Name :- Pritam Patel

Roll No:- 1901145

EC362 VLSI Design

Objective:-Design of 12×12-bit Baugh Wooley Array Multiplier.

Theory:-

Array Multiplier:-An array multiplier is a digital combinational circuit used for multiplying two binary numbers by employing an array of full adders and half adders.This array is used for the nearly simultaneous addition of the various product terms involved.For the product terms, an array of AND gates is used before the Adder array.

Baugh Wooley Array Multiplier:-In signed multiplication the length of the partial products and the number of partial products will be very high. The BaughWooley algorithm was introduced for signed multiplication. 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 complement numbers.

Baugh-Wooley Algorithm:-

Let two n-bit numbers, number (X) and number (Y), X and Ycan be written as:-

$$X=-x_{n-1}2^{n-1}+\sum_{i=0}^{n-2} x_i2^i$$

$$Y=-y_{n-1}2^{n-1}+\sum_{i=0}^{n-2} y_i2^i$$

Where x_{n-1} and y_{n-1} are the sign bits.

The product, P = X × Y, is

$$P=[(-x_{n-1}2^{n-1}+\sum_{i=0}^{n-2} x_i2^i)\times(-y_{n-1}2^{n-1}+\sum_{i=0}^{n-2} y_i2^i)]$$

$$=x_{n-1}y_{n-1}2^{2n-2}+\sum_{i=0}^{n-1} x_i2^i \sum_{j=0}^{n-2} y_j2^j -2^{n-1}\sum_{i=0}^{n-2} x_iy_{n-1}2^i-2^{n-1}\sum_{j=0}^{n-2} x_{n-1}y_j2^j$$

The first two terms of the above equation are positive and the last two terms are negative. In order to calculate the product, instead of subtracting the last two terms, it is possible to add the opposite values. The above equation signifies the Baugh-Wooley algorithm for the multiplication process in two's complement form.

Multiplication example of baugh-wooley algorithm:-

•Let's consider 4 bit signed operands X and Y and 8 bit product P:-

$X\Leftrightarrow(X_3,X_2,X_1,X_0)$, $Y\Leftrightarrow(Y_3,Y_2,Y_1,Y_0)$,$P\Leftrightarrow(P_7,P_6,P_5,P_4,P_3,P_2,P_1,P_0)$

These are in the numerical form ,we can calculate their numerical value :-

$X=-x_32^3+\sum_{i=0}^{2} x_i2^i$ $\quad Y=-y_32^3+\sum_{j=0}^{2} y_i2^j$ $\quad P=-p_72^7+\sum_{i=0}^{6} p_i2^i$

$$P=(-x_32^3+\sum_{i=0}^{2} x_i2^i)(-y_32^3+\sum_{j=0}^{2} y_i2)$$

$$=x_3y_32^6 +\sum_{i=0}^{2}\sum_{j=0}^{2} x_iy_j2^{i+j} - \sum_{i=0}^{2} x_iy_32^{i+3} - \sum_{j=0}^{2} x_3y_j2^{j+3}$$

				X3	X2	X1	X0
				Y3	Y2	Y1	Y0
				$\overline{X3Y0}$	X2Y0	X1Y0	X0Y0
			$\overline{X3Y1}$	X2Y1	X1Y1	X0Y1	
		$\overline{X3Y2}$	X2Y2	X1Y2	X0Y2		
1	X3Y3	$\overline{X2Y3}$	$\overline{X1Y3}$	$\overline{X0Y3}$			
P7	P6	P5	P4	P3	P2	P1	P0

Baugh wooley multiplier architecture for 4×4 bit:-

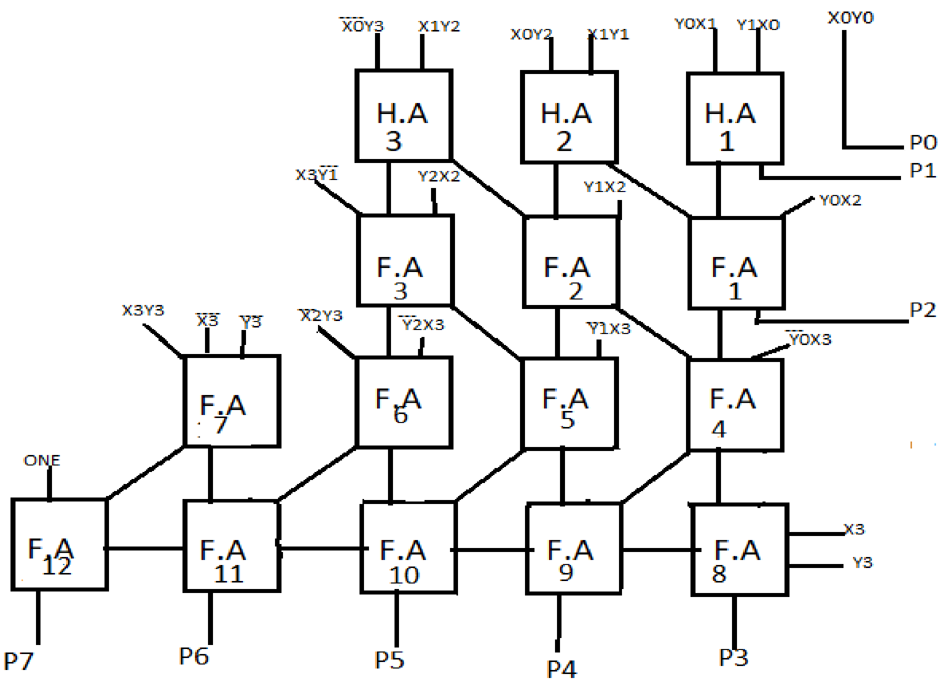


Fig 1:-BW architecture for 4 bit

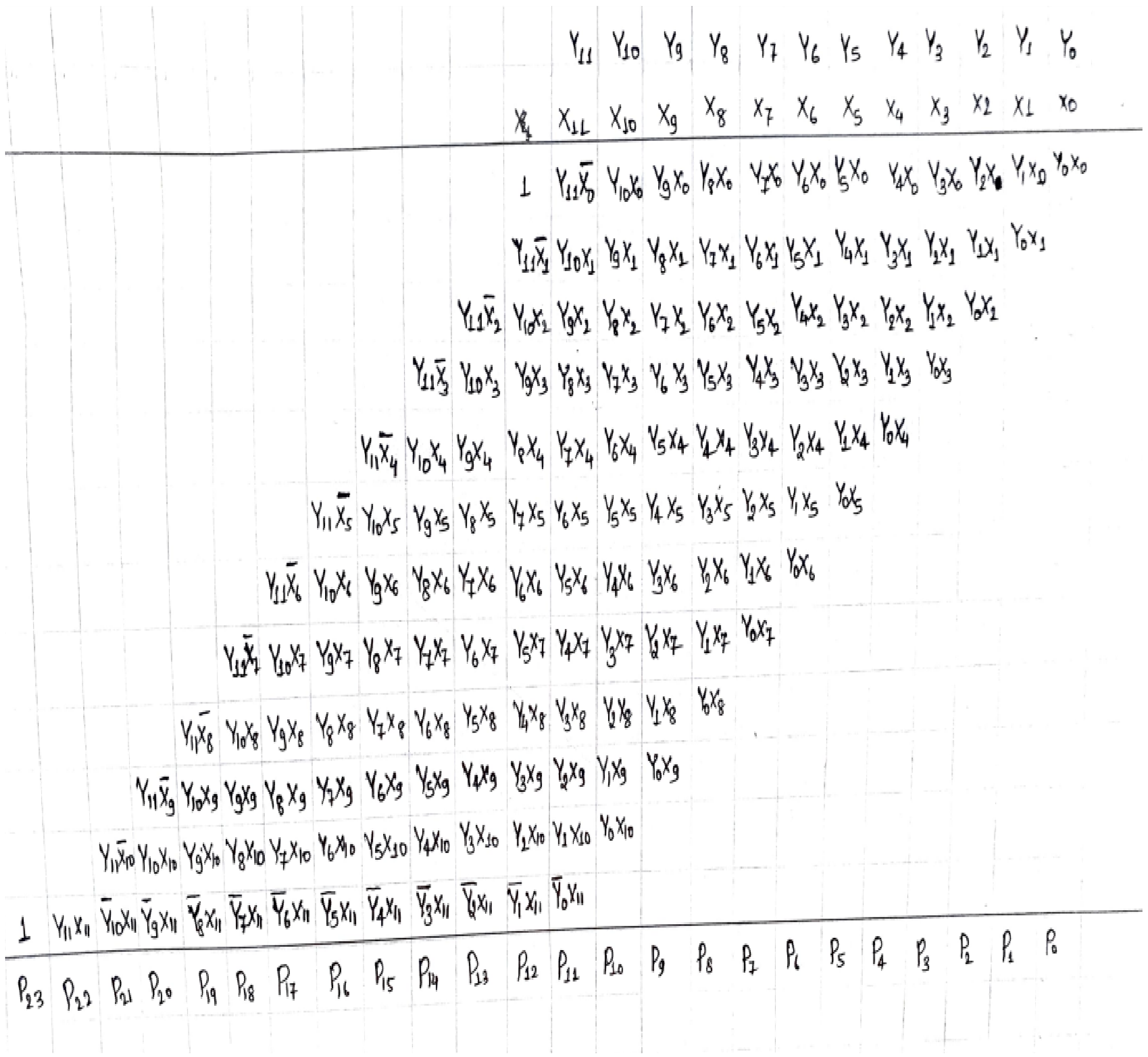
Multiplication of 12×12 signed bit:-

Let’s consider 12 bit signed operands X and Y and 24 bit product P:-

$X \Leftrightarrow (X_{11},X_{10},X_9,X_{8,}X_7,X_6,X_5,X_4X_3,X_2,X_1,X_0)$,

$Y \Leftrightarrow (Y_{11},Y_{10},Y_9,Y_8,Y_7,Y_6,Y_5,Y_4,Y_3,Y_2,Y_1,Y_0)$,

$P \Leftrightarrow (P_{23},P_{22},P_{21},P_{20},P_{19},P_{18},P_{17},P_{16}P_{,15},P_{14},P_{13},P_{12},P_{11},P_{10},P_9,P_8,P_7,P_6,P_5,$
$P_4,P_3,P_2,P_1,P_0)$



													Y_{11}	Y_{10}	Y_9	Y_8	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
												X_4	X_{11}	X_{10}	X_9	X_8	X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0
												1	$Y_{11}\bar{X}_0$	$Y_{10}X_0$	Y_9X_0	Y_8X_0	Y_7X_0	Y_6X_0	Y_5X_0	Y_4X_0	Y_3X_0	Y_2X_0	Y_1X_0	Y_0X_0
												$Y_{11}\bar{X}_1$	$Y_{10}X_1$	Y_9X_1	Y_8X_1	Y_7X_1	Y_6X_1	Y_5X_1	Y_4X_1	Y_3X_1	Y_2X_1	Y_1X_1	Y_0X_1	
											$Y_{11}\bar{X}_2$	$Y_{10}X_2$	Y_9X_2	Y_8X_2	Y_7X_2	Y_6X_2	Y_5X_2	Y_4X_2	Y_3X_2	Y_2X_2	Y_1X_2	Y_0X_2		
										$Y_{11}\bar{X}_3$	$Y_{10}X_3$	Y_9X_3	Y_8X_3	Y_7X_3	Y_6X_3	Y_5X_3	Y_4X_3	Y_3X_3	Y_2X_3	Y_1X_3	Y_0X_3			
									$Y_{11}\bar{X}_4$	$Y_{10}X_4$	Y_9X_4	Y_8X_4	Y_7X_4	Y_6X_4	Y_5X_4	Y_4X_4	Y_3Y_4	Y_2X_4	Y_1X_4	Y_0X_4				
								$Y_{11}\bar{X}_5$	$Y_{10}X_5$	Y_9X_5	Y_8X_5	Y_7X_5	Y_6X_5	Y_5X_5	Y_4X_5	Y_3X_5	Y_2X_5	Y_1X_5	Y_0X_5					
							$Y_{11}\bar{X}_6$	$Y_{10}X_6$	Y_9X_6	Y_8X_6	Y_7X_6	Y_6X_6	Y_5X_6	Y_4X_6	Y_3X_6	Y_2X_6	Y_1X_6	Y_0X_6						
						$Y_{11}\bar{X}_7$	$Y_{10}X_7$	Y_9X_7	Y_8X_7	Y_7X_7	Y_6X_7	Y_5X_7	Y_4X_7	Y_3X_7	Y_2X_7	Y_1X_7	Y_0X_7							
					$Y_{11}\bar{X}_8$	$Y_{10}X_8$	Y_9X_8	Y_8X_8	Y_7X_8	Y_6X_8	Y_5X_8	Y_4X_8	Y_3X_8	Y_2X_8	Y_1X_8	Y_0X_8								
				$Y_{11}\bar{X}_9$	$Y_{10}X_9$	Y_9X_9	Y_8X_9	Y_7X_9	Y_6X_9	Y_5X_9	Y_4X_9	Y_3X_9	Y_2X_9	Y_1X_9	Y_0X_9									
			$Y_{11}\bar{X}_{10}$	$Y_{10}X_{10}$	Y_9X_{10}	Y_8X_{10}	Y_7X_{10}	Y_6X_{10}	Y_5X_{10}	Y_4X_{10}	Y_3X_{10}	Y_2X_{10}	Y_1X_{10}	Y_0X_{10}										
	1	$Y_{11}X_{11}$	$\bar{Y}_{10}X_{11}$	$\bar{Y}_9X_{11}$	$\bar{Y}_8X_{11}$	$\bar{Y}_7X_{11}$	$\bar{Y}_6X_{11}$	$\bar{Y}_5X_{11}$	$\bar{Y}_4X_{11}$	$\bar{Y}_3X_{11}$	$\bar{Y}_2X_{11}$	$\bar{Y}_1X_{11}$	$\bar{Y}_0X_{11}$											
	P_{23}	P_{22}	P_{21}	P_{20}	P_{19}	P_{18}	P_{17}	P_{16}	P_{15}	P_{14}	P_{13}	P_{12}	P_{11}	P_{10}	P_9	P_8	P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0

Baugh wooley multiplier architecture for 12×12 bit:-

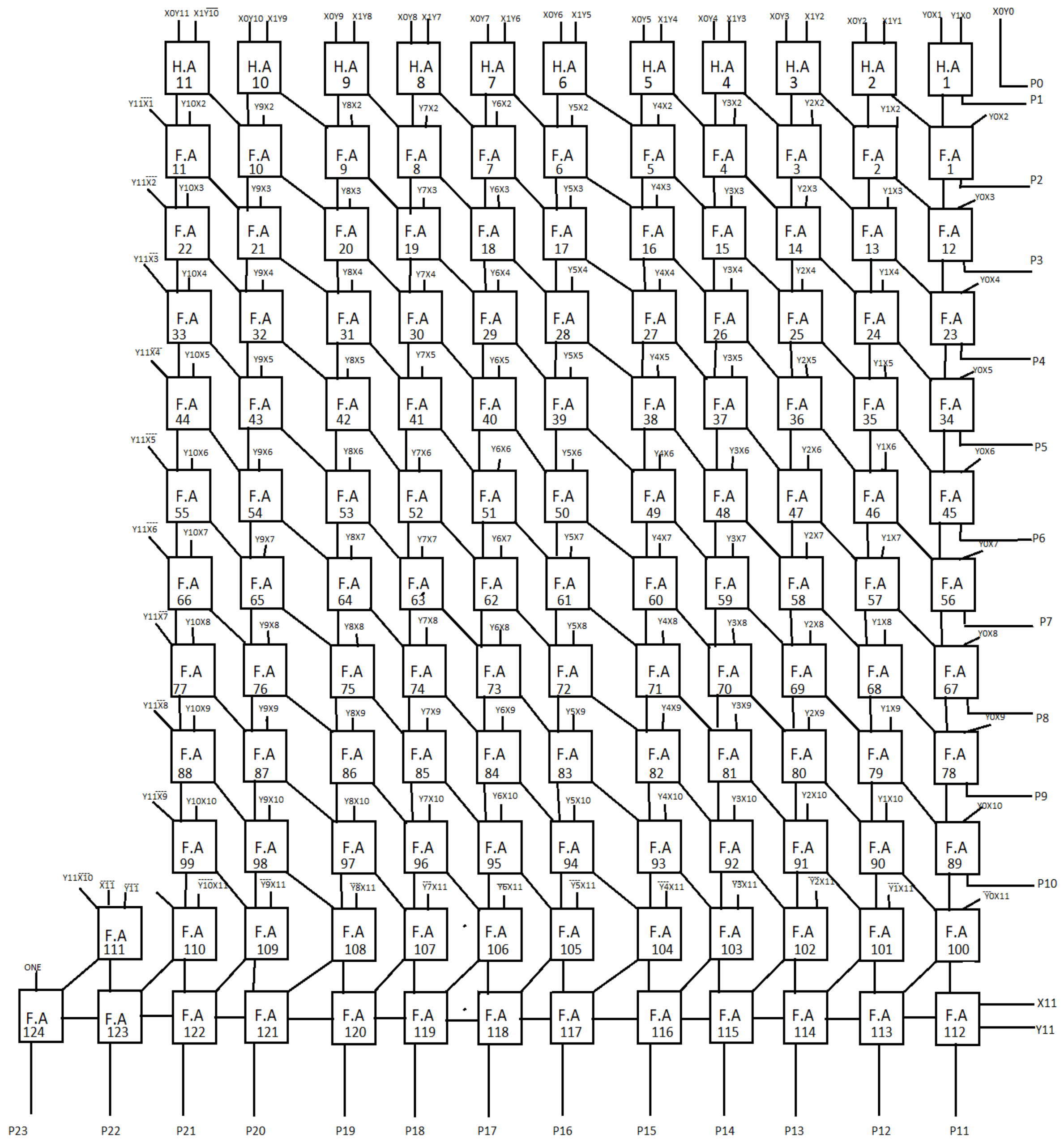


Fig2 :-BW architecture for 12×12 bit

Results:-

Output Waveform For Some Random Number Taken In Test Bench:-

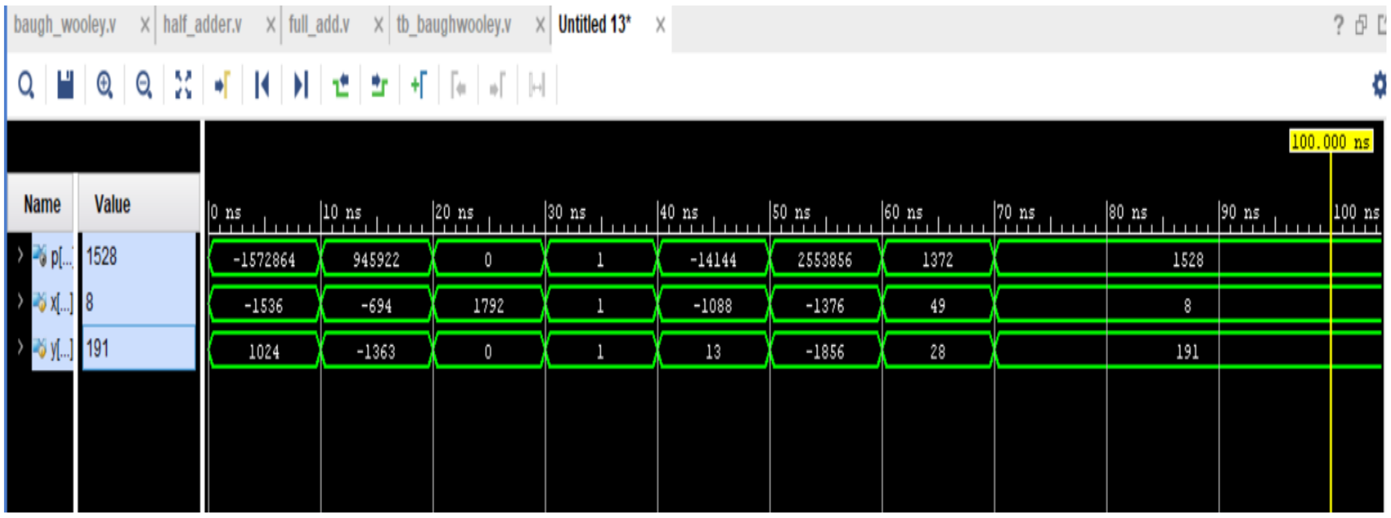


Fig 3:-Output waveform in signed decimal

In binary:-

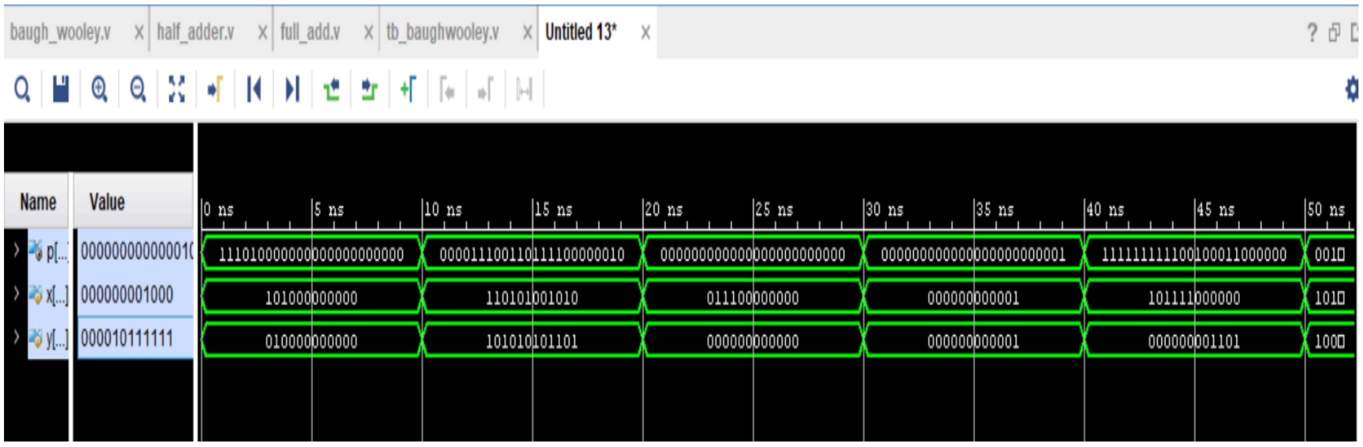


Fig 4:-Output waveform in binary

Schematic of 12x12 Bit Baugh-Wooley Multiplier:-

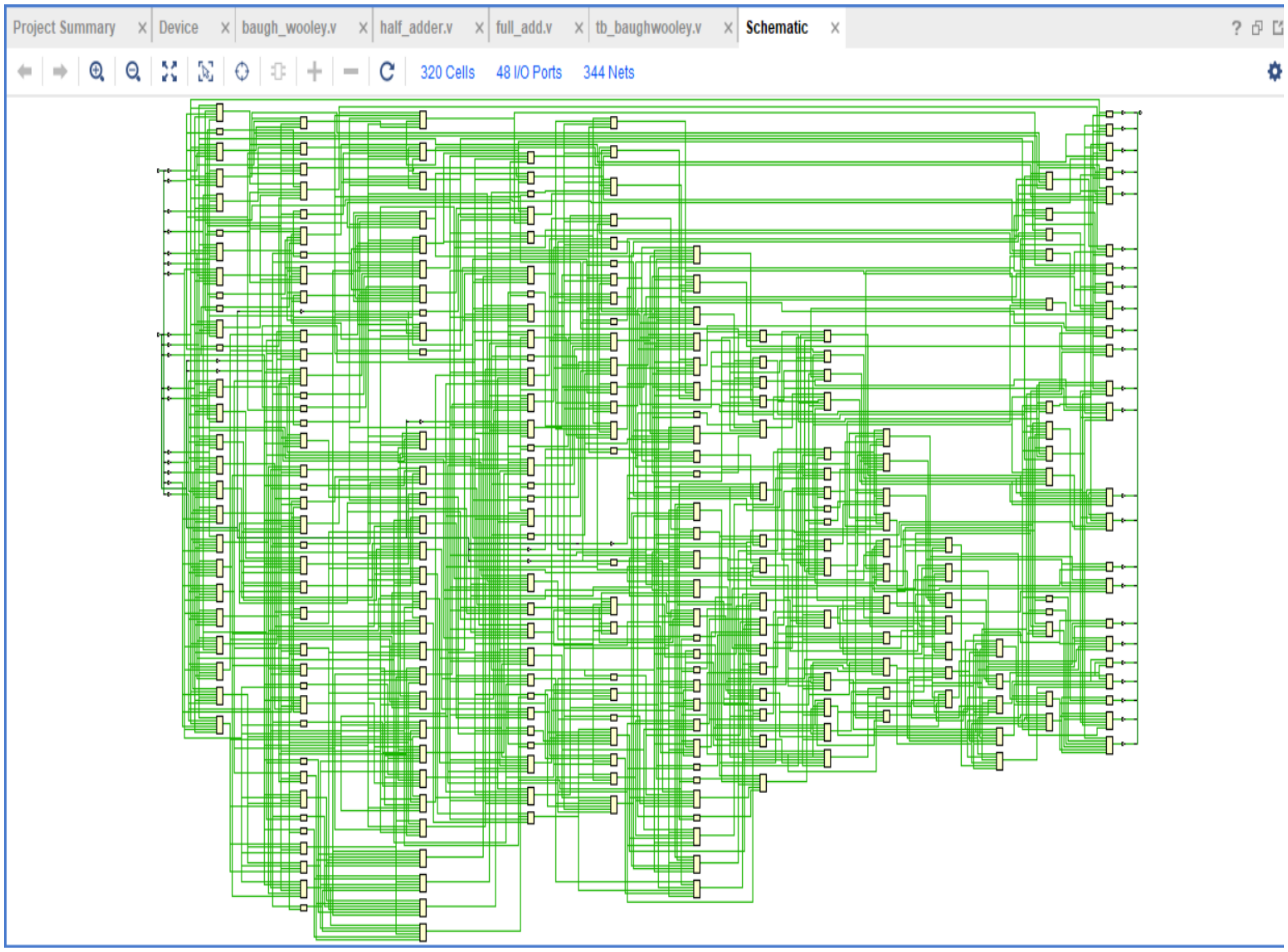
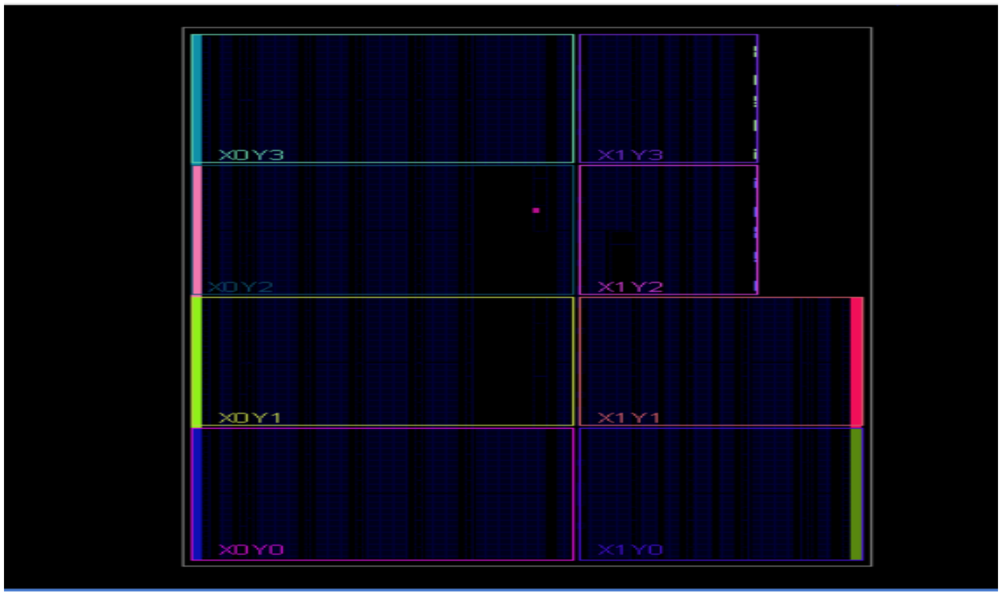


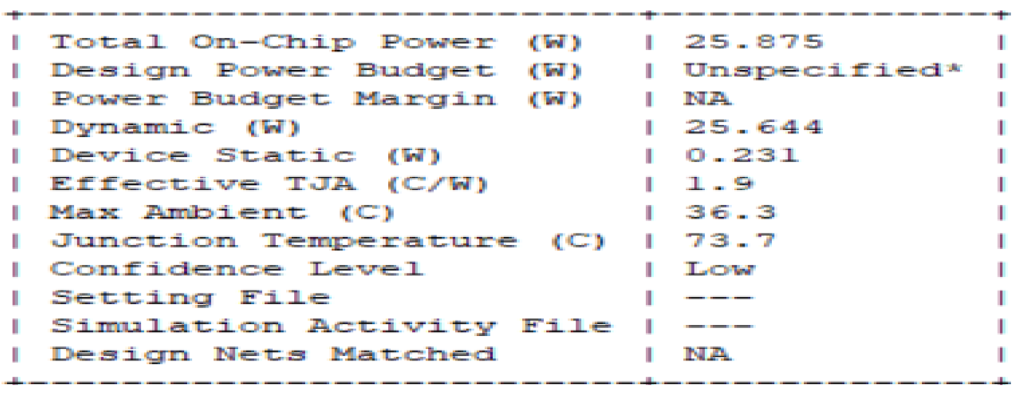
Fig 5:-Schematic for 12x12 bit BW multiplier

Hardware Result:-

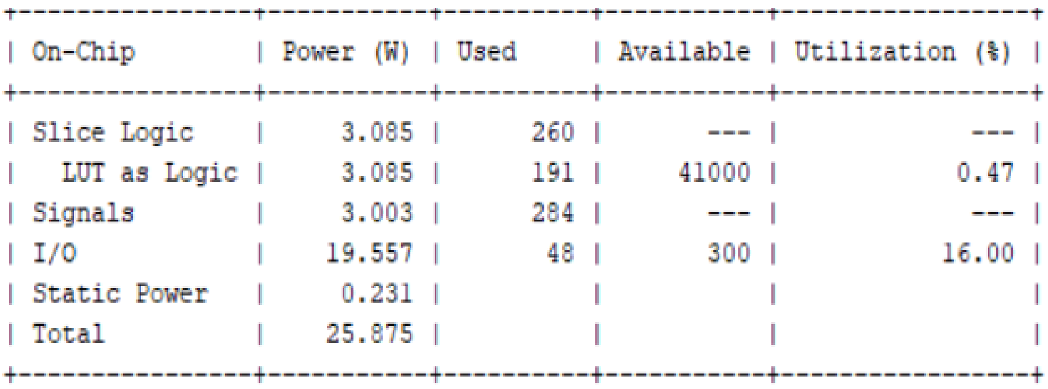
Device:-



Hardware implementation:-



```
+--------------------------------+----------------+
| Total On-Chip Power (W)        | 25.875         |
| Design Power Budget (W)        | Unspecified*   |
| Power Budget Margin (W)        | NA             |
| Dynamic (W)                    | 25.644         |
| Device Static (W)              | 0.231          |
| Effective TJA (C/W)            | 1.9            |
| Max Ambient (C)                | 36.3           |
| Junction Temperature (C)       | 73.7           |
| Confidence Level               | Low            |
| Setting File                   | ---            |
| Simulation Activity File       | ---            |
| Design Nets Matched            | NA             |
+--------------------------------+----------------+
```



```
+----------------+-----------+----------+-----------+-----------------+
| On-Chip        | Power (W) | Used     | Available | Utilization (%) |
+----------------+-----------+----------+-----------+-----------------+
| Slice Logic    |     3.085 |      260 |       --- |             --- |
|   LUT as Logic |     3.085 |      191 |     41000 |            0.47 |
| Signals        |     3.003 |      284 |       --- |             --- |
| I/O            |    19.557 |       48 |       300 |           16.00 |
| Static Power   |     0.231 |          |           |                 |
| Total          |    25.875 |          |           |                 |
+----------------+-----------+----------+-----------+-----------------+
```

Conclusion:-

Baugh Wooley multiplier exhibits less delay, low power dissipation and the area occupied is also small compared to other array multipliers.

CODE:-

DESIGN MODULE

```
// half adder component used in the multiplier
module half_adder(a, b, s, cout);
input a, b;
output s, cout;
assign s = a^b;
assign cout = a&b;
   endmodule

// full adder component used in the multiplier
module full_add(a, b, cin, s, cout);
input a, b, cin;
output s, cout;
assign s = a^b^cin;
assign cout = (a&b) | (b&cin) | (a&cin);
   endmodule

// 12-bit by 12-bit Baugh-Wooley signed multiplier
module baugh_wooley(x, y, p);
input [11:0] x, y;
output [23:0] p;
```

```
// constant logic-one value
supply1 one;
// Internal results which is given to next stage adders
wire t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12,t13, t14, t15, t16, t17, t18, t19, t20, t21, t22, t23,
t24,t25,t26,t27,t28,t29,t30,t31,t32,t33,t34,t35,t36,t37,t38,t39,t40,t41,t42,t43,t44,t45,t46,t47,t48,
T49,t50,t51,t52,t53,t54,t55,t56,t57,t58,t59,t60,t61,t62,t63,t64,t65,t66,t67,t68,t69,t70,t71,t72,t73,
t74,t75,t76,t77,t78,t79,t80,t81,t82,t83,t84,t85,t86,t87,t88,t89,t90,t91,t92,t93,t94,t95,t96,t97,t98,
t99,t100,t101,t102,t103,t104,t105,t106,t107,t108,t109,t110,t111,t112,t113,t114,t115,t116,t117,
t118,t119,t120,t121,t122,t123,t124,t125,t126,t127,t128,t129,t130,t131,t132,t133,t134,t135,t136,
t137,t138,t139,t140,t141,t142,t143,t144,t145,t146,t147,t148,t149,t150,t151,t152,t153,t154,t155,
t156,t157,t158,t159,t160,t161,t162,t163,t164,t165,t166,t167,t168,t169,t170,t171,t172,t173,t174,
t175,t176,t177,t178,t179,t180,t181,t182,t183,t184,t185,t186,t187,t188,t189,t190,t191,t192
,t193,t194,t195,t196,t197,t198,t199,t200,t201,t202,t203,t204,t205,t206,t207,t208,t209,t210,t211,
t212,t213,t214,t215,t216,t217,t218,t219,t220,t221,t222,t223,t224,t225,t226,t227,t228,t229,
t230,t231,t232,t233,t234,t235,t236,t237,t238,t239,t240,t241,t242,t243,t244,t245,t246,t247;
// structural description of the multiplier circuit
assign p[0] = x[0]&y[0];

half_adder ha1(x[1]&y[0], x[0]&y[1], p[1], t1);
half_adder ha2(x[2]&y[0], x[1]&y[1], t2, t3);
half_adder ha3(x[3]&y[0], x[2]&y[1], t4, t5);
half_adder ha4(x[4]&y[0], x[3]&y[1], t6, t7);
half_adder ha5(x[5]&y[0], x[4]&y[1], t8, t9);
half_adder ha6(x[6]&y[0], x[5]&y[1], t10, t11);
half_adder ha7(x[7]&y[0], x[6]&y[1], t12, t13);
half_adder ha8(x[8]&y[0], x[7]&y[1], t14, t15);
half_adder ha9(x[9]&y[0], x[8]&y[1], t16, t17);
half_adder ha10(x[10]&y[0], x[9]&y[1], t18,t19);
half_adder ha11(x[11]&~y[0], x[10]&y[1], t20, t21);
```

```
full_add fa1(t2, t1, x[0]&y[2], p[2], t22);
full_add fa2(t4, t3, x[1]&y[2], t23, t24);
full_add fa3( t6,t5, x[2]&y[2], t25, t26);
full_add fa4( t8, t7,x[3]&y[2], t27, t28);
full_add fa5( t10,t9,x[4]&y[2], t29, t30);
full_add fa6(t12,t11,x[5]&y[2], t31, t32);
full_add fa7(t14,t13,x[6]&y[2], t33,t34);
full_add fa8(t16,t15,x[7]&y[2], t35, t36);
full_add fa9(t18,t17,x[8]&y[2], t37, t38);
full_add fa10(t20,t19,x[9]&y[2], t39, t40);
full_add fa11(t21,x[11]&~y[1],x[10]&y[2],t41,t42);

full_add fa12(t23,t22, x[0]&y[3], p[3], t43);
full_add fa13(t25,t24, x[1]&y[3], t44, t45);
full_add fa14(t27,t26, x[2]&y[3], t46, t47);
full_add fa15(t29,t28, x[3]&y[3], t48, t49);
full_add fa16(t31,t30,x[4]&y[3], t50, t51);
full_add fa17(t33,t32,x[5]&y[3], t52, t53);
full_add fa18(t35,t34,x[6]&y[3], t54,t55);
full_add fa19(t37,t36,x[7]&y[3], t56, t57);
full_add fa20(t39,t38,x[8]&y[3], t58, t59);
full_add fa21(t41,t40,x[9]&y[3], t60, t61);
full_add fa22(t42,x[11]&~y[2],x[10]&y[3],t62,t63);


full_add fa23( t44,t43, x[0]&y[4], p[4], t64);
full_add fa24( t46,t45, x[1]&y[4], t65, t66);
full_add fa25( t48,t47, x[2]&y[4], t67, t68);
full_add fa26( t50,t49, x[3]&y[4], t69, t70);
full_add fa27( t52,t51, x[4]&y[4], t71, t72);
full_add fa28(t54,t53,  x[5]&y[4], t73, t74);
full_add fa29(t56,t55,  x[6]&y[4], t75,t76);
full_add fa30(t58,t57,x[7]&y[4], t77, t78);
full_add fa31(t60,t59,x[8]&y[4], t79, t80);
full_add fa32(t62,t61,x[9]&y[4], t81, t82);
full_add fa33(t63,x[11]&~y[3],x[10]&y[4],t83,t84);

full_add fa34(t65,t64, x[0]&y[5], p[5], t85);
full_add fa35(t67,t66, x[1]&y[5], t86, t87);
full_add fa36(t69,t68, x[2]&y[5], t88, t89);
full_add fa37(t71,t70, x[3]&y[5], t90, t91);
full_add fa38(t73,t72,x[4]&y[5], t92, t93);
full_add fa39(t75,t74,x[5]&y[5], t94, t95);
full_add fa40(t77,t76,x[6]&y[5], t96,t97);
full_add fa41(t79,t78,x[7]&y[5], t98, t99);
full_add fa42(t81,t80,x[8]&y[5], t100, t101);
full_add fa43(t83,t82,x[9]&y[5], t102, t103);
full_add fa44(t84,x[11]&~y[4],x[10]&y[5],t104,t105);

full_add fa45( t86,t85, x[0]&y[6], p[6], t106);
full_add fa46( t88,t87, x[1]&y[6], t107, t108);
full_add fa47( t90,t89, x[2]&y[6], t109, t110);
full_add fa48( t92,t91, x[3]&y[6], t111, t112);
full_add fa49( t94,t93,x[4]&y[6], t113, t114);
full_add fa50(t96,t95,x[5]&y[6], t115, t116);
full_add fa51(t98,t97,x[6]&y[6], t117,t118);

full_add fa52(t100,t99,x[7]&y[6], t119, t120);

full_add fa53(t102,t101,x[8]&y[6], t121, t122);

full_add fa54(t104,t103,x[9]&y[6], t123, t124);

full_add fa55(t105,x[11]&~y[5],x[10]&y[6],t125,t126);


full_add fa56(t107,t106,  x[0]&y[7], p[7], t127);

full_add fa57(t109,t108,  x[1]&y[7], t128, t129);

full_add fa58( t111,t110, x[2]&y[7], t130, t131);

full_add fa59( t113,t112, x[3]&y[7], t132, t133);

full_add fa60( t115,t114,x[4]&y[7], t134, t135);

full_add fa61(t117,t116,x[5]&y[7], t136, t137);

full_add fa62(t119,t118,x[6]&y[7], t138,t139);

full_add fa63(t121,t120,x[7]&y[7], t140, t141);

full_add fa64(t123,t122,x[8]&y[7], t142, t143);

full_add fa65(t125,t124,x[9]&y[7], t144, t145);

full_add fa66(t126,x[11]&~y[6],x[10]&y[7],t146,t147);


full_add fa67(t128,t127,  x[0]&y[8], p[8], t148);

full_add fa68( t130,t129, x[1]&y[8], t149, t150);

full_add fa69( t132,t131, x[2]&y[8], t151, t152);

full_add fa70( t134,t133, x[3]&y[8], t153, t154);

full_add fa71( t136,t135,x[4]&y[8], t155, t156);

full_add fa72(t138,t137,x[5]&y[8], t157, t158);

full_add fa73(t140,t139,x[6]&y[8], t159,t160);

full_add fa74(t142,t141,x[7]&y[8], t161, t162);

full_add fa75(t144,t143,x[8]&y[8], t163, t164);

full_add fa76(t146,t145,x[9]&y[8], t165, t166);
full_add fa77(t147,x[11]&~y[7],x[10]&y[8],t167,t168);

full_add fa78( t149,t148, x[0]&y[9], p[9], t169);
full_add fa79( t151,t150, x[1]&y[9], t170, t171);
full_add fa80( t153,t152, x[2]&y[9], t172, t173);
full_add fa81( t155,t154, x[3]&y[9], t174, t175);
full_add fa82( t157,t156,x[4]&y[9], t176, t177);
full_add fa83(t159,t158,x[5]&y[9], t178, t179);
full_add fa84(t161,t160,x[6]&y[9], t180,t181);
full_add fa85(t163,t162,x[7]&y[9], t182, t183);
full_add fa86(t165,t164,x[8]&y[9], t184, t185);
full_add fa87(t167,t166,x[9]&y[9], t186, t187);
full_add fa88(t168,x[11]&~y[8],x[10]&y[9],t188,t189);

full_add fa89( t170,t169, x[0]&y[10], p[10], t190);
full_add fa90( t172,t171, x[1]&y[10], t191, t192);
full_add fa91( t174,t173, x[2]&y[10], t193, t194);
full_add fa92( t176,t175, x[3]&y[10], t195, t196);
full_add fa93( t178,t177,x[4]&y[10], t197, t198);
full_add fa94(t180,t179,x[5]&y[10], t199, t200);
full_add fa95(t182,t181,x[6]&y[10], t201,t202);
full_add fa96(t184,t183,x[7]&y[10], t203, t204);
full_add fa97(t186,t185,x[8]&y[10], t205, t206);
full_add fa98(t188,t187,x[9]&y[10], t207, t208);
full_add fa99(t189,x[11]&~y[9],x[10]&y[10],t209,t210);


full_add fa100( t191,t190, ~x[0]&y[11], t211, t212);
full_add fa101( t193,t192, ~x[1]&y[11], t213, t214);
full_add fa102( t195,t194, ~x[2]&y[11], t215, t216);
full_add fa103( t197,t196, ~x[3]&y[11], t217, t218);
full_add fa104( t199,t198,~x[4]&y[11], t219, t220);
full_add fa105(t201,t200,~x[5]&y[11], t221, t222);
full_add fa106(t203,t202,~x[6]&y[11], t223,t224);
full_add fa107(t205,t204,~x[7]&y[11], t225, t226);
full_add fa108(t207,t206,~x[8]&y[11], t227, t228);
full_add fa109(t209,t208,~x[9]&y[11], t229, t230);
full_add fa110(t210,x[11]&~y[10],~x[10]&y[11],t231,t232);
full_add fa111(~x[11],~y[11],x[11]&y[11],t233, t234);

full_add fa112(t211, x[11],y[11], p[11], t235);
full_add fa113( t213,t212,t235, p[12], t236);
full_add fa114( t215,t214,t236 ,p[13], t237);
full_add fa115( t217,t216,t237 ,p[14], t238);
full_add fa116( t219,t218,t238, p[15], t239);
full_add fa117(t221,t220,t239, p[16], t240);
full_add fa118(t223,t222,t240,p[17],t241);
full_add fa119(t225,t224,t241,p[18], t242);
full_add fa120(t227,t226,t242,p[19], t243);
full_add fa121(t229,t228,t243,p[20], t244);
full_add fa122(t231,t230,t244,p[21],t245);
full_add fa123(t233,t232,t245,p[22],t246);
full_add fa124(1,t234,t246,p[23],t247);

endmodule
```

TEST BENCH:-

```
module test;
wire signed [23:0] p;
reg signed [11:0] x,y;
baugh_wooley my_booth(.x(x),.y(y),.p(p));
initial
begin
$monitor($time,"       ",x," *",y," = ", p);
x = 12'b101000000000;
y = 12'b010000000000;
#10
x =12'b110101001010;
y= 12'b101010101101;
#10
x= 12'b011100000000;
y= 12'b000000000000;
#10
x= 12'b1;
y= 12'b1;
#10
x= 12'b101111000000;
y = 12'b1101;
#10
x = 12'b101010100000;
y = 12'b100011000000;
#10
x= 12'b110001;
y= 12'b11100;
```

```
#10
x = 12'b1000;
y = 12'b10111111;

end
endmodule
```