



GEBZE TEKNİK ÜNİVERSİTESİ
ELEKTRONİK MÜHENDİSLİĞİ

ELM235

LOJİK DEVRE TASARIM LABORATUVARI

LAB 0x1 Deney Raporu

Lojik Devreler ve Tasarım Laboratuvarı Dersine Hazırlık

Hazırlayanlar
1) 1801022035 – Ruveyda Dilara Günal
2) 200102002087 – Alican Bayındır

1. Problemler

1.1. Problem I – Basit Bir Devre Tasarımı ve Simülasyonu

$$Y = AB + A\bar{C} + \bar{B}C$$

Denklem 1

1.1.A

Denklem 1’de verilen Boolean denkleminin HDL ile tasarlanmıştır.

```
/*lab1_g29_p1.sv
*
*Hazırlayanlar:
* Ruveyda Dilara Günal
* Alican Bayındır
*/

module lab1_g29_p1 (
input logic a, b, c,
output logic y
);

assign y = (a & b) | (a & ~c) | (~b & ~c & a) | (~b & c);

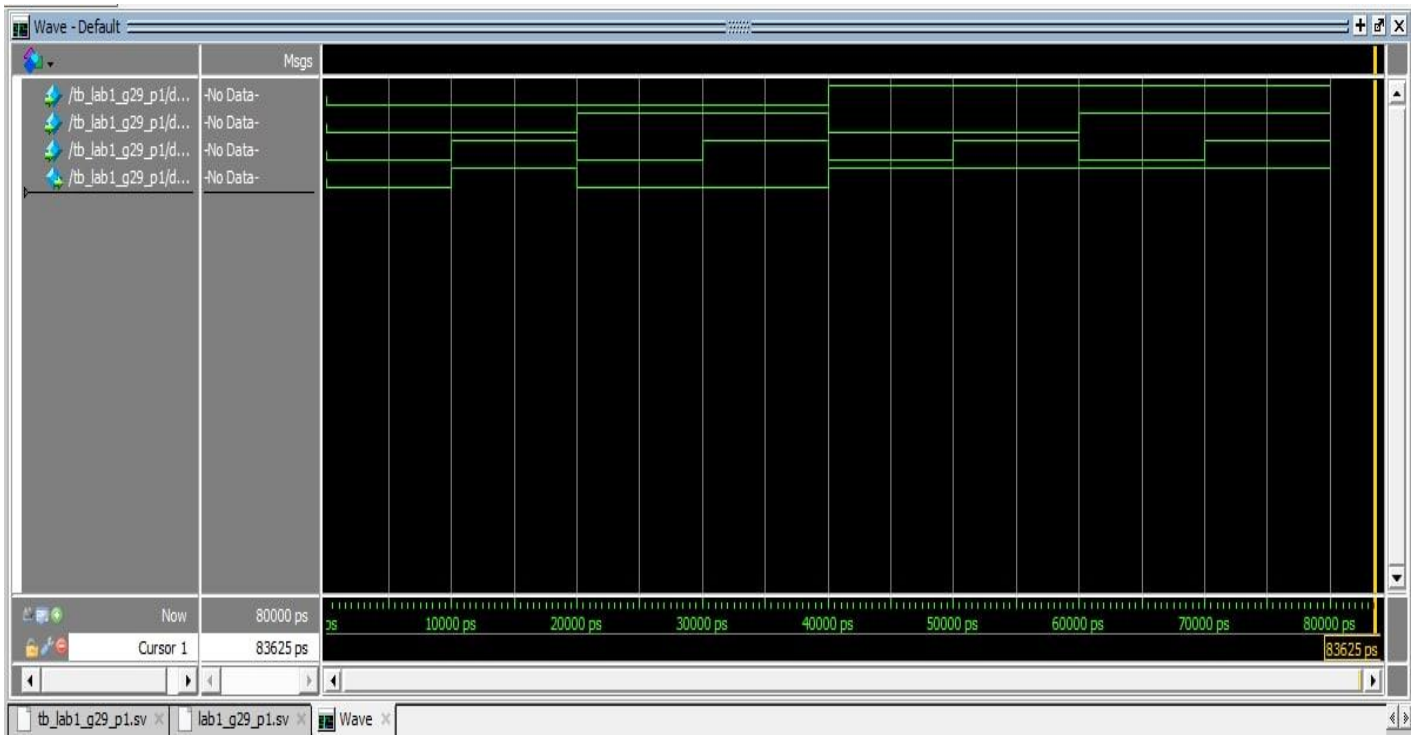
endmodule
```

1.1.B

Denklem 1’deki Boolean denklemi için tasarlanan HDL için Testbench oluşturularak, devrenin bütün girişlere karşı nasıl davrandığı gözlemlenmiştir.

```
// tb_lab1_g29_p1

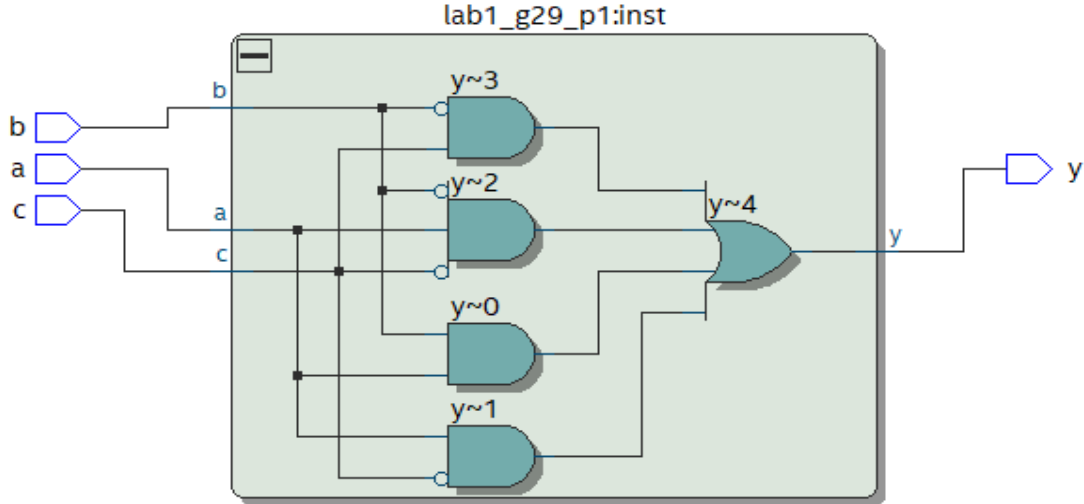
`timescale 1ns/1ps
module tb_lab1_g29_p1 ();
logic a, b, c;
logic y;
soru_iki dut0(a, b, c, y);
initial begin
a = 0; b = 0; c = 0; #10;
c = 1; #10;
b = 1; c = 0; #10;
c = 1; #10;
a = 1; b = 0; c = 0; #10;
c = 1; #10;
b = 1; c = 0; #10;
c = 1; #10;
$stop;
end
endmodule
```



Şekil 1: Modelsim Çıktısı


1.1.C

Devre Quartus programında sentezlenmiş ve devrenin ne kadar yer kapladığı, sentezlenen RTL ve eşleştirme ardı devre şemaları gösterilmiştir.



Şekil 2: Quartus Devre Sentezi

Analysis & Synthesis Resource Utilization by Entity

 <<Filter>>

	Compilation Hierarchy Node	Combinational ALUTs		Dedicated Logic Registers		Memory Bits	
1	> lab1_g29_p1_top	1 (0)		0 (0)		0	
UFM Blocks		DSP Elements	DSP 9x9	DSP 18x18	Pins	Virtual Pins	ADC blocks
0		0	0	0	4	0	0
Virtual Pins	ADC blocks	Full Hierarchy Name			Entity Name	Library Name	
0	0	lab1_g29_p1_top			lab1_g29_p1_top	work	

Tablo 1: Resource Utilization Report

Analysis & Synthesis Resource Usage Summary		
<<Filter>>		
	Resource	Usage
1	Estimated Total logic elements	1
2		
3	Total combinational functions	1
4	▼ Logic element usage by number of LUT inputs	
1	-- 4 input functions	0
2	-- 3 input functions	1
3	-- <=2 input functions	0
5		
6	▼ Logic elements by mode	
1	-- normal mode	1
2	-- arithmetic mode	0
7		
8	▼ Total registers	0
1	-- Dedicated logic registers	0
2	-- I/O registers	0
9		
10	I/O pins	4
11		
12	Embedded Multiplier 9-bit elements	0
13		
14	Maximum fan-out node	lab1_g29_p1:inst y~0
15	Maximum fan-out	1
16	Total fan-out	8
17	Average fan-out	0.89

Tablo 2: Resource Usage Summary

Yukarıdaki boolean denklemi a, b, c adlı üç tane girişi olan bir lojik devrede istenen performansı sağlayacak şekilde bağlanmıştır. Bu problemde amaçlanan, föyde verilen boolean denklemini HDL kullanarak tasarlamak ve testbenchi ile birlikte simüle etmektir. Şekil 1 ve Şekil 2 de görüldüğü üzere istenen boolean denklemi başarılı bir şekilde devre haline getirilmiştir.

1.2. Problem II – Toplayıcı Tasarımı

1.2.A-B-C

Half-adder, half-adder kullanılarak full-adder ve full-adder kullanılarak ripple carry adder tasarlanmıştır.

```
/*lab1_g29_p2.sv
*Hazırlayanlar:
*  Ruveyda Dilara Günal
*  Alican Bayındır
*/

module half_adder (
    input logic a, b,
    output logic s, c
);

    assign s = a ^ b;
    assign c = a & b;
endmodule

module full_adder (
    input logic x, y, cin,
    output logic sout, cout
);
    wire sum0;
    wire carry0;
    wire carry1;

    half_adder ha0 (.a(x), .b(y), .s(sum0), .c(carry0));
    half_adder ha1 (.a(sum0), .b(cin), .s(sout), .c(carry1));
    assign cout = carry0 | carry1;
endmodule

module lab1_g29_p2 (
    input logic [3:0] A,B,
    input logic cin,
    output logic [3:0] S,
    output logic cout
);

    wire [2:0] carries;

    full_adder fa0 (.x(A[0]), .y(B[0]), .cin(cin), .sout(S[0]), .cout(carries[0]));
    full_adder fa1 (.x(A[1]), .y(B[1]), .cin(carries[0]), .sout(S[1]),
    .cout(carries[1]));
    full_adder fa2 (.x(A[2]), .y(B[2]), .cin(carries[1]), .sout(S[2]),
    .cout(carries[2]));
    full_adder fa3 (.x(A[3]), .y(B[3]), .cin(carries[2]), .sout(S[3]),
    .cout(cout));
endmodule
```

1.2.C

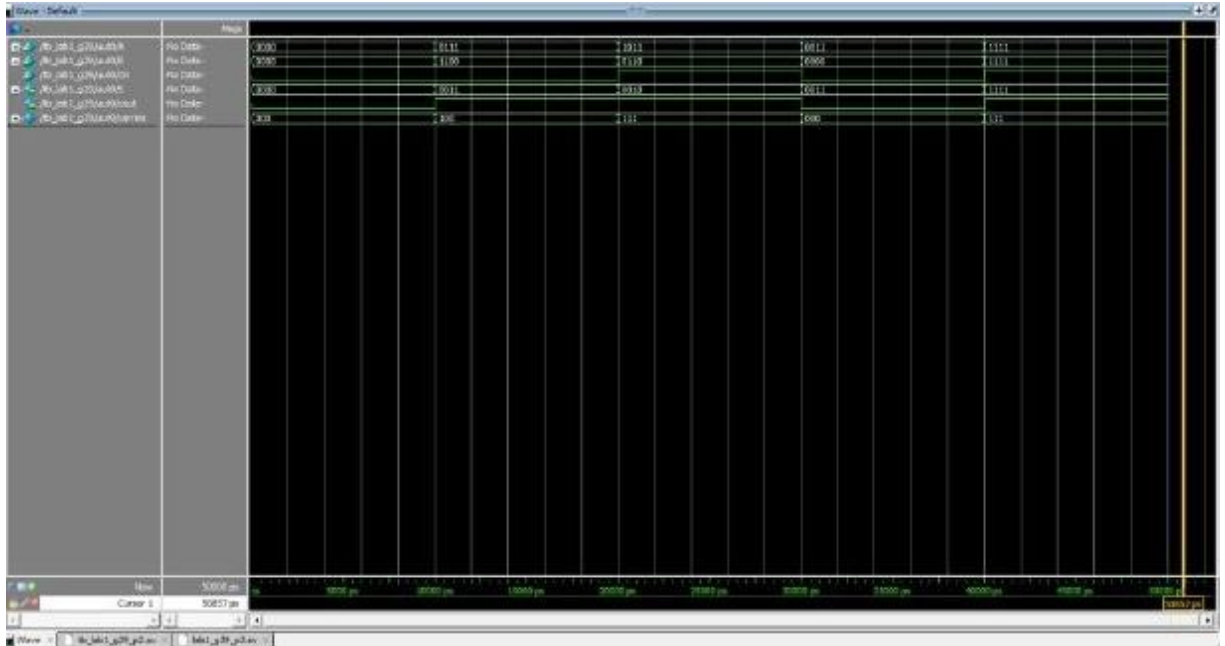
Testbench oluşturularak, devrenin bütün girişlere karşı nasıl davrandığını ve sonuçların doğruluğu gösterilmiştir.

```
// tb_lab1_g29_p2

`timescale 1ns/1ps
module tb_lab1_g29_p2();
  reg[3:0] A, B;
  reg cin;
  wire[3:0] S;
  wire cout;

  lab1_g29_p2 uut0(.A(A), .B(B), .cin(cin), .S(S), .cout(cout));

  initial begin
    A = 4'b0000; B = 4'b0000; cin = 0; #10;
    A = 7; B = 12; #10;
    A = 4'b1011; B = 4'b0110; cin = 1; #10;
    A = 3; B = 16; cin = 0; #10;
    A = 4'b1111; B = 4'b1111; cin = 1; #10;
    $stop;
  end
endmodule
```




Şekil 3: Modelsim Çıktısı

1.2.D

Devre Quartus programında sentezlenmiş ve devrenin ne kadar yer kapladığı, sentezlenen RTL ve eşleştirme ardı devre şemaları gösterilmiştir.

Analysis & Synthesis Resource Utilization by Entity

 <<Filter>>

Compilation Hierarchy Node		Combinational ALUTs		Dedicated Logic Registers		Memory Bits
1	▼ lab1_g29_p2_top	12 (0)		0 (0)		0
1	> full_adder:inst2	2 (1)		0 (0)		0
2	half_adder:inst	2 (2)		0 (0)		0
3	> lab1_g29_p2:inst3	8 (0)		0 (0)		0

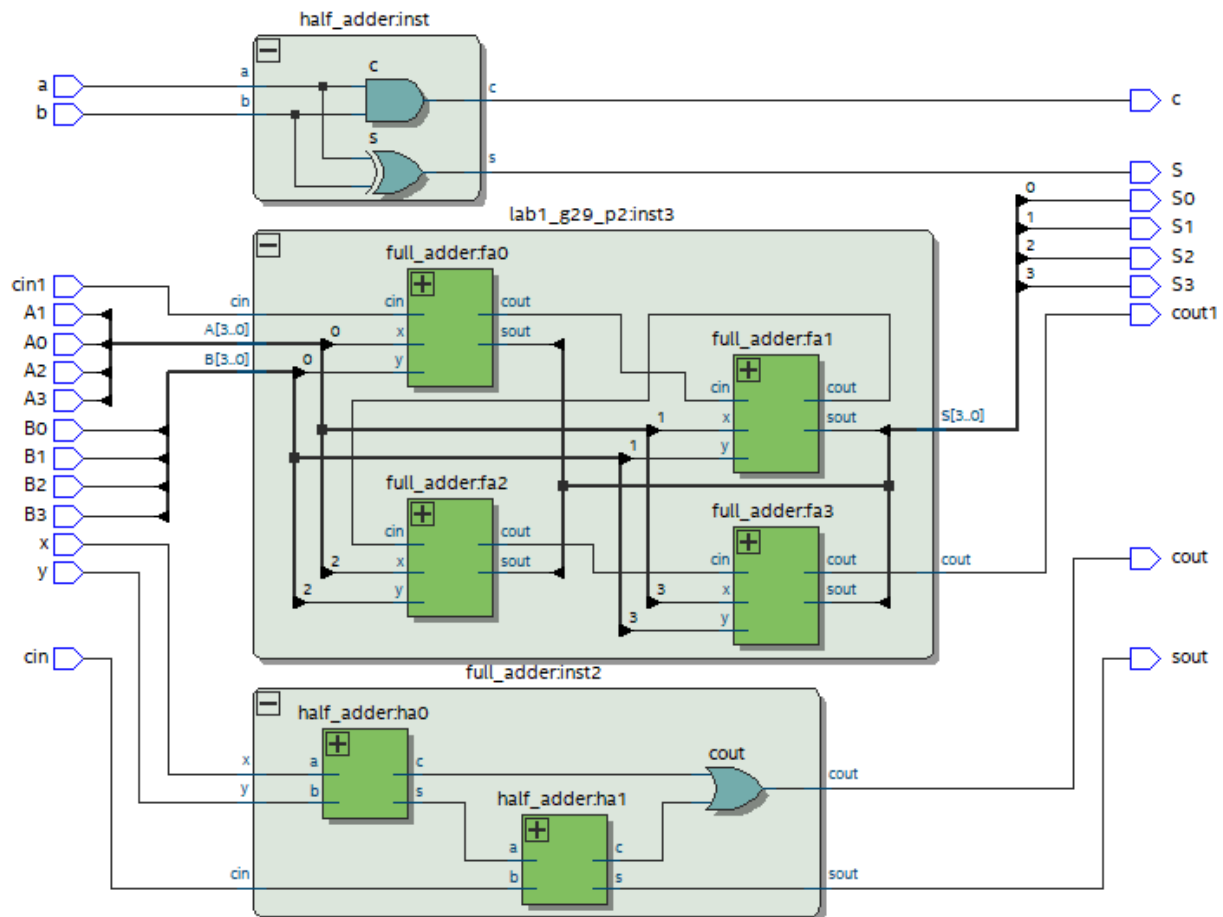
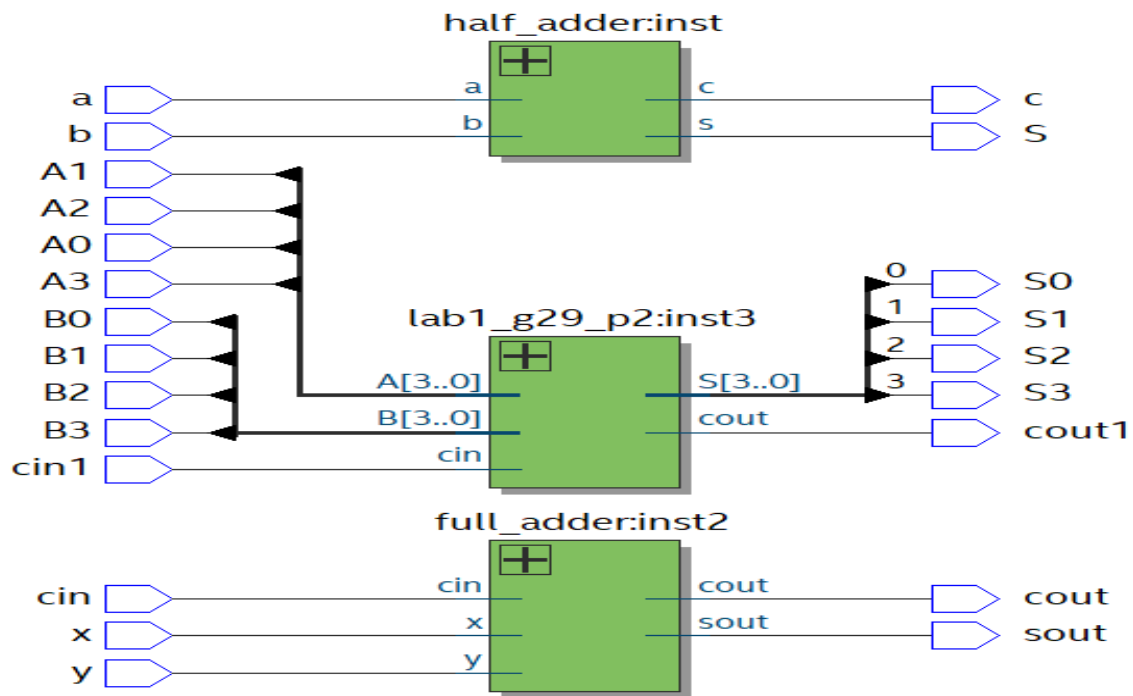
UFM Blocks	DSP Elements	DSP 9x9	DSP 18x18	Pins	Virtual Pins	ADC blocks
0	0	0	0	23	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

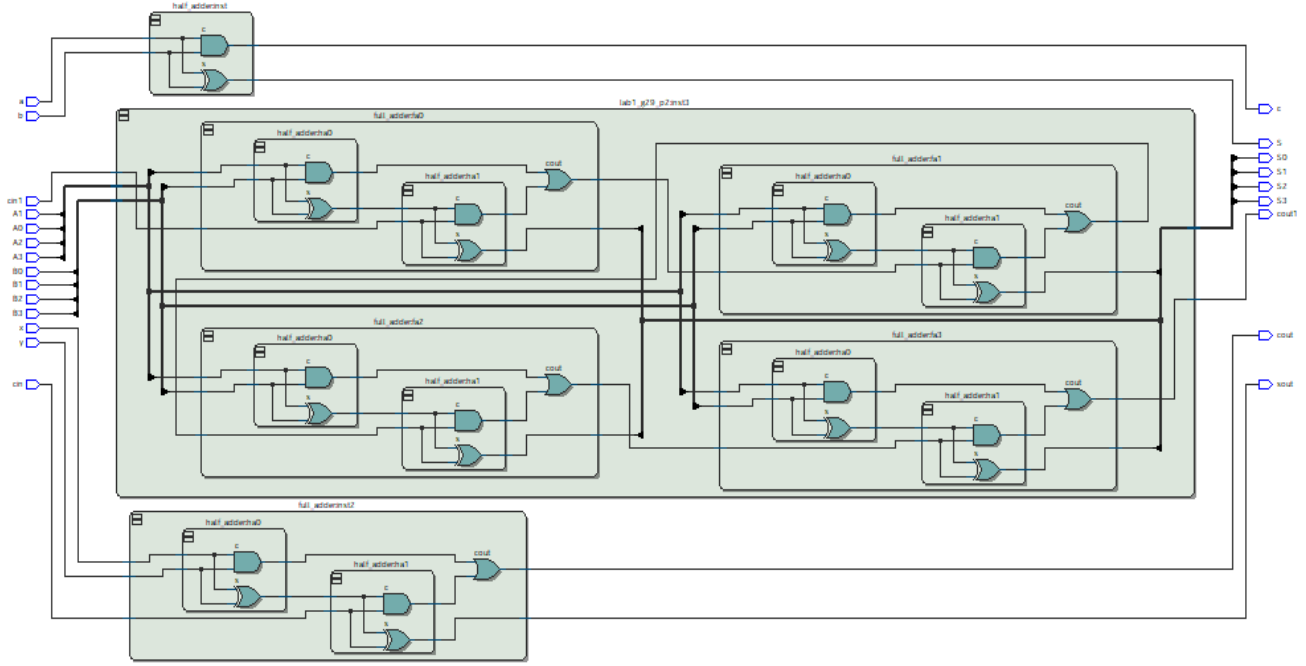
Full Hierarchy Name	Entity Name	Library Name
lab1_g29_p2_top	lab1_g29_p2_top	work
lab1_g29_p2_top full_adder:inst2	full_adder	work
lab1_g29_p2_top half_adder:inst	half_adder	work
lab1_g29_p2_top lab1_g29_p2:inst3	lab1_g29_p2	work

Tablo 3: Resource Utilization Report

	Resource	Usage
1	Estimated Total logic elements	12
2		
3	Total combinational functions	12
4	▼ Logic element usage by number of LUT inputs	
1	-- 4 input functions	0
2	-- 3 input functions	10
3	-- <=2 input functions	2
5		
6	▼ Logic elements by mode	
1	-- normal mode	12
2	-- arithmetic mode	0
7		
8	▼ Total registers	0
1	-- Dedicated logic registers	0
2	-- I/O registers	0
9		
10	I/O pins	23
11		
12	Embedded Multiplier 9-bit elements	0
13		
14	Maximum fan-out node	lab1_g29_p...fa0 cout~0
15	Maximum fan-out	2
16	Total fan-out	66
17	Average fan-out	1.14

Tablo 4: Resource Usage Summary





Şekil 4: Quartus Devre Sentezi

Bu problemde ilk olarak half-adder basit lojik kapılar(transistörler) kullanılarak tasarlanmıştır. Daha sonra tasarlanan half-adder kullanılarak full-adder tasarlanmıştır. Bu full-adder ile de ripple carry adder tasarlanmıştır. Aşamalar yapılırken her bir yeni modüle bir önceki modüle uygun olacak şekilde tüm gerekli bağlantılar gerekli portlara yapılmıştır. İstenen 4 bitlik ripple carry adder tasarlanmadan önce 1 bitlik half adder tasarlanmıştır. Sonrasında ise 2 tane half adder kullanılarak full adder'ın “sum” outputu hesaplanmıştır ve kalan cout outputu ise half adderların outputlarının or kapısı ile birbirine bağlanmasıyla full adder devresi tamamlanmıştır. Yapılan bu full adder'lar 1 bitlik olduğu için 4 adet full adder kullanılarak 4 bitlik ripple carry adder tasarlanmıştır. Modelsimde tasarlanan bütün scriptler daha sonra “Quartus” programında sentezlenerek devrenin RTL şeması, Resource Utilization Report ve Summary'leri bulunmuştur. Şekil 3'ten de görüldüğü gibi yapılan 4 bitlik ripple-carry adder doğru bir şekilde çalıştığı anlaşılmıştır. Daha sonra yapılan her şey raporlanmış ve pdf haline getirilmiştir.

1.3 Problem III – Çözücü tasarımı

1.3.A

X_3	X_2	X_1	X_0	A	B	C	D	E	F	G	Y
0	0	0	0	1	1	1	1	1	1	0	-
0	0	0	1	X	X	X	X	X	X	X	x
0	0	1	0	0	1	1	0	0	0	0	E
0	0	1	1	1	1	1	0	0	0	1	L
0	1	0	0	X	X	X	X	X	X	X	X
0	1	0	1	X	X	X	X	X	X	X	X
0	1	1	0	X	X	X	X	X	X	X	X
0	1	1	1	0	1	1	0	0	0	1	C
1	0	0	0	X	X	X	X	X	X	X	X
1	0	0	1	0	0	0	0	1	1	0	2
1	0	1	0	X	X	X	X	X	X	X	X
1	0	1	1	0	0	0	0	1	1	0	3
1	1	0	0	X	X	X	X	X	X	X	X
1	1	0	1	X	X	X	X	X	X	X	X
1	1	1	0	X	X	X	X	X	X	X	X
1	1	1	1	0	1	0	0	1	0	0	5

A:

X_1X_0/X_3X_2	00	01	11	10
00	1	X	1	0
01	X	X	0	X
11	X	X	0	X
10	X	0	0	X

$$A = \sim X_3 \sim X_1 + \sim X_3 \sim X_2 X_0$$

B:

X_1X_0/X_3X_2	00	01	11	10
00	1	X	1	1
01	X	X	1	X
11	X	X	1	X
10	X	0	0	X

$$B = \sim X_3 + X_2 X_3$$

C:

X_1X_0/X_3X_2	00	01	11	10
00	1	X	1	1
01	X	X	1	X
11	X	X	0	X
10	X	1	0	X

$$C = \sim X_3 + \sim X_1 X_0$$

D:

X_1X_0/X_3X_2	00	01	11	10
00	1	X	0	0
01	X	X	0	X
11	X	X	0	X
10	X	0	0	X

$$D = \sim X_3 \sim X_1$$

E:

X_1X_0/X_3X_2	00	01	11	10
00	1	X	0	0
01	X	X	0	X
11	X	X	1	X
10	X	0	1	X

$$E = \sim X_3 \sim X_1 + X_3 X_1$$

F:

X_1X_0/X_3X_2	00	01	11	10
00	1	X	0	0
01	X	X	0	X
11	X	X	0	X
10	X	1	1	X

$$F = \sim X_1 + X_3 \sim X_2$$

G:

X_1X_0/X_3X_2	00	01	11	10
00	1	X	1	0
01	X	X	1	X
11	X	X	0	X
10	X	0	0	X

$$G = \sim X_3 \sim X_1 + \sim X_3 X_0$$

1.3.B

Decoder HDL ile tasarlanmıştır.

```
/*lab1_g29_p3.sv
*
*Hazırlayanlar:
* Ruveyda Dilara Günal
* Alican Bayındır
*/

module lab1_g29_p3 (
    input logic x3, x2, x1, x0,
    output logic a, b, c, d, e, f, g
);
    assign a = (~x3 & ~x1) | x0 & (x0 & ~x3 & ~x2);
    assign b = ~x3 | (x3 & x2);
    assign c = ~x3 + (~x1 & x0);
    assign d = ~x3 & ~x1;
    assign e = (~x3 & ~x1) | (x1 & x3);
    assign f = ~x1 | (x3 & ~x2);
    assign g = ~x3 & ~x1 | ~x3 & x0;
endmodule
```

1.3.C

Testbench oluşturarak, 16 farklı giriş kombinasyonuna göre çıkış dalga şeklini gözlemlendi.

```

// tb_lab1_g29_p2

`timescale 1ns/1ps
module tb_lab1_g29_p3();
    logic x3, x2, x1, x0;
    logic a, b, c, d, e, f, g;


    lab1_g29_p3 dut0(x3, x2, x1, x0, a, b, c, d, e, f, g);
    initial begin
        x3 = 0; x2 = 0; x1 = 0; x0 = 0; #10;
        x0 = 1; #10;
        x1 = 1; #10;
        x0 = 0; #10;
        x2 = 1; x1 = 0; #10;
        x0 = 1; #10;
        x1 = 1; x0 = 0; #10;
        x0 = 1; #10;
        x3 = 1; x2 = 0; x1 = 0; x0 = 0; #10;
        x0 = 1; #10;
        x1 = 1; #10;
        x0 = 0; #10;
        x2 = 1; x1 = 0; #10;
        x0 = 1; #10;
        x1 = 1; x0 = 0; #10;
        x0 = 1; #10;
        $stop;
    end
endmodule

```

1.3.D

Devre Quartus programında sentezlenmiş ve devrenin ne kadar yer kapladığı, sentezlenen RTL ve eşleştirme ardı devre şemaları gösterilmiştir

Analysis & Synthesis Resource Utilization by Entity

 <<Filter>>

	Compilation Hierarchy Node	Combinational ALUTs	Dedicated Logic Registers	Memory Bits
1	lab1_g29_p3top	7 (0)	0 (0)	0
1	lab1_g29_p3:inst	7 (7)	0 (0)	0

UFM Blocks	DSP Elements	DSP 9x9	DSP 18x18	Pins	Virtual Pins	ADC blocks
0	0	0	0	11	0	0
0	0	0	0	0	0	0

Full Hierarchy Name	Entity Name	Library Name
lab1_g29_p3top	lab1_g29_p3top	work
lab1_g29_p3toplab1_g29_p3:inst	lab1_g29_p3	work

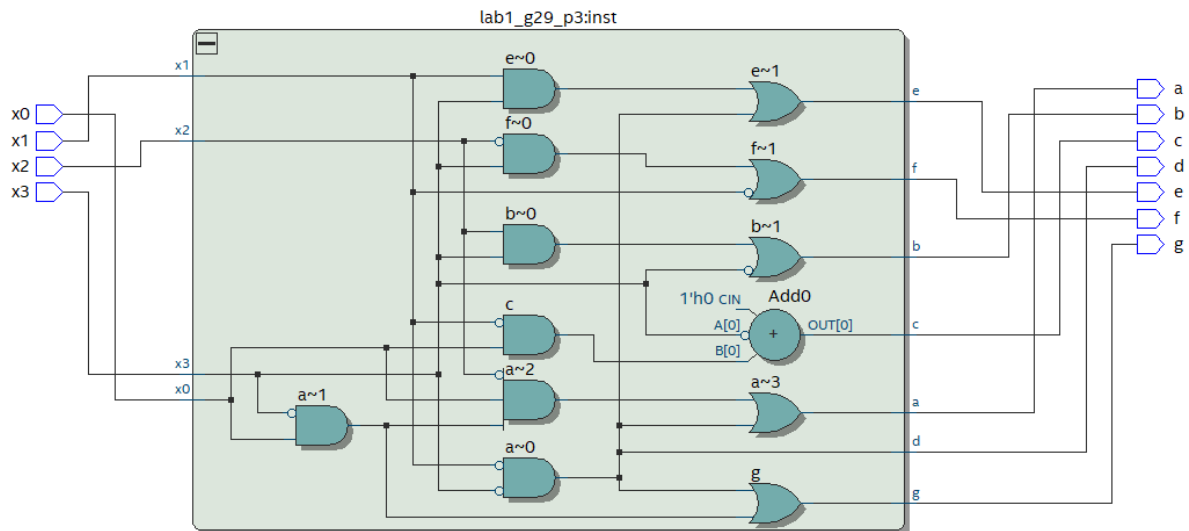
Tablo 5: Resource Utilization Report

Analysis & Synthesis Resource Usage Summary

<<Filter>>

	Resource	Usage
1	Estimated Total logic elements	7
2		
3	Total combinational functions	7
4	▼ Logic element usage by number of LUT inputs	
1	-- 4 input functions	1
2	-- 3 input functions	3
3	-- <=2 input functions	3
5		
6	▼ Logic elements by mode	
1	-- normal mode	7
2	-- arithmetic mode	0
7		
8	▼ Total registers	0
1	-- Dedicated logic registers	0
2	-- I/O registers	0
9		
10	I/O pins	11
11		
12	Embedded Multiplier 9-bit elements	0
13		
14	Maximum fan-out node	x3~input
15	Maximum fan-out	7
16	Total fan-out	37
17	Average fan-out	1.28

Tablo 6: Resource Usage Summary



Şekil 5: Quartus Devre Sentezi

1. Sonular ve Genel Yorumlar

Verilen deney yapılırken ilk olarak boolean denklemi başarılı bir şekilde .sv uzantılı dosyaya geçirilmiş ve sonrasında ise istenen değeri kullanarak yazılan testbench devresi sayesinde devrenin başarılı bir şekilde çalıştığı gözlemlenmiştir. Daha sonraki 2. Soruda ise half adder tasarlanmıştır. Sonrasında tasarlanan half adder kullanılarak full adder tasarlanmıştır. Tasarlanan full adder'dan 4 adet kullanılarak ripple carry adder tasarlanmıştır. İlk soru yapılırken tasarlanan tüm bu devreleri test etmek için yazılan testbench sayesinde hazırlanan bütün devrelerin doğruluğu test edilmiştir. Testbench'te görüldüğü üzere sadece birkaç değeri test edilmiştir çünkü bütün değeri test edilmesi gereksizdir. Generate ve propagate kavramlarını kullanan ripple carry adder'ın "cin" i dışarıya vermesi veya dışarıdan alması testbenchte sayesinde başarılı bir şekilde gerçekleştirildiği gözlemlenmiştir. Aynı şekilde yapılan son testbench değeri yüklemesinde tasarlanan ripple carry adder'ın başarılı bir şekilde 16+16 işlemini (4 bit binary ile gösterilebilecek maksimum sayı) cin = 1 iken başarılı bir şekilde yaptığı gözlemlenmiştir. Son olarak isteri verilen 7 segment display devresinin doğruluk tablosu active-low'a göre yapılmış ve doğruluk tablosundan faydalanılarak her bir diyot için boolean denklemleri bulunmuştur. Bu tablolar daha sonra devre tasarlanırken 1.3B ilk kısımda kod üzerine uygulanmış ve sonrasında yazılan testbench dosyası ile devrenin doğruluğu test edilmiştir. Tüm modelsim çıktıları hazır ve doğru olduğu düşünöldükten sonra quartus üzerinde devre sentezlemeye başlanmıştır. Sentezlenen devrelerin Post-Fitting, RTL şemaları Resource utilization report/summary raporları ekran görüntüsü halinde rapora eklenmiştir.

Referanslar

[1] Intel® Quartus® Prime Software Suite. URL: <https://www.intel.com.tr/content/www/tr/tr/software/programmable/quartus-prime/overview.html> Accessed: 21.02.2020

[2] Quartus Prime Introduction Using Schematic Designs. URL: ftp://ftp.intel.com/Pub/fpgaup/pub/Intel_Material/16.0/Tutorials/Schematic/Quartus_II_Introduction.pdf Accessed: 21.02.2020

[3]Furkan aycı Örnek Kodlar. URL: <https://github.com/fcayci/sv-digital-design>: 28.02.2020

[4]ELM234 Ödev 1 2020-2021