COAL LAB PROJECT REPORT

32-Bit Prefix Adder

GROUP MEMBERS:

MOHEEZ CS211254

M. Anas Siddiqui CS211251

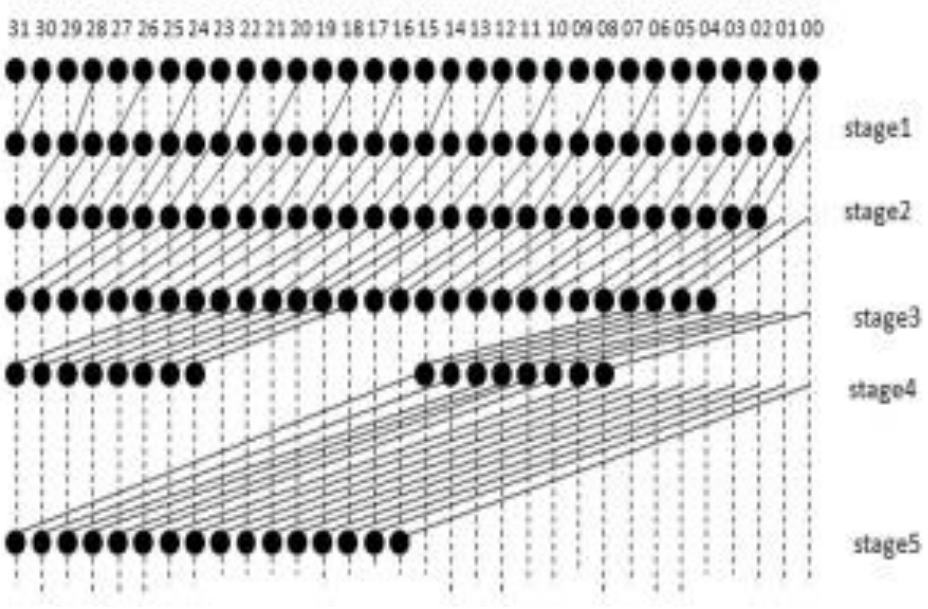
DANIYAL CS211229

Literature Review:

A 32-bit prefix adder is a digital circuit that can perform the addition of two 32-bit binary numbers and generate a 32-bit output. It is typically implemented using a series of full adder circuits, one for each bit of the input numbers. The prefix adder is used to add the carry-out bit from one full adder to the carry-in bit of the next full adder, resulting in a more efficient design. The circuit can also have a carry-lookahead adder which used more advanced technique to perform the addition which will improve the performance of the adder.

Parallel prefix adder is a type of carry look ahead adder structure. It is widely considered as the fastest adder and used for high performance arithmetic circuits in the digital signal processors

Diagram:



Verilog Code:

```
`include "cirmod.v"
`include "dff.v"
`include "dotmod.v"
`include "piped.v"
module pfa32( x[31:0] , y[31:0] , cin , s[31:0] , cout , clk );
        input [31:0]x,y;
        output [31:0]s;
        input cin;
        output cout;
        input clk;

        wire [31:0]p,g,a,pt,gt,at;

        wire [31:0] s1 , s1t , s12 , s12t , s21 , s21t , s22 , s22t , s31 , s31t , s32 , s32t , s41 , s41t , s42 ,
s42t , s51 , s51t;

        assign pt = (x^y);
        assign gt = (x&y);
        assign at = (g|p);

        piped p11( pt , clk , p );
        piped p12( gt , clk , g );
        piped p13( at , clk , a );

        dotmod d11 ( g[0] ,  a[0] ,  cin ,   s1t[0] );
        cirmod c11 ( g[2] ,  a[2] ,  g[1] ,  a[1] ,  s1t[1] ,  s12t[0] );
        cirmod c12 ( g[4] ,  a[4] ,  g[3] ,  a[3] ,  s1t[2] ,  s12t[1] );
        cirmod c13 ( g[6] ,  a[6] ,  g[5] ,  a[5] ,  s1t[3] ,  s12t[2] );
        cirmod c14 ( g[8] ,  a[8] ,  g[7] ,  a[7] ,  s1t[4] ,  s12t[3] );
```

```
cirmod c15 ( g[10] , a[10] , g[9] ,  a[9] ,  s1t[5] ,  s12t[4] );
cirmod c16 ( g[12] , a[12] , g[11] , a[11] , s1t[6] ,  s12t[5] );
cirmod c17 ( g[14] , a[14] , g[13] , a[13] , s1t[7] ,  s12t[6] );
cirmod c18 ( g[16] , a[16] , g[15] , a[15] , s1t[8] ,  s12t[7] );
cirmod c19 ( g[18] , a[18] , g[17] , a[17] , s1t[9] ,  s12t[8] );
cirmod c110( g[20] , a[20] , g[19] , a[19] , s1t[10] , s12t[9] );
cirmod c111( g[22] , a[22] , g[21] , a[21] , s1t[11] , s12t[10] );
cirmod c112( g[24] , a[24] , g[23] , a[23] , s1t[12] , s12t[11] );
cirmod c113( g[26] , a[26] , g[25] , a[25] , s1t[13] , s12t[12] );
cirmod c114( g[28] , a[28] , g[27] , a[27] , s1t[14] , s12t[13] );
cirmod c115( g[30] , a[30] , g[29] , a[29] , s1t[15] , s12t[14] );

//second level

piped p21( s1t  , clk , s1  );
piped p22( s12t , clk , s12 );

dotmod d21 (g[1]   , a[1]    , s1[0]  , s21t[0] );
dotmod d22 (s1[1]  , s12[0]  , s1[0]  , s21t[1] );
cirmod c21 (g[5]   , a[5]    , s1[2]  , s12[1]  , s21t[2]  , s22t[0] );
cirmod c22 (s1[3]  , s12[2]  , s1[2]  , s12[1]  , s21t[3]  , s22t[1] ); //till second level 4/8 circle
mods
cirmod c23 (g[9]   , a[9]    , s1[4]  , s12[3]  , s21t[4]  , s22t[2] );
cirmod c24 (s1[5]  , s12[4]  , s1[4]  , s12[3]  , s21t[5]  , s22t[3] );
cirmod c25 (g[13]  , a[13]   , s1[6]  , s12[5]  , s21t[6]  , s22t[4] );
cirmod c26 (s1[7]  , s12[6]  , s1[6]  , s12[5]  , s21t[7]  , s22t[5] );  //till second level 8 mods
cirmod c27 (g[7]   , a[17]   , s1[8]  , s12[7]  , s21t[8]  , s22t[6] );
cirmod c28 (s1[9]  , s12[8]  , s1[8]  , s12[7]  , s21t[9]  , s22t[7] );
cirmod c29 (g[21]  , a[21]   , s1[10] , s12[9]  , s21t[10] , s22t[8] );
cirmod c210(s1[11] , s12[10] , s1[10] , s12[9]  , s21t[11] , s22t[9] );
```

```
cirmod c211(g[25]  , a[25]   , s1[12] , s12[11] , s21t[12] , s22t[10] );
cirmod c212(s1[13] , s12[12] , s1[12] , s12[11] , s21t[13] , s22t[11] );
cirmod c213(g[29]  , a[29]   , s1[14] , s12[13] , s21t[14] , s22t[12] );
cirmod c214(s1[15] , s12[14] , s1[14] , s12[13] , s21t[15] , s22t[13] );                  //level 2 done

//third level

piped p31( s21t , clk , s21 );
piped p32( s22t , clk , s22 );

dotmod d31 (g[3]    , a[3]    , s21[1]  , s31t[0] );
dotmod d32 (s1[2]   , s12[1]  , s21[1]  , s31t[1] );
dotmod d33 (s21[2]  , s22[0]  , s21[1]  , s31t[2] );
dotmod d34 (s21[3]  , s22[1]  , s21[1]  , s31t[3] ); //for the 8 bit thing given
cirmod c31 (s21[4]  , s22[2]  , g[7]    , a[7]    , s31t[4]  , s32t[0] );
cirmod c32 (s21[5]  , s22[3]  , g[7]    , a[7]    , s31t[5]  , s32t[1] );
cirmod c33 (s21[6]  , s22[4]  , g[11]   , a[11]   , s31t[6]  , s32t[2] );
cirmod c34 (s21[7]  , s22[5]  , g[11]   , a[11]   , s31t[7]  , s32t[3] );
cirmod c35 (g[19]   , a[19]   , s21[9]  , s22[7]  , s31t[8]  , s32t[4] );
cirmod c36 (s1[10]  , s12[9]  , s21[9]  , s22[7]  , s31t[9]  , s32t[5] );
cirmod c37 (s21[10] , s22[8]  , s21[9]  , s22[7]  , s31t[10] , s32t[6] );
cirmod c38 (s21[11] , s22[9]  , s21[9]  , s22[7]  , s31t[11] , s32t[7] );                 //gg
cirmod c39 (g[27]   , a[27]   , s21[13] , s22[11] , s31t[12] , s32t[8] );
cirmod c310(s1[14]  , s12[13] , s21[13] , s22[11] , s31t[13] , s32t[9] );
cirmod c311(s21[14] , s22[12] , s21[13] , s22[11] , s31t[14] , s32t[10] );
cirmod c312(s21[15] , s22[13] , s21[13] , s22[11] , s31t[15] , s32t[11] );                //level 3 done

//fourth level apparently...
//for sure of form dotmod d4x( ,  , s31[3] , s41[x]);
```

```
piped p41( s31t , clk , s31 );
piped p42( s32t , clk , s32 );

dotmod d41(s21[4]  , s22[2]  , s31[3]  , s41t[0] );
dotmod d42(s21[5]  , s22[3]  , s31[3]  , s41t[1] );
dotmod d43(s21[6]  , s22[4]  , s31[3]  , s41t[2] );
dotmod d44(s21[7]  , s22[5]  , s31[3]  , s41t[3] );
dotmod d45(s31[4]  , s32[0]  , s31[3]  , s41t[4] );
dotmod d46(s31[5]  , s32[1]  , s31[3]  , s41t[5] );
dotmod d47(s31[6]  , s32[2]  , s31[3]  , s41t[6] );
dotmod d48(s31[7]  , s32[3]  , s31[3]  , s41t[7] );
cirmod c41(g[23]   , a[23]   , s31[11] , s32[7] , s41t[8]  , s42t[0] );
cirmod c42(s1[12]  , s12[11] , s31[11] , s32[7] , s41t[9]  , s42t[1] );
cirmod c43(s21[12] , s22[10] , s31[11] , s32[7] , s41t[10] , s42t[2] );
cirmod c44(s21[13] , s22[11] , s31[11] , s32[7] , s41t[11] , s42t[3] );
cirmod c45(s31[12] , s32[8]  , s31[11] , s32[7] , s41t[12] , s42t[4] );
cirmod c46(s31[13] , s32[9]  , s31[11] , s32[7] , s41t[13] , s42t[5] );
cirmod c47(s31[14] , s32[10] , s31[11] , s32[7] , s41t[14] , s42t[6] );
cirmod c48(s31[15] , s32[11] , s31[11] , s32[7] , s41t[15] , s42t[7] );                    //level 4 done

//dotmod d49(g[15] , a[15] , s41[7] , cout);

//level 5

piped p51( s41t , clk , s41 );
piped p52( s42t , clk , s42 );

dotmod d51 (g[15]   , a[15]  , s41[7] , s51t[0]);
dotmod d52 (s1[8]   , s12[7] , s41[7] , s51t[1]);
dotmod d53 (s21[8]  , s22[6] , s41[7] , s51t[2]);
```

```
dotmod d54 (s21[9]  , s22[7] , s41[7] , s51t[3]);
dotmod d55 (s31[8]  , s32[4] , s41[7] , s51t[4]);
dotmod d56 (s31[9]  , s32[5] , s41[7] , s51t[5]);
dotmod d57 (s31[10] , s32[6] , s41[7] , s51t[6]);
dotmod d58 (s31[11] , s32[7] , s41[7] , s51t[7]);
dotmod d59 (s41[8]  , s42[0] , s41[7] , s51t[8]);
dotmod d510(s41[9]  , s42[1] , s41[7] , s51t[9]);
dotmod d511(s41[10] , s42[2] , s41[7] , s51t[10]);
dotmod d512(s41[11] , s42[3] , s41[7] , s51t[11]);
dotmod d513(s41[12] , s42[4] , s41[7] , s51t[12]);
dotmod d514(s41[13] , s42[5] , s41[7] , s51t[13]);
dotmod d515(s41[14] , s42[6] , s41[7] , s51t[14]);
dotmod d516(s41[15] , s42[7] , s41[7] , s51t[15]);                    //level 5 done

dotmod d517(g[31] , a[31] , s51[15] , cout);

//last sum generation

piped pfin( s51t , clk , s51 );

assign s[0]  = cin   ^ p[0];
assign s[1]  = p[1]  ^ s1[0];
assign s[2]  = p[2]  ^ s21[0];
assign s[3]  = p[3]  ^ s21[1];
assign s[4]  = p[4]  ^ s31[0];
assign s[5]  = p[5]  ^ s31[1];
assign s[6]  = p[6]  ^ s31[2];
assign s[7]  = p[7]  ^ s31[3];
assign s[8]  = p[8]  ^ s41[0];
assign s[9]  = p[9]  ^ s41[1];
```

```
assign s[10] = p[10] ^ s41[2];
assign s[11] = p[11] ^ s41[3];
assign s[12] = p[12] ^ s41[4];
assign s[13] = p[13] ^ s41[5];
assign s[14] = p[14] ^ s41[6];
assign s[15] = p[15] ^ s41[7];
assign s[16] = p[16] ^ s51[0];
assign s[17] = p[17] ^ s51[1];
assign s[18] = p[18] ^ s51[2];
assign s[19] = p[19] ^ s51[3];
assign s[20] = p[20] ^ s51[4];
assign s[21] = p[21] ^ s51[5];
assign s[22] = p[22] ^ s51[6];
assign s[23] = p[23] ^ s51[7];
assign s[24] = p[24] ^ s51[8];
assign s[25] = p[25] ^ s51[9];
assign s[26] = p[26] ^ s51[10];
assign s[27] = p[27] ^ s51[11];
assign s[28] = p[28] ^ s51[12];
assign s[29] = p[29] ^ s51[13];
assign s[30] = p[30] ^ s51[14];
assign s[31] = p[31] ^ s51[15];

endmodule
```

Modules:

- ## Module cirmod:

```
module cirmod(gl , al , gr , ar , gout , aout);
input gr , ar , gl , al ;
output gout , aout;
```

```
assign gout = gl + (al & gr);

assign aout = al & ar;



endmodule
```

- ## Module DFF:

```
module dff(d , clk , q);
        input d , clk;
        output q;
        reg q;

        initial begin

                q = 0;

        end

        always @(posedge(clk))

                q<=d;

endmodule
```

- ## Module dotmod:

```
module dotmod(gl , al , gr , gout);
        input gl , al , gr;
        output gout;

        assign gout = gl + (al & gr);

endmodule
```

- ## Module piped:

```
module piped(a , clk , b);
        input[31:0] a;
```

```
        input clk;
        output[31:0] b;
        wire[31:0] b;

        dff d0( a[0] , clk , b[0]);
        dff d1( a[1] , clk , b[1]);
        dff d2( a[2] , clk , b[2]);
        dff d3( a[3] , clk , b[3]);
        dff d4( a[4] , clk , b[4]);
        dff d5( a[5] , clk , b[5]);
        dff d6( a[6] , clk , b[6]);
        dff d7( a[7] , clk , b[7]);
        dff d8( a[8] , clk , b[8]);
        dff d9( a[9] , clk , b[9]);
        dff d10(a[10] , clk , b[10]);
        dff d11(a[11] , clk , b[11]);
        dff d12(a[12] , clk , b[12]);
        dff d13(a[13] , clk , b[13]);
        dff d14(a[14] , clk , b[14]);
        dff d15(a[15] , clk , b[15]);
        dff d16(a[16] , clk , b[16]);
        dff d17(a[17] , clk , b[17]);
        dff d18(a[18] , clk , b[18]);
        dff d19(a[19] , clk , b[19]);
        dff d20(a[20] , clk , b[20]);
        dff d21(a[21] , clk , b[21]);
        dff d22(a[22] , clk , b[22]);
        dff d23(a[23] , clk , b[23]);
        dff d24(a[24] , clk , b[24]);
        dff d25(a[25] , clk , b[25]);
        dff d26(a[26] , clk , b[26]);
        dff d27(a[27] , clk , b[27]);
        dff d28(a[28] , clk , b[28]);
        dff d29(a[29] , clk , b[29]);
        dff d30(a[30] , clk , b[30]);
        dff d31(a[31] , clk , b[31]);

endmodule // piped
```

Test bench Code:

```
module pfa32_tb;
```

```
reg[31:0] a , b;
reg cin;

wire[31:0] s;
wire cout;

reg clk;
pfa32 calc( a , b , cin , s , cout , clk );

initial begin

        clk = 0;

end

always begin

        #1
        clk = ~clk;

end

initial begin

        a = 32'b00000000111111110000000011111111;
        b = 32'b11111111000000001111111100000000;
        cin = 0;

        #2
```

```
            a = 32'b11110011111111110000000011111111;
            b = 32'b00001100000000000111111100000000;
            cin = 1;



        end

        always begin

            #48
            $finish;

        end

        initial begin

            $monitor($time , " , a = %b , b = %b , cin = %b , s = %b , cout = %b" , a , b , cin , s ,
cout);

        end
    initial begin
        $dumpfile("dump.vcd");
        $dumpvars(0);
    end

endmodule // pfa32_tb
```

Output Screenshots: