	0	0

Groups

(2,3)	**D'C'B**
(2,6)	**D'BA'**
(4,5)	**D'CB'**
(8,9)	**DC'B'**

g = D'C'B + D'BA' + D'CB' + DC'B'

IMPLEMENTATION AND CODE:

HDL code in nand2tetris software

```
CHIP ClkGrp8
{
OUT sec1[3],sec10[4],min10[4],min1[3],as,bs,cs,ds,es,fs,gs;

PARTS:
//MOD10-SECONDS
Not(in=Q1,out=NQ1);
Not(in=Q2,out=NQ2);
Not(in=Q3,out=NQ3);
Not(in=Q4,out=NQ4);

//D1
And4(a=NQ1,b=Q2,c=Q3,d=Q4,out=t1);
```

```
And4(a=Q1,b=NQ2,c=NQ3,d=NQ4,out=t2);
Or(a=t1,b=t2,out=D1);

//D2
And3(a=NQ1,b=Q2,c=NQ3,out=t3);
And3(a=NQ1,b=Q2,c=NQ4,out=t4);
And4(a=NQ1,b=NQ2,c=Q3,d=Q4,out=t5);
Or3(a=t3,b=t4,c=t5,out=D2);

//D3
And3(a=NQ1,b=NQ3,c=Q4,out=t6);
And3(a=NQ1,b=Q3,c=NQ4,out=t7);
Or(a=t6,b=t7,out=D3);

//D4
And(a=NQ1,b=NQ4,out=t8);
And3(a=NQ2,b=NQ3,c=NQ4,out=t9);
Or(a=t8,b=t9,out=D4);

DFF(in=D1,out=sec10[3],out=Q1);
DFF(in=D2,out=sec10[2],out=Q2);
DFF(in=D3,out=sec10[1],out=Q3);
DFF(in=D4,out=sec10[0],out=Q4);

SevenSegmentDecoder(A=Q4,B=Q3,C=Q2,D=Q1,a=as,b=bs,c=cs,d=ds,e=es
,f=fs,g=gs);
//MOD-6 SECONDS
//Not(in=q1,out=nq1);
```

```
//Not(in=q2,out=nq2);
//Not(in=q3,out=nq3);

//d1
//And3(a=nq1,b=q2,c=q3,out=tt1);
//And3(a=q1,b=nq2,c=nq3,out=tt2);
//Or(a=t1,b=tt2,out=d1);

//d2
//And3(a=nq1,b=nq2,c=q3,out=tt3);
//And3(a=nq1,b=q2,c=nq3,out=tt4);
//Or(a=tt3,b=tt4,out=d2);

//d3
//And(a=nq1,b=nq3,out=tt5);
//And(a=nq2,b=nq3,out=tt6);
//Or(a=tt5,b=tt6,out=d3);

//DFfmy(D=d1,clk=NQ4,Q=sec1[2],Q=q3);
//DFfmy(D=d2,clk=NQ4,Q=sec1[1],Q=q2);
//DFfmy(D=d3,clk=NQ4,Q=sec1[0],Q=q1);
//SevenSegmentDecoder(A=q3,B=q2,C=q1,D=false,a=as1,b=bs1,c=cs1,d=
ds1,e=es1,f=fs1,g=gs1);



//Min
//MOD-10 MINUTE
```

```
//Not(in=mQ1,out=mNQ1);
//Not(in=mQ2,out=mNQ2);
//Not(in=mQ3,out=mNQ3);
//Not(in=mQ4,out=mNQ4);

//mD1
//And4(a=mNQ1,b=mQ2,c=mQ3,d=mQ4,out=mt1);
//And4(a=mQ1,b=mNQ2,c=mNQ3,d=mNQ4,out=mt2);
//Or(a=mt1,b=mt2,out=mD1);

//mD2
//And3(a=mNQ1,b=mQ2,c=mNQ3,out=mt3);
//And3(a=mNQ1,b=mQ2,c=mNQ4,out=mt4);
//And4(a=mNQ1,b=mNQ2,c=mQ3,d=mQ4,out=mt5);
//Or3(a=mt3,b=mt4,c=mt5,out=mD2);

//mD3
//And3(a=mNQ1,b=mNQ3,c=mQ4,out=mt6);
//And3(a=mNQ1,b=mQ3,c=mNQ4,out=mt7);
//Or(a=mt6,b=mt7,out=mD3);

//mD4
//And(a=mNQ1,b=mNQ4,out=mt8);
//And3(a=mNQ2,b=mNQ3,c=mNQ4,out=mt9);
//Or(a=mt8,b=mt9,out=mD4);

//DFfmy(D=mD1,clk=mNq3,Q=min10[3],Q=mQ4);
//DFfmy(D=mD2,clk=mNq3,Q=min10[2],Q=mQ3);
```

```
//DFfmy(D=mD3,clk=mNq3,Q=min10[1],Q=mQ2);
//DFfmy(D=mD4,clk=mNq3,Q=min10[0],Q=mQ1);

//SevenSegmentDecoder(A=mQ4,B=mQ3,C=mQ2,D=mQ1,a=am,b=bm,c=cm,d=dm,e=em,f=fm,g=gm);

//MOD-6 MINUTE
//Not(in=mq1,out=mnq1);
//Not(in=mq2,out=mnq2);
//Not(in=mq3,out=mnq3);

//md1
//And3(a=mnq1,b=mq2,c=mq3,out=mtt1);
//And3(a=mq1,b=mnq2,c=mnq3,out=mtt2);
//Or(a=mtt1,b=mtt2,out=md1);

//d2
//And3(a=mnq1,b=mnq2,c=mq3,out=mtt3);
//And3(a=mnq1,b=mq2,c=mnq3,out=mtt4);
//Or(a=mtt3,b=mtt4,out=md2);

//d3
//And(a=mnq1,b=mnq3,out=mtt5);
//And(a=mnq2,b=mnq3,out=mtt6);
//Or(a=mtt5,b=mtt6,out=md3);

//DFfmy(D=md1,clk=mNQ4,Q=min1[2],Q=mq3);
//DFfmy(D=md2,clk=mNQ4,Q=min1[1],Q=mq2);
```

```
//DFfmy(D=md3,clk=mNQ4,Q=min1[0],Q=mq1);


//SevenSegmentDecoder(A=mq3,B=mq2,C=mq1,D=false,a=am1,b=bm1,c
=cm1,d=dm1,e=em1,f=fm1,g=gm1);
}
```

Note: some statements have been commented such that chips loads in hdl.

SUB CIRCUIT:DFfmy

```
 CHIP DFfmy{
    IN D,clk;
    OUT Q,Qb;


    PARTS:
// Put your code here:
Nand(a=D,b=b1,out=a,out=a1,out=a2);
Nand3(a=a1,b=clk,c=c3,out=b,out=b1,out=b2);
Nand(a=d1,b=clk,out=c,out=c1,out=c2,out=c3);
Nand(a=a2,b=c1,out=d,out=d1);
Nand(a=c2,b=qb,out=Q,out=q);
Nand(a=b2,b=q,out=Qb,out=qb);
}
```

Sub Circuit: SevenSegmentDecoder

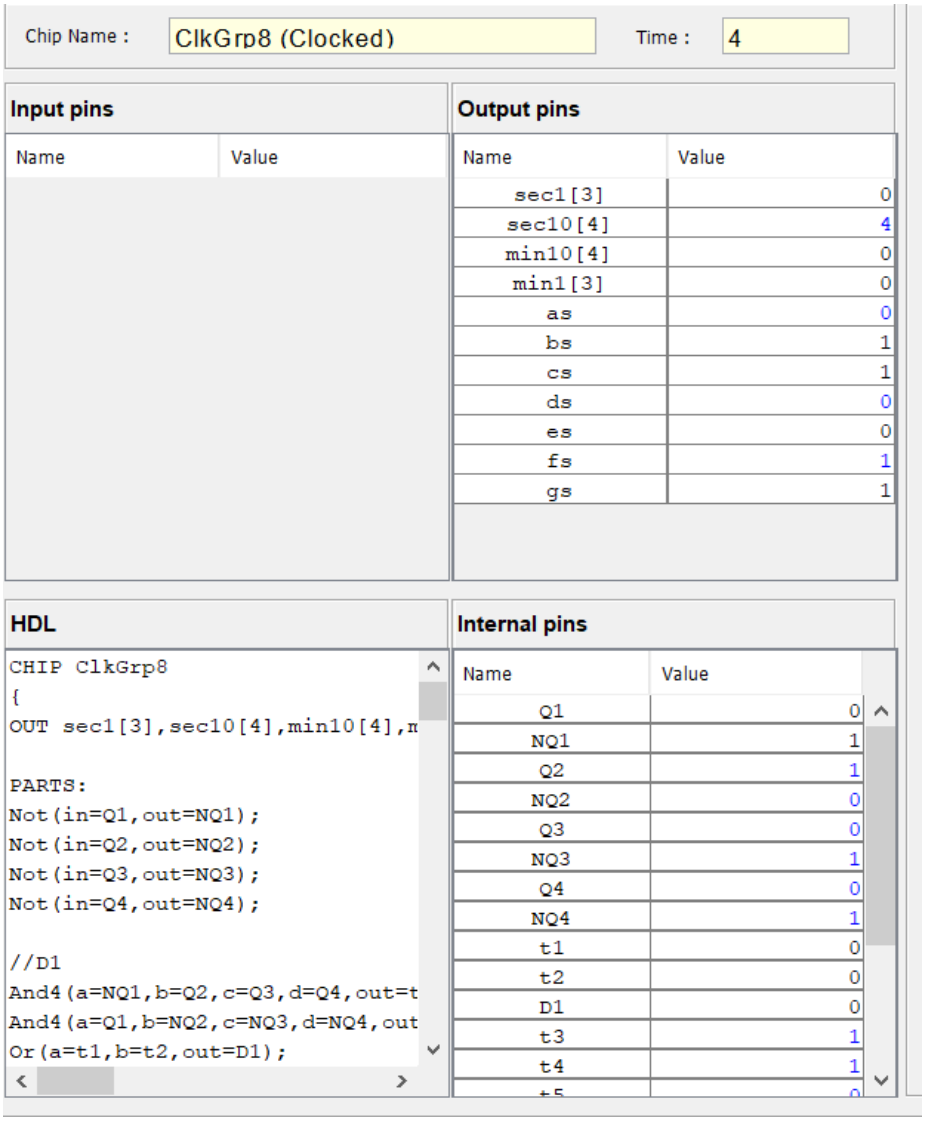
```
CHIP SevenSegmentDecoder{
    IN A,B,C,D;
    OUT a,b,c,d,e,f,g;
```

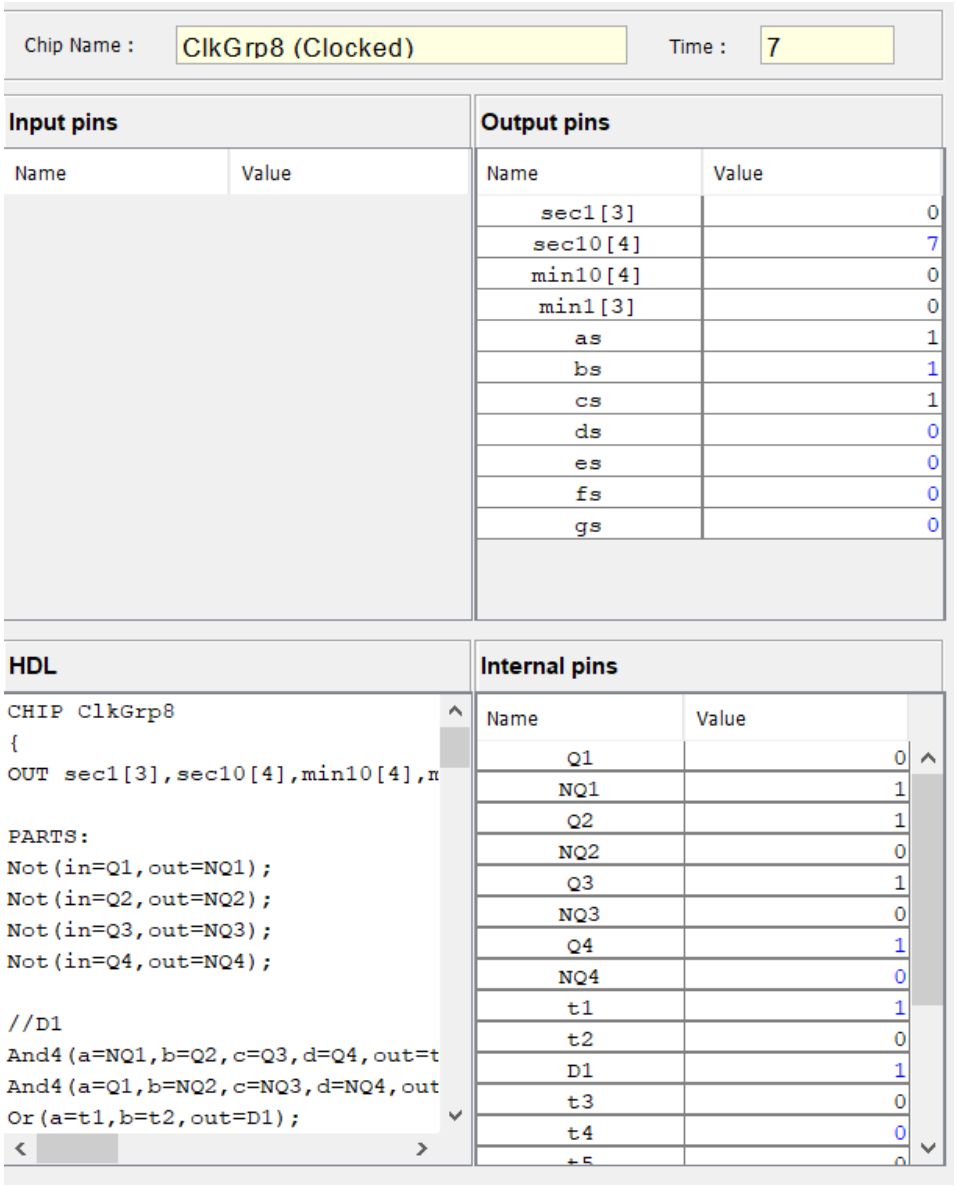
PARTS:

// Put your code here:

```
Not(in=A,out=notA);
Not(in=B,out=notB);
Not(in=C,out=notC);
Not(in=D,out=notD);

And(a=C,b=A,out=aand1);
And(a=notC,b=notA,out=aand2);
And(a=true,b=D,out=aand3);
And(a=true,b=B,out=aand4);
Or(a=aand1,b=aand2,out=aor1);
Or3(a=aand3,b=aand4,c=aor1,out=a);

And(a=B,b=A,out=band1);
And(a=notB,b=notA,out=band2);
Or3(a=band1,b=band2,c=notC,out=b);

Or3(a=C,b=notB,c=A,out=c);

And(a=true,b=D,out=dand1);
And(a=notC,b=notA,out=dand2);
And(a=B,b=notA,out=dand3);
And(a=notC,b=B,out=dand4);
And3(a=C,b=notB,c=A,out=dand5);
Or3(a=dand1,b=dand2,c=dand3,out=dor1);
Or(a=dand4,b=dand5,out=dor2)
```

```
    Or(a=dor1,b=dor2,out=d);

    And(a=B,b=notA,out=eand1);
    And(a=notC,b=notA,out=eand2);
    Or(a=eand1,b=eand2,out=e);

    And(a=notB,b=notA,out=fand1);
    And(a=C,b=notA,out=fand2);
    And(a=C,b=notB,out=fand3);
    And(a=true,b=D,out=fand4);
    Or3(a=fand1,b=fand2,c=fand3,out=for1);
    Or(a=for1,b=fand4,out=f);

    And(a=B,b=notA,out=gand1);
    And(a=C,b=notB,out=gand2);  And(a=true,b=D,out=gand3);
    And(a=notC,b=B,out=gand4);
    Or3(a=gand1,b=gand2,c=gand3,out=gor1);
    Or(a=gor1,b=gand4,out=g);

}
```


Chip Name :
ClkGrp8 (Clocked)
Time :
4
Input pins
Name
Value
Output pins
Name
Value
sec1[3] 0
sec10[4] 4
min10[4] 0
min1[3] 0
as 0
bs 1
cs 1
ds 0
es 0
fs 1
gs 1
HDL
CHIP ClkGrp8
{
OUT sec1[3],sec10[4],min10[4],m
PARTS:
Not(in=Q1,out=NQ1);
Not(in=Q2,out=NQ2);
Not(in=Q3,out=NQ3);
Not(in=Q4,out=NQ4);
//D1
And4(a=NQ1,b=Q2,c=Q3,d=Q4,out=t
And4(a=Q1,b=NQ2,c=NQ3,d=NQ4,out
Or(a=t1,b=t2,out=D1);
Internal pins
Name
Value
Q1 0
NQ1 1
Q2 1
NQ2 0
Q3 0
NQ3 1
Q4 0
NQ4 1
t1 0
t2 0
D1 0
t3 1
t4 1



Chip Name : ClkGrp8 (Clocked) Time : 7

Input pins

Name	Value

Output pins

Name	Value
sec1[3]	0
sec10[4]	7
min10[4]	0
min1[3]	0
as	1
bs	1
cs	1
ds	0
es	0
fs	0
gs	0

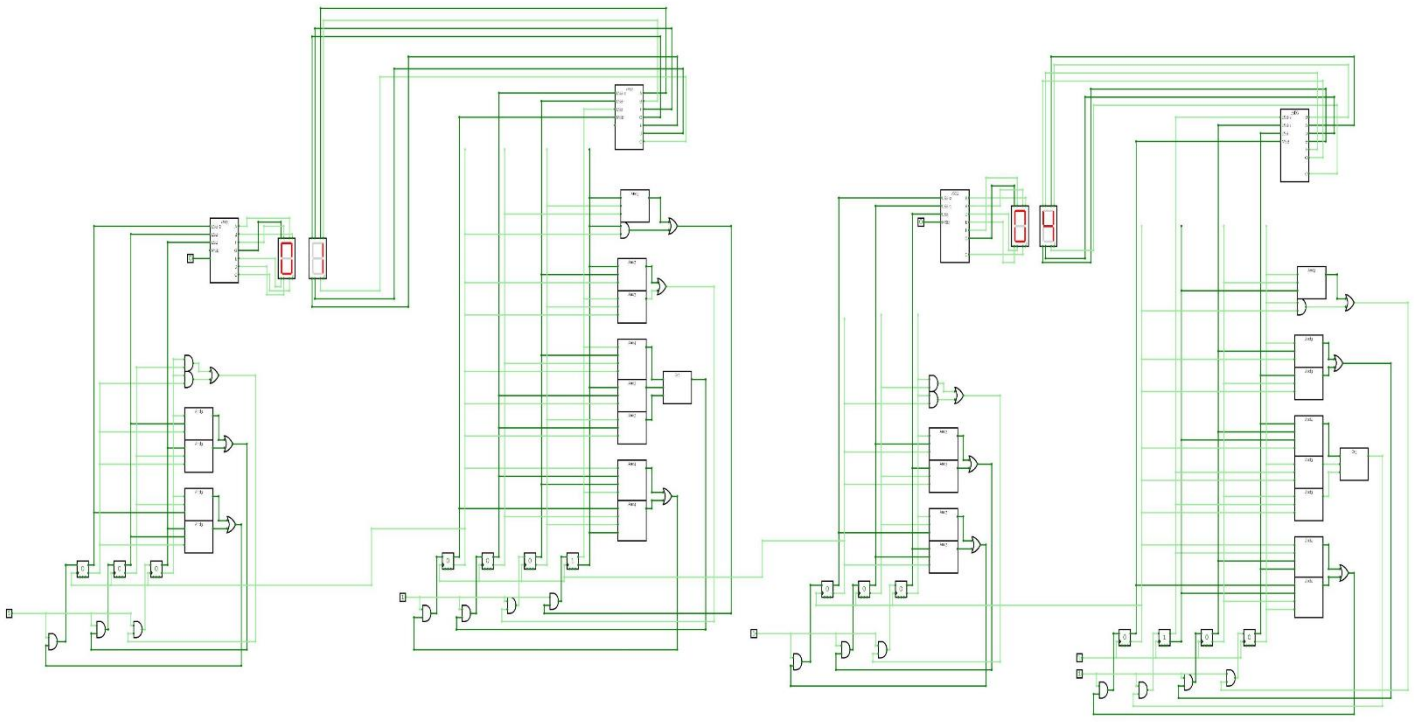
HDL

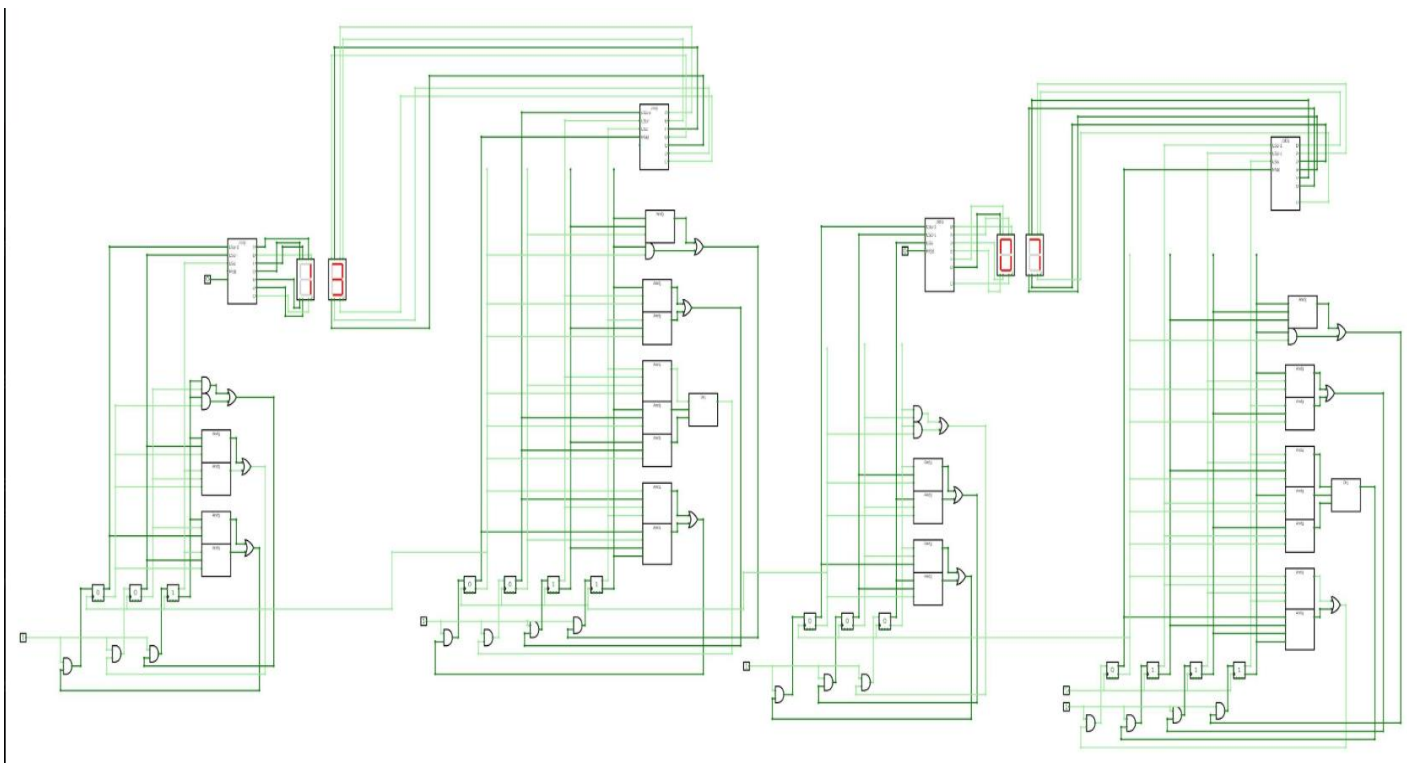
```
CHIP ClkGrp8
{
OUT sec1[3],sec10[4],min10[4],m

PARTS:
Not(in=Q1,out=NQ1);
Not(in=Q2,out=NQ2);
Not(in=Q3,out=NQ3);
Not(in=Q4,out=NQ4);

//D1
And4(a=NQ1,b=Q2,c=Q3,d=Q4,out=t
And4(a=Q1,b=NQ2,c=NQ3,d=NQ4,out
Or(a=t1,b=t2,out=D1);
```

Internal pins

Name	Value
Q1	0
NQ1	1
Q2	1
NQ2	0
Q3	1
NQ3	0
Q4	1
NQ4	0
t1	1
t2	0
D1	1
t3	0
t4	0
t5	0

CIRCUIT VERSE

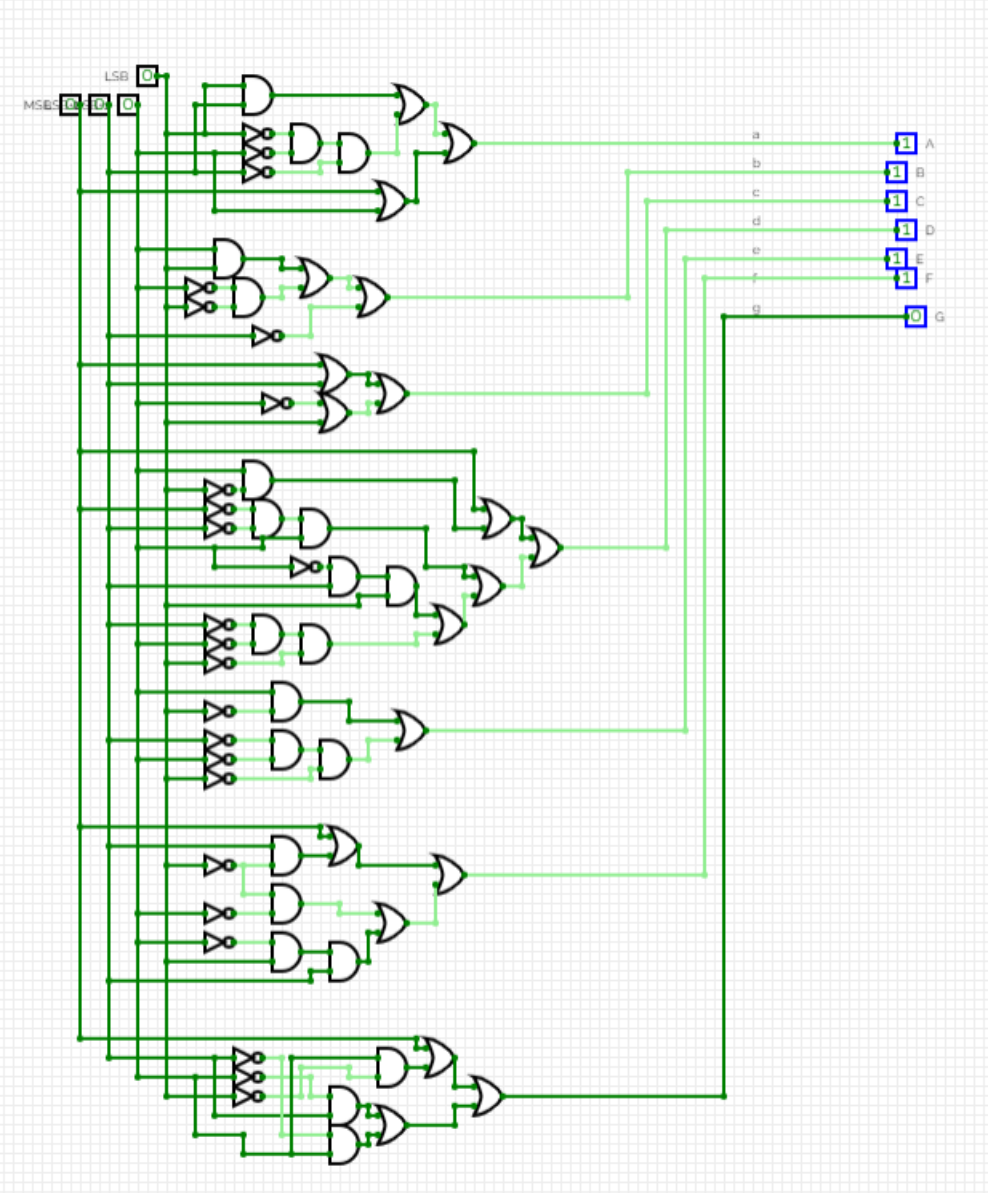




DESIGN IN CIRCUITVERSE

- USING THE EXPRESSIONS OBTAINED AFTER SIMPLIFYING THE KMAPS ,WE DESIGNED THE MOD-6, MOD-10 COUNTERS IN THE ONLINE PLATFORM USING INBUILT AND, OR, NOT GATES AND DFF.
- THE OUTPUTS OF THE MOD-10, MOD-6 COUNTERS WERE FED INTO A BCD TO 7 SEGMENT DECODER AND THEREBY THE OUTPUT FROM THE DECODER IS CONNECTED TO THE RESPECTIVE PINS OF THE DISPLAY.

BCD TO 7 SEGMENT



CHALLENGES FACED :

- DURING THE DESIGN OF A MOD-6 COUNTER FOR DISPLAYING THE SECONDS AFTER 9 THE DFF OF THE MOD 6 COUNTER HAS TO GET TRIGGERED ONLY WHEN THE MOD-10 COUNTER GOES FROM 9 TO 0 (1001 TO 0000) .
- IN THE NAND2TETRIS PLATFORM THE DFF IS CLOCKED AUTOMATICALLY BY THE SYSTEM AND HENCE THE MOD-6 COUNTER DESIGNED WITH THE SAME DFF COUNTS AFTER EVERY TICK-TOCK.
- IN THIS CASE WE WANT THE COUNTER TO COUNT ONLY WHEN THE MOD-10 COUNTER GOES FROM 9 TO 0 (1001 TO 0000). THIS CAN BE ACHIEVED BY BUILDING A DFF USING NAND GATES AND CLOCKING IT MANUALLY.

- DURING THE LOADING OF THE CHIP DFfMY THE HARDWARE SIMULATOR THREW AN ERROR “THE CHIP HAS CIRCLE IN ITS PARTS CONNECTION.”
- THE ABOVE CIRCUIT WAS NOT LOADING IN NAND2TETRIS EVEN AFTER REPEATED ATTEMPTS TO APPEND THE OUTPUT/INTERNAL PINS.
- HENCE THE MOD-6 (TENS DIDGIT) SECOND COUNTER AND THE MOD-10 , MOD-6 MINUTE COUNTER WAS NOT LOADING FOR THE DESIGN DEPENDS ON THE DFF THAT HAS BEEN CREATED.
- THEREBY THE OUTPUT IN THE HACK PLATFORM HAS NOT BEEN SHOWN AFTER 9 SECONDS.

CONCLUSION

THUS, THIS IS HOW A CLOCK CAN BE DESIGNED USING COUNTERS AND FLIP FLOPS IN NAND2TETRIS AND CIRCUITVERSE.

REFERENCE

- NOAM NISAN AND SHIMON SCHOCKEN BOOK
- WIKIPEDIA (FOR CIRCUIT DIAGRAM FOR DFF)

THANK YOU!