

**DIGITAL SYSTEM DESIGN APPLICATION**

**EHB436E CRN: 11280**

**Salih Ömer Ongün**

**040220780**

**Experiment 4**

**HALF ADDER**

**Design Sources**

Half Adder design source code

**module** HA

**(**

**input** x**,**

**input** y**,**

**output** cout**,**

**output** sum

**);**

**assign** sum **=** x **^** y **;**

**assign** cout **=** x **&** y**;**

**endmodule**

**Simulation Sources**

Half Adder simulation source codes

**module** HA\_tb**();**

**reg** X**=** 1'b0**;**

**reg** Y**=** 1'b0**;**

**wire** COUT**;**

**wire** SUM**;**

HA uut

**(**

**.**x**(**X**),**

**.**y**(**Y**),**

**.**cout**(**COUT**),**

**.**sum**(**SUM**)**

**);**

**initial**

**begin**

X **=** 0**;**

Y **=** 0**;**

**#**10**;**

X **=** 0**;**

Y **=** 1**;**

**#**10**;**

X **=** 1**;**

Y **=** 0**;**

**#**10**;**

X **=** 1**;**

Y **=** 1**;**

**#**10**;**

$finish**;**

**end**

**endmodule**

**Simulation Wave**

Behavioral simulation wave screenshot

ekran görüntüsü, çizgi, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

As seen in the simulation, the module works as it should. Sum bit has logic 1 value if one of two inputs has logic 1 value. Cout has logic 1 value if two bit have logic 1 value.

**FULL ADDER**

**Design Sources**

Full Adder design source code

**module** FA

**(**

**input** x**,**

**input** y**,**

**input** cin**,**

**output** cout**,**

**output** sum

**);**

**wire** ha1\_sum**,** ha1\_cout**,** ha2\_cout**;**

HA half1

**(**

**.**x**(**x**),**

**.**y**(**y**),**

**.**sum**(**ha1\_sum**),**

**.**cout**(**ha1\_cout**)**

**);**

HA half2

**(**

**.**x**(**ha1\_sum**),**

**.**y**(**cin**),**

**.**sum**(**sum**),**

**.**cout**(**ha2\_cout**)**

**);**

OR or1

**(**

**.**l1**(**ha1\_cout**),**

**.**l2**(**ha2\_cout**),**

**.**O**(**cout**)**

**);**

**endmodule**

**Simulation Sources**

Full Adder simulation source codes

**module** FA\_tb**();**

**reg** X**=** 1'b0**;**

**reg** Y**=** 1'b0**;**

**reg** CIN**=** 1'b0**;**

**wire** COUT**;**

**wire** SUM**;**

FA uut

**(**

**.**x**(**X**),**

**.**y**(**Y**),**

**.**cin**(**CIN**),**

**.**cout**(**COUT**),**

**.**sum**(**SUM**)**

**);**

**initial**

**begin**

X **=** 0**;**

Y **=** 0**;**

CIN **=** 0**;**

**#**10**;**

X **=** 0**;**

Y **=** 0**;**

CIN **=** 1**;**

**#**10**;**

X **=** 0**;**

Y **=** 1**;**

CIN **=** 0**;**

**#**10**;**

X **=** 0**;**

Y **=** 1**;**

CIN **=** 1**;**

**#**10**;**

X **=** 1**;**

Y **=** 0**;**

CIN **=** 0**;**

**#**10**;**

X **=** 1**;**

Y **=** 0**;**

CIN **=** 1**;**

**#**10**;**

X **=** 1**;**

Y **=** 1**;**

CIN **=** 0**;**

**#**10**;**

X **=** 1**;**

Y **=** 1**;**

CIN **=** 1**;**

**#**10**;**

$finish**;**

**end**

**endmodule**

**Simulation Wave**

Behavioral simulation wave screenshot

ekran görüntüsü, multimedya yazılımı, grafik yazılımı, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

As seen in the simulation, the module works as it should. Sum bit has logic 1 value if one of three (X, Y, CIN) inputs has logic 1 value. Cout has logic 1 if two of three (X, Y, CIN) inputs has logic 1 value. If all inputs have logic 1 values, then both outputs have logic 1 values.

**RIPPLE CARRY ADDER**

**Design Sources**

Ripple Carry Adder design source code

**module** RCA

**(**

**input** **[**3**:**0**]** x**,**

**input** **[**3**:**0**]** y**,**

**input** cin**,**

**output** cout**,**

**output** **[**3**:**0**]** sum

**);**

**wire** cout1**,**cout2**,**cout3**;**

FA fa1

**(**

**.**x**(**x**[**0**]),**

**.**y**(**y**[**0**]),**

**.**cin**(**cin**),**

**.**sum**(**sum**[**0**]),**

**.**cout**(**cout1**)**

**);**

FA fa2

**(**

**.**x**(**x**[**1**]),**

**.**y**(**y**[**1**]),**

**.**cin**(**cout1**),**

**.**sum**(**sum**[**1**]),**

**.**cout**(**cout2**)**

**);**

FA fa3

**(**

**.**x**(**x**[**2**]),**

**.**y**(**y**[**2**]),**

**.**cin**(**cout2**),**

**.**sum**(**sum**[**2**]),**

**.**cout**(**cout3**)**

**);**

FA fa4

**(**

**.**x**(**x**[**3**]),**

**.**y**(**y**[**3**]),**

**.**cin**(**cout3**),**

**.**sum**(**sum**[**3**]),**

**.**cout**(**cout**)**

**);**

**endmodule**

**Simulation Sources**

Ripple Carry Adder simulation source codes

**module** RCA\_tb**();**

**reg** **[**3**:**0**]** X**=** 4'b0**;**

**reg** **[**3**:**0**]** Y**=** 4'b0**;**

**reg** CIN**=** 1'b0**;**

**wire** COUT**;**

**wire** **[**3**:**0**]** SUM**;**

RCA uut

**(**

**.**x**(**X**),**

**.**y**(**Y**),**

**.**cin**(**CIN**),**

**.**cout**(**COUT**),**

**.**sum**(**SUM**)**

**);**

**initial**

**begin**

X **=** 4'b0000**;**

Y **=** 4'b0000**;**

CIN **=** 0**;**

**#**10**;**

X **=** 4'b0000**;**

Y **=** 4'b0000**;**

CIN **=** 1**;**

**#**10**;**

X **=** 4'b0001**;**

Y **=** 4'b0010**;**

CIN **=** 0**;**

**#**10**;**

X **=** 4'b0011**;**

Y **=** 4'b0011**;**

CIN **=** 0**;**

**#**10**;**

X **=** 4'b0100**;**

Y **=** 4'b0100**;**

CIN **=** 0**;**

**#**10**;**

X **=** 4'b1000**;**

Y **=** 4'b1000**;**

CIN **=** 0**;**

**#**10**;**

X **=** 4'b1001**;**

Y **=** 4'b1011**;**

CIN **=** 0**;**

**#**10**;**

X **=** 4'b1100**;**

Y **=** 4'b1010**;**

CIN **=** 0**;**

**#**10**;**

X **=** 4'b1100**;**

Y **=** 4'b1010**;**

CIN **=** 1**;**

**#**10**;**

X **=** 4'b1111**;**

Y **=** 4'b1111**;**

CIN **=** 1**;**

**#**10**;**

$finish**;**

**end**

**endmodule**

**Simulation Wave**

Behavioral simulation wave screenshot

ekran görüntüsü, multimedya yazılımı, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Ripple Carry Adder has the same working principle as Full Adder. It have three inputs (X, Y, CIN) and two outputs (COUT, SUM). The main difference between the two modules is that Ripple Carry Adder has multiple bit inputs and outputs. Therefore, it uses 4 Full Adder for the module. As seen in the simulation, the module works as it should. If sum of X, Y and CIN have more than 15, COUT has logic 1 values. Because SUM has a 4- bit representation.

**PARAMETRIC RIPPLE CARRY ADDER**

**Design Sources**

Parametric Ripple Carry Adder design source code

**module** parametric\_RCA **#(parameter** SIZE **=** 8**)**

**(**

**input** **[**SIZE**-**1**:**0**]** x**,**

**input** **[**SIZE**-**1**:**0**]** y**,**

**input** cin**,**

**output** cout**,**

**output** **[**SIZE**-**1**:**0**]** sum

**);**

**wire** **[**SIZE**:**0**]** cout\_gen**;**

**assign** cout\_gen**[**0**]** **=** cin**;**

**genvar** i**;**

**generate**

**for(**i **=** 0**;** i**<**SIZE**;** i **=** i**+**1**)** **begin** **:** gen\_full\_adder

FA gen\_full

**(**

x**[**i**],**

y**[**i**],**

cout\_gen**[**i**],**

cout\_gen**[**i**+**1**],**

sum**[**i**]**

**);**

**end**

**endgenerate**

**assign** cout **=** cout\_gen**[**SIZE**];**

**endmodule**

**Simulation Sources**

Parametric Ripple Carry Adder simulation source codes

**module** parametric\_RCA\_tb**();**

**parameter** SIZE **=** 8**;**

**reg** **[**SIZE**-**1**:**0**]** X**;**

**reg** **[**SIZE**-**1**:**0**]** Y**;**

**reg** CIN**;**

**wire** COUT**;**

**wire** **[**SIZE**-**1**:**0**]** SUM**;**

parametric\_RCA **#(.**SIZE**(**SIZE**))** uut

**(**

**.**x**(**X**),**

**.**y**(**Y**),**

**.**cin**(**CIN**),**

**.**cout**(**COUT**),**

**.**sum**(**SUM**)**

**);**

**initial**

**begin**

X **=** 8'b00000000**;**

Y **=** 8'b00000000**;**

CIN **=** 0**;**

**#**10**;**

X **=** 8'b00000000**;**

Y **=** 8'b00000000**;**

CIN **=** 1**;**

**#**10**;**

X **=** 8'b00000001**;**

Y **=** 8'b00000011**;**

CIN **=** 0**;**

**#**10**;**

X **=** 8'b00001000**;**

Y **=** 8'b00001000**;**

CIN **=** 0**;**

**#**10**;**

X **=** 8'b00001100**;**

Y **=** 8'b00001011**;**

CIN **=** 0**;**

**#**10**;**

X **=** 8'b00010000**;**

Y **=** 8'b00011011**;**

CIN **=** 0**;**

**#**10**;**

X **=** 8'b001001100**;**

Y **=** 8'b01001011**;**

CIN **=** 0**;**

**#**10**;**

X **=** 8'b10000000**;**

Y **=** 8'b10000000**;**

CIN **=** 0**;**

**#**10**;**

X **=** 8'b01001100**;**

Y **=** 8'b00101011**;**

CIN **=** 0**;**

**#**10**;**

X **=** 8'b11111111**;**

Y **=** 8'b11111111**;**

CIN **=** 0**;**

**#**10**;**

X **=** 8'b11111111**;**

Y **=** 8'b11111111**;**

CIN **=** 1**;**

**#**10**;**

$finish**;**

**end**

**endmodule**

**Simulation Wave**

ekran görüntüsü, multimedya yazılımı, grafik yazılımı, yazılım içeren bir resim

Açıklama otomatik olarak oluşturulduBehavioral simulation wave screenshot

As seen in the simulation, the module works as it should. The main difference between Ripple Carry Adder and Parametric Ripple Carry Adder is their use of parameters. Therefore, users design any number of bits adder design. SIZE parameter is 8 for the design. If sum of X, Y and CIN have more than 255, COUT has logic 1 values.

**RTL Schematic**

diyagram, metin, çizgi, plan içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Technology Schematic**

metin, diyagram, plan, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Utilization Report**

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

8 LUT’s is used in the design.

**Timing Report**metin, ekran görüntüsü, sayı, numara, menü içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, ekran görüntüsü, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, ekran görüntüsü, sayı, numara, paralel içeren bir resim

Açıklama otomatik olarak oluşturuldu

There are maximum delays between the input paths y and output paths count. The greatest delay is between y[0] and the cout path.

**PARAMETRIC CARRY LOOKAHEAD ADDER**

**Design Sources**

Parametric Carry Lookahead Adder design source code

**module** CLA **#(parameter** SIZE **=** 8**)**

**(**

**input** **[**SIZE**-**1**:**0**]** x**,**

**input** **[**SIZE**-**1**:**0**]** y**,**

**input** cin**,**

**output** cout**,**

**output** **[**SIZE**-**1**:**0**]** s

**);**

**wire** **[**SIZE**:**0**]** p**;**

**wire** **[**SIZE**:**0**]** g**;**

**wire** **[**SIZE**+**1**:**0**]** c**;**

**assign** c**[**0**]** **=** cin**;**

**genvar** i**;**

**genvar** j**;**

**generate**

**for(**i **=** 0**;** i**<**SIZE**;** i **=** i**+**1**)** **begin:** gen\_p\_g

**assign** p**[**i**]** **=** x**[**i**]** **^** y**[**i**];**

**assign** g**[**i**]** **=** x**[**i**]** **&** y**[**i**];**

**end**

**endgenerate**

**generate**

**for(**j **=** 0**;** j**<**SIZE**+**1**;** j **=** j**+**1**)** **begin** **:** gen\_c

**assign** c**[**j**+**1**]** **=** g**[**j**]|(**p**[**j**]** **&** c**[**j**]);**

**end**

**endgenerate**

**assign** cout **=** c**[**SIZE**];**

**generate**

**for(**j **=** 0**;** j**<**SIZE**;** j **=** j**+**1**)** **begin:** gen\_s

**assign** s**[**j**]** **=** p**[**j**]** **^** c**[**j**];**

**end**

**endgenerate**

**endmodule**

**Simulation Sources**

Parametric Carry Lookahead Adder simulation source codes

**module** CLA\_tb**();**

**parameter** SIZE **=** 8**;**

**reg** **[**SIZE**-**1**:**0**]** X**;**

**reg** **[**SIZE**-**1**:**0**]** Y**;**

**reg** CIN**;**

**wire** COUT**;**

**wire** **[**SIZE**-**1**:**0**]** S**;**

CLA **#(.**SIZE**(**SIZE**))** uut

**(**

**.**x**(**X**),**

**.**y**(**Y**),**

**.**cin**(**CIN**),**

**.**cout**(**COUT**),**

**.**s**(**S**)**

**);**

**initial**

**begin**

X **=** 8'b00000000**;**

Y **=** 8'b00000000**;**

CIN **=** 0**;**

**#**10**;**

X **=** 8'b00000000**;**

Y **=** 8'b00000000**;**

CIN **=** 1**;**

**#**10**;**

X **=** 8'b00000001**;**

Y **=** 8'b00000011**;**

CIN **=** 0**;**

**#**10**;**

X **=** 8'b00001000**;**

Y **=** 8'b00001000**;**

CIN **=** 0**;**

**#**10**;**

X **=** 8'b00001100**;**

Y **=** 8'b00001011**;**

CIN **=** 0**;**

**#**10**;**

X **=** 8'b00010000**;**

Y **=** 8'b00011011**;**

CIN **=** 0**;**

**#**10**;**

X **=** 8'b001001100**;**

Y **=** 8'b01001011**;**

CIN **=** 0**;**

**#**10**;**

X **=** 8'b10000000**;**

Y **=** 8'b10000000**;**

CIN **=** 0**;**

**#**10**;**

X **=** 8'b01001100**;**

Y **=** 8'b00101011**;**

CIN **=** 0**;**

**#**10**;**

X **=** 8'b11111111**;**

Y **=** 8'b11111111**;**

CIN **=** 0**;**

**#**10**;**

X **=** 8'b11111111**;**

Y **=** 8'b11111111**;**

CIN **=** 1**;**

**#**10**;**

$finish**;**

**end**

**endmodule**

**Simulation Wave**

ekran görüntüsü, multimedya yazılımı, grafik yazılımı, yazılım içeren bir resim

Açıklama otomatik olarak oluşturulduBehavioral simulation wave screenshot

As seen in the simulation, the module works as it should. There are not any differences between parametric ripple carry adder with parametric carry lookahead adder between results. There are differences in the algorithm for these two adders.

**RTL Schematic**çizgi, diyagram, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, paralel içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, diyagram, plan, şematik içeren bir resim

Açıklama otomatik olarak oluşturuldu**Technology Schematic**

**Utilization Report**

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Timing Report**

metin, ekran görüntüsü, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

There are maximum delays between the input paths y and output paths count. The greatest delay is between y[0] and the cout path. Carry Lookahead Adder calculates all results almost simultaneously. On the contrary, Ripple Carry Adder performs parallel computation, for subsequent computations it must wait for the previous computation to finish. There are not any differences for delay time for this code. However, had their input been greater, perhaps we would have achieved different results.

**BEHAVIORAL ADDER**

**Design Sources**

Behavioral Adder design source code

**module** Behav\_adder **#(parameter** SIZE **=** 8**)**

**(**

**input** **[**SIZE**-**1**:**0**]** x**,**

**input** **[**SIZE**-**1**:**0**]** y**,**

**output** cout**,**

**output** **[**SIZE**-**1**:**0**]** sum

**);**

**assign** **{**cout**,**sum**}** **=** x **+** y**;**

**endmodule**

**Simulation Sources**

Behavioral Adder simulation source codes

**module** Behav\_adder\_tb**();**

**parameter** SIZE **=** 8**;**

**reg** **[**SIZE**-**1**:**0**]** X**;**

**reg** **[**SIZE**-**1**:**0**]** Y**;**

**wire** COUT**;**

**wire** **[**SIZE**-**1**:**0**]** SUM**;**

Behav\_adder **#(.**SIZE**(**SIZE**))** uut

**(**

**.**x**(**X**),**

**.**y**(**Y**),**

**.**cout**(**COUT**),**

**.**sum**(**SUM**)**

**);**

**initial**

**begin**

X **=** 8'b00000000**;**

Y **=** 8'b00000000**;**

**#**10**;**

X **=** 8'b00000000**;**

Y **=** 8'b00000000**;**

**#**10**;**

X **=** 8'b00000001**;**

Y **=** 8'b00000011**;**

**#**10**;**

X **=** 8'b00001000**;**

Y **=** 8'b00001000**;**

**#**10**;**

X **=** 8'b00001100**;**

Y **=** 8'b00001011**;**

**#**10**;**

X **=** 8'b00010000**;**

Y **=** 8'b00011011**;**

**#**10**;**

X **=** 8'b001001100**;**

Y **=** 8'b01001011**;**

**#**10**;**

X **=** 8'b10000000**;**

Y **=** 8'b10000000**;**

**#**10**;**

X **=** 8'b01001100**;**

Y **=** 8'b00101011**;**

**#**10**;**

X **=** 8'b11111111**;**

Y **=** 8'b11111111**;**

**#**10**;**

X **=** 8'b11111111**;**

Y **=** 8'b11111111**;**

**#**10**;**

$finish**;**

**end**

**endmodule**

**Simulation Wave**

Behavioral simulation wave screenshotekran görüntüsü, multimedya yazılımı, grafik yazılımı, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

As seen in the simulation, the module works as it should. There are not any differences between parametric ripple carry adder with behavioral adder between results. There are differences in the algorithm for these two adders.

**RTL Schematic**

diyagram, çizgi, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Technology Schematic**

metin, diyagram, plan, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Utilization Report**

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Timing Report**

metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, sayı, numara, menü içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, menü, ekran görüntüsü, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

There are different algorithms. Vivado includes these algorithms. If we apply any of these algorithms, Vivado will perform the design with the algorithms we applied. If we use arithmetic operators like add (+) for input, Vivado will choose the most optimal algorithm. The algorithm has optimum delay, power usage and resource usage for our code. Therefore, Behavioral Adder has the minimum delays in all paths. It uses fewer I/O ports than other algorithms.

**DONT\_TOUCH Implementation**

Design Source Code

**(\*** DONT\_TOUCH **=** "yes" **\*)**

**module** Behav\_adder **#(parameter** SIZE **=** 8**)**

**(**

**input** **[**SIZE**-**1**:**0**]** x**,**

**input** **[**SIZE**-**1**:**0**]** y**,**

**output** cout**,**

**output** **[**SIZE**-**1**:**0**]** sum

**);**

**assign** **{**cout**,**sum**}** **=** x **+** y**;**

**endmodule**

**Behavioral Adder**

There are not any differences in RTL and Technology Schematics between with and without "DONT\_TOUCH". They have same delays and resource usage. A small number of inputs may have resulted in the same results. If we will have greater input, we can see effect of "DONT\_TOUCH".

metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**RCA**metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturulduWith DONT\_TOCUH

Without DONT\_TOCUH

metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

There are differences between with and without DONT\_TOUCH. Without DONT\_TOUCH implementation have less delays and less resource usage.

**CLA**

Without DONT\_TOUCH

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

With DONT\_TOUCH metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

There aren’t any significant differences in terms of delay for with and without DON’T\_TOUCH CLA implementation. However, there are big differences in terms of resource usage. Without the DON’T\_TOUCH implementation, much less resources are used.

**ADDER – SUBTRACTOR CIRCUIT**

**Design Sources**

Adder – Subtractor Circuit design source code

**module** Add\_Sub

**(**

**input** **[**3**:**0**]** A**,**

**input** **[**3**:**0**]** B**,**

**input** cin**,**

**output** **[**3**:**0**]** sum**,**

**output** cout**,**

**output** overflow

**);**

**wire** **[**3**:**0**]** b\_xor**;**

**wire** **[**4**:**0**]** c\_gen**;**

**assign** c\_gen**[**0**]** **=** cin**;**

**genvar** i**;**

**generate**

**for(**i **=** 0**;** i**<**4**;** i**=**i**+**1**)** **begin:** gen\_xor

**assign** b\_xor**[**i**]** **=** B**[**i**]** **^** cin **;**

**end**

**for(**i **=** 0**;** i**<**4**;** i**=**i**+**1**)** **begin:** gen\_full

FA gen\_full

**(**

A**[**i**],**

b\_xor**[**i**],**

c\_gen**[**i**],**

c\_gen**[**i**+**1**],**

sum**[**i**]**

**);**

**end**

**endgenerate**

**assign** cout **=** c\_gen**[**4**];**

**assign** overflow **=** c\_gen**[**4**]** **^** c\_gen**[**3**];**

**endmodule**

**Simulation Sources**

Adder – Subtractor Circuit simulation source code

**module** Add\_Sub\_tb**();**

**reg** **[**3**:**0**]** A**=** 4'b0**;**

**reg** **[**3**:**0**]** B**=** 4'b0**;**

**reg** CIN**=** 1'b0**;**

**wire** COUT**;**

**wire** **[**3**:**0**]** SUM**;**

**wire** OVERFLOW**;**

Add\_Sub uut

**(**

**.**A**(**A**),**

**.**B**(**B**),**

**.**cin**(**CIN**),**

**.**cout**(**COUT**),**

**.**overflow**(**OVERFLOW**),**

**.**sum**(**SUM**)**

**);**

**integer** i**;**

**integer** k**;**

**integer** j**;**

**initial**

**begin**

**for(**k **=** 0**;** k**<**2**;** k **=** k**+**1**)** **begin**

CIN **=** k**;**

**for(**i **=** **-**8**;** i**<**8**;** i **=** i**+**1**)** **begin**

A **=** i**;**

**for(**j **=** **-**8**;** j**<**8**;** j **=** j**+**1**)** **begin**

B **=** j**;**

**#**5**;**

**end**

**end**

**end**

$finish**;**

**end**

**endmodule**

**RTL Schematic**

diyagram, çizgi, plan, metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Technology Schematic**metin, diyagram, plan, şematik içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Utilization Report**

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Timing Reportmetin, ekran görüntüsü, sayı, numara, menü içeren bir resim

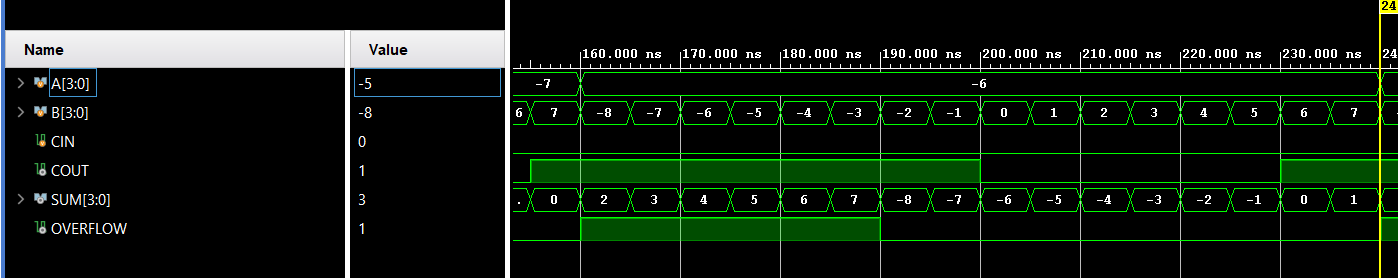
Açıklama otomatik olarak oluşturuldu**

**metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

multimedya yazılımı, yazılım, çizgi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturulduekran görüntüsü, multimedya yazılımı, yazılım, grafik yazılımı içeren bir resim

Açıklama otomatik olarak oluşturulduekran görüntüsü, metin, çizgi, yazılım içeren bir resim

Açıklama otomatik olarak oluşturulduekran görüntüsü, metin, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu**Simulation Waves**

ekran görüntüsü, metin, çizgi içeren bir resim

Açıklama otomatik olarak oluşturulduekran görüntüsü, çizgi, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturulduekran görüntüsü, çizgi, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturulduekran görüntüsü, multimedya yazılımı, yazılım içeren bir resim

Açıklama otomatik olarak oluşturulduekran görüntüsü, multimedya yazılımı, yazılım içeren bir resim

Açıklama otomatik olarak oluşturulduekran görüntüsü, metin, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

ekran görüntüsü, metin, multimedya yazılımı, çizgi içeren bir resim

Açıklama otomatik olarak oluşturulduekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturulduekran görüntüsü, multimedya yazılımı, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldumultimedya yazılımı, yazılım, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, yazılım, multimedya yazılımı, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

ekran görüntüsü, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturulduekran görüntüsü, multimedya yazılımı, yazılım, grafik yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

ekran görüntüsü, multimedya yazılımı, grafik yazılımı, yazılım içeren bir resim

Açıklama otomatik olarak oluşturulduekran görüntüsü, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturulduekran görüntüsü, multimedya yazılımı, grafik yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

ekran görüntüsü, multimedya yazılımı, grafik yazılımı içeren bir resim

Açıklama otomatik olarak oluşturulduekran görüntüsü, multimedya yazılımı, grafik yazılımı içeren bir resim

Açıklama otomatik olarak oluşturulduekran görüntüsü, multimedya yazılımı, yazılım, grafik yazılımı içeren bir resim

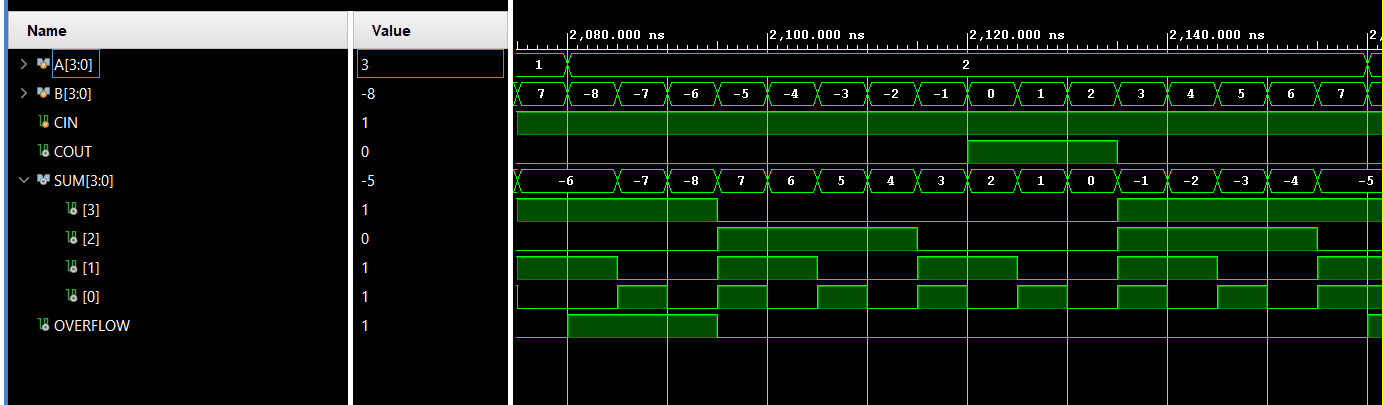
Açıklama otomatik olarak oluşturuldu

ekran görüntüsü, multimedya yazılımı, grafik yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

ekran görüntüsü, multimedya yazılımı, grafik yazılımı, yazılım içeren bir resim

Açıklama otomatik olarak oluşturulduekran görüntüsü, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturulduekran görüntüsü, multimedya yazılımı, yazılım, grafik yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

ekran görüntüsü, multimedya yazılımı, grafik yazılımı, yazılım içeren bir resim

Açıklama otomatik olarak oluşturulduekran görüntüsü, multimedya yazılımı, diyagram içeren bir resim

Açıklama otomatik olarak oluşturulduekran görüntüsü, multimedya yazılımı, grafik yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

**RESEARCH**

1. DSP Block has different tasks in FPGA. One of them multipliers implementation. Especially, DSP Blocks accelerate and facilitate multiplying. Therefore, multiplication and multiplication related operations are performed in DSP Blocks. If operations cannot be performed in a single DSP Blocks, Vivado use different DSP Blocks or use DSP Blocks and Slice logic together. [1]

**Example code for usage DSP Blocks**

**module** DSP\_IMP

**(**

**input** clk**,**

**input** **[**15**:**0**]** a**,**

**input** **[**15**:**0**]** b**,**

**input** **[**15**:**0**]** c**,**

**output** **[**31**:**0**]** result

**);**

**reg** **[**31**:**0**]** result\_reg**;**

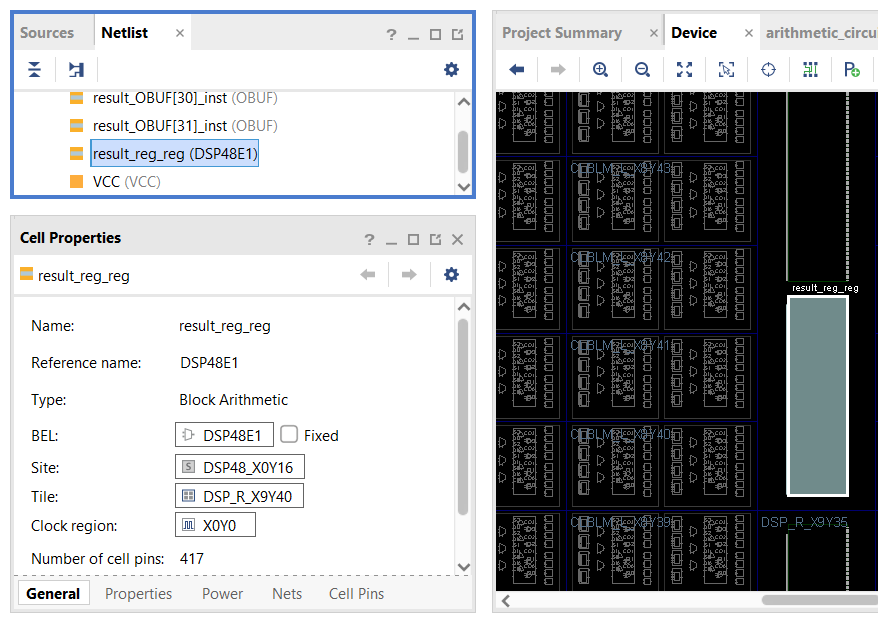
**always** **@(posedge** clk**)** **begin**

result\_reg **<=** **(**a **\*** b**)** **+** c**;**

**end**

**assign** result **=** result\_reg**;**

**endmodule**



**Placement**

DSP Blocks accelerate and facilitate multiplying. They use FPGA resources efficiently. They have optimal results in term of power consumption.

1. Fixed point representation is a method used to represent fractional numbers. It is used in DSP operations and arithmetic operations. It has significant advantages compared with other representations for fractional numbers.

It is represented with Qm.n format.

Q: Format of fixed point. M

m : Number of integer bits

n : Number of fractionalbits

For Q3.5 format, it have 3 bit integer number, 5 bit fractional number. [2]

**FRACTIONAL**

**INTEGER**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 |  | 1 | 1 | 0 | 0 | 1 |

5 is integer part

0.5+0.25+0+0+0.03125=0.78125 is fractional part

101.11001 = 5.78125

Fixed point hardware have less complex than floating points. Therefore, it reduces the size of the chip and results in less power consumption. Generally, fixed-point representation uses fewer bits. Therefore, it requires less memory usage. It may increase the speed of FPGA. The size of the chip and the use of less memory can reduce the price of the FPGA. [3]

**References**

[1] *DSP Block Implementation • VivaDo Design Suite User Guide: Synthesis (UG901) • Reader • AMD Technical Information Portal*. (n.d.). <https://docs.amd.com/r/en-US/ug901-vivado-synthesis/DSP-Block-Implementation>

[2] Harvie, L. (2024, August 22). How to perform Fixed-Point Arithmetic on an FPGA. *RunTime Recruitment*. <https://runtimerec.com/how-to-perform-fixed-point-arithmetic-on-an-fpga/>

[3] *Benefits of Fixed-Point Hardware*. (n.d.). <https://www.mathworks.com/help/fixedpoint/gs/benefits-of-fixed-point-hardware.html>