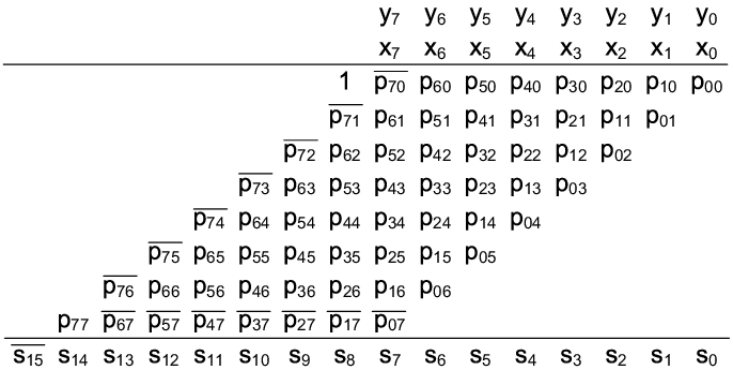
Design and Implementation of Baugh-Wooley Multiplier

Introduction

The Baugh-Wooley multiplication is one amongst the cost-effective ways to handle the sign bits. This method has been developed so as to style regular multipliers, suited to 2's compliment numbers.

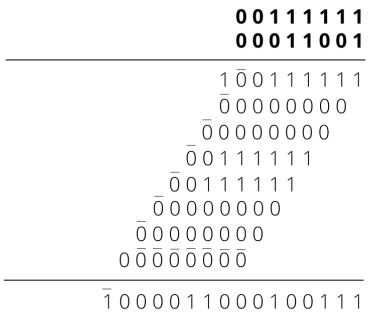
Description

The multiplication proceeds as shown in the description below, where p_{ij} stands for $x_i.y_j$. The required terms that are to be complemented is also shown clearly in this figure



Example

$(63)_{10}$ x $(110)_{10}$ as shown below: wherever a bar is present, the complement is taken



The final answer is $(1000011000100111)_2$ which is 1575.

Implementation

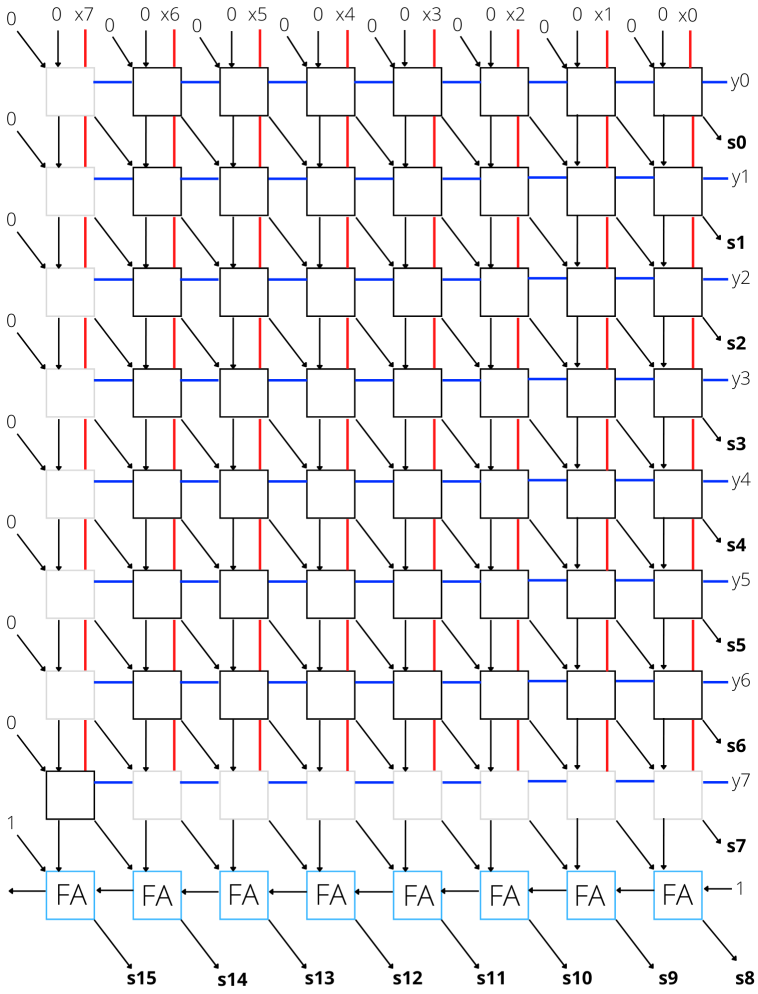


Figure 1: Block diagram of an 8 x 8 Baugh-Wooley multiplier

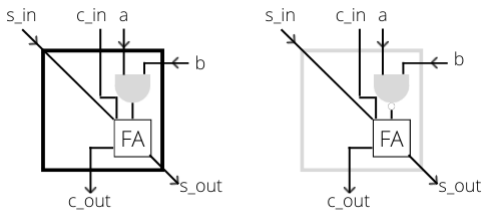


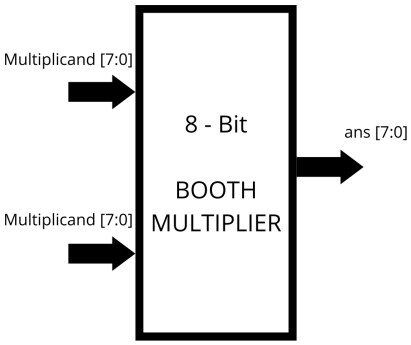
Figure 2: Contents of black and grey colored boxes in the given block diagram

Design Details

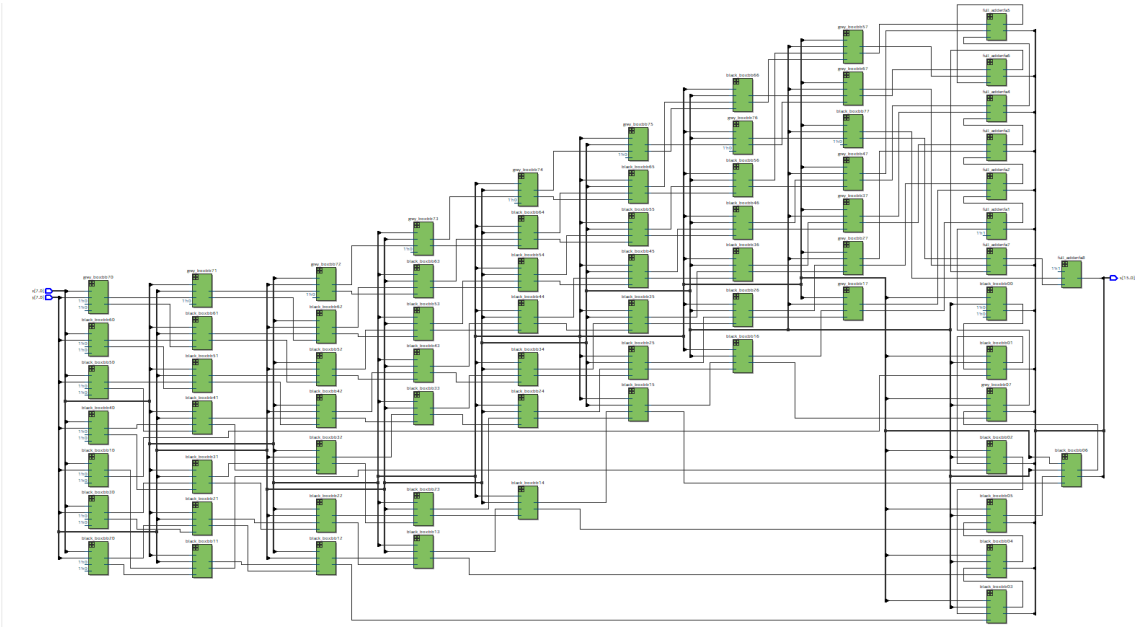
Pin Description Table

Pin/Signal	Direction	Size	Nature
multiplicand[7:0]	Input	8-bit	Signed / unsigned
multiplicand[7:0]	Input	8-bit	Signed / unsigned
ans[15:0]	Output	16-bit	Signed / unsigned

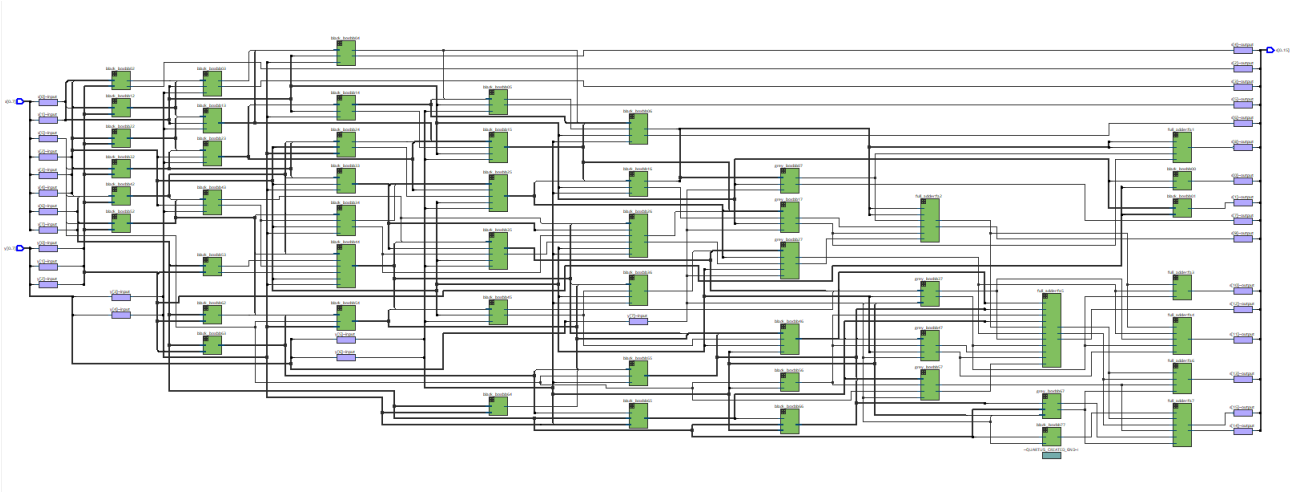
Block Diagram



RTL Design



Post Fitting Design



Testbench Simulation Results

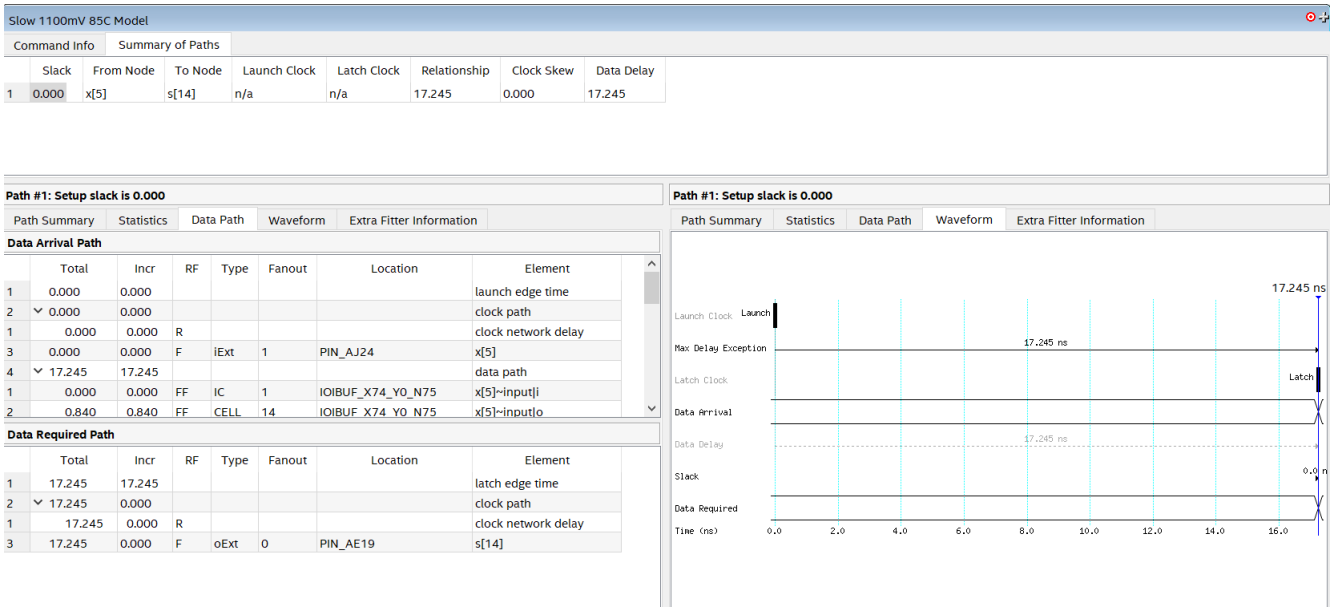
Two cases were considered while testbenching the multiplier in ModelSim Altera.
The first case: x = 00111111 and y = 00011001 (x = 63 and y = 25).
The result came out as 0000011000100111 which is 1575 in decimal.
The second case: x = 00110111 and y = 00011011 (x = 55 and y = 27).
The result came out as 0000010111001101 which is 1485 in decimal.

Figure 3: RTL Simulation

Operating Frequency

The operating frequency is found by using the timing analyzer in Quartus and it has been found out that baugh-wooley multiplier requires an operating frequency of 57.988 MHz.

Path Summary for Maximum Delay



Slow 1100mV 85C Model

Command Info | Summary of Paths

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	0.000	x[5]	s[14]	n/a	n/a	17.245	0.000	17.245

Path #1: Setup slack is 0.000

Path Summary | Statistics | Data Path | Waveform | Extra Fitter Information

Data Arrival Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	0.000	0.000					launch edge time
2	0.000	0.000					clock path
1	0.000	0.000	R				clock network delay
3	0.000	0.000	F	iExt	1	PIN_AJ24	x[5]
4	17.245	17.245					data path
1	0.000	0.000	FF	IC	1	IOIBUF_X74_Y0_N75	x[5]~input|i
2	0.840	0.840	FF	CELL	14	IOIBUF_X74_Y0_N75	x[5]~input|o

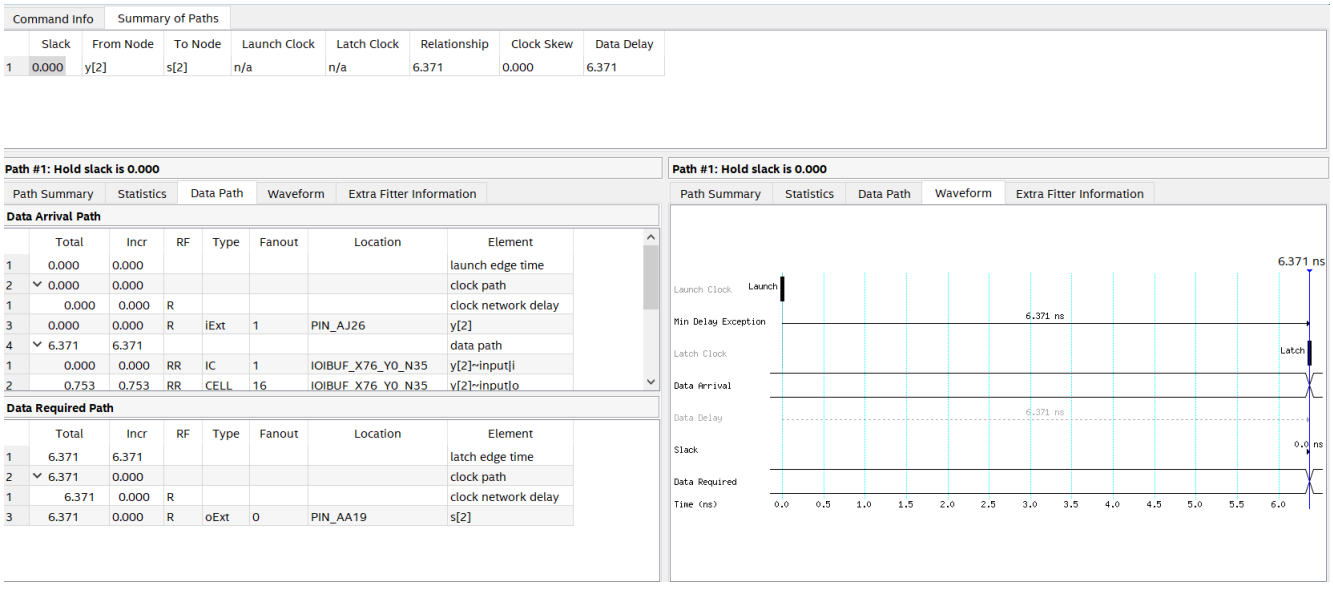
Data Required Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	17.245	17.245					latch edge time
2	17.245	0.000					clock path
1	17.245	0.000	R				clock network delay
3	17.245	0.000	F	oExt	0	PIN_AE19	s[14]

Path #1: Setup slack is 0.000

Path Summary | Statistics | Data Path | Waveform | Extra Fitter Information

Path Summary for Minimum Delay



Command Info | Summary of Paths

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	0.000	y[2]	s[2]	n/a	n/a	6.371	0.000	6.371

Path #1: Hold slack is 0.000

Path Summary | Statistics | Data Path | Waveform | Extra Fitter Information

Data Arrival Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	0.000	0.000					launch edge time
2	0.000	0.000					clock path
1	0.000	0.000	R				clock network delay
3	0.000	0.000	R	iExt	1	PIN_AJ26	y[2]
4	6.371	6.371					data path
1	0.000	0.000	RR	IC	1	IOIBUF_X76_Y0_N35	y[2]~input|i
2	0.753	0.753	RR	CELL	16	IOIBUF_X76_Y0_N35	y[2]~input|o

Data Required Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	6.371	6.371					latch edge time
2	6.371	0.000					clock path
1	6.371	0.000	R				clock network delay
3	6.371	0.000	R	oExt	0	PIN_AA19	s[2]

Path #1: Hold slack is 0.000

Path Summary | Statistics | Data Path | Waveform | Extra Fitter Information

Power Analysis

Power Analyzer Summary

<<Filter>>

Power Analyzer Status	Successful - Sat Jun 11 00:19:19 2022
Quartus Prime Version	18.0.0 Build 614 04/24/2018 SJ Lite Edition
Revision Name	baugh_wooley
Top-level Entity Name	baugh_wooley
Family	Cyclone V
Device	5CSEMA5F31C6
Power Models	Final
Total Thermal Power Dissipation	420.48 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	411.23 mW
I/O Thermal Power Dissipation	9.25 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

Conclusion

The flow summary for the Baugh-Wooley Multiplier can be seen below:

Flow Summary

<<Filter>>

Flow Status	Successful - Sat Jun 11 00:21:33 2022
Quartus Prime Version	18.0.0 Build 614 04/24/2018 SJ Lite Edition
Revision Name	baugh_wooley
Top-level Entity Name	baugh_wooley
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	76 / 32,070 (< 1 %)
Total registers	0
Total pins	32 / 457 (7 %)
Total virtual pins	0
Total block memory bits	0 / 4,065,280 (0 %)
Total DSP Blocks	0 / 87 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 4 (0 %)

Performance summary:

Power consumed	Area Used	Maximum Delay	Minimum Delay	Operating Frequency
420.48 mW	76/32070(1%) logic elements	17.245 ns	6.371 ns	57.988 MHz

Appendix

Verilog code

```
// First defining the full adder component that will be used in the circuit
module full_adder(a, b, cin, s, cout);

  input a, b, cin;
  output s, cout;

  assign s = a ^ b ^ cin;
  assign cout = (a&b) | (b&cin) | (cin&a);

endmodule

// Defining black block as shown in the circuit diagram
module black_box(sin, cin, a, b, sout, cout);

  input a, b, cin, sin;
  output cout, sout;

  full_adder f(a&b, sin, cin, sout, cout);

endmodule

// Defining grey block as shown in the circuit diagram
module grey_box(sin, cin, a, b, sout, cout);

  input a, b, cin, sin;
  output cout, sout;

  full_adder f(!(a&b), sin, cin, sout, cout);

endmodule

module baugh_wooley(x, y, s);

  input [7:0] x, y;
  output [15:0] s;

  // defining constant logic values
  supply1 one;
  supply0 zero;

  // internal results which are required by next stage adders
  wire c00, c01, c02, c03, c04, c05, c06, c07, c10, c11, c12, c13, c14, c15, c16,
     c17, c20, c21, c22, c23, c24, c25, c26, c27, c30, c31, c32, c33, c34, c35,
    c36, c37, c40, c41, c42, c43, c44, c45, c46, c47, c50, c51, c52, c53, c54,
    c55, c56, c57, c60, c61, c62, c63, c64, c65, c66, c67, c70, c71, c72, c73,
```

```
  c74, c75, c76, c77;
wire s00, s01, s02, s03, s04, s05, s06, s07, s10, s11, s12, s13, s14, s15, s16,
   s17, s20, s21, s22, s23, s24, s25, s26, s27, s30, s31, s32, s33, s34, s35,
  s36, s37, s40, s41, s42, s43, s44, s45, s46, s47, s50, s51, s52, s53, s54,
  s55, s56, s57, s60, s61, s62, s63, s64, s65, s66, s67, s70, s71, s72, s73,
  s74, s75, s76, s77;
wire s1, s2, s3, s4, s5, s6, s7, s8;
wire c1, c2, c3, c4, c5, c6, c7, c8;

// Naming the block in the diagram corresponsing to xi and yj inputs as bbij

// s0
black_box bb00(zero, zero, x[0], y[0], s00, c00);
assign s[0] = s00;

// s1
black_box bb10(zero, zero, x[1], y[0], s10, c10);
black_box bb01(s10, c00, x[0], y[1], s01, c01);
assign s[1] = s01;

//s2
black_box bb20(zero, zero, x[2], y[0], s20, c20);
black_box bb11(s20, c10, x[1], y[1], s11, c11);
black_box bb02(s11, c01, x[0], y[2], s02, c02);
assign s[2] = s02;

//s3
black_box bb30(zero, zero, x[3], y[0], s30, c30);
black_box bb21(s30, c20, x[2], y[1], s21, c21);
black_box bb12(s21, c11, x[1], y[2], s12, c12);
black_box bb03(s12, c02, x[0], y[3], s03, c03);
assign s[3] = s03;

//s4
black_box bb40(zero, zero, x[4], y[0], s40, c40);
black_box bb31(s40, c30, x[3], y[1], s31, c31);
black_box bb22(s31, c21, x[2], y[2], s22, c22);
black_box bb13(s22, c12, x[1], y[3], s13, c13);
black_box bb04(s13, c03, x[0], y[4], s04, c04);
assign s[4] = s04;

//s5
black_box bb50(zero, zero, x[5], y[0], s50, c50);
black_box bb41(s50, c40, x[4], y[1], s41, c41);
 black_box bb32(s41, c31, x[3], y[2], s32, c32);
 black_box bb23(s32, c22, x[2], y[3], s23, c23);
black_box bb14(s23, c13, x[1], y[4], s14, c14);
black_box bb05(s14, c04, x[0], y[5], s05, c05);
assign s[5] = s05;

//s6
```

```
black_box bb60(zero, zero, x[6], y[0], s60, c60);
 black_box bb51(s60, c50, x[5], y[1], s51, c51);
 black_box bb42(s51, c41, x[4], y[2], s42, c42);
 black_box bb33(s42, c32, x[3], y[3], s33, c33);
 black_box bb24(s33, c23, x[2], y[4], s24, c24);
 black_box bb15(s24, c14, x[1], y[5], s15, c15);
 black_box bb06(s15, c05, x[0], y[6], s06, c06);
assign s[6] = s06;

//s7
grey_box bb70(zero, zero, x[7], y[0], s70, c70);
 black_box bb61(s70, c60, x[6], y[1], s61, c61);
 black_box bb52(s61, c51, x[5], y[2], s52, c52);
 black_box bb43(s52, c42, x[4], y[3], s43, c43);
 black_box bb34(s43, c33, x[3], y[4], s34, c34);
 black_box bb25(s34, c24, x[2], y[5], s25, c25);
 black_box bb16(s25, c15, x[1], y[6], s16, c16);
 grey_box bb07(s16, c06, x[0], y[7], s07, c07);
assign s[7] = s07;

//s8
grey_box bb71(zero, c70, x[7], y[1], s71, c71);
 black_box bb62(s71, c61, x[6], y[2], s62, c62);
 black_box bb53(s62, c52, x[5], y[3], s53, c53);
 black_box bb44(s53, c43, x[4], y[4], s44, c44);
 black_box bb35(s44, c34, x[3], y[5], s35, c35);
 black_box bb26(s35, c25, x[2], y[6], s26, c26);
 grey_box bb17(s26, c16, x[1], y[7], s17, c17);
full_adder fa1(s17, c07, one, s1, c1);
assign s[8] = s1;

//s9
grey_box bb72(zero, c71, x[7], y[2], s72, c72);
 black_box bb63(s72, c62, x[6], y[3], s63, c63);
 black_box bb54(s63, c53, x[5], y[4], s54, c54);
 black_box bb45(s54, c44, x[4], y[5], s45, c45);
 black_box bb36(s45, c35, x[3], y[6], s36, c36);
 grey_box bb27(s36, c26, x[2], y[7], s27, c27);
full_adder fa2(s27, c17, c1, s2, c2);
assign s[9] = s2;

//s10
grey_box bb73(zero, c72, x[7], y[3], s73, c73);
 black_box bb64(s73, c63, x[6], y[4], s64, c64);
 black_box bb55(s64, c54, x[5], y[5], s55, c55);
 black_box bb46(s55, c45, x[4], y[6], s46, c46);
 grey_box bb37(s46, c36, x[3], y[7], s37, c37);
 full_adder fa3(s37, c27, c2, s3, c3);
assign s[10] = s3;

//s11
```

```
  grey_box bb74(zero, c73, x[7], y[4], s74, c74);
   black_box bb65(s74, c64, x[6], y[5], s65, c65);
   black_box bb56(s65, c55, x[5], y[6], s56, c56);
   grey_box bb47(s56, c46, x[4], y[7], s47, c47);
   full_adder fa4(s47, c37, c3, s4, c4);
  assign s[11] = s4;

  //s12
  grey_box bb75(zero, c74, x[7], y[5], s75, c75);
   black_box bb66(s75, c65, x[6], y[6], s66, c66);
   grey_box bb57(s66, c56, x[5], y[7], s57, c57);
   full_adder fa5(s57, c47, c4, s5, c5);
  assign s[12] = s5;

  //s13
  grey_box bb76(zero, c75, x[7], y[6], s76, c76);
   grey_box bb67(s76, c66, x[6], y[7], s67, c67);
   full_adder fa6(s67, c57, c5, s6, c6);
  assign s[13] = s6;

  //s14
  black_box bb77(zero, c76, x[7], y[7], s77, c77);
   full_adder fa7(s77, c67, c6, s7, c7);
  assign s[14] = s7;

  //s15
  full_adder fa8(one, c77, c7, s8, c8);
  assign s[15] = s8;


endmodule
```

Testbenching Verilog code

```
module test;

reg [7:0] x, y;   // 8 bit inputs

wire [15:0] s; // 16 bit output of multiplier circuit

baugh_wooley mult(.x(x), .y(y), .s(s));

initial
begin
x = 8'b00111111; y = 8'b00011001;
#10 x = 8'b00110111; y = 8'b00011011;
end
```

```
initial
$monitor($time, "%b * %b = %b", x, y, s);

endmodule
```