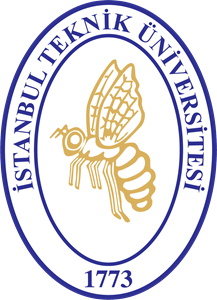
**DIGITAL SYSTEM DESIGN APPLICATIONS**

**(CRN: 11275)**

**THE REPORT OF EXPERIMENT – 4**



Faculty of Electrical and Electronics Engineering

Electronics and Communication Engineering

Yusuf Tekin – 040200043

1. **Half Adder**

Half Adder Verilog Code

**module** HA**(**

**input** x**,**

**input** y**,**

**output** cout**,**

**output** sum

**);**

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

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

**endmodule**

Half Adder Testbench Code

`timescale 1ns **/** 1ps

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

**reg** x**,** y**;**

**wire** cout**,** sum**;**

HA ha1**(**

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

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

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

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

**);**

**initial** **begin**

x **=** 1'b0**;** y **=** 1'b0**;** **#**10**;**

x **=** 1'b0**;** y **=** 1'b1**;** **#**10**;**

x **=** 1'b1**;** y **=** 1'b0**;** **#**10**;**

x **=** 1'b1**;** y **=** 1'b1**;** **#**10**;**

$finish**();**

**end**

**endmodule**

A screenshot of a computer

Description automatically generated

Half Adder Behavioral Simulation

1. **Full Adder**

Full Adder Verilog Code

**module** FA**(**

**input** x**,**

**input** y**,**

**input** cin**,**

**output** cout**,**

**output** sum

**);**

**wire** **[**2**:**0**]** fa\_temp\_wire**;**

HA ha1**(.**x**(**x**),** **.**y**(**y**),** **.**sum**(**fa\_temp\_wire**[**0**]),** **.**cout**(**fa\_temp\_wire**[**1**]));**

HA ha2**(.**x**(**fa\_temp\_wire**[**0**]),** **.**y**(**cin**),** **.**sum**(**sum**),** **.**cout**(**fa\_temp\_wire**[**2**]));**

**assign** cout **=** fa\_temp\_wire**[**1**]** **|** fa\_temp\_wire**[**2**];**

**endmodule**

Full Adder Testbench Code

`timescale 1ns **/** 1ps

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

**wire** cout**,** sum**;**

**reg** x**,** y**,** cin**;**

FA uut**(**

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

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

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

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

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

**);**

**initial** **begin**

x **=** 1'b0**;** y **=** 1'b0**;** cin **=** 1'b0**;** **#**10**;**

x **=** 1'b0**;** y **=** 1'b0**;** cin **=** 1'b1**;** **#**10**;**

x **=** 1'b0**;** y **=** 1'b1**;** cin **=** 1'b0**;** **#**10**;**

x **=** 1'b0**;** y **=** 1'b1**;** cin **=** 1'b1**;** **#**10**;**

x **=** 1'b1**;** y **=** 1'b0**;** cin **=** 1'b0**;** **#**10**;**

x **=** 1'b1**;** y **=** 1'b0**;** cin **=** 1'b1**;** **#**10**;**

x **=** 1'b1**;** y **=** 1'b1**;** cin **=** 1'b0**;** **#**10**;**

x **=** 1'b1**;** y **=** 1'b1**;** cin **=** 1'b1**;** **#**10**;**

$finish**();**

**end**

**endmodule**

A screenshot of a video game

Description automatically generated

Full Adder Behavioral Simulation

1. **Ripple Carry Adder**

RCA Verilog Code

**module** RCA**(**

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

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

**input** cin**,**

**output** cout**,**

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

**);**

**wire** **[**2**:**0**]** rca\_wire**;**

FA fa0**(.**x**(**x**[**0**]),** **.**y**(**y**[**0**]),** **.**cin**(**cin**),** **.**cout**(**rca\_wire**[**0**]),** **.**sum**(**sum**[**0**]));**

FA fa1**(.**x**(**x**[**1**]),** **.**y**(**y**[**1**]),** **.**cin**(**rca\_wire**[**0**]),** **.**cout**(**rca\_wire**[**1**]),** **.**sum**(**sum**[**1**]));**

FA fa2**(.**x**(**x**[**2**]),** **.**y**(**y**[**2**]),** **.**cin**(**rca\_wire**[**1**]),** **.**cout**(**rca\_wire**[**2**]),** **.**sum**(**sum**[**2**]));**

FA fa3**(.**x**(**x**[**3**]),** **.**y**(**y**[**3**]),** **.**cin**(**rca\_wire**[**2**]),** **.**cout**(**cout**),** **.**sum**(**sum**[**3**]));**

**endmodule**

RCA Testbench Code

`timescale 1ns **/** 1ps

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

**wire** cout**;**

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

**reg** **[**3**:**0**]** x**,** y**;**

**reg** cin**;**

RCA uut**(**

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

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

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

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

**.**sum**(**sum**));**

**initial** **begin**

x **=** 4'b0000**;** y **=** 4'b0000**;** cin **=** 1'b0**;** **#**10**;**

x **=** 4'b0001**;** y **=** 4'b0001**;** cin **=** 1'b0**;** **#**10**;**

x **=** 4'b0010**;** y **=** 4'b0010**;** cin **=** 1'b0**;** **#**10**;**

x **=** 4'b0100**;** y **=** 4'b0100**;** cin **=** 1'b0**;** **#**10**;**

x **=** 4'b1000**;** y **=** 4'b1000**;** cin **=** 1'b0**;** **#**10**;**

x **=** 4'b0000**;** y **=** 4'b0000**;** cin **=** 1'b1**;** **#**10**;**

x **=** 4'b0001**;** y **=** 4'b0001**;** cin **=** 1'b1**;** **#**10**;**

x **=** 4'b0010**;** y **=** 4'b0010**;** cin **=** 1'b1**;** **#**10**;**

x **=** 4'b0100**;** y **=** 4'b0100**;** cin **=** 1'b1**;** **#**10**;**

x **=** 4'b1000**;** y **=** 4'b1000**;** cin **=** 1'b1**;** **#**10**;**

$finish**();**

**end**

**endmodule**

A screenshot of a computer

Description automatically generated

RCA Behavioral Simulation

1. **Ripple Carry Adder “parametric”**

Parametric RCA Verilog 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 **-** 2**:**0**]** carry**;**

**genvar** i**;**

**generate**

**for(**i **=** 0**;** i **<** SIZE**;** i **=** i **+** 1**)** **begin** **:** RCA\_generate

**if** **(**i **==** 0**)** **begin**

FA fa\_first**(.**x**(**x**[**i**]),**

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

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

**.**cout**(**carry**[**i**]),**

**.**sum**(**sum**[**i**]));**

**end**

**else** **if** **(**i **==** SIZE **-** 1**)** **begin**

FA fa\_last **(.**x**(**x**[**i**]),**

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

**.**cin**(**carry**[**i**-**1**]),**

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

**.**sum**(**sum**[**i**]));**

**end**

**else** **begin**

FA fa\_middle**(.**x**(**x**[**i**]),**

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

**.**cin**(**carry**[**i**-**1**]),**

**.**cout**(**carry**[**i**]),**

**.**sum**(**sum**[**i**]));**

**end**

**end**

**endgenerate**

**endmodule**

Parametric RCA Testbench Code

`timescale 1ns **/** 1ps

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

**reg** **[**7**:**0**]** x**,** y**;**

**reg** cin**;**

**wire** **[**7**:**0**]** sum**;**

**wire** cout**;**

parametric\_RCA **#(.**SIZE**(**8**))** uut**(**

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

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

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

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

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

**);**

**initial** **begin**

x **=** 8'b0000\_0001**;** y **=** 8'b0000\_1000**;** cin **=** 1'b0**;** **#**10**;**

x **=** 8'b0000\_1100**;** y **=** 8'b0110\_0000**;** cin **=** 1'b0**;** **#**10**;**

x **=** 8'b1110\_0011**;** y **=** 8'b1100\_0001**;** cin **=** 1'b0**;** **#**10**;**

x **=** 8'b0000\_0001**;** y **=** 8'b0000\_1000**;** cin **=** 1'b1**;** **#**10**;**

x **=** 8'b0000\_1100**;** y **=** 8'b0110\_0000**;** cin **=** 1'b1**;** **#**10**;**

x **=** 8'b1110\_0011**;** y **=** 8'b1100\_0001**;** cin **=** 1'b1**;** **#**10**;**

$finish**();**

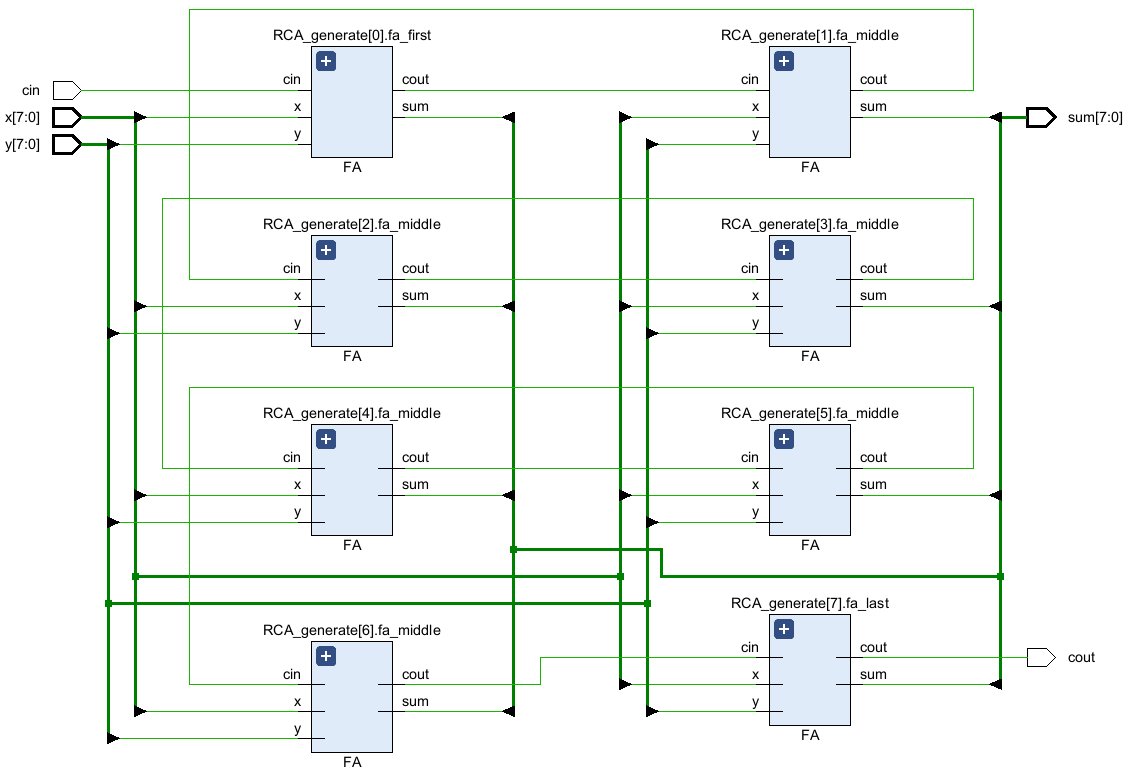
**end**

**endmodule**

A screenshot of a computer

Description automatically generated

Parametric RCA Behavioral Simulation



RTL Schematic - Parametric RCA

A diagram of a computer

Description automatically generated

Technology Schematic - Parametric RCA

A screenshot of a graph

Description automatically generated

Utilization Summary - Parametric RCA

A screenshot of a computer

Description automatically generated

Setup Timing - Parametric RCA

A screenshot of a computer

Description automatically generated

Hold Timing - Parametric RCA

1. **Carry Lookahead Adder “parametric”**

CLA Verilog 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**-**1**:**0**]** wire\_g**,** wire\_p**,** wire\_c**;**

**genvar** i**;**

**generate**

**for** **(**i **=** 0**;** i**<**SIZE**;** i **=** i **+** 1**)** **begin**

**assign** wire\_g**[**i**]** **=** x**[**i**]** **&** y**[**i**];**

**assign** wire\_p**[**i**]** **=** x**[**i**]** **^** y**[**i**];**

**end**

**endgenerate**

**assign** wire\_c**[**0**]** **=** cin**;**

**generate**

**for** **(**i **=** 1**;** i**<**SIZE**;** i **=** i **+** 1**)** **begin**

**assign** wire\_c**[**i**]** **=** wire\_g**[**i**-**1**]** **|** **(**wire\_p**[**i**-**1**]** **&** wire\_c**[**i**-**1**]);**

**end**

**endgenerate**

**generate**

**for** **(**i **=** 0**;** i**<**SIZE**;** i **=** i **+** 1**)** **begin**

**assign** s**[**i**]** **=** wire\_p**[**i**]** **^** wire\_c**[**i**];**

**end**

**endgenerate**

**assign** cout **=** wire\_g**[**SIZE**-**1**]** **|** **(**wire\_p**[**SIZE**-**1**]** **&** wire\_c**[**SIZE**-**1**]);**

**endmodule**

CLA Testbench Code

`timescale 1ns **/** 1ps

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

**wire** **[**7**:**0**]** s**;**

**wire** cout**;**

**reg** cin**;**

**reg** **[**7**:**0**]** x**,** y**;**

CLA **#(.**SIZE**(**8**))** uut**(**

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

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

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

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

**.**s**(**s**));**

**initial** **begin**

x **=** 4'b0000**;** y **=** 4'b0000**;** cin **=** 1'b0**;** **#**10**;**

x **=** 4'b0001**;** y **=** 4'b0001**;** cin **=** 1'b0**;** **#**10**;**

x **=** 4'b0010**;** y **=** 4'b0010**;** cin **=** 1'b0**;** **#**10**;**

x **=** 4'b0100**;** y **=** 4'b0100**;** cin **=** 1'b0**;** **#**10**;**

x **=** 4'b1000**;** y **=** 4'b1000**;** cin **=** 1'b0**;** **#**10**;**

x **=** 4'b0000**;** y **=** 4'b0000**;** cin **=** 1'b1**;** **#**10**;**

x **=** 4'b0001**;** y **=** 4'b0001**;** cin **=** 1'b1**;** **#**10**;**

x **=** 4'b0010**;** y **=** 4'b0010**;** cin **=** 1'b1**;** **#**10**;**

x **=** 4'b0100**;** y **=** 4'b0100**;** cin **=** 1'b1**;** **#**10**;**

x **=** 4'b1000**;** y **=** 4'b1000**;** cin **=** 1'b1**;** **#**10**;**

$finish**();**

**end**

**endmodule**

A screenshot of a computer

Description automatically generated

CLA Behavioral Simulation

A diagram of a computer

Description automatically generated

CLA RTL Schematic

A diagram of a computer

Description automatically generated

CLA Technology Schematic

A screenshot of a computer

Description automatically generated

Setup Delays - CLA

A screenshot of a computer

Description automatically generated

Hold Delays - CLA

A screenshot of a graph

Description automatically generated

Utilization Summary - CLA

1. **Behavioral Adder**

Behavioral Adder Verilog Code

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

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

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

**output** cout**,**

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

**);**

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

**assign** temp **=** x **+** y**;**

**assign** sum **=** temp**[**SIZE**-**1**:**0**];**

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

**endmodule**

Behavioral Adder Testbench Code

`timescale 1ns **/** 1ps

**module** BA\_tb**();**

**wire** **[**7**:**0**]** sum**;**

**wire** cout**;**

**reg** **[**7**:**0**]** x**,** y**;**

BA **#(.**SIZE**(**8**))** uut**(**

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

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

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

**.**sum**(**sum**));**

**initial** **begin**

x **=** 8'b0000\_0000**;** y **=** 8'b0000\_0000**;** **#**10**;**

x **=** 8'b0001\_0001**;** y **=** 8'b0001\_0001**;** **#**10**;**

x **=** 8'b0010\_0010**;** y **=** 8'b0010\_0010**;** **#**10**;**

x **=** 8'b0100\_0100**;** y **=** 8'b0100\_0100**;** **#**10**;**

x **=** 8'b1000\_1001**;** y **=** 8'b1000\_1000**;** **#**10**;**

x **=** 8'b0000\_0100**;** y **=** 8'b0110\_1100**;** **#**10**;**

x **=** 8'b1001\_0001**;** y **=** 8'b0001\_0001**;** **#**10**;**

x **=** 8'b1010\_0010**;** y **=** 8'b0110\_0010**;** **#**10**;**

x **=** 8'b0100\_0100**;** y **=** 8'b0100\_0111**;** **#**10**;**

x **=** 8'b1111\_1111**;** y **=** 8'b1010\_1000**;** **#**10**;**

$finish**();**

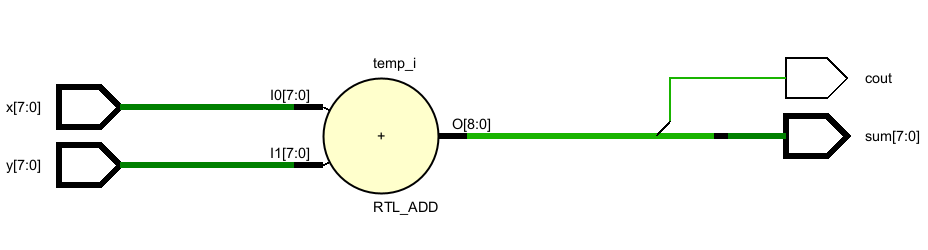
**end**

**endmodule**

A screenshot of a computer

Description automatically generated

Behavioral Adder Behavioral Simulation



Behavioral Adder RTL Schematic

A diagram of a computer

Description automatically generated

Behavioral Adder Technology Schematic

A screenshot of a graph

Description automatically generated

Utilization Summary

A screenshot of a computer

Description automatically generated

Setup Delays - Behavioral Adder

A screenshot of a computer

Description automatically generated

Hold Delays - Behavioral Adder

To compare the behavioral adder with previous adders, it can be seen that: I/O ports are dropped by one because the Cin did not implement to the behavioral adder. Even though all the designs use 8 LUTs, the delays are different. For the RCA and CLA designs, Cout has the longest delay which is around 14ns for both designs. However, that’s not the case for the behavioral design. The values of the delays of the behavioral design are close numbers around 11 ns and all of them are nearly the same.

Both RCA and CLA design uses the LUT5s and LUT3s in the technology implementation however, in the behavioral design LUT2s and CARRY4s are observed. Also, the behavioral design has the simplest RTL schematic of all. RCA has the full-adders and CLA has the basic logic gates in the RTL schematic.

**After “DON’T\_TOUCH” Attribute:**

**A diagram of a circular object with a green line

Description automatically generated**

RTL Schematic of BA (Behavioral Adder)

A diagram of a computer

Description automatically generated

Technology Schematic of BA

A screenshot of a computer

Description automatically generated

Setup Delays - BA

A screenshot of a computer

Description automatically generated

Hold Delays - BA

A screenshot of a graph

Description automatically generated

Utilization Summary - BA

It is observed that there is no change for the behavioral design. This means, Vivado do not apply any optimization to the behavioral design. The optimizations occur after synthesis therefore, any difference cannot be seen in any RTL schematics.

A diagram of a computer scheme

Description automatically generated with medium confidence

Technology Schematic - parametric RCA

A screenshot of a graph

Description automatically generated

Utilization Summary - parametric RCA

A screenshot of a computer

Description automatically generated

Setup Delays - parametric RCA

A screenshot of a computer

Description automatically generated

Hold Delays - parametric RCA

For RCA, the utilization did not change however, technology schematic enlarged by non-optimized full-adders and the setup delays are increased significantly. No difference between RTLs.

A diagram of a computer network

Description automatically generated

Technology Schematic of CLA

A screenshot of a graph

Description automatically generated

Utilization Summary of CLA

A screenshot of a computer

Description automatically generated

Setup Delays - CLA

Both resource usage and delays have increased significantly for the CLA design. This is because of the non-optimization of LUT5s into a LUT2 and a LUT3. The number of LUTs increases therefore the path delays increases.

1. **Adder-Subtractor Circuit**

Add-Sub Verilog Code

**module** Add\_Sub**(**

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

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

**input** cin**,**

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

**output** cout**,**

**output** overflow

**);**

**wire** **[**2**:**0**]** wire\_c**;**

**wire** **[**3**:**0**]** wire\_fa\_b**;**

**genvar** i**;**

**generate**

**for(**i**=**0**;** i**<**4**;** i **=** i **+**1**)** **begin**

**assign** wire\_fa\_b**[**i**]** **=** cin **^** B**[**i**];**

**end**

**endgenerate**

FA fa0**(.**x**(**A**[**0**]),** **.**y**(**wire\_fa\_b**[**0**]),** **.**cin**(**cin**),** **.**cout**(**wire\_c**[**0**]),** **.**sum**(**sum**[**0**]));**

FA fa1**(.**x**(**A**[**1**]),** **.**y**(**wire\_fa\_b**[**1**]),** **.**cin**(**wire\_c**[**0**]),** **.**cout**(**wire\_c**[**1**]),** **.**sum**(**sum**[**1**]));**

FA fa2**(.**x**(**A**[**2**]),** **.**y**(**wire\_fa\_b**[**2**]),** **.**cin**(**wire\_c**[**1**]),** **.**cout**(**wire\_c**[**2**]),** **.**sum**(**sum**[**2**]));**

FA fa3**(.**x**(**A**[**3**]),** **.**y**(**wire\_fa\_b**[**3**]),** **.**cin**(**wire\_c**[**2**]),** **.**cout**(**cout**),** **.**sum**(**sum**[**3**]));**

**assign** overflow **=** wire\_c**[**2**]** **^** cout**;**

**endmodule**

Add-Sub Testbench Code

`timescale 1ns **/** 1ps

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

**wire** cout**,** overflow**;**

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

**reg** **[**3**:**0**]** A**,** B**;**

**reg** cin**;**

Add\_Sub uut**(**

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

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

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

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

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

**.**overflow**(**overflow**));**

**initial** **begin**

A **=** 4'b0000**;** B **=** 4'b0000**;** cin **=** 1'b0**;** **#**10**;**

A **=** 4'b1010**;** B **=** 4'b0101**;** cin **=** 1'b0**;** **#**10**;**

A **=** 4'b0000**;** B **=** 4'b1111**;** cin **=** 1'b0**;** **#**10**;**

A **=** 4'b1100**;** B **=** 4'b1100**;** cin **=** 1'b0**;** **#**10**;**

A **=** 4'b1110**;** B **=** 4'b0011**;** cin **=** 1'b0**;** **#**10**;**

A **=** 4'b0000**;** B **=** 4'b0000**;** cin **=** 1'b1**;** **#**10**;**

A **=** 4'b1010**;** B **=** 4'b0101**;** cin **=** 1'b1**;** **#**10**;**

A **=** 4'b0000**;** B **=** 4'b1111**;** cin **=** 1'b1**;** **#**10**;**

A **=** 4'b1100**;** B **=** 4'b1100**;** cin **=** 1'b1**;** **#**10**;**

A **=** 4'b1110**;** B **=** 4'b0011**;** cin **=** 1'b1**;** **#**10**;**

A **=** 4'b1100**;** B **=** 4'b1111**;** cin **=** 1'b1**;** **#**10**;**

$finish**();**

**end**

**endmodule**

A screenshot of a computer

Description automatically generated

Add-Sub Behavioral Simulation

A diagram of a computer program

Description automatically generated

Add-Sub RTL Schematic

A diagram of a computer

Description automatically generated

Add-Sub Technology Schematic

A screenshot of a graph

Description automatically generated

Add-Sub Utilization Summary

A screenshot of a computer

Description automatically generated

Setup Delays - Add-Sub

A screenshot of a computer

Description automatically generated

Hold Delays - Add-Sub

1. **Research**

**DSP Blocks:** DSP (Digital Signal Processing) blocks are build-in hardware units in FPGAs designed for compute intensive operations like Multiplication, MAC (Multiply-Accumulate), Filtering, FFT (Fast Fourier Transform). However, operands must fit 25x18-bit multipliers in Xilinx 7-Series FPGAs. To be able to use DPS blocks the multiplication **(\*)** operation must be performed. After the synthesis, open “Utilization Report” to look DSP block usage. It can also be seen in the technology schematic.1 Here is a RTL code example:

**module** mult\_unsigned **(**clk**,** A**,** B**,** RES**);**

**parameter** WIDTHA **=** 16**;**

**parameter** WIDTHB **=** 24**;**

**input** clk**;**

**input** **[**WIDTHA**-**1**:**0**]** A**;**

**input** **[**WIDTHB**-**1**:**0**]** B**;**

**output** **[**WIDTHA**+**WIDTHB**-**1**:**0**]** RES**;**

**reg** **[**WIDTHA**-**1**:**0**]** rA**;**

**reg** **[**WIDTHB**-**1**:**0**]** rB**;**

**reg** **[**WIDTHA**+**WIDTHB**-**1**:**0**]** M **[**3**:**0**];**

**integer** i**;**

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

**begin**

rA **<=** A**;**

rB **<=** B**;**

M**[**0**]** **<=** rA **\*** rB**;**

**for** **(**i **=** 0**;** i **<** 3**;** i **=** i**+**1**)**

M**[**i**+**1**]** **<=** M**[**i**];**

**end**

**assign** RES **=** M**[**3**];**

**endmodule**

Multiply-Accumulate (MAC) Operation

**Fixed-Point Representation in FPGA Design:** Fixed-point representation is a way of encoding real numbers as integers, where the position of the decimal point is fixed. This design is commonly used in FPGA designs due to the efficiency in arithmetic operations. In binary 0110.1010 represents 6.625 in decimal. Compared with the floating-point design, fixed point design has a lower power consumption, faster operation speed in simpler operations, minimalization in LUT and DSP usage and easier design/implement. However, it has limited precision which makes it unsuitable for some complex computation and scientific calculations.2

**References**

1. Vivado Design Suite User Guide: Synthesis (UG901), Multipliers, DSP Block Implementations
2. Vivado Design Suite User Guide: Logic Simulation (UG900), Fixed and Floating Point Packages