Konstruktion av ett logiskt grindnät

Rapport av: Jacob Lundkvist
Klass: Ela21
Lärare: Erik Jagre, Erik Pihl
Kurs: Digital konstruktion
Rapportens datum: 2022-05-04
YRGO HÖGRE YRKESUTBILDNING GÖTEBORG

Introduktion

Under detta projekt så skulle en sanningstabell avläsas genom att skapa ett Karnaugh diagram. Diagrammet resulterade i ett par logiska uttryck som gjorde det möjligt att rita upp en krets i LTspice för simulering. Efter att simuleringen bekräftade sanningstabellen mot det uppritade grindnätet så skrevs en hårdvarunära kod i VHDL.

När koden var färdigskriven så gick det att verifiera det förväntade grindnätet med det genererade grindnät som uppstod från VHDL-koden och kontrollera mot ModelSim för samtliga binära kombinationer. Till sist programmerades ett FPGA-kort med VHDL-koden.

Tabell 1 - Sanningstabell för projektet. ABCD är insignaler, XYZ är utsignaler.

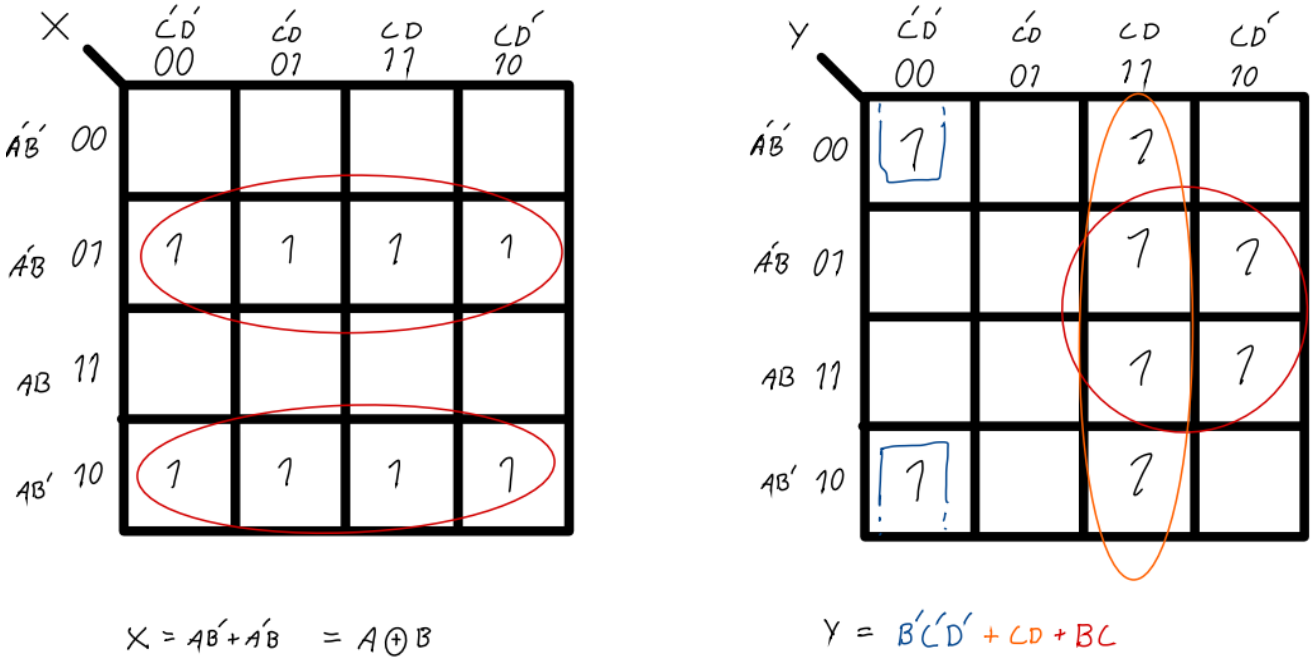
ABCD	XYZ
0000	010
0001	000
0010	001
0011	011
0100	101
0101	100
0110	110
0111	111
1000	110
1001	100
1010	101
1011	111
1100	001
1101	000
1110	010
1111	011

Innehållsförteckning

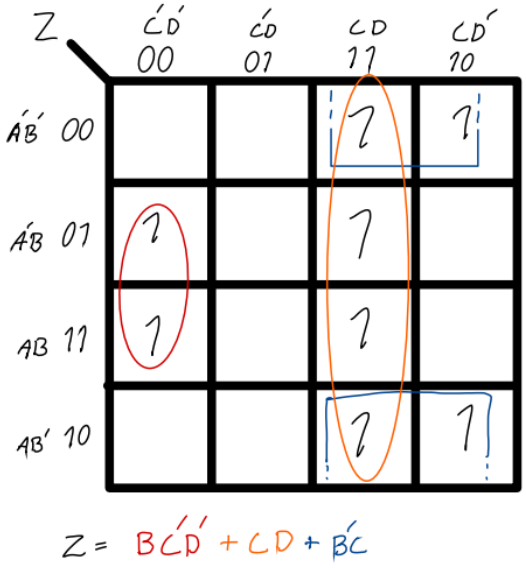
1. Från sanningstabell till logiskt grindnät

1.1. Tolkning av sanningstabell med Karnaugh-diagram

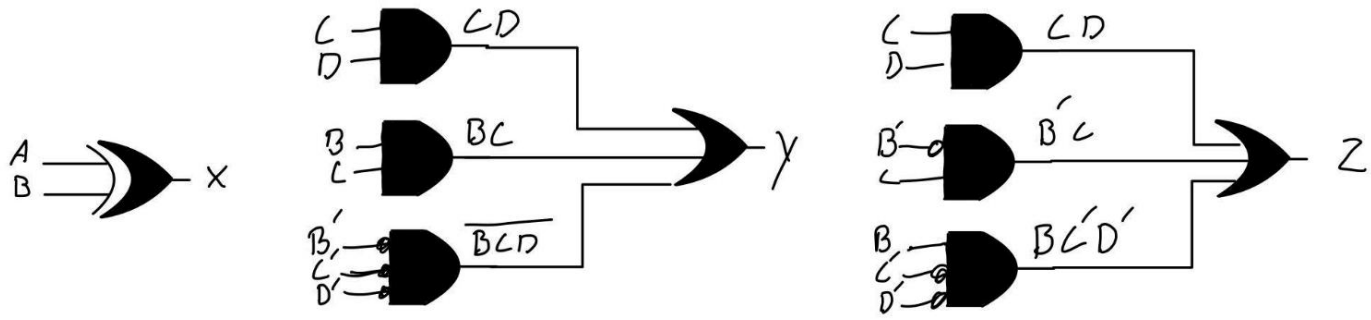
För att göra uppgiften att tydligare så var första steget att separera utsignalerna X, Y och Z. Efter detta så gick det att rita upp Karnaugh-diagram och sätta in signalerna A, B, C, D i ett rutnät av 4x4. Sedan ringade man in så stora angränsande par som möjligt som följer tvåpotens (2^n)



Figur 1 & 2 - Karnaugh-diagram för signal X och Y



Figur 2 - Karnaugh-diagram för signal Z

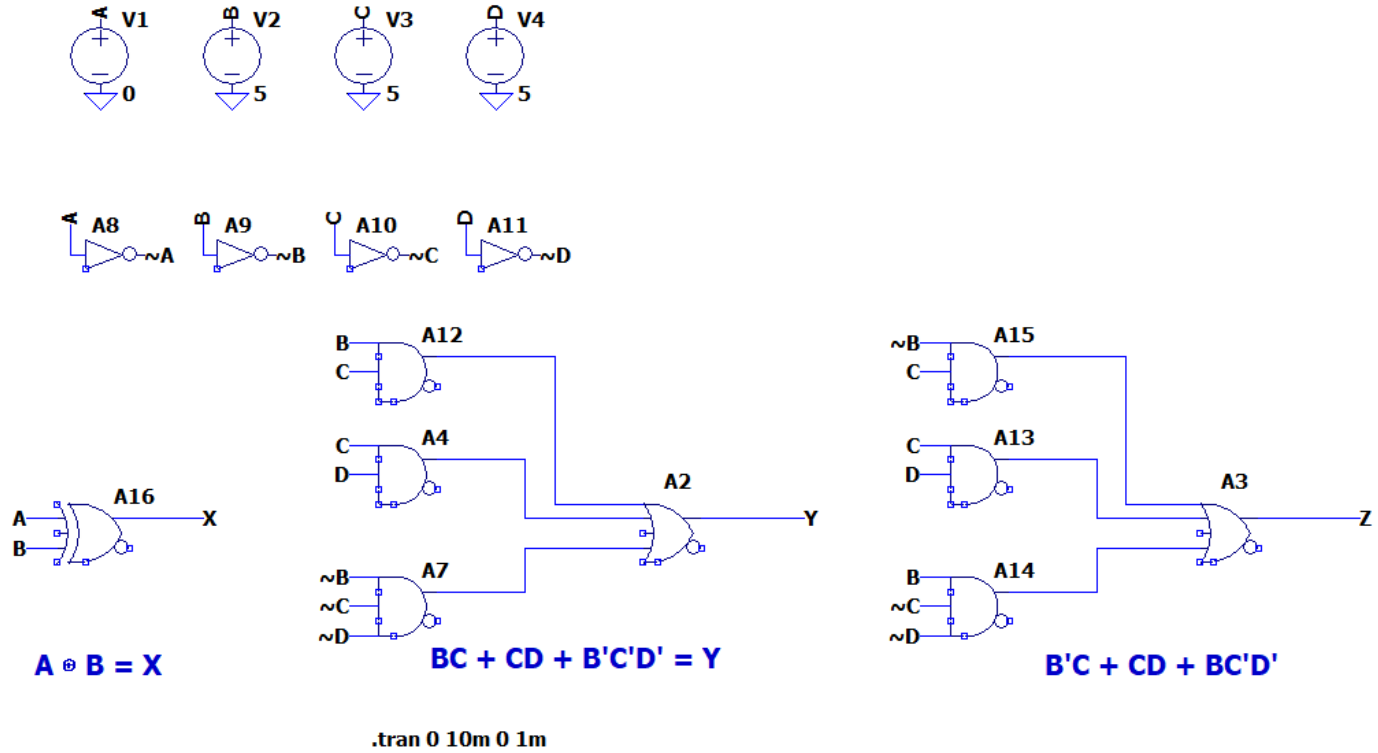


Figur 3 - Handritat grindnät för signaler XYZ

Med hjälp av Karnaugh-diagrammen och de nya logiska uttrycken så kunde grindnätet realiseras i LTspiceXVII för simulering.

1.2. Validering av krets i LTspiceXVII

Grindnätet ritades upp utefter ekvationerna från Karnaugh-diagrammet. Varje addition markerade en signal till en OR-grind, multiplikation resulterar i en AND grind och formeln för utsignal X ledde till en XOR grind där bara en signal kan vara hög för att resultera i hög utsignal.



Figur 4 - Uppritad krets i LTspiceXVII för validering av sanningstabell.

Genom att skifta värden på signalerna ABCD från noll till fem så gick det att få utsignaler som gick att jämföra mot sanningstabellen (tabell 1). Efter att tillräckligt många variationer hade testats så ansågs grindnätet vara korrekt ritat och därmed kunde nästa steg inledas, realisera konstruktionen i VHDL.

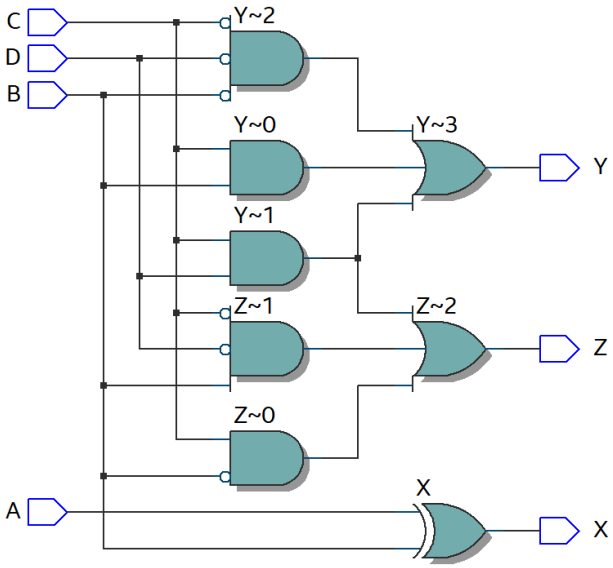
1.3. Konstruktion och simulering i VHDL.

Från de två tidigare stegen så fanns nu logiska ekvationer för ut- och insignalerna och hur det förväntade grindnätet bör se ut. Med tanke på det fanns tillgång till både sanningstabell och logiska ekvationer så fanns det två tydliga vägar man kunde ta med VHDL-koden. En väg hade varit att använda sanningstabellen och med det sagt använt sig av vektorer för ut- och insignaler, som exempel ”`std_logic_vector (3 downto 0);` ”.

I detta projekt så valdes det andra alternativet, användandet av de logiska uttrycken som hade tagits fram från tidigare diagram. Det var då enkelt att tilldela utsignalerna X, Y och Z genom att mata in respektive ekvation från Karnaugh-diagrammen.

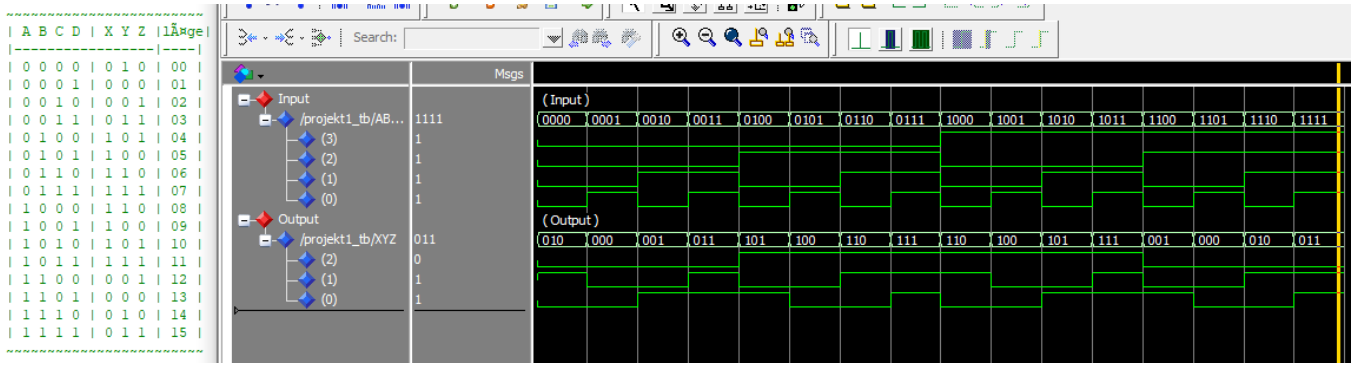
```
X <=  (A xor B);
Y <=  (B and C)      or (C and D) or (not B and not C and not D);
Z <=  (not B and C) or (C and D) or (B and not C and not D);
```

Figur 5 - Utportar tilldelas respektive ekvation med sina inportar.



Figur 6 - Syntetiserat grindnät i RTL Viewer (Quartus prime lite 18.1)

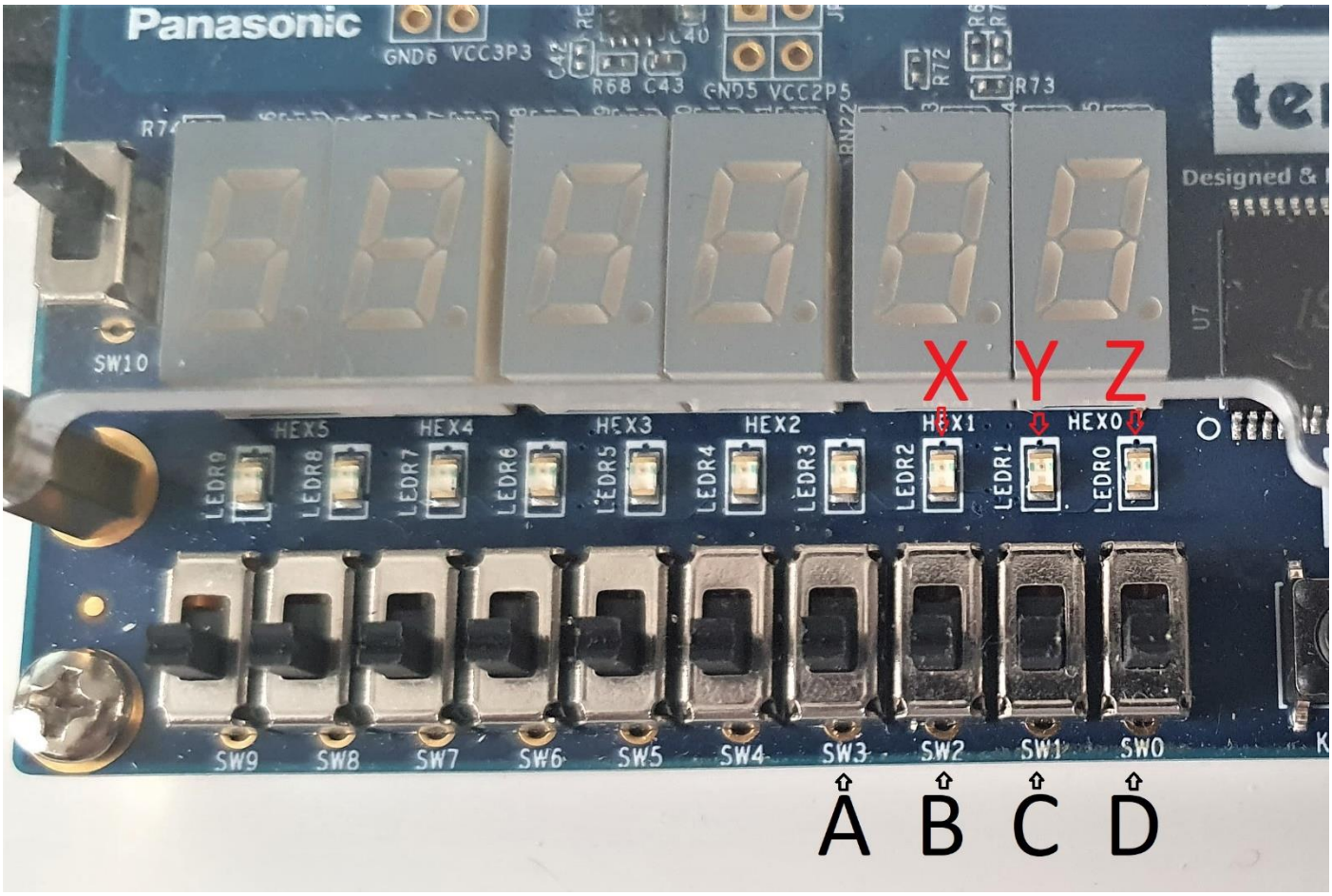
I testbänken så valdes det att använda vektorer som signaler under simulering. Detta krävde enbart en process som itererar igenom ett fyra bitars tal för att simulera samtliga 16 lägen under 160 nanosekunder. Simuleringsdatan från ModelSim gick då att betrakta binärt precis som i den givna sanningstabellen (se figur 7).



Figur 7 - Data från Modelsim baserat på filen projekt1_tb.vht

2. Resultat

Konstruktionen fungerade som planerat, när VHDL koden väl var implementerad på FPGA-kortet så gick det att följa sanningstabellen genom att använda kortets slide switches som i sin tur tände korrekt lampa enligt sanningstabellen. Detta visade då att karnaugh-diagrammen, simuleringarna och programmeringen var korrekt och gav efterfrågat resultat.



Figur 8 - PIN-placement för samtliga in- och utportar.

3. Diskussion

Ett projekt på en bra nivå där man fick gå igenom samtliga steg i processen. Där man kunde bryta ner konstruktionen i mindre delar vilket gjorde det enkelt att minimera logiska ekvationer. Hade sanningstabellen varit större, med fler insignaler så hade det inte varit lika trevligt med att göra om det till ett Karnaugh diagram. Antagligen så finns det en balans på när sanningstabellen bör implementeras som en case-sats, men i detta fall så kändes det lämpligt.

Det var lärorikt att nyttja vektorer i testbänken då det uppfattas mer effektivt att inkrementera en vektor än att skapa en process för varje enskild utsignal. För övrigt så fanns inga direkta motgångar under detta projekt bortsett från en tidig miss som upptäcktes snabbt, två värden missades ur sanningstabellen vid uppritande av Karnaugh-diagrammet för signalen X. Hade det inte upptäckts så hade X signalen blivit en or-gate kopplats via fyra and-gates.

Referenser

[1] terasIC, ”Teraisc DE0-CV user manual,” 08 05 2015. [Online]. Available: https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=163&No=921&PartNo=4. [Använd 2022 05 04].