Expt. No: 3 Biancaa.R

Date: 14.2.2023 2210329

# Design and Synthesis of Carry Lookahead Adders

## Aim:

* To model a 4-bit Carry Lookahead Adder using Dataflow modeling.
* To compile, simulate and plot the results using Xilinx ISE Tools.
* To implement the proposed systems using Xilinx Tools and generate the synthesis report.

## Software used:

Xilinx ISE Tools

## Functional Description: Full Adder:

Truth Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **Carry-in** | **Sum** | **Carry-out** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Boolean Equation: Inputs: A, B & Cin

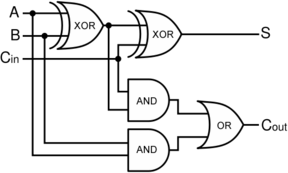
Outputs: Carry generate: Gi= A(i) & B(i) Carry propogate: Pi= A(i) ^ B(i) Sum: Si= P(i) ^ C(i)

Carry output: C(i+1)= G(i)+P(i) &C(i) where i-bit position

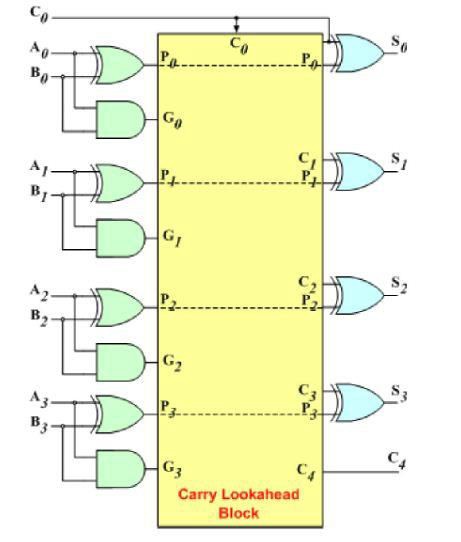
## Full adder: Block Diagram:



**Logic Diagram:**

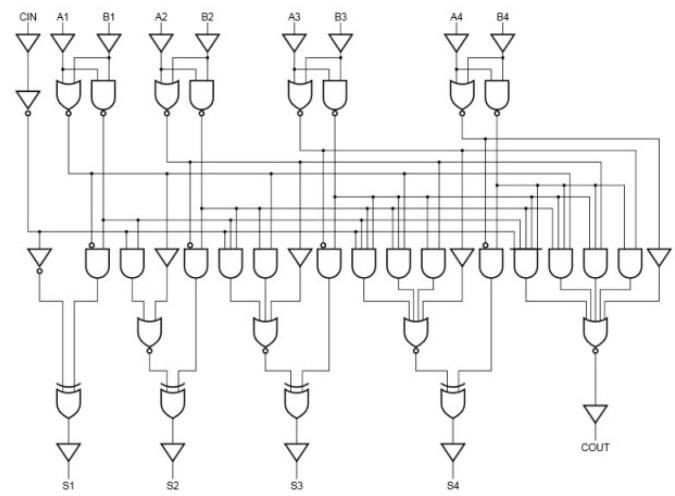


**4-Bit Carry Lookahead Adder:**



**Schematic Diagram:**

**4-Bit Carry Lookahead Adder:**



**Modeling using Verilog HDL: 4-Bit Carry Lookahead Adder: Design module:**

module CLA\_Adder(a,b,cin,sum,cout); input[3:0] a,b;

input cin; output[3:0] sum; output cout; wire[3:0] p,g,c;

assign p[3:0]=a[3:0]^b[3:0];

assign g[3:0]=a[3:0]&b[3:0]; assign c[0]=cin;

assign c[3:1]=g[2:0]|(p[2:0]&c[2:0]); assign cout=c[3];

assign sum[3:0]=p[3:0]^c[3:0];

endmodule **Stimulus file:** module test\_cla;

// Inputs reg [3:0] A;

reg [3:0] B; reg CIN;

// Outputs

wire [3:0] SUM; wire COUT;

// Instantiate the Unit Under Test (UUT) CLA\_Adder uut(A,B,CIN,SUM,COUT);

//Print statement initial begin

$monitor($time," A= %b,B=%b,CIN= %b,COUT= %b, SUM= %b",A, B, CIN, COUT, SUM);

end initial begin

end endmodule

A = 4'd0; B = 4'd0; CIN = 1'b0;

#100 A= 4'd3; B= 4'd4;

#100 A= 4'd3; B= 4'd4;

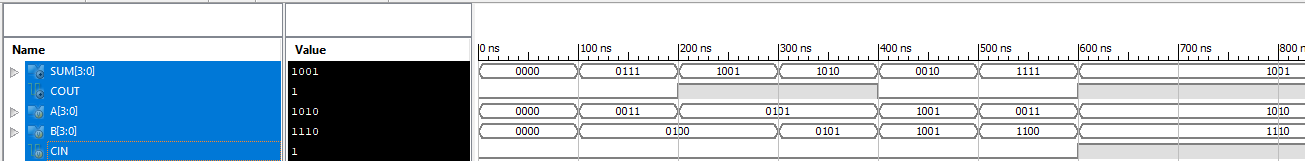
#100 A = 4'd5; B = 4'd5;

#100 A = 4'd9; B = 4'd9;

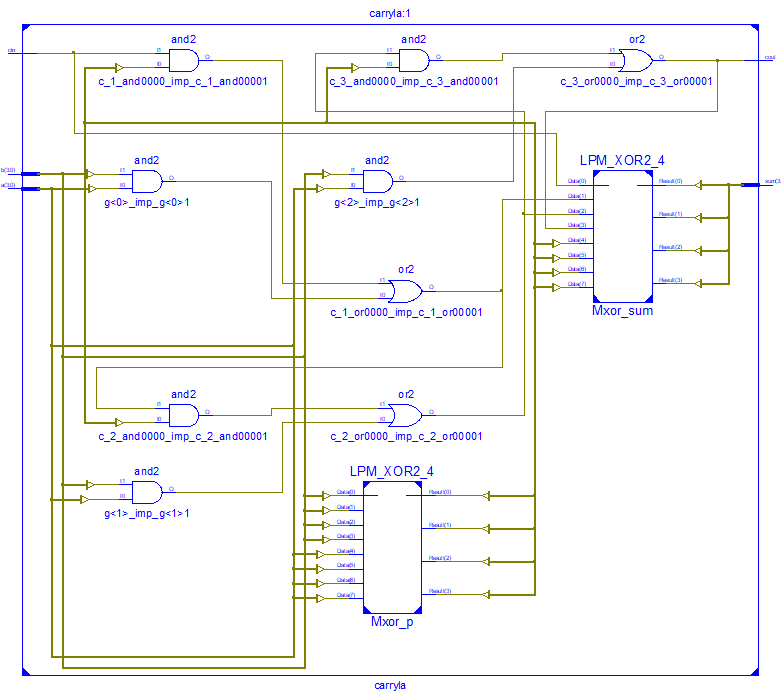
#100 A = 4'd3; B= 4'd12;

#100 A = 4'd10; B = 4'd14; CIN = 1'b1;

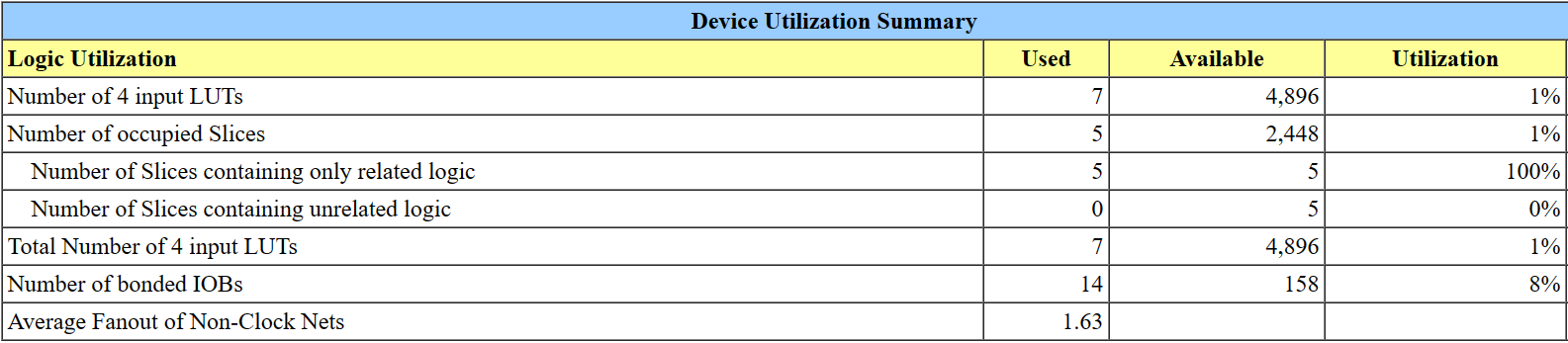
## Simulation Results:



**RTL Schematic Diagram:**



**Design Summary:**



**Result:**

Thus, a model for 4-bit Carry Lookahead Adder using Dataflow modelling was compiled, simulated, synthesized, and implemented.

Expt. No: 1 BIANCAA.R

Date: 07.02.2023 2210329

# Study of Xilinx FPGA Trainer Kit

## Aim:

To Study the Xilinx FPGA Trainer Kit

## Software used:

Xilinx ISE Tools

## Kit Specifications:

* 1. **FPGA Specifications: Hardware:**

Family: Spartan3E Device: XC3S250E Package: PQ208 Speed Grade: -4

## Software:

Synthesis Tool: XST (VHDL/Verilog) Simulator: ISE Simulator (VHDL/Verilog)

## Other hardware:

* 1. **Clock Generation:**

The trainer kit has two clock sources:

* + - Fixed clock of 4MHz connected to PIN No: 181
    - Manual clock by push-to-on switch connected to PIN No: 178. When the key is pressed once, one positive going pulse will be applied to the clock pin of FPGA.

## Input Signal Generation:

The input signal level is generated using DIP switches. The DIP-switch has 8 separate switches. The switch diagram is shown below:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Switch S2 | | | | | | | | |
| Signal | S2-1 | S2-2 | S2-3 | S2-4 | S2-5 | S2-6 | S2-7 | S2-8 |
| Pin | 159 | 169 | 174 | 175 | 183 | 184 | 194 | 204 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Switch S3 | | | | | | | | |
| Signal | S3-1 | S3-2 | S3-3 | S3-4 | S3-5 | S3-6 | S3-7 | S3-8 |
| Pin | 159 | 169 | 174 | 175 | 183 | 184 | 194 | 204 |

When the switch is ON position the output will be ‘0’ level, which is fed to

FPGA

as input. When the switch is OFF position the output of this will be ‘1’ level. The ‘RC’ is used to limit the current, while connecting to ground point.

## Output’s:

The FPGA device outputs are connected to bar-graph LEDs which shows the output level. The output is ‘1’ level the LED will be glowing and when the output is at ‘0’ level the LED will be in off.

## Bi-directional Lines:

The PIN Nos. 106, 107, 108, 109, 112, 113, 115 & 116 of FPGA can be used as bi-directional, in which the output can be viewed at DS7 and the input can be set by switch S1. The circuit diagram of single line is given below:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Switch S1 | | | | | | | | |
| Signal | S1-1 | S1-2 | S1-3 | S1-4 | S1-5 | S1-6 | S1-7 | S1-8 |
| Pin | 106 | 107 | 108 | 109 | 112 | 113 | 115 | 116 |

## Edge Triggered Signals:

PIN Nos. 126, 127 and 128 are connected to push-to-on switches, which generate a positive going pulse.

## Keyboard:

The trainer kit has a 4\*4 key matrix connected to the FPGA I/O lines. The connection details are given below:

|  |  |
| --- | --- |
| Scan Lines | |
| SC0 | 123 |
| SC1 | 122 |
| SC2 | 120 |
| SC3 | 119 |

|  |  |
| --- | --- |
| Return Lines | |
| RT0 | 137 |
| RT1 | 138 |
| RT2 | 139 |
| RT3 | 140 |

## Seven Segment Display:

The trainer kit has 4 digit seven segment displays, which are multiplexed. The connection detail is given below.

|  |  |
| --- | --- |
| 7-segment display select  lines | |
| DS1 | 119 |
| DS2 | 120 |
| DS3 | 122 |
| DS4 | 123 |

|  |  |
| --- | --- |
| Segments | |
| A | 144 |
| B | 145 |
| C | 146 |
| D | 147 |
| E | 150 |
| F | 151 |
| G | 152 |
| DP | 153 |

## LCD:

The trainer kit has one 16\*2 LCD display. The connection details are given below:

|  |  |  |  |
| --- | --- | --- | --- |
| LCD Control Signals | | | |
| Signal | RS | R/W | EN |
| Pin | 55 | 61 | 62 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LCD Data lines | | | | | | | | |
| Signal | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 |
| Pin | 63 | 64 | 65 | 68 | 69 | 76 | 89 | 90 |

RS - LCD Register Select Signal R/W - LCD Read / Write Signal EN - LCD Enable Signal

D7-D0 - LCD Data Lines

## 26-pin FRC Lines:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| FRC Pin | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | | 13 | |
| FPGA  Pin | 2 | 3 | 4 | 5 | 8 | 9 | 11 | 12 | 15 | 16 | 18 | 19 | | 33 | |
| FRC Pin | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | | 26 | |
| FPGA  Pin | 34 | 35 | 36 | 39 | 40 | 41 | 42 | 45 | 47 | 129 | 132 | +5 | V | GN | D |

**FPGA Kit Interfacing Diagram:**

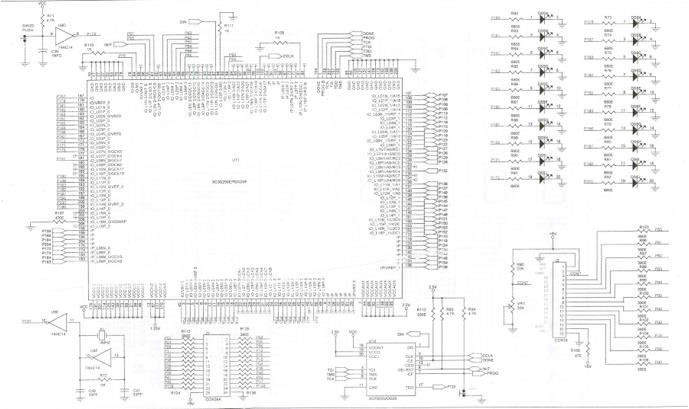


Fig. 1 – FPGA Spartan3E-250K

## XILINX SPARTAN 3E TRAINER KIT SPECIFICATIONS Model No: VSK-SPARTAN 3E

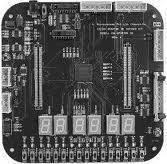


Fig. 2 – FPGA Spartan3E-500K

**Key components and features:** Family: **Xilinx Spartan3E FPGA** Device: **XC3S500E**

Package: **FT256**

Speed Grade: **-4**

16 Nos. of digital input using slide switches

16 Nos. of digital output using discrete LEDs FPGA configuration through

* JTAG port
* Slave serial
* Onboard Flash PROM XCFO4S

Onboard programmable oscillator from 3 MHz to 200 MHz

## 16 Nos. of digital input using slide switches:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input switches | | | | | | | | |
| Switches | SW4 | SW5 | SW6 | SW7 | SW8 | SW9 | SW10 | SW11 |
| FPGA Pin | T14 | T12 | T9 | T7 | T2 | G12 | H1 | R3 |
| Switches | SW12 | SW13 | SW14 | SW15 | SW16 | SW17 | SW18 | SW19 |
| FPGA Pin | N11 | N3 | M13 | M7 | M3 | K4 | J12 | J11 |

**16 Nos. of digital output using discrete LEDs**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Output LEDs | | | | | | | | |
| LEDs | L1 | L2 | L3 | L4 | L5 | L6 | L7 | L8 |
| FPGA Pin | P14 | T13 | R13 | P13 | N12 | N9 | P12 | N10 |
| LEDs | L9 | L10 | L11 | L12 | L13 | L14 | L15 | L16 |
| FPGA Pin | R10 | T8 | R6 | T5 | T4 | K3 | R2 | R1 |

**Result:**

Hence the Xilinx FPGA Trainer Kit has been studied and been familiarized with.

Expt. No: 2 BIANCAA.R

Date: 07.02.2023 2210329

# Design and Synthesis of Ripple Carry Adders

## Aim:

* To model a 4-bit Ripple Carry Adder and an 8-bit Ripple Carry Adder using structural modeling.
* To compile, simulate and plot the results using Xilinx ISE Tools.
* To implement the proposed systems using Xilinx Tools and generate the synthesis report.

## Software used:

Xilinx ISE Tools

## Functional Description: Full Adder:

Truth Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **Carry-in** | **Sum** | **Carry-out** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

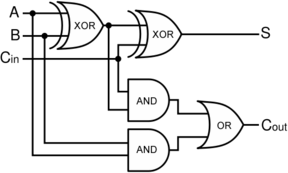
Boolean Equation: Inputs: A, B & Cin

Outputs: Sum: A ^ B ^ Cin

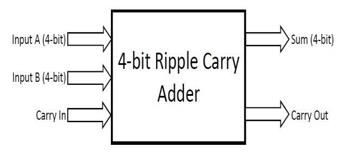
Carry: (A & B) || (B & Cin) || (Cin & A)



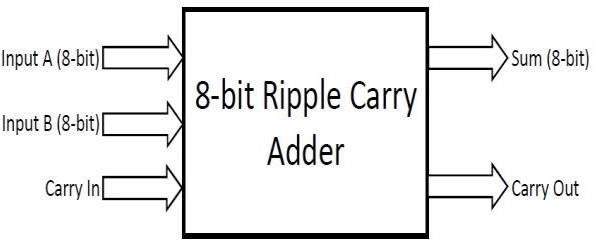
## Full Adder: Logic Diagram:



**4-Bit Ripple Carry Adder:**

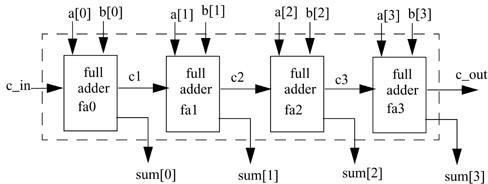


**8-Bit Ripple Carry Adder:**

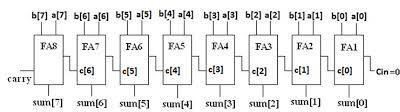


**Schematic Diagram:**

**4-Bit Ripple Carry Adder:**



**8-Bit Ripple Carry Adder:**



**Modelling using Verilog HDL: 4-Bit Ripple Carry Adder: Design module:**

module RCA(sum,cout,a,b,cin); output [3:0] sum; output cout; input [3:0] a,b; input cin;

wire c1,c2,c3;

fulladd f1(sum[0],c1,a[0],b[0],cin);

fulladd f2(sum[1],c2,a[1],b[1],c1);

fulladd f3(sum[2],c3,a[2],b[2],c2);

fulladd f4(sum[3],cout,a[3],b[3],c3); endmodule

module fulladd(sum,cout,a,b,c); output sum,cout; input a,b,c;

assign sum=a^b^c;

assign cout=(a&b)|(b&c)|(c&a); endmodule

## Stimulus file:

module stimulus; reg [3:0] A, B; reg CIN;

wire [3:0] SUM; wire COUT;

RCA RippleCarryAdder(SUM, COUT,A,B,CIN); Initial

Begin

$monitor($time," A= %b, B=%b, CIN= %b, COUT= %b, SUM= %b", A, B, CIN, COUT, SUM);

end

Initial Begin

A = 4'd0; B = 4'd0; CIN = 1'b0;

#10 A = 4'd3; B = 4'd4;

#10 A = 4'd2; B = 4'd5;

#10 A = 4'd9; B = 4'd9;

#10 A = 4'd10; B= 4'd5;

#10 A = 4'd10; B = 4'd5; CIN = 1'b1;

end endmodule

## 8-Bit Ripple Carry Adder: Design module:

module RippleCarryAdder(sum,cout,a,b,cin); output [7:0] sum;

Output cout; Input[7:0] a,b;

wire c1,c2,c3,c4,c5,c6,c7;

fulladd f1(sum[0],c1,a[0],b[0],cin);

fulladd f2(sum[1],c2,a[1],b[1],c1);

fulladd f3(sum[2],c3,a[2],b[2],c2);

fulladd f4(sum[3],c4,a[3],b[3],c3);

fulladd f5(sum[4],c5,a[4],b[4],c4);

fulladd f6(sum[5],c6,a[5],b[5],c5);

fulladd f7(sum[6],c7,a[6],b[6],c6);

fulladd f8(sum[7],cout,a[7],b[7],c7); endmodule

module fulladd (sum,cout,a,b,c); output sum,cout;

input a,b,c;

assign sum=a^b^c;

assign cout=(a&b)|(b&c)|(c&a); endmodule

**Stimulus File:** module stimulus; reg [7:0] A, B;

reg C\_IN;

wire [7:0] SUM; wire C\_OUT;

RippleCarryAdder FA1\_4(SUM, C\_OUT, A, B, C\_IN); initial

begin

$monitor($time," A= %b,B=%b,C\_IN= %b,C\_OUT= %b, SUM= %b",A, B, C\_IN, C\_OUT, SUM);

end

Initial begin

A = 8'd0; B = 8'd0; C\_IN = 1'b0;

#10 A = 8'd3; B = 8'd4;

#10 A = 8'd2; B = 8'd5;

#10 A = 8'd9; B = 8'd9;

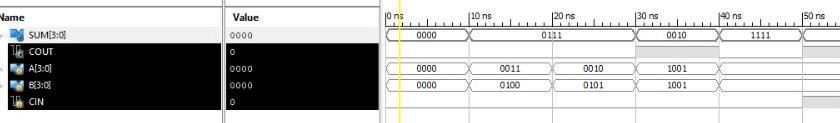
#10 A = 8'd10; B= 8'd5;

#10 A = 8'd10; B = 8'd5; C\_IN = 1'b1;

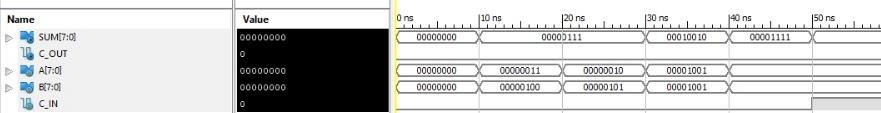
end endmodule

## Simulation Results:

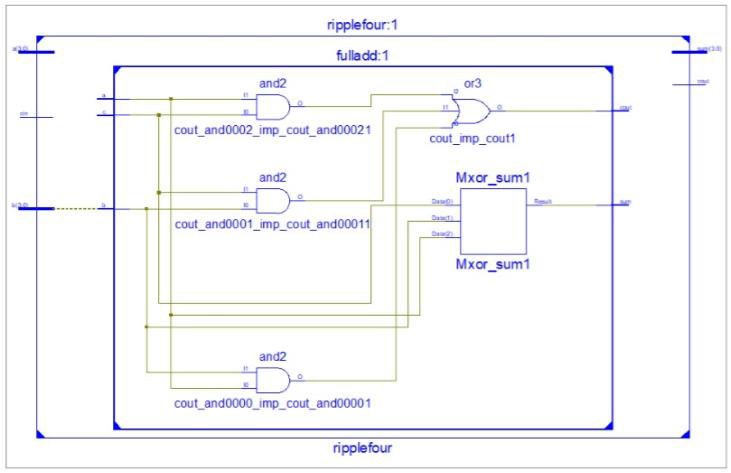
**4-Bit Ripple Carry Adder:**



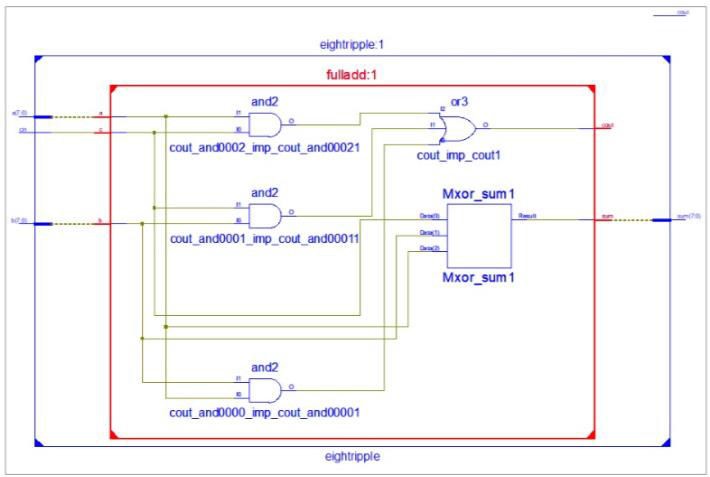
**8-Bit Ripple Carry Adder:**



**RTL Schematic Diagram: 4-Bit Ripple Carry Adder:**

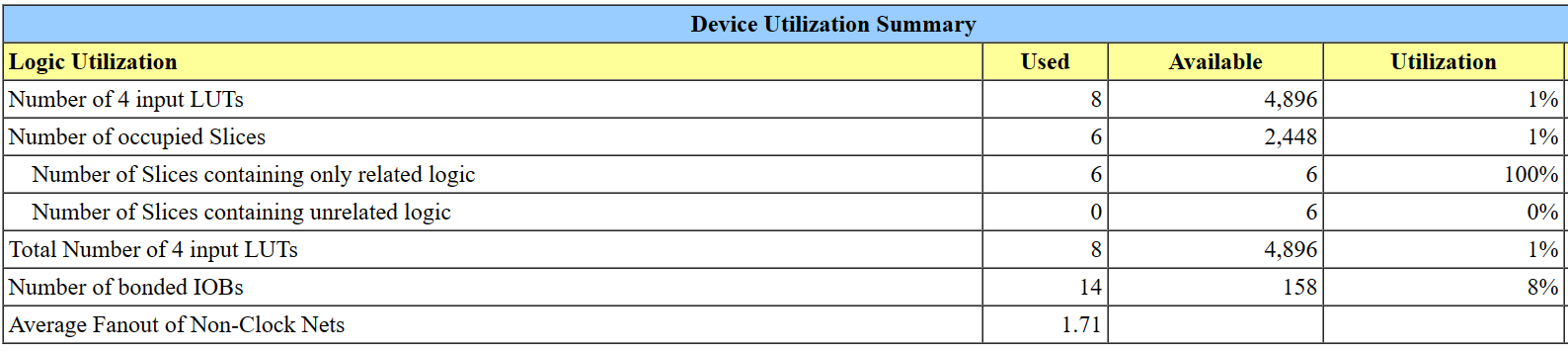


**8-Bit Ripple Carry Adder:**

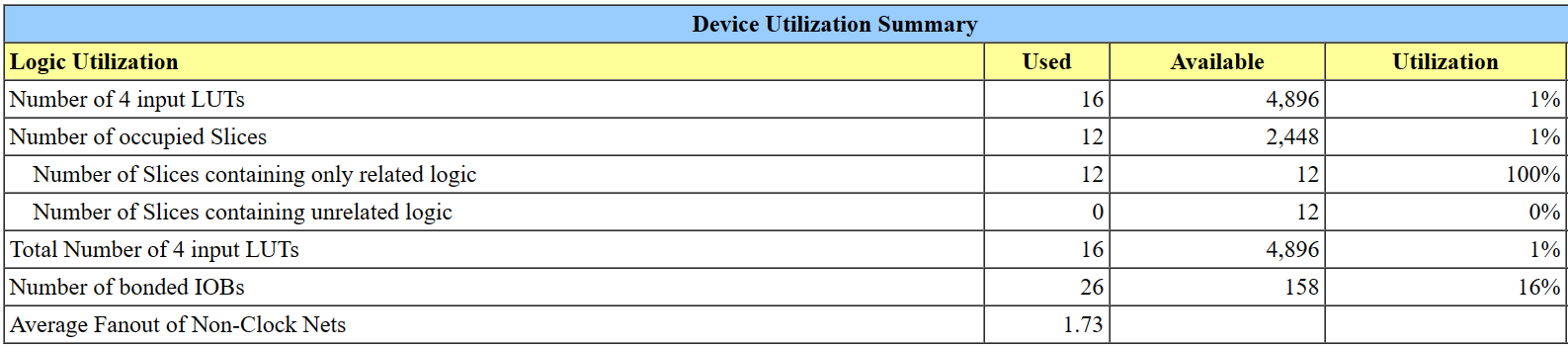


**Design Summary:**

**4-Bit Ripple Carry Adder:**



1. **bit Ripple Carry Adder:**



**Result:**

Thus, a model for Ripple Carry Adder using Structural modelling was compiled, simulated, synthesized, and implemented.

**Date: 13.09.2024**

**DESIGN OF CMOS D-FLIP FLOP**

**AIM:**

To design and implement a D-Flip Flop using Cadence Virtuoso 90nm CMOS technology

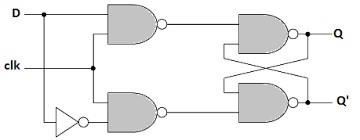
## TOOLS REQUIRED:

Cadence Virtuoso Analog Design Environment

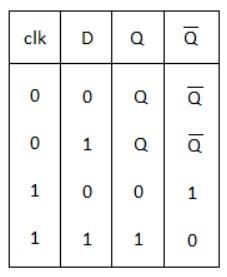
## PROCEDURE:

* + Open Virtuoso and create a new library with existing gpdk090nm technology.
  + File -> New -> Cellview -> Schematic
  + With the help of the circuit diagram, implement the D-Flip Flop using cmos by adding instances and connect the instances using wire
  + Add the source and ground to the required pins
  + Complete the circuit with the wireconnections
  + Launch ->ADE L
  + Choose Transient analysis
    - Set the stop time
    - Click on moderate
  + Select input and output pins from the design by clicking on the corresponding wires
  + Then run the simulation process to get the transient response
  + Thus, the pre layout simulation results are obtained

## CIRCUIT DIAGRAM AND TRUTH TABLE:

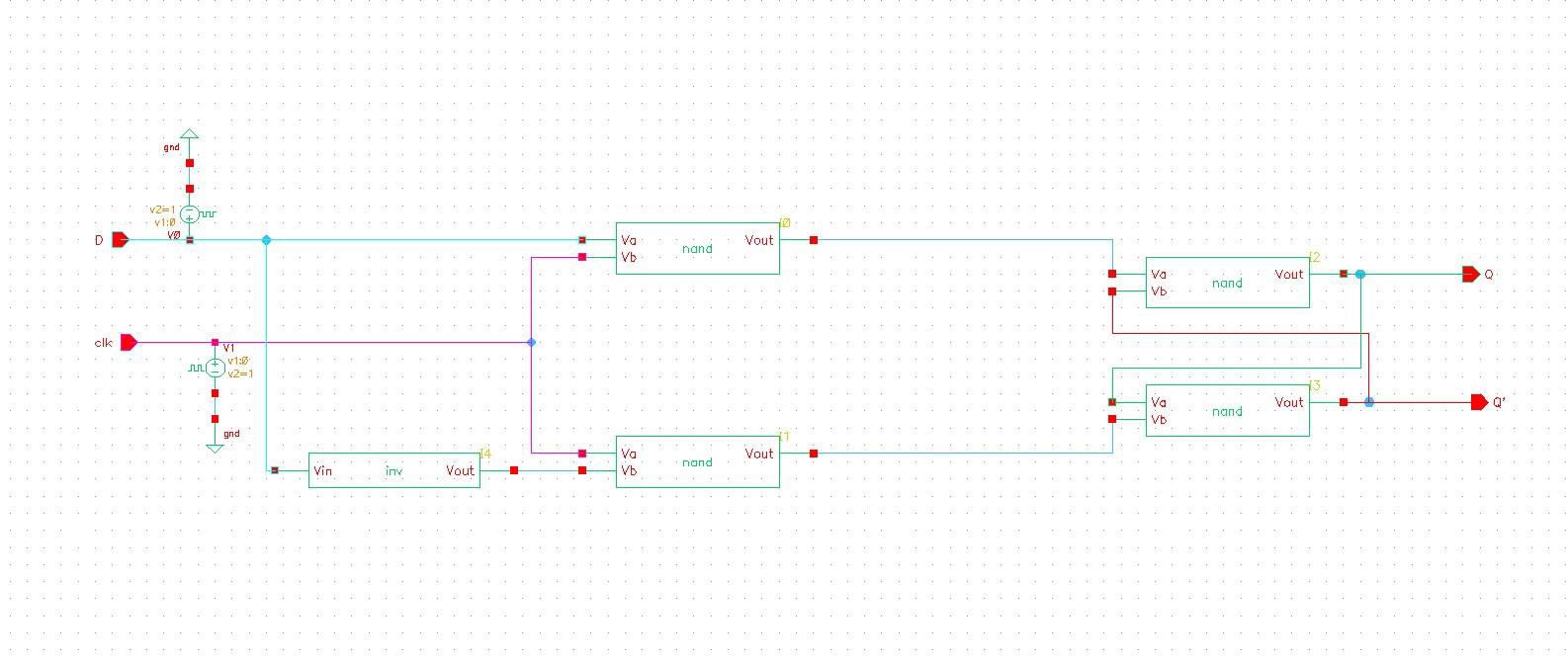


**Fig 4.1 Circuit diagram of CMOS D-Flip Flop**

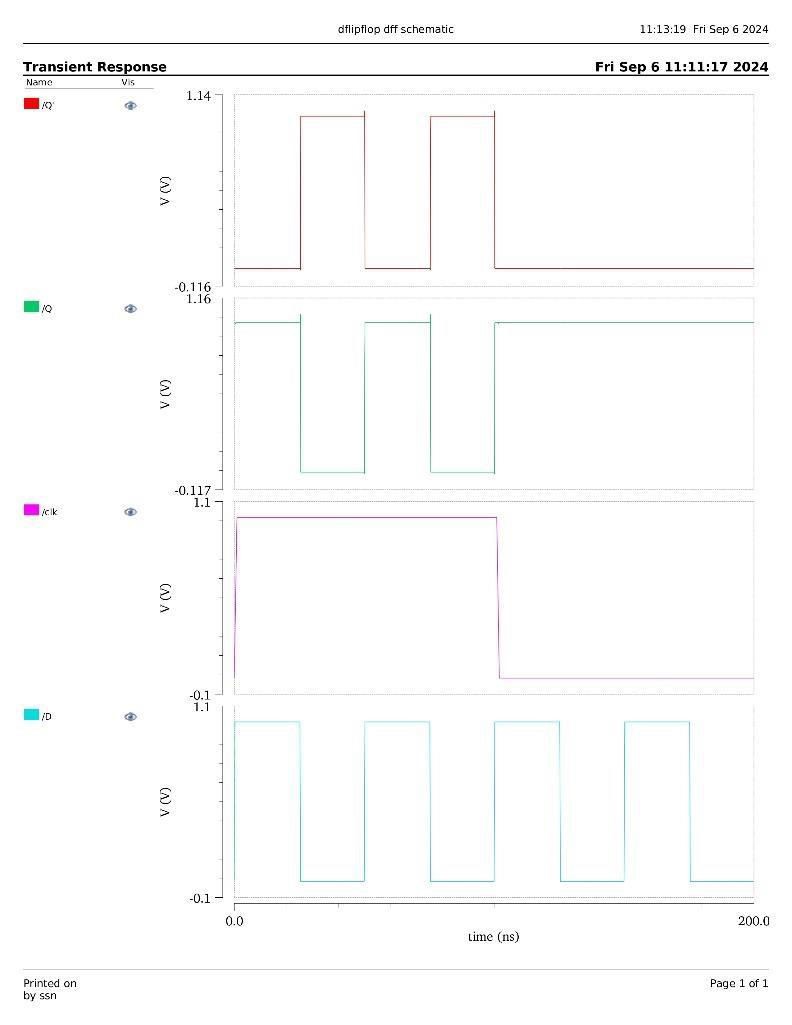


**Table 4.1 Truth Table for D-Flip Flop**

**TRANSIENT ANALYSIS:**



**Fig 4.2 Schematic diagram of CMOS D-Flip Flop**



**Fig 4.3 Transient Analysis of CMOS D-Flip Flop**

**RESULT:**

Thus, the CMOS based D-Flip Flop was implemented and verified using Cadence Virtuoso Analog Design Environment.