**==================================================================================**

**Original adder:**

`timescale 1ns / 1ps

//------------------ 1) 4-Bit Han-Carlson Adder ------------------

module hancarlson\_4bit(

input [3:0] A,

input [3:0] B,

input cin,

output [3:0] sum,

output cout

);

wire [3:0] g = A & B;

wire [3:0] p = A ^ B;

wire c0 = cin;

wire c1 = g[0] | (p[0] & c0);

wire c2 = g[1] | (p[1] & g[0]) | (p[1] & p[0] & c0);

wire c3 = g[2] | (p[2] & g[1]) | (p[2] & p[1] & g[0]) | (p[2] & p[1] & p[0] & c0);

wire c4 = g[3] | (p[3] & g[2]) | (p[3] & p[2] & g[1]) |

(p[3] & p[2] & p[1] & g[0]) | (p[3] & p[2] & p[1] & p[0] & c0);

assign sum[0] = p[0] ^ c0;

assign sum[1] = p[1] ^ c1;

assign sum[2] = p[2] ^ c2;

assign sum[3] = p[3] ^ c3;

assign cout = c4;

endmodule

//------------------ 2) 4-Bit Ling Adder ------------------

module ling\_4bit(

input [3:0] A,

input [3:0] B,

input cin,

output [3:0] sum,

output cout

);

wire [3:0] g = A & B;

wire [3:0] p = A | B;

wire c0 = cin;

wire c1 = g[0] | (p[0] & c0);

wire c2 = g[1] | (p[1] & g[0]) | (p[1] & p[0] & c0);

wire c3 = g[2] | (p[2] & g[1]) | (p[2] & p[1] & g[0]) | (p[2] & p[1] & p[0] & c0);

wire c4 = g[3] | (p[3] & g[2]) | (p[3] & p[2] & g[1]) |

(p[3] & p[2] & p[1] & g[0]) | (p[3] & p[2] & p[1] & p[0] & c0);

assign sum[0] = A[0] ^ B[0] ^ c0;

assign sum[1] = A[1] ^ B[1] ^ c1;

assign sum[2] = A[2] ^ B[2] ^ c2;

assign sum[3] = A[3] ^ B[3] ^ c3;

assign cout = c4;

endmodule

//------------------ 3) Binary to Excess-1 Converter ------------------

module bec\_4bit(

input [3:0] in,

output [3:0] out

);

assign out = in + 1;

endmodule

//------------------ 4) Weinberger Adder ------------------

module weinberger\_4bit(

input [3:0] A,

input [3:0] B,

input cin,

output [3:0] sum,

output cout

);

wire [3:0] sum0;

wire c1, c2, c3, c4;

wire c0 = 1'b0;

assign sum0[0] = A[0] ^ B[0] ^ c0;

assign c1 = (A[0] & B[0]) | ((A[0] ^ B[0]) & c0);

assign sum0[1] = A[1] ^ B[1] ^ c1;

assign c2 = (A[1] & B[1]) | ((A[1] ^ B[1]) & c1);

assign sum0[2] = A[2] ^ B[2] ^ c2;

assign c3 = (A[2] & B[2]) | ((A[2] ^ B[2]) & c2);

assign sum0[3] = A[3] ^ B[3] ^ c3;

assign c4 = (A[3] & B[3]) | ((A[3] ^ B[3]) & c3);

wire [3:0] sum1;

bec\_4bit U\_bec(.in(sum0), .out(sum1));

wire [3:0] dummy;

wire cc1, cc2, cc3, cc4;

assign dummy[0] = A[0] ^ B[0] ^ 1'b1;

assign cc1 = (A[0] & B[0]) | ((A[0] ^ B[0]) & 1'b1);

assign dummy[1] = A[1] ^ B[1] ^ cc1;

assign cc2 = (A[1] & B[1]) | ((A[1] ^ B[1]) & cc1);

assign dummy[2] = A[2] ^ B[2] ^ cc2;

assign cc3 = (A[2] & B[2]) | ((A[2] ^ B[2]) & cc2);

assign dummy[3] = A[3] ^ B[3] ^ cc3;

assign cc4 = (A[3] & B[3]) | ((A[3] ^ B[3]) & cc3);

assign sum = (cin == 1'b0) ? sum0 : sum1;

assign cout = (cin == 1'b0) ? c4 : cc4;

endmodule

//------------------ 5) Han-Carlson Adder + BEC ------------------

module hancarlson\_bec\_4bit(

input [3:0] A,

input [3:0] B,

input cin,

output [3:0] sum,

output cout

);

wire [3:0] sum0;

wire c0out;

hancarlson\_4bit hc0(

.A(A), .B(B), .cin(1'b0), .sum(sum0), .cout(c0out)

);

wire [3:0] sum1;

bec\_4bit U\_bec(.in(sum0), .out(sum1));

wire [3:0] dummy;

wire d1, d2, d3, d4;

assign dummy[0] = A[0] ^ B[0] ^ 1'b1;

assign d1 = (A[0] & B[0]) | ((A[0] ^ B[0]) & 1'b1);

assign dummy[1] = A[1] ^ B[1] ^ d1;

assign d2 = (A[1] & B[1]) | ((A[1] ^ B[1]) & d1);

assign dummy[2] = A[2] ^ B[2] ^ d2;

assign d3 = (A[2] & B[2]) | ((A[2] ^ B[2]) & d2);

assign dummy[3] = A[3] ^ B[3] ^ d3;

assign d4 = (A[3] & B[3]) | ((A[3] ^ B[3]) & d3);

assign sum = (cin == 1'b0) ? sum0 : sum1;

assign cout = (cin == 1'b0) ? c0out : d4;

endmodule

//------------------ 6) 16-Bit Hybrid Adder ------------------

module hybrid\_16bit\_adder(

input [15:0] A,

input [15:0] B,

input cin,

output [15:0] sum,

output cout

);

wire [3:0] sum0, sum1, sum2, sum3;

wire c0, c1, c2;

hancarlson\_4bit u0(.A(A[3:0]), .B(B[3:0]), .cin(cin), .sum(sum0), .cout(c0));

weinberger\_4bit u1(.A(A[7:4]), .B(B[7:4]), .cin(c0), .sum(sum1), .cout(c1));

weinberger\_4bit u2(.A(A[11:8]), .B(B[11:8]), .cin(c1), .sum(sum2), .cout(c2));

hancarlson\_bec\_4bit u3(.A(A[15:12]), .B(B[15:12]), .cin(c2), .sum(sum3), .cout(cout));

assign sum = {sum3, sum2, sum1, sum0};

Endmodule

Adp 425 - 426 with delay of 11.215 or luts 38  
  
===================================================================================

**Hybrid adder most optimized:**

`timescale 1ns / 1ps

// 4-bit Adder Modules

module brent\_kung\_4bit(

input [3:0] a,

input [3:0] b,

input cin,

output [3:0] sum,

output cout

);

// Brent-Kung implementation

wire [3:0] p, g;

wire [4:0] c;

assign c[0] = cin;

assign p = a ^ b;

assign g = a & b;

// Carry generation

assign c[1] = g[0] | (p[0] & c[0]);

assign c[2] = g[1] | (p[1] & c[1]);

assign c[3] = g[2] | (p[2] & c[2]);

assign c[4] = g[3] | (p[3] & c[3]);

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

assign cout = c[4];

endmodule

module han\_carlson\_4bit(

input [3:0] a,

input [3:0] b,

input cin,

output [3:0] sum,

output cout

);

// Han-Carlson implementation

wire [3:0] p, g;

wire [4:0] c;

assign c[0] = cin;

assign p = a ^ b;

assign g = a & b;

// Prefix tree

assign c[1] = g[0] | (p[0] & c[0]);

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

assign c[3] = g[2] | (p[2] & c[2]);

assign c[4] = g[3] | (p[3] & c[3]);

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

assign cout = c[4];

endmodule

module ling\_4bit(

input [3:0] a,

input [3:0] b,

input cin,

output [3:0] sum,

output cout

);

// Ling adder implementation

wire [3:0] p, g;

wire [4:0] h, c;

assign c[0] = cin;

assign p = a ^ b;

assign g = a & b;

// Ling carry equations

assign h[0] = g[0] | (p[0] & c[0]);

assign h[1] = g[1] | (p[1] & h[0]);

assign h[2] = g[2] | (p[2] & h[1]);

assign h[3] = g[3] | (p[3] & h[2]);

assign c[1] = h[0];

assign c[2] = h[1];

assign c[3] = h[2];

assign c[4] = h[3];

assign sum = p ^ {c[3], c[2], c[1], c[0]};

assign cout = c[4];

endmodule

module kogge\_stone\_4bit(

input [3:0] a,

input [3:0] b,

input cin,

output [3:0] sum,

output cout

);

// Kogge-Stone implementation

wire [3:0] p, g;

wire [4:0] c;

assign c[0] = cin;

assign p = a ^ b;

assign g = a & b;

// Prefix tree

wire [3:0] p1, g1;

wire [3:0] p2, g2;

// First stage

assign g1[0] = g[0];

assign p1[0] = p[0];

assign g1[1] = g[1] | (p[1] & g[0]);

assign p1[1] = p[1] & p[0];

assign g1[2] = g[2] | (p[2] & g[1]);

assign p1[2] = p[2] & p[1];

assign g1[3] = g[3] | (p[3] & g[2]);

assign p1[3] = p[3] & p[2];

// Second stage

assign g2[1] = g1[1];

assign p2[1] = p1[1];

assign g2[2] = g1[2] | (p1[2] & g1[0]);

assign p2[2] = p1[2] & p1[0];

assign g2[3] = g1[3] | (p1[3] & g1[1]);

assign p2[3] = p1[3] & p1[1];

// Carry generation

assign c[1] = g1[0] | (p1[0] & c[0]);

assign c[2] = g2[1] | (p2[1] & c[0]);

assign c[3] = g2[2] | (p2[2] & c[0]);

assign c[4] = g2[3] | (p2[3] & c[0]);

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

assign cout = c[4];

endmodule

// 16-bit Hybrid Adder

module hybrid\_adder\_16bit(

input [15:0] a,

input [15:0] b,

input cin,

output [15:0] sum,

output cout

);

// Internal carry signals

wire c3, c7, c11;

// 0-3: Brent-Kung

brent\_kung\_4bit bk0 (

.a(a[3:0]),

.b(b[3:0]),

.cin(cin),

.sum(sum[3:0]),

.cout(c3)

);

// 4-7: Han-Carlson

han\_carlson\_4bit hc4 (

.a(a[7:4]),

.b(b[7:4]),

.cin(c3),

.sum(sum[7:4]),

.cout(c7)

);

// 8-11: Ling

ling\_4bit ling8 (

.a(a[11:8]),

.b(b[11:8]),

.cin(c7),

.sum(sum[11:8]),

.cout(c11)

);

// 12-15: Kogge-Stone

kogge\_stone\_4bit ks12 (

.a(a[15:12]),

.b(b[15:12]),

.cin(c11),

.sum(sum[15:12]),

.cout(cout)

);

endmodule

**Luts 26 delay 13.215 and adp is 359.17**

**BK + HC+ LING+ KS made booth multiplier:**

`timescale 1ns / 1ps

//======================= MODULES FOR HYBRID ADDER ======================

module brent\_kung\_4bit(

input [3:0] a,

input [3:0] b,

input cin,

output [3:0] sum,

output cout

);

// Brent-Kung implementation

wire [3:0] p, g;

wire [4:0] c;

assign c[0] = cin;

assign p = a ^ b;

assign g = a & b;

// Carry generation

assign c[1] = g[0] | (p[0] & c[0]);

assign c[2] = g[1] | (p[1] & c[1]);

assign c[3] = g[2] | (p[2] & c[2]);

assign c[4] = g[3] | (p[3] & c[3]);

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

assign cout = c[4];

endmodule

module han\_carlson\_4bit(

input [3:0] a,

input [3:0] b,

input cin,

output [3:0] sum,

output cout

);

// Han-Carlson implementation

wire [3:0] p, g;

wire [4:0] c;

assign c[0] = cin;

assign p = a ^ b;

assign g = a & b;

// Prefix tree

assign c[1] = g[0] | (p[0] & c[0]);

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

assign c[3] = g[2] | (p[2] & c[2]);

assign c[4] = g[3] | (p[3] & c[3]);

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

assign cout = c[4];

endmodule

module ling\_4bit(

input [3:0] a,

input [3:0] b,

input cin,

output [3:0] sum,

output cout

);

// Ling adder implementation

wire [3:0] p, g;

wire [4:0] h, c;

assign c[0] = cin;

assign p = a ^ b;

assign g = a & b;

// Ling carry equations

assign h[0] = g[0] | (p[0] & c[0]);

assign h[1] = g[1] | (p[1] & h[0]);

assign h[2] = g[2] | (p[2] & h[1]);

assign h[3] = g[3] | (p[3] & h[2]);

assign c[1] = h[0];

assign c[2] = h[1];

assign c[3] = h[2];

assign c[4] = h[3];

assign sum = p ^ {c[3], c[2], c[1], c[0]};

assign cout = c[4];

endmodule

module kogge\_stone\_4bit(

input [3:0] a,

input [3:0] b,

input cin,

output [3:0] sum,

output cout

);

// Kogge-Stone implementation

wire [3:0] p, g;

wire [4:0] c;

assign c[0] = cin;

assign p = a ^ b;

assign g = a & b;

// Prefix tree

wire [3:0] p1, g1;

wire [3:0] p2, g2;

// First stage

assign g1[0] = g[0];

assign p1[0] = p[0];

assign g1[1] = g[1] | (p[1] & g[0]);

assign p1[1] = p[1] & p[0];

assign g1[2] = g[2] | (p[2] & g[1]);

assign p1[2] = p[2] & p[1];

assign g1[3] = g[3] | (p[3] & g[2]);

assign p1[3] = p[3] & p[2];

// Second stage

assign g2[1] = g1[1];

assign p2[1] = p1[1];

assign g2[2] = g1[2] | (p1[2] & g1[0]);

assign p2[2] = p1[2] & p1[0];

assign g2[3] = g1[3] | (p1[3] & g1[1]);

assign p2[3] = p1[3] & p1[1];

// Carry generation

assign c[1] = g1[0] | (p1[0] & c[0]);

assign c[2] = g2[1] | (p2[1] & c[0]);

assign c[3] = g2[2] | (p2[2] & c[0]);

assign c[4] = g2[3] | (p2[3] & c[0]);

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

assign cout = c[4];

endmodule

// 16-bit Hybrid Adder

module hybrid\_adder\_16bit(

input [15:0] a,

input [15:0] b,

input cin,

output [15:0] sum,

output cout

);

// Internal carry signals

wire c3, c7, c11;

// 0-3: Brent-Kung

brent\_kung\_4bit bk0 (

.a(a[3:0]),

.b(b[3:0]),

.cin(cin),

.sum(sum[3:0]),

.cout(c3)

);

// 4-7: Han-Carlson

han\_carlson\_4bit hc4 (

.a(a[7:4]),

.b(b[7:4]),

.cin(c3),

.sum(sum[7:4]),

.cout(c7)

);

// 8-11: Ling

ling\_4bit ling8 (

.a(a[11:8]),

.b(b[11:8]),

.cin(c7),

.sum(sum[11:8]),

.cout(c11)

);

// 12-15: Kogge-Stone

kogge\_stone\_4bit ks12 (

.a(a[15:12]),

.b(b[15:12]),

.cin(c11),

.sum(sum[15:12]),

.cout(cout)

);

endmodule