

Labbrappo

10 oktober 2020

Philip Salqvist

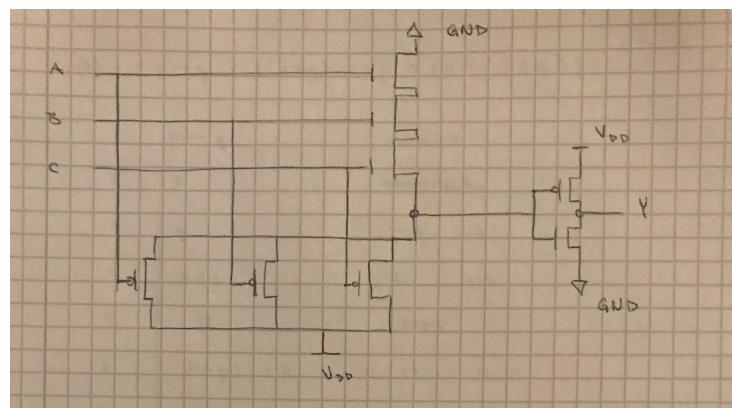
phisal@kth.se
CINTE
1992-04-17

Innehåll

1 Modul 1	1
2 Modul 2	3
3 Modul 3	5
4 Modul 4	8

1 Modul 1

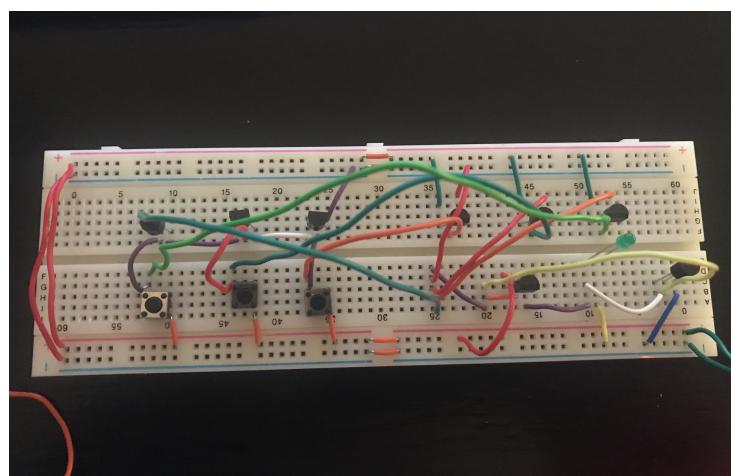
Målet med laborationen var att bygga en 3-ingångars CMOS-grind. Beroende på om studentens födelsedatum var udda eller jämnt valdes antingen en CMOS AND eller en CMOS-OR, för den här labbrapporten valdes AND. Som vi kan se i figur 1 så krävs en NAND-grind bestående av tre p-mos- och tre n-mos-transistorer, följt av en inverterare bestående av en p-mos- och en n-mos-transistor. Vi kan se i figur 2 att $Y = 1$ då $A = B = C = 1$. För alla andra kompositioner av A, B och C så kommer $Y = 0$. För att åstadkomma $Y = 1$ vill vi att NAND-grinden genererar 0. Det kommer bara att ske då alla inputs är 1, eftersom det är det enda fallet då våra seriekopplade n-mos transistorer kommer att öppna vägen för 0V att ta sig till inverteraren. I de fall då någon input är 0 kommer motsvarande n-mos att stänga vägen för GND, medan det räcker med minst en 0:a till någon av våra p-mos-transistorer att släppa igenom 5V till inverteraren. Om 5V tar sig till inverteraren kommer vår p-mos att stänga vägen till Y för 5V medan 5V till n-mos innebär en öppen väg för 0V till Y . I motsatt fall där 0V tar sig till inverteraren kommer p-mos att släppa fram 5V medan n-mos kommer att stänga möjligheten för 0V att ta sig till Y . Figur 3 visar hur detta har implementerats på breadboard.



Figur 1: Kopplingsschema labb 1

A	B	C		Y
1	0	0		0
1	1	0		0
1	1	1		1
0	1	1		0
0	0	1		0
0	0	0		0
0	1	0		0
1	0	1		0

Figur 2: Sanningstabell labb 1



Figur 3: Breadboard labb 1

2 Modul 2

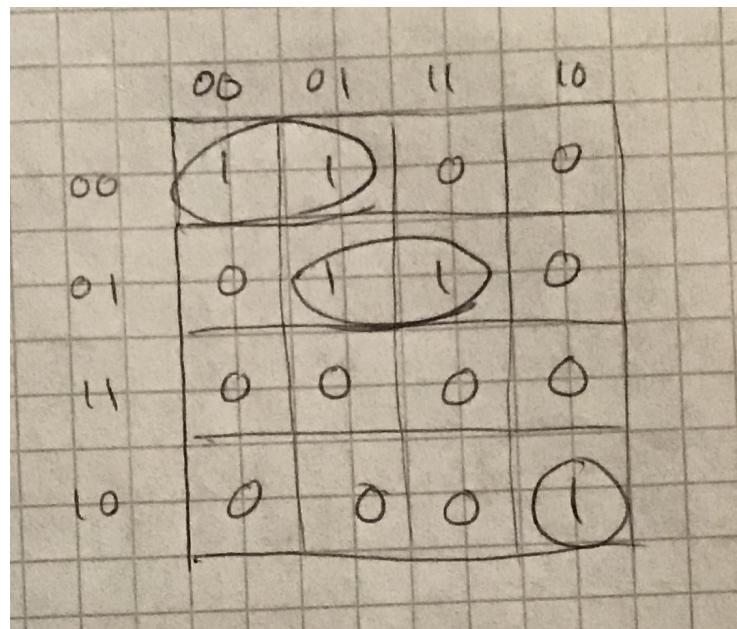
Laborationen bestod av att gå från sanningstabell, till K-map, till kopplings-schema och till sist implementera kopplingsschemat på breadboard genom att använda integrerade kretsar. I momentets start genererades individuella outputs från ens födelsedatum vilka redovisas i figur 4. Informationen från sanningstabellen kunde sedan överföras till en K-map där kolumner representeras av q_3q_2 och rader av q_1q_0 . Vidare har kolumner och rader placerats i graykod-ordning för att förenkla optimeringen av vårt logiska uttryck för Y . Vi förenklar genom att ringa in 1:or i samlingar av 1 och 2, som vi kan se i figur 5, vilket ger uttrycket:

$$Y = \overline{y_3}\overline{y_1}y_0 + y_2\overline{y_1}y_0 + y_3\overline{y_0}y_1\overline{y_0}$$

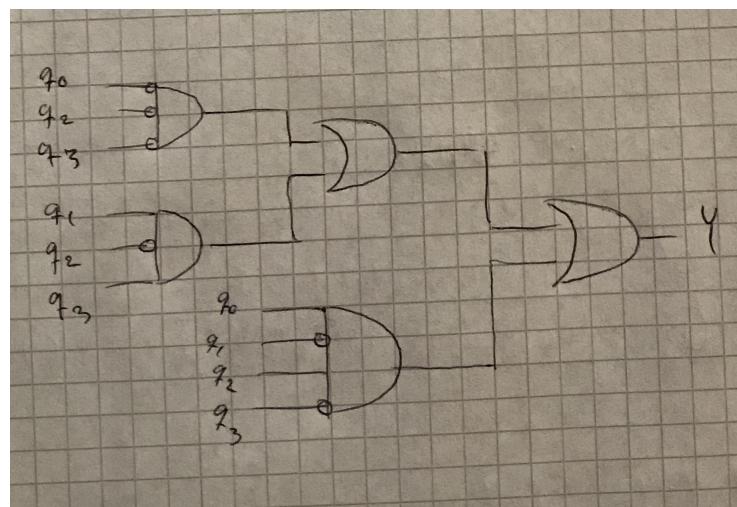
Kopplingsschemat som man kan se i figur 6 visar att lösningen består av två 2-ingångars OR-grindar, två 3-ingångars AND-grindar samt en 4-ingångars AND-grind. Figur 7 visar den slutgiltiga implementationen på breadboard.

q_0	q_1	q_2	q_3	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

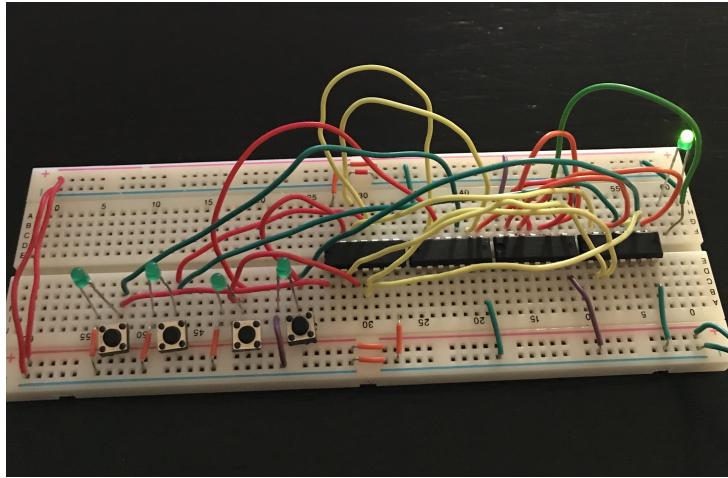
Figur 4: Sanningstabell labb 2



Figur 5: K-map labb 2



Figur 6: Kopplingsschema labb 2



Figur 7: Breadboard labb 2

3 Modul 3

Modul 3 behandlade Finite State Machines (FSM). Återigen så utgick varje student från sitt födelsedatum för att generera en individuell uppgift, i detta fall det state diagram som man kan se i figur 8. Därefter genererades en state table som beskrev next states, $q_1^+ q_0^+$ i förhållande till present states, $q_1 q_0$ samt inputs uttryckt som ba . Denna state table kunde sedan brytas ned till en K-map för q_1^+ som vi ser i figur 9 och ytterligare en K-map för q_0^+ som vi ser i figur 10, där båda våra K-maps har kolumner och rader i graykod-ordning. Vidare så ringades 1:or in i figur 9 och 0:or i figur 10 vilket gav dem förenklade uttryckena:

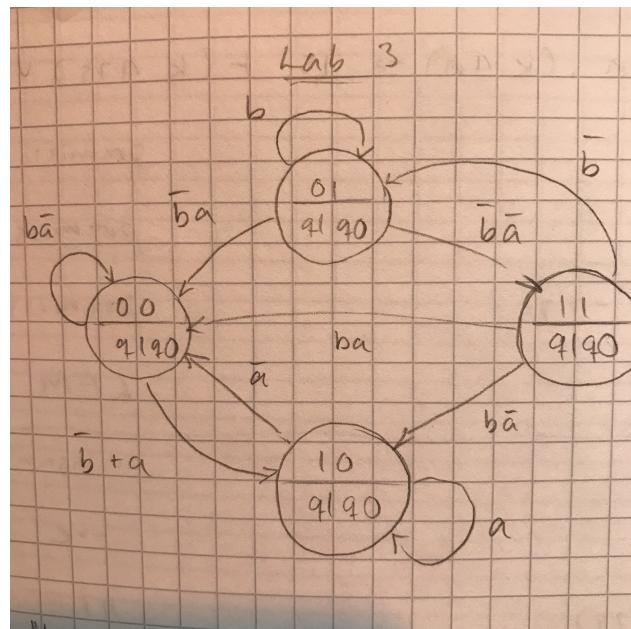
$$q_1^+ = \overline{q_0}a + q_1 q_0 \bar{a}b + \bar{a}\bar{b}\overline{q_1}$$

$$q_0^+ = q_0(\overline{q_1} + \bar{b})(\bar{a} + b + q_1)$$

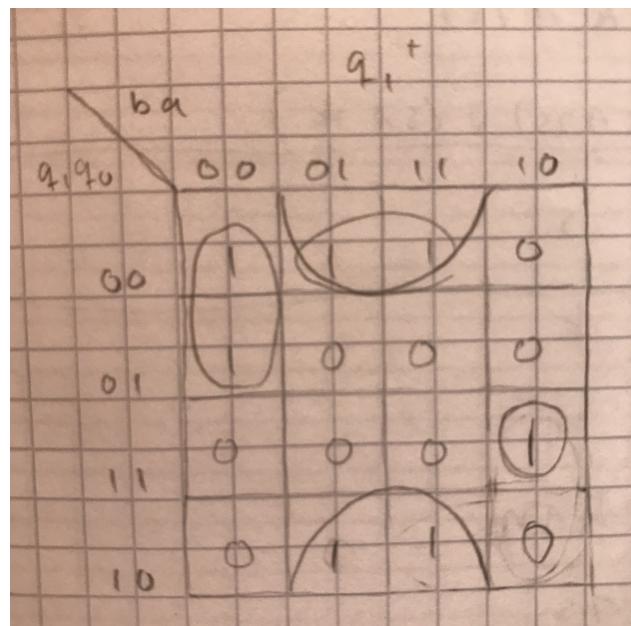
Om vi använder två negationssträck över q_0^+ och knäcker får vi genom De Morgans:

$$q_0^+ = \overline{\overline{q_0}(\overline{q_1} + b)(\bar{a} + b + q_1)} = \overline{\overline{q_0} + q_1 b + a \bar{b} \overline{q_1}}$$

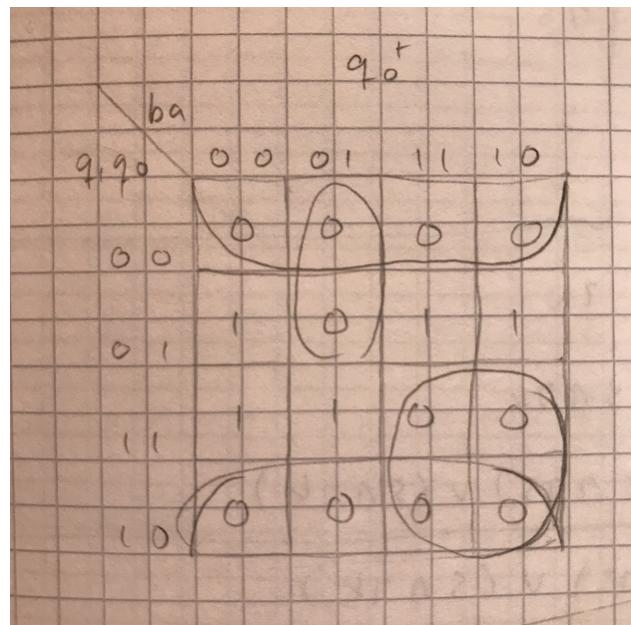
Vi har nu våra uttryck för next state logic som kan kopplas till vår DFF. Eftersom att vi inte har någon output logic så är uttryckena kompletta för att bygga systemet på en breadboard, vilket redovisas i figur 11.



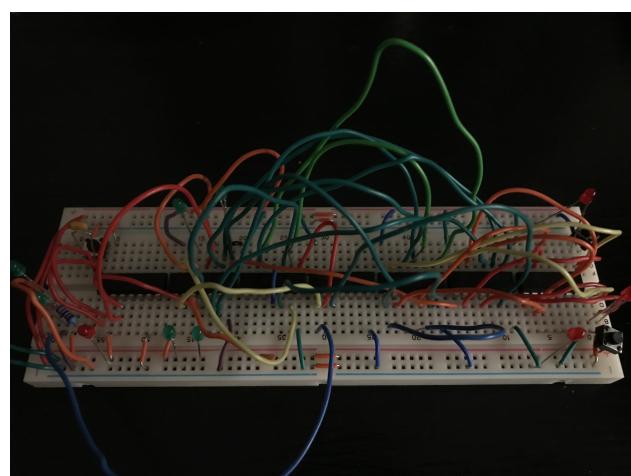
Figur 8: State diagram labb 3



Figur 9: K-map q_1^+ labb 3



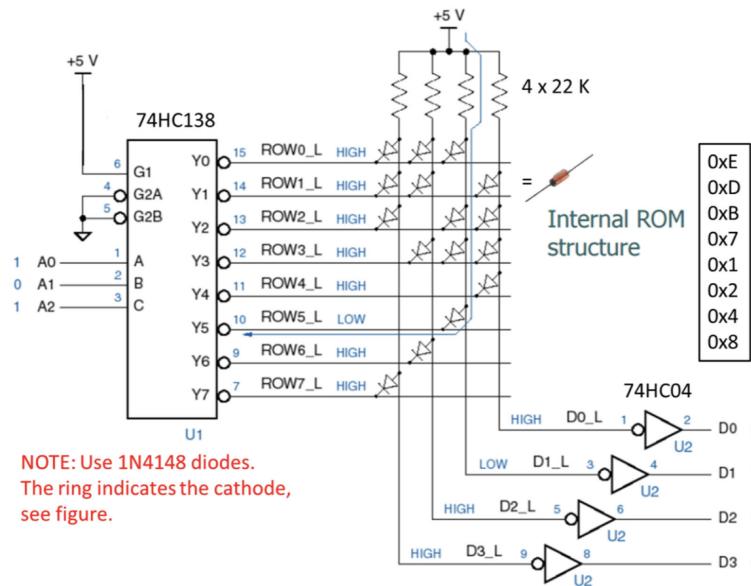
Figur 10: K-map q_0^+ labb 3



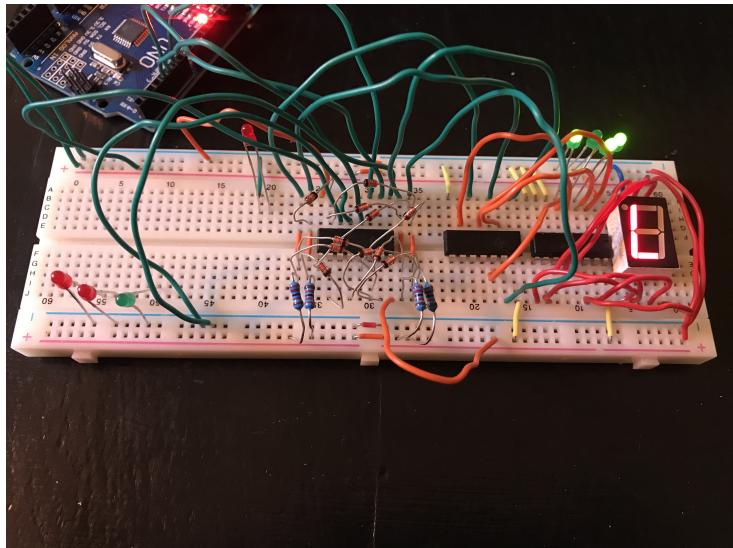
Figur 11: Breadboard labb 3

4 Modul 4

I den avsultande modulen handlade arbetet om att bygga ett system som genom inputs A_0, A_1, A_2 , skulle visa studentens 8-siffriga personnummer löpande på en 7 segment display. Som vi kan se i figur 12 så kopplas A_0, A_1, A_2 till en 1 to 8 decoder. Inputen kan generera $2^3 = 8$ olika tal där varje tal korresponderar till en output eller wordline. Decodern kommer beroende på input aktivera en motsvarande wordline, där aktiverad innebär låg och resterande alltså kommer att vara höga. För varje wordline har vi möjligheten att koppla vidare till fyra bitlines mha small signal diodes, som i sin tur är kopplade till inverterare. En påkopplad small signal diode kommer alltså, genom inverteraren innebära en 1:a som output och avsaknaden av en sådan diod kommer innebära att outputen för motsvarande bitline blir 0. Antalet dioder kopplade till en wordline kommer alltså att bestämma den 4-bitars utsignal som önskas, vilket motsvarar ett av de åtta tal som utgör det sökta personumret. Alla möjliga kombinationer av A_0, A_1, A_2 genereras av vår Arduino. Implementationen av systemet på breadboard ges i figur 13.



Figur 12: Kopplingsschema Labb 4



Figur 13: Breadboard labb 4