CA_Lab7

K. Ram Mohan

COE19B055

Verilog Code:

```
`include "rdca_str.v"
`include "output_select.v"

module LOGIC_UNIT (in1, in2, sel, out);
   input[63:0]  in1, in2;
   input[2:0]  sel;
   output[63:0]  out;

   wire[63:0] and_, nand_, or_, nor_, xor_, not_, t_comp, xnor_, temp;
   wire temp_c;

   // AND
   and(and_[0], in1[0], in2[0]);
   and(and_[1], in1[1], in2[1]);
   and(and_[2], in1[2], in2[2]);
   and(and_[3], in1[3], in2[3]);
   and(and_[4], in1[4], in2[4]);
   and(and_[5], in1[5], in2[5]);
   and(and_[6], in1[6], in2[6]);
   and(and_[7], in1[7], in2[7]);
   and(and_[8], in1[8], in2[8]);
   and(and_[9], in1[9], in2[9]);
   and(and_[10], in1[10], in2[10]);
   and(and_[11], in1[11], in2[11]);
   and(and_[12], in1[12], in2[12]);
```

```
and(and_[13], in1[13], in2[13]);
and(and_[14], in1[14], in2[14]);
and(and_[15], in1[15], in2[15]);
and(and_[16], in1[16], in2[16]);
and(and_[17], in1[17], in2[17]);
and(and_[18], in1[18], in2[18]);
and(and_[19], in1[19], in2[19]);
and(and_[20], in1[20], in2[20]);
and(and_[21], in1[21], in2[21]);
and(and_[22], in1[22], in2[22]);
and(and_[23], in1[23], in2[23]);
and(and_[24], in1[24], in2[24]);
and(and_[25], in1[25], in2[25]);
and(and_[26], in1[26], in2[26]);
and(and_[27], in1[27], in2[27]);
and(and_[28], in1[28], in2[28]);
and(and_[29], in1[29], in2[29]);
and(and_[30], in1[30], in2[30]);
and(and_[31], in1[31], in2[31]);
and(and_[32], in1[32], in2[32]);
and(and_[33], in1[33], in2[33]);
and(and_[34], in1[34], in2[34]);
and(and_[35], in1[35], in2[35]);
and(and_[36], in1[36], in2[36]);
and(and_[37], in1[37], in2[37]);
and(and_[38], in1[38], in2[38]);
and(and_[39], in1[39], in2[39]);
and(and_[40], in1[40], in2[40]);
and(and_[41], in1[41], in2[41]);
```

```
and(and_[42], in1[42], in2[42]);
and(and_[43], in1[43], in2[43]);
and(and_[44], in1[44], in2[44]);
and(and_[45], in1[45], in2[45]);
and(and_[46], in1[46], in2[46]);
and(and_[47], in1[47], in2[47]);
and(and_[48], in1[48], in2[48]);
and(and_[49], in1[49], in2[49]);
and(and_[50], in1[50], in2[50]);
and(and_[51], in1[51], in2[51]);
and(and_[52], in1[52], in2[52]);
and(and_[53], in1[53], in2[53]);
and(and_[54], in1[54], in2[54]);
and(and_[55], in1[55], in2[55]);
and(and_[56], in1[56], in2[56]);
and(and_[57], in1[57], in2[57]);
and(and_[58], in1[58], in2[58]);
and(and_[59], in1[59], in2[59]);
and(and_[60], in1[60], in2[60]);
and(and_[61], in1[61], in2[61]);
and(and_[62], in1[62], in2[62]);
and(and_[63], in1[63], in2[63]);

// XOR
xor(xor_[0], in1[0], in2[0]);
xor(xor_[1], in1[1], in2[1]);
xor(xor_[2], in1[2], in2[2]);
xor(xor_[3], in1[3], in2[3]);
xor(xor_[4], in1[4], in2[4]);
```

```
xor(xor_[5], in1[5], in2[5]);
xor(xor_[6], in1[6], in2[6]);
xor(xor_[7], in1[7], in2[7]);
xor(xor_[8], in1[8], in2[8]);
xor(xor_[9], in1[9], in2[9]);
xor(xor_[10], in1[10], in2[10]);
xor(xor_[11], in1[11], in2[11]);
xor(xor_[12], in1[12], in2[12]);
xor(xor_[13], in1[13], in2[13]);
xor(xor_[14], in1[14], in2[14]);
xor(xor_[15], in1[15], in2[15]);
xor(xor_[16], in1[16], in2[16]);
xor(xor_[17], in1[17], in2[17]);
xor(xor_[18], in1[18], in2[18]);
xor(xor_[19], in1[19], in2[19]);
xor(xor_[20], in1[20], in2[20]);
xor(xor_[21], in1[21], in2[21]);
xor(xor_[22], in1[22], in2[22]);
xor(xor_[23], in1[23], in2[23]);
xor(xor_[24], in1[24], in2[24]);
xor(xor_[25], in1[25], in2[25]);
xor(xor_[26], in1[26], in2[26]);
xor(xor_[27], in1[27], in2[27]);
xor(xor_[28], in1[28], in2[28]);
xor(xor_[29], in1[29], in2[29]);
xor(xor_[30], in1[30], in2[30]);
xor(xor_[31], in1[31], in2[31]);
xor(xor_[32], in1[32], in2[32]);
xor(xor_[33], in1[33], in2[33]);
```

```
xor(xor_[34], in1[34], in2[34]);
xor(xor_[35], in1[35], in2[35]);
xor(xor_[36], in1[36], in2[36]);
xor(xor_[37], in1[37], in2[37]);
xor(xor_[38], in1[38], in2[38]);
xor(xor_[39], in1[39], in2[39]);
xor(xor_[40], in1[40], in2[40]);
xor(xor_[41], in1[41], in2[41]);
xor(xor_[42], in1[42], in2[42]);
xor(xor_[43], in1[43], in2[43]);
xor(xor_[44], in1[44], in2[44]);
xor(xor_[45], in1[45], in2[45]);
xor(xor_[46], in1[46], in2[46]);
xor(xor_[47], in1[47], in2[47]);
xor(xor_[48], in1[48], in2[48]);
xor(xor_[49], in1[49], in2[49]);
xor(xor_[50], in1[50], in2[50]);
xor(xor_[51], in1[51], in2[51]);
xor(xor_[52], in1[52], in2[52]);
xor(xor_[53], in1[53], in2[53]);
xor(xor_[54], in1[54], in2[54]);
xor(xor_[55], in1[55], in2[55]);
xor(xor_[56], in1[56], in2[56]);
xor(xor_[57], in1[57], in2[57]);
xor(xor_[58], in1[58], in2[58]);
xor(xor_[59], in1[59], in2[59]);
xor(xor_[60], in1[60], in2[60]);
xor(xor_[61], in1[61], in2[61]);
xor(xor_[62], in1[62], in2[62]);
```

```
xor(xor_[63], in1[63], in2[63]);

// NAND
nand(nand_[0], in1[0], in2[0]);
nand(nand_[1], in1[1], in2[1]);
nand(nand_[2], in1[2], in2[2]);
nand(nand_[3], in1[3], in2[3]);
nand(nand_[4], in1[4], in2[4]);
nand(nand_[5], in1[5], in2[5]);
nand(nand_[6], in1[6], in2[6]);
nand(nand_[7], in1[7], in2[7]);
nand(nand_[8], in1[8], in2[8]);
nand(nand_[9], in1[9], in2[9]);
nand(nand_[10], in1[10], in2[10]);
nand(nand_[11], in1[11], in2[11]);
nand(nand_[12], in1[12], in2[12]);
nand(nand_[13], in1[13], in2[13]);
nand(nand_[14], in1[14], in2[14]);
nand(nand_[15], in1[15], in2[15]);
nand(nand_[16], in1[16], in2[16]);
nand(nand_[17], in1[17], in2[17]);
nand(nand_[18], in1[18], in2[18]);
nand(nand_[19], in1[19], in2[19]);
nand(nand_[20], in1[20], in2[20]);
nand(nand_[21], in1[21], in2[21]);
nand(nand_[22], in1[22], in2[22]);
nand(nand_[23], in1[23], in2[23]);
nand(nand_[24], in1[24], in2[24]);
nand(nand_[25], in1[25], in2[25]);
```

```
nand(nand_[26], in1[26], in2[26]);
nand(nand_[27], in1[27], in2[27]);
nand(nand_[28], in1[28], in2[28]);
nand(nand_[29], in1[29], in2[29]);
nand(nand_[30], in1[30], in2[30]);
nand(nand_[31], in1[31], in2[31]);
nand(nand_[32], in1[32], in2[32]);
nand(nand_[33], in1[33], in2[33]);
nand(nand_[34], in1[34], in2[34]);
nand(nand_[35], in1[35], in2[35]);
nand(nand_[36], in1[36], in2[36]);
nand(nand_[37], in1[37], in2[37]);
nand(nand_[38], in1[38], in2[38]);
nand(nand_[39], in1[39], in2[39]);
nand(nand_[40], in1[40], in2[40]);
nand(nand_[41], in1[41], in2[41]);
nand(nand_[42], in1[42], in2[42]);
nand(nand_[43], in1[43], in2[43]);
nand(nand_[44], in1[44], in2[44]);
nand(nand_[45], in1[45], in2[45]);
nand(nand_[46], in1[46], in2[46]);
nand(nand_[47], in1[47], in2[47]);
nand(nand_[48], in1[48], in2[48]);
nand(nand_[49], in1[49], in2[49]);
nand(nand_[50], in1[50], in2[50]);
nand(nand_[51], in1[51], in2[51]);
nand(nand_[52], in1[52], in2[52]);
nand(nand_[53], in1[53], in2[53]);
nand(nand_[54], in1[54], in2[54]);
```

```
nand(nand_[55], in1[55], in2[55]);
nand(nand_[56], in1[56], in2[56]);
nand(nand_[57], in1[57], in2[57]);
nand(nand_[58], in1[58], in2[58]);
nand(nand_[59], in1[59], in2[59]);
nand(nand_[60], in1[60], in2[60]);
nand(nand_[61], in1[61], in2[61]);
nand(nand_[62], in1[62], in2[62]);
nand(nand_[63], in1[63], in2[63]);

// OR
or(or_[0], in1[0], in2[0]);
or(or_[1], in1[1], in2[1]);
or(or_[2], in1[2], in2[2]);
or(or_[3], in1[3], in2[3]);
or(or_[4], in1[4], in2[4]);
or(or_[5], in1[5], in2[5]);
or(or_[6], in1[6], in2[6]);
or(or_[7], in1[7], in2[7]);
or(or_[8], in1[8], in2[8]);
or(or_[9], in1[9], in2[9]);
or(or_[10], in1[10], in2[10]);
or(or_[11], in1[11], in2[11]);
or(or_[12], in1[12], in2[12]);
or(or_[13], in1[13], in2[13]);
or(or_[14], in1[14], in2[14]);
or(or_[15], in1[15], in2[15]);
or(or_[16], in1[16], in2[16]);
or(or_[17], in1[17], in2[17]);
```

```
or(or_[18], in1[18], in2[18]);
or(or_[19], in1[19], in2[19]);
or(or_[20], in1[20], in2[20]);
or(or_[21], in1[21], in2[21]);
or(or_[22], in1[22], in2[22]);
or(or_[23], in1[23], in2[23]);
or(or_[24], in1[24], in2[24]);
or(or_[25], in1[25], in2[25]);
or(or_[26], in1[26], in2[26]);
or(or_[27], in1[27], in2[27]);
or(or_[28], in1[28], in2[28]);
or(or_[29], in1[29], in2[29]);
or(or_[30], in1[30], in2[30]);
or(or_[31], in1[31], in2[31]);
or(or_[32], in1[32], in2[32]);
or(or_[33], in1[33], in2[33]);
or(or_[34], in1[34], in2[34]);
or(or_[35], in1[35], in2[35]);
or(or_[36], in1[36], in2[36]);
or(or_[37], in1[37], in2[37]);
or(or_[38], in1[38], in2[38]);
or(or_[39], in1[39], in2[39]);
or(or_[40], in1[40], in2[40]);
or(or_[41], in1[41], in2[41]);
or(or_[42], in1[42], in2[42]);
or(or_[43], in1[43], in2[43]);
or(or_[44], in1[44], in2[44]);
or(or_[45], in1[45], in2[45]);
or(or_[46], in1[46], in2[46]);
```

```
or(or_[47], in1[47], in2[47]);
or(or_[48], in1[48], in2[48]);
or(or_[49], in1[49], in2[49]);
or(or_[50], in1[50], in2[50]);
or(or_[51], in1[51], in2[51]);
or(or_[52], in1[52], in2[52]);
or(or_[53], in1[53], in2[53]);
or(or_[54], in1[54], in2[54]);
or(or_[55], in1[55], in2[55]);
or(or_[56], in1[56], in2[56]);
or(or_[57], in1[57], in2[57]);
or(or_[58], in1[58], in2[58]);
or(or_[59], in1[59], in2[59]);
or(or_[60], in1[60], in2[60]);
or(or_[61], in1[61], in2[61]);
or(or_[62], in1[62], in2[62]);
or(or_[63], in1[63], in2[63]);

// // NOT
not(not_[0], in1[0]);
not(not_[1], in1[1]);
not(not_[2], in1[2]);
not(not_[3], in1[3]);
not(not_[4], in1[4]);
not(not_[5], in1[5]);
not(not_[6], in1[6]);
not(not_[7], in1[7]);
not(not_[8], in1[8]);
not(not_[9], in1[9]);
```

```
not(not_[10], in1[10]);
not(not_[11], in1[11]);
not(not_[12], in1[12]);
not(not_[13], in1[13]);
not(not_[14], in1[14]);
not(not_[15], in1[15]);
not(not_[16], in1[16]);
not(not_[17], in1[17]);
not(not_[18], in1[18]);
not(not_[19], in1[19]);
not(not_[20], in1[20]);
not(not_[21], in1[21]);
not(not_[22], in1[22]);
not(not_[23], in1[23]);
not(not_[24], in1[24]);
not(not_[25], in1[25]);
not(not_[26], in1[26]);
not(not_[27], in1[27]);
not(not_[28], in1[28]);
not(not_[29], in1[29]);
not(not_[30], in1[30]);
not(not_[31], in1[31]);
not(not_[32], in1[32]);
not(not_[33], in1[33]);
not(not_[34], in1[34]);
not(not_[35], in1[35]);
not(not_[36], in1[36]);
not(not_[37], in1[37]);
not(not_[38], in1[38]);
```

```
not(not_[39], in1[39]);
not(not_[40], in1[40]);
not(not_[41], in1[41]);
not(not_[42], in1[42]);
not(not_[43], in1[43]);
not(not_[44], in1[44]);
not(not_[45], in1[45]);
not(not_[46], in1[46]);
not(not_[47], in1[47]);
not(not_[48], in1[48]);
not(not_[49], in1[49]);
not(not_[50], in1[50]);
not(not_[51], in1[51]);
not(not_[52], in1[52]);
not(not_[53], in1[53]);
not(not_[54], in1[54]);
not(not_[55], in1[55]);
not(not_[56], in1[56]);
not(not_[57], in1[57]);
not(not_[58], in1[58]);
not(not_[59], in1[59]);
not(not_[60], in1[60]);
not(not_[61], in1[61]);
not(not_[62], in1[62]);
not(not_[63], in1[63]);

// // NOR
nor(nor_[0], in1[0], in2[0]);
nor(nor_[1], in1[1], in2[1]);
```

```
nor(nor_[2], in1[2], in2[2]);
nor(nor_[3], in1[3], in2[3]);
nor(nor_[4], in1[4], in2[4]);
nor(nor_[5], in1[5], in2[5]);
nor(nor_[6], in1[6], in2[6]);
nor(nor_[7], in1[7], in2[7]);
nor(nor_[8], in1[8], in2[8]);
nor(nor_[9], in1[9], in2[9]);
nor(nor_[10], in1[10], in2[10]);
nor(nor_[11], in1[11], in2[11]);
nor(nor_[12], in1[12], in2[12]);
nor(nor_[13], in1[13], in2[13]);
nor(nor_[14], in1[14], in2[14]);
nor(nor_[15], in1[15], in2[15]);
nor(nor_[16], in1[16], in2[16]);
nor(nor_[17], in1[17], in2[17]);
nor(nor_[18], in1[18], in2[18]);
nor(nor_[19], in1[19], in2[19]);
nor(nor_[20], in1[20], in2[20]);
nor(nor_[21], in1[21], in2[21]);
nor(nor_[22], in1[22], in2[22]);
nor(nor_[23], in1[23], in2[23]);
nor(nor_[24], in1[24], in2[24]);
nor(nor_[25], in1[25], in2[25]);
nor(nor_[26], in1[26], in2[26]);
nor(nor_[27], in1[27], in2[27]);
nor(nor_[28], in1[28], in2[28]);
nor(nor_[29], in1[29], in2[29]);
nor(nor_[30], in1[30], in2[30]);
```

```
nor(nor_[31], in1[31], in2[31]);
nor(nor_[32], in1[32], in2[32]);
nor(nor_[33], in1[33], in2[33]);
nor(nor_[34], in1[34], in2[34]);
nor(nor_[35], in1[35], in2[35]);
nor(nor_[36], in1[36], in2[36]);
nor(nor_[37], in1[37], in2[37]);
nor(nor_[38], in1[38], in2[38]);
nor(nor_[39], in1[39], in2[39]);
nor(nor_[40], in1[40], in2[40]);
nor(nor_[41], in1[41], in2[41]);
nor(nor_[42], in1[42], in2[42]);
nor(nor_[43], in1[43], in2[43]);
nor(nor_[44], in1[44], in2[44]);
nor(nor_[45], in1[45], in2[45]);
nor(nor_[46], in1[46], in2[46]);
nor(nor_[47], in1[47], in2[47]);
nor(nor_[48], in1[48], in2[48]);
nor(nor_[49], in1[49], in2[49]);
nor(nor_[50], in1[50], in2[50]);
nor(nor_[51], in1[51], in2[51]);
nor(nor_[52], in1[52], in2[52]);
nor(nor_[53], in1[53], in2[53]);
nor(nor_[54], in1[54], in2[54]);
nor(nor_[55], in1[55], in2[55]);
nor(nor_[56], in1[56], in2[56]);
nor(nor_[57], in1[57], in2[57]);
nor(nor_[58], in1[58], in2[58]);
nor(nor_[59], in1[59], in2[59]);
```

```
nor(nor_[60], in1[60], in2[60]);
nor(nor_[61], in1[61], in2[61]);
nor(nor_[62], in1[62], in2[62]);
nor(nor_[63], in1[63], in2[63]);

// 2's complement
assign temp = 64'b1;
RCDA rcda1(not_, temp, 1'b0, t_comp, temp_c);

// XNOR
xnor(xnor_[0], in1[0], in2[0]);
xnor(xnor_[1], in1[1], in2[1]);
xnor(xnor_[2], in1[2], in2[2]);
xnor(xnor_[3], in1[3], in2[3]);
xnor(xnor_[4], in1[4], in2[4]);
xnor(xnor_[5], in1[5], in2[5]);
xnor(xnor_[6], in1[6], in2[6]);
xnor(xnor_[7], in1[7], in2[7]);
xnor(xnor_[8], in1[8], in2[8]);
xnor(xnor_[9], in1[9], in2[9]);
xnor(xnor_[10], in1[10], in2[10]);
xnor(xnor_[11], in1[11], in2[11]);
xnor(xnor_[12], in1[12], in2[12]);
xnor(xnor_[13], in1[13], in2[13]);
xnor(xnor_[14], in1[14], in2[14]);
xnor(xnor_[15], in1[15], in2[15]);
xnor(xnor_[16], in1[16], in2[16]);
xnor(xnor_[17], in1[17], in2[17]);
xnor(xnor_[18], in1[18], in2[18]);
```

```
xnor(xnor_[19], in1[19], in2[19]);
xnor(xnor_[20], in1[20], in2[20]);
xnor(xnor_[21], in1[21], in2[21]);
xnor(xnor_[22], in1[22], in2[22]);
xnor(xnor_[23], in1[23], in2[23]);
xnor(xnor_[24], in1[24], in2[24]);
xnor(xnor_[25], in1[25], in2[25]);
xnor(xnor_[26], in1[26], in2[26]);
xnor(xnor_[27], in1[27], in2[27]);
xnor(xnor_[28], in1[28], in2[28]);
xnor(xnor_[29], in1[29], in2[29]);
xnor(xnor_[30], in1[30], in2[30]);
xnor(xnor_[31], in1[31], in2[31]);
xnor(xnor_[32], in1[32], in2[32]);
xnor(xnor_[33], in1[33], in2[33]);
xnor(xnor_[34], in1[34], in2[34]);
xnor(xnor_[35], in1[35], in2[35]);
xnor(xnor_[36], in1[36], in2[36]);
xnor(xnor_[37], in1[37], in2[37]);
xnor(xnor_[38], in1[38], in2[38]);
xnor(xnor_[39], in1[39], in2[39]);
xnor(xnor_[40], in1[40], in2[40]);
xnor(xnor_[41], in1[41], in2[41]);
xnor(xnor_[42], in1[42], in2[42]);
xnor(xnor_[43], in1[43], in2[43]);
xnor(xnor_[44], in1[44], in2[44]);
xnor(xnor_[45], in1[45], in2[45]);
xnor(xnor_[46], in1[46], in2[46]);
xnor(xnor_[47], in1[47], in2[47]);
```

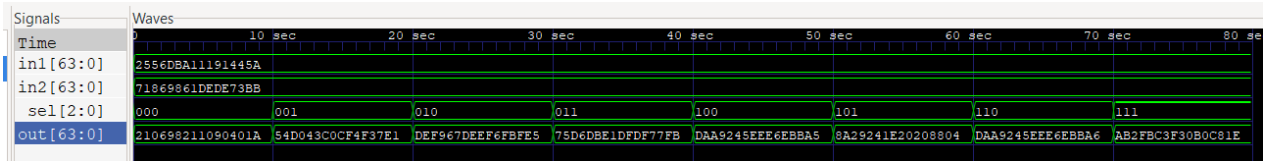
```
    xnor(xnor_[48], in1[48], in2[48]);
    xnor(xnor_[49], in1[49], in2[49]);
    xnor(xnor_[50], in1[50], in2[50]);
    xnor(xnor_[51], in1[51], in2[51]);
    xnor(xnor_[52], in1[52], in2[52]);
    xnor(xnor_[53], in1[53], in2[53]);
    xnor(xnor_[54], in1[54], in2[54]);
    xnor(xnor_[55], in1[55], in2[55]);
    xnor(xnor_[56], in1[56], in2[56]);
    xnor(xnor_[57], in1[57], in2[57]);
    xnor(xnor_[58], in1[58], in2[58]);
    xnor(xnor_[59], in1[59], in2[59]);
    xnor(xnor_[60], in1[60], in2[60]);
    xnor(xnor_[61], in1[61], in2[61]);
    xnor(xnor_[62], in1[62], in2[62]);
    xnor(xnor_[63], in1[63], in2[63]);

    SELECT select1(sel, and_, nand_, or_, nor_, xor_, not_, t_comp, xnor_, out);
endmodule

module LOGIC_UNIT_TB ;
    reg[63:0] in1, in2;
    reg[2:0] sel;

    wire[63:0] out;

    LOGIC_UNIT logic_unit1(in1, in2, sel, out);

    initial begin
```

```
        #0
in1=64'b0010010101010110110110111010000100010001100100010100010001011010;in2
=64'b0111000110000110100110000110000111011110110111100111001110111011;
sel=3'b000;


        #10 sel=3'b001;
        #10 sel=3'b010;
        #10 sel=3'b011;
        #10 sel=3'b100;
        #10 sel=3'b101;
        #10 sel=3'b110;
        #10 sel=3'b111;


        #10;
    end


    initial begin
        $monitor("in1=%b \nin2=%b \nsel=%b \nout=%b\n",  in1, in2, sel, out);
        $dumpfile("logic_unit.vcd");
        $dumpvars(0, logic_unit1);
    end
endmodule
```

Output:

```
C:\Users\rammo\OneDrive\Documents\CA\Lab\Lab7>iverilog -o lg logic_unit.v

C:\Users\rammo\OneDrive\Documents\CA\Lab\Lab7>vvp lg
VCD info: dumpfile logic_unit.vcd opened for output.
in1=0010010101010110110110111010000100010001100100010100010001011010
in2=0111000110000110100110000110000111011110110111001110011101110111
sel=000
out=0010000100000110100110000010000100010000100100000100000000011010

in1=0010010101010110110110111010000100010001100100010100010001011010
in2=0111000110000110100110000110000111011110110111001110011101110111
sel=001
out=0101010011010000010000111100000011001111010011110011011111100001

in1=0010010101010110110110111010000100010001100100010100010001011010
in2=0111000110000110100110000110000111011110110111001110011101110111
sel=010
out=1101111011111001011001111101111011101111011011111011111111100101

in1=0010010101010110110110111010000100010001100100010100010001011010
in2=0111000110000110100110000110000111011110110111001110011101110111
sel=011
out=0111010111010110110110111110000111011111110111110111011111111011

in1=0010010101010110110110111010000100010001100100010100010001011010
in2=0111000110000110100110000110000111011110110111001110011101110111
sel=100
out=1101101010101001001001000101111011101110011011101011101110100101

in1=0010010101010110110110111010000100010001100100010100010001011010
in2=0111000110000110100110000110000111011110110111001110011101110111
sel=101
out=1000101000101001001001000001111000100000001000001000100000000100

in1=0010010101010110110110111010000100010001100100010100010001011010
in2=0111000110000110100110000110000111011110110111001110011101110111
sel=110
out=1101101010101001001001000101111011101110011011101011101110100110

in1=0010010101010110110110111010000100010001100100010100010001011010
in2=0111000110000110100110000110000111011110110111001110011101110111
sel=111
out=1010101100101111101111000011111100110000101100001100100000011110
```

Wave:


Signals
Waves
Time
in1[63:0]
in2[63:0]
sel[2:0]
out[63:0]
10 sec
20 sec
30 sec
40 sec
50 sec
60 sec
70 sec
2556DBA11191445A
71869861DEDE73BB
000
001
010
011
100
101
110
111
2106982110904O1A
54D043C0CF4F37E1
DEF967DEEF6FBFE5
75D6DBE1DFDF77FB
DAA9245EEE6EBBA5
8A29241E20208804
DAA9245EEE6EBBA6
AB2FBC3F30B0C81E