Class: CSE320 Digital Design and Synthesis

Time: T-TH 4:30PM

Semester: Spring 2014

Lab 1 Report

The purpose of this assignment is two-fold. First, it is to gain experience in writing Verilog behavioral and structural descriptions of a most common component in digital systems, namely, combinational adders, and to perform functional and (abstract) timing simulation. Second is to learn some remarkably simple ways of increasing the performance of adders.

**Part 1**

**Answers:**

a)

module a\_full\_adder(sum,c\_out,a,b,c\_in);

input a,b,c\_in;

output sum, c\_out;

wire g,p,pc;

// Internal nets

wire s1, c1, c2;

// Logic gates

xor #2 (s1, a, b);

and #1 (c1, a, b);

xor #2 (sum, s1, c\_in);

and #1 (c2, s1, c\_in);

or #1 (c\_out, c2, c1);

endmodule

b)

module a\_n\_rca(sum, c\_out, a, b, c\_in);

parameter n = 4;

input [n-1:0] a,b;

input c\_in;

output [n-1:0] sum;

output c\_out;

wire [n-1:0] gen, prop;

wire [n:0] cc; // internal carry bits

assign cc[0] = c\_in;

assign c\_out = cc[n];

genvar j;

generate

for(j=0 ; j<n ; j=j+1) begin

a\_full\_adder FA(sum[j],cc[j+1],a[j],b[j],cc[j]);

end

endgenerate

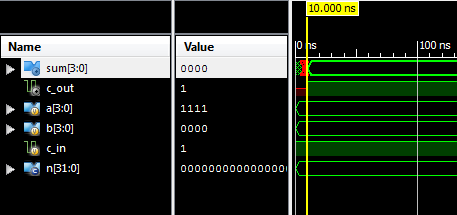
endmodule

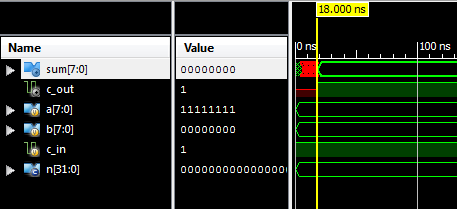
c)

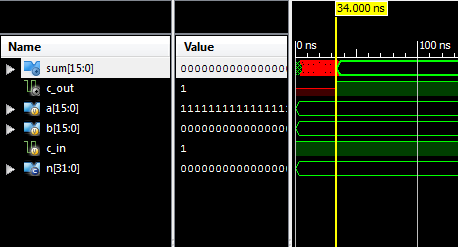
|  |  |
| --- | --- |
| n-bit rca | ns |
| 4 | 10 |
| 8 | 18 |
| 16 | 34 |
| 32 | 66 |
| 64 | 130 |
| 128 | 258 |

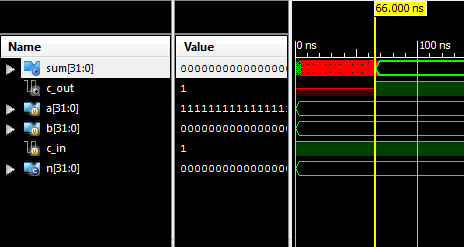
Explanation: The delay time (in nanosecond) equals nearly the double of the n-bits of the RCA. As shown in the plot, when the RCA bit capacity increases, the delay times increases.

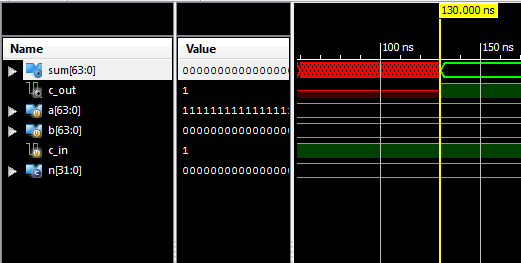
**Wave forms:**

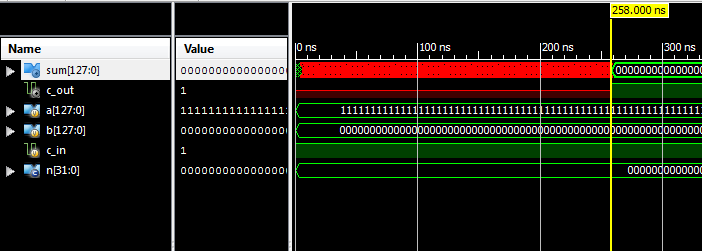
**N=4 **

**N=8 **

**N=16 **

**N=32 **

**N=64 **

**N=128 **

**Code:**

module a\_full\_adder(sum,c\_out,a,b,c\_in);

input a,b,c\_in;

output sum, c\_out;

// Internal nets

wire s1, c1, c2;

// Logic gates

xor #2 (s1, a, b);

and #1 (c1, a, b);

xor #2 (sum, s1, c\_in);

and #1 (c2, s1, c\_in);

or #1 (c\_out, c2, c1);

endmodule

module a\_n\_rca(sum, c\_out, a, b, c\_in);

parameter n = 4;

input [n-1:0] a,b;

input c\_in;

output [n-1:0] sum;

output c\_out;

wire [n-1:0] gen, prop;

wire [n:0] cc; // internal carry bits

assign cc[0] = c\_in;

assign c\_out = cc[n];

genvar j;

generate

for(j=0 ; j<n ; j=j+1) begin

a\_full\_adder FA(sum[j],cc[j+1],a[j],b[j],cc[j]);

end

endgenerate

endmodule

module a\_n\_rca\_tb;

parameter n=4;

// Inputs

reg [n-1:0] a;

reg [n-1:0] b;

reg c\_in;

// Outputs

wire [n-1:0] sum;

wire c\_out;

// Instantiate the Unit Under Test (UUT)

a\_n\_rca #(n) uut (

.sum(sum),

.c\_out(c\_out),

.a(a),

.b(b),

.c\_in(c\_in)

);

initial begin

// Add stimulus here

a=4'hF;b=4'h0;c\_in=1;

#700 $finish;

End

endmodule

**Part 2**

**Answers:**

a)

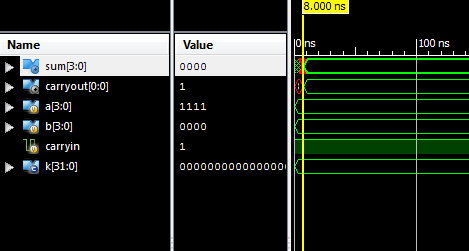
|  |  |  |
| --- | --- | --- |
| K | N-bit cba | ns |
| 1 | 4 | 8 |
| 2 | 8 | 16 |
| 4 | 16 | 32 |
| 8 | 32 | 64 |
| 16 | 64 | 128 |
| 32 | 128 | 256 |

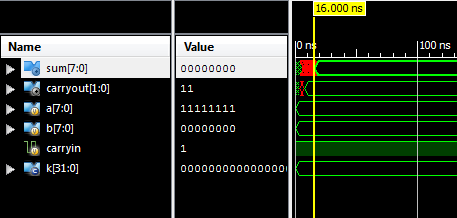
Explanation: The delay time (in nanosecond) M (4 in this case) multiplied with the half value of N (bits). As shown in the plot, when the RCA bit capacity increases, the delay times increases by the double.

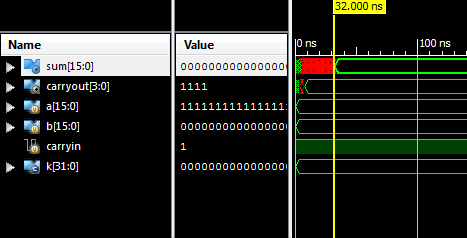
b)

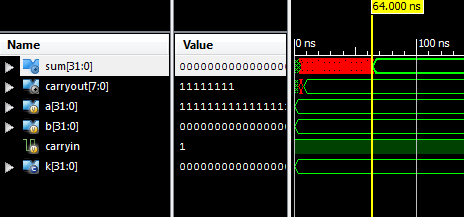
delay = M \* (N/2) (M=4 in this example)

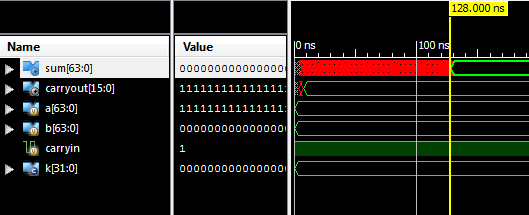
**Wave forms:**

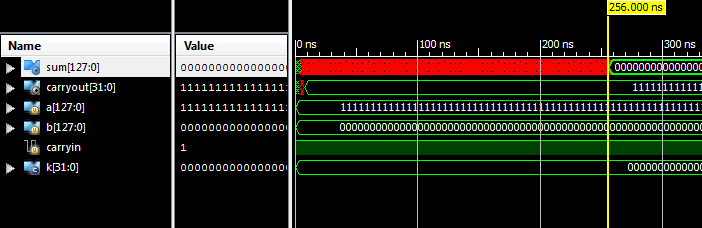
**K=1, N=4 **

**K=2, N=8 **

**K=4, N=16 **

**K=8, N=32 **

**K=16, N=64 **

**K=32, N=128 **

**Code:**

// CARRY-SKIP ADDER

module a\_n\_cba(a,b,carryin,sum,carryout);

parameter m = 4;

parameter k = 2;

parameter n = k\*m;

input [n-1:0] a, b; /\* add these bits \*/

input carryin; /\* carry in\*/

output [n-1:0] sum; /\* result \*/

output [k-1:0] carryout;

wire [n:0] carry; /\* transfers the carry between bits \*/

wire [n-1:0] p; /\* propagate for each bit \*/

assign carry[0] = carryin;

genvar i,j;

generate

for(i=1 ; i<=k ; i=i+1) begin

for(j=0 ; j<i\*m ; j=j+1) begin

a\_skip\_fulladder A(a[j],b[j],carry[j],sum[j],carry[j+1],p[j]);

end

assign #2 carryout[i-1] = carry[j] | ((p[j-4] & p[j-3] & p[j-2] & p[j-1] & carryin));

end

endgenerate

endmodule

module a\_skip\_fulladder(a,b,carryin,sum,carryout,p);

input a,b,carryin;

output sum,carryout,p;

assign #2 {carryout,sum}=a+b+carryin;

assign #2 p=a^b;

endmodule

module a\_n\_cba\_tb;

parameter k=1;

// Inputs

reg [4\*k-1:0] a;

reg [4\*k-1:0] b;

reg carryin;

// Outputs

wire [4\*k-1:0] sum;

wire [k-1:0] carryout;

// Instantiate the Unit Under Test (UUT)

a\_n\_cba #(.k(k)) uut (

.a(a),

.b(b),

.carryin(carryin),

.sum(sum),

.carryout(carryout)

);

initial begin

// Add stimulus here

a=8'hFF;b=8'h00;carryin=1;

#700 $finish;

end

endmodule

**Part 3**

**Answers:**

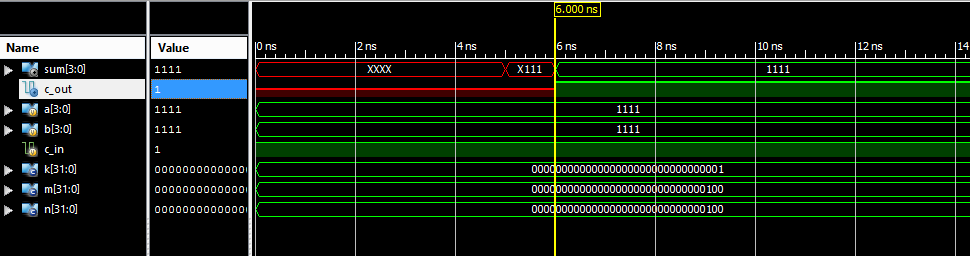
a)

|  |  |  |
| --- | --- | --- |
| k | N-bit CSA | ns |
| 1 | 4 | 6 |
| 2 | 8 | 10 |
| 4 | 16 | 18 |
| 8 | 32 | 34 |
| 16 | 64 | 66 |
| 32 | 128 | 130 |

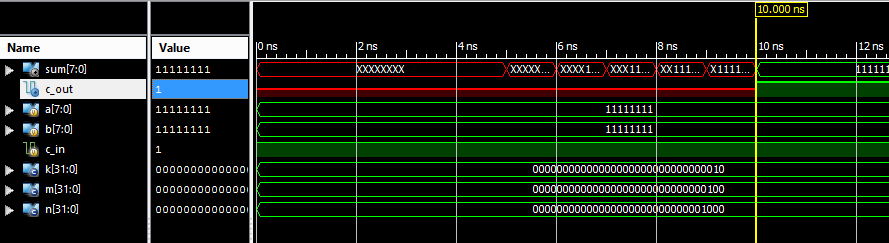
1. delay = (k\*M)+2 (M=4 In Examples)

**Wave forms:**

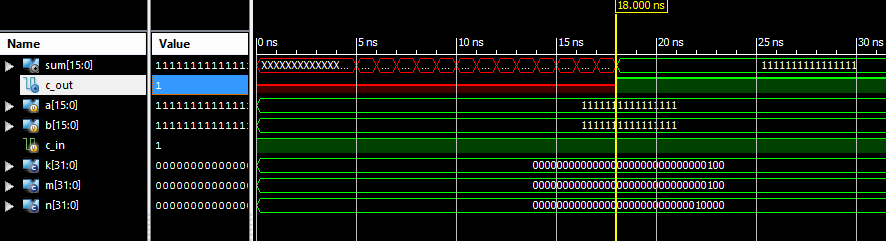
**K=1, N=4**



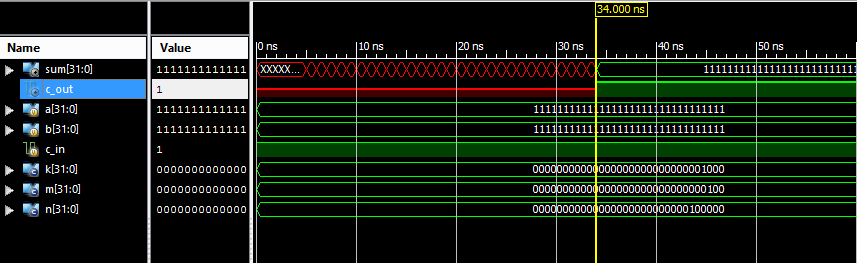
**K=2, N=8**



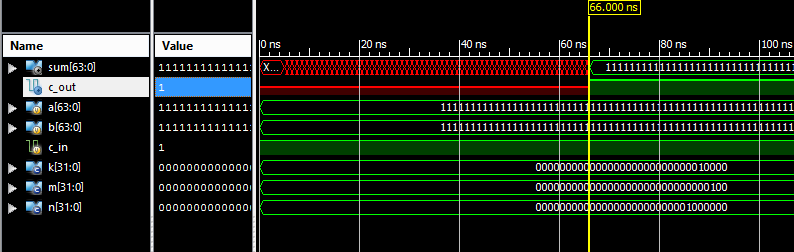
**K=4, N=16**



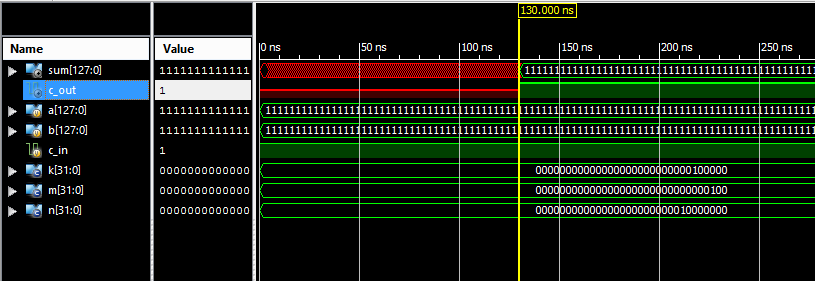
**K=8, N=32**



**K=16, N=64**



**K=32, N=128**



**Code:**

module z\_n\_csa(sum,c\_out,a,b,c\_in);

parameter k = 8;

parameter m = 4;

parameter n = k\*m;

input [n-1:0] a, b;

input c\_in;

output [n-1:0] sum;

output c\_out;

wire [n:0] carry;

wire [n-1:0] sum\_w;

assign carry[0] = c\_in;

assign c\_out = carry[n];

genvar j;

generate

for(j=0 ; j<k ; j=j+1) begin

z\_m\_sca\_stage CSAM(sum[j\*m+3:j\*m],carry[j\*m+4],a[j\*m+3:j\*m],b[j\*m+3:j\*m],carry[j\*m]);

defparam CSAM[j].m=m;

end

endgenerate

endmodule

module z\_m\_sca\_stage(sum,c\_out,a,b,c\_in);

parameter m = 4;

input [m-1:0] a, b;

input c\_in;

output [m-1:0] sum;

output c\_out;

wire [m:0] carry;

wire [m-1:0] sum\_w;

assign carry[0] = c\_in;

assign c\_out = carry[m];

genvar j;

generate

for(j=0 ; j<m ; j=j+1) begin

z\_csa\_stage CSA(sum[j],carry[j+1],a[j],b[j],carry[j]);

end

endgenerate

endmodule

module z\_csa\_stage(sum,c\_out,a,b,c\_in);

input a,b,c\_in;

output sum,c\_out;

wire [1:0] sum\_r, c\_out\_r;

// Full Adders for both c\_in = 0 & 1

a\_full\_adder FA1(sum\_r[0],c\_out\_r[0],a,b,0);

a\_full\_adder FA2(sum\_r[1],c\_out\_r[1],a,b,1);

// MUX - Starts at the same time as the FA so no delay

z\_2\_1\_mux M1(sum,sum\_r[0],sum\_r[1],c\_in);

z\_2\_1\_mux M2(c\_out,c\_out\_r[0],c\_out\_r[1],c\_in);

Endmodule

module a\_full\_adder(sum,c\_out,a,b,c\_in);

input a,b,c\_in;

output sum, c\_out;

wire g,p,pc;

// Internal nets

wire s1, c1, c2;

// Logic gates

xor #2 (s1, a, b);

and #1 (c1, a, b);

xor #2 (sum, s1, c\_in);

and #1 (c2, s1, c\_in);

or #1 (c\_out, c2, c1);

endmodule

module z\_2\_1\_mux(out,a,b,sel);

output out;

input a,b,sel;

wire sel\_,a1,b1,out;

not not1(sel\_,sel);

and #1 and1(a1,a,sel\_);

and #1 and2(b1,b,sel);

or or1(out,a1,b1);

endmodule

**Part 4**

**Answers:**

a)

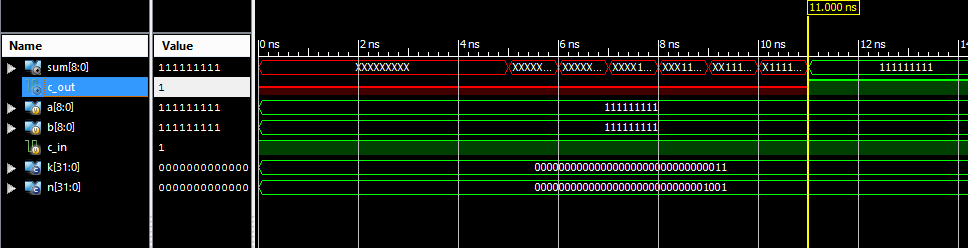
|  |  |  |
| --- | --- | --- |
| k | N-bit CSA | ns |
| 3 | 9 | 11 |
| 5 | 20 | 22 |
| 7 | 35 | 37 |
| 10 | 65 | 67 |
| 15 | 135 | 137 |

b) delay = ((k+1)(k+2)/2)+1

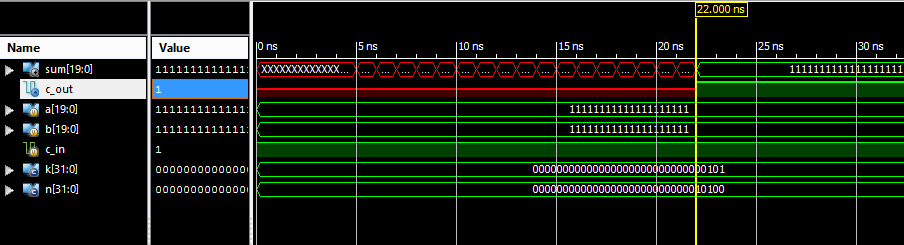
This result is because by the time that the carry out has left section n-1 the inputs for the mux of section 2 are then ready for the carry in. This cycle continues up to the end and all of the delay is generated by the MUX.

**Wave forms:**

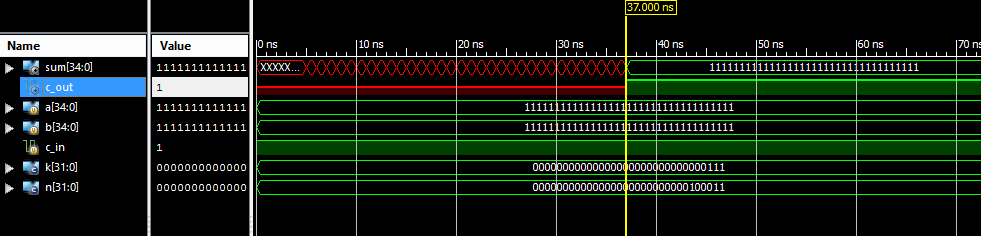
**K=3, N=9**



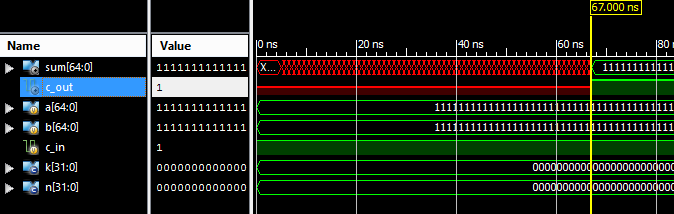
**K=5, N=20**



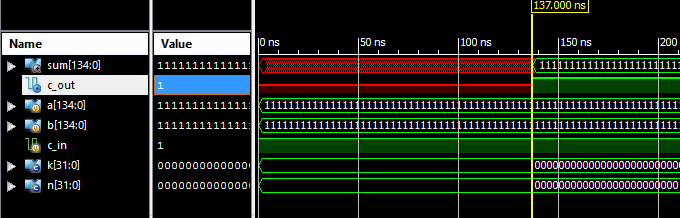
**K=7, N=35**



**K=10, N=65**



**K=15, N=135**



**Code:**

module z\_sqcsa(sum,c\_out,a,b,c\_in);

parameter k = 15;

parameter n = ((k+1)\*(k+2)/2)-1;

input [n-1:0] a, b;

input c\_in;

output [n-1:0] sum;

output c\_out;

wire [n:0] carry;

wire [n-1:0] sum\_w;

assign carry[0] = c\_in;

assign c\_out = carry[k];

genvar j;

generate

for(j=0 ; j<k ; j=j+1) begin

parameter m = ((j+2)\*(j+3)/2)-2;

parameter m\_last = ((j+1)\*(j+2)/2)-1;

z\_m\_sca\_stage CSAM(sum[m:m\_last],carry[j+1],a[m:m\_last],b[m:m\_last],carry[j]);

defparam CSAM[j].m=m+1-m\_last;

end

endgenerate

endmodule

module z\_m\_sca\_stage(sum,c\_out,a,b,c\_in);

parameter m = 4;

input [m-1:0] a, b;

input c\_in;

output [m-1:0] sum;

output c\_out;

wire [m:0] carry;

wire [m-1:0] sum\_w;

assign carry[0] = c\_in;

assign c\_out = carry[m];

genvar j;

generate

for(j=0 ; j<m ; j=j+1) begin

z\_csa\_stage CSA(sum[j],carry[j+1],a[j],b[j],carry[j]);

end

endgenerate

endmodule

module z\_csa\_stage(sum,c\_out,a,b,c\_in);

input a,b,c\_in;

output sum,c\_out;

wire [1:0] sum\_r, c\_out\_r;

// Full Adders for both c\_in = 0 & 1

a\_full\_adder FA1(sum\_r[0],c\_out\_r[0],a,b,0);

a\_full\_adder FA2(sum\_r[1],c\_out\_r[1],a,b,1);

// MUX - Starts at the same time as the FA so no delay

z\_2\_1\_mux M1(sum,sum\_r[0],sum\_r[1],c\_in);

z\_2\_1\_mux M2(c\_out,c\_out\_r[0],c\_out\_r[1],c\_in);

endmodule

module a\_full\_adder(sum,c\_out,a,b,c\_in);

input a,b,c\_in;

output sum, c\_out;

wire g,p,pc;

// Internal nets

wire s1, c1, c2;

// Logic gates

xor #2 (s1, a, b);

and #1 (c1, a, b);

xor #2 (sum, s1, c\_in);

and #1 (c2, s1, c\_in);

or #1 (c\_out, c2, c1);

endmodule

module z\_2\_1\_mux(out,a,b,sel);

output out;

input a,b,sel;

wire sel\_,a1,b1,out;

not not1(sel\_,sel);

and #1 and1(a1,a,sel\_);

and #1 and2(b1,b,sel);

or or1(out,a1,b1);

endmodule

Last Page:

This is entirely our work and we have not received assistance in any form whatsoever.

Signatures:

Zachary Priddy Abdulla Al Braiki

------------------- ----------------------