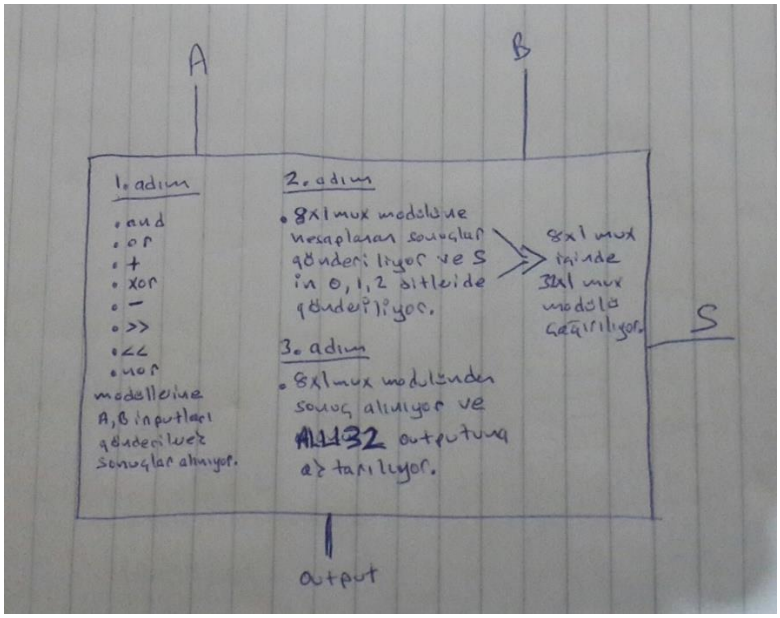
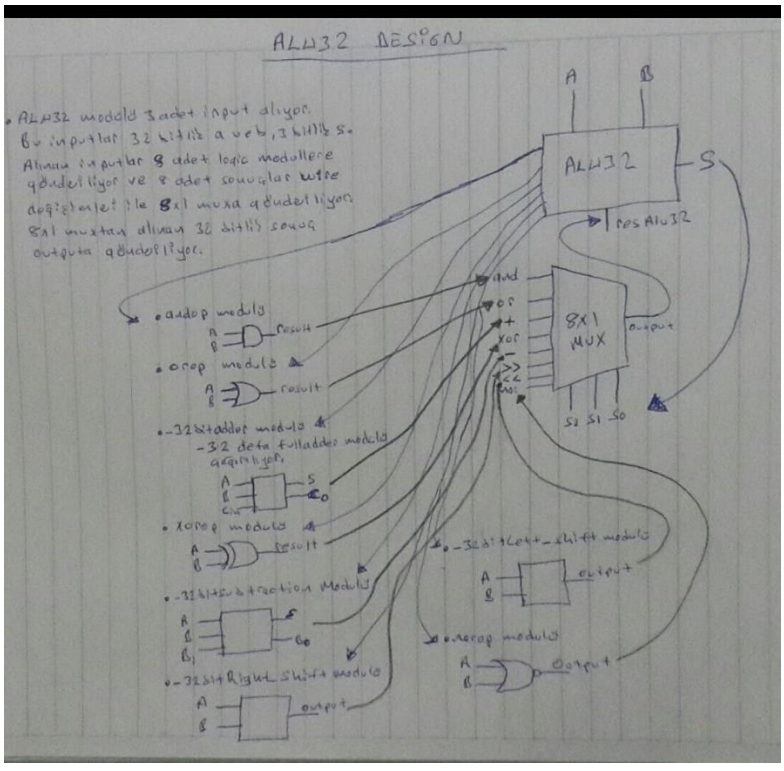
HOMEWORK 2 – 151044085 – EMRE KAVAK

MODÜLLER:

1- alu32 modülü : Bu modülde 32 bitlik 2 adet **a** ve **b**, 3 bitlik **s(select)** inputları alınmaktadır. Bu inputlar alındıktan sonra modül içinde yapmamız istenen **8 adet(and,or,+,xor, -,>>,<<,nor)** işlemin modülleri çağırılmaktadır. Çağırılan bu modüllerin sonuçları **wire** değişkenlerinde tutularak , _**8x1mux** modülüne gönderilmektedir. _8x1 modülü 8 adet 32 bitlik , 1 adet 3 bitlik input almaktadır. Alınan bu inputlar _**32bitmux** modülüne, sırasıyla I0,I1…I7, şeklinde gönderiliyor ve s bitinin sırasıyla s[0], s[1] ve s[2] bitleri gönderilerek hangi inputun outputa gönderileceği belirleniyor. _8x1mux modülünde en son çağırılan _32bitmux modülünün outputu alınarak, _8x1mux modülünün outputuna gönderiliyor. Oluşan bu output, son olarakda alu32 icinde ki resultAlu32 outputuna atanarak istenilen işlemin sonucu elde edilmiş oluyor.

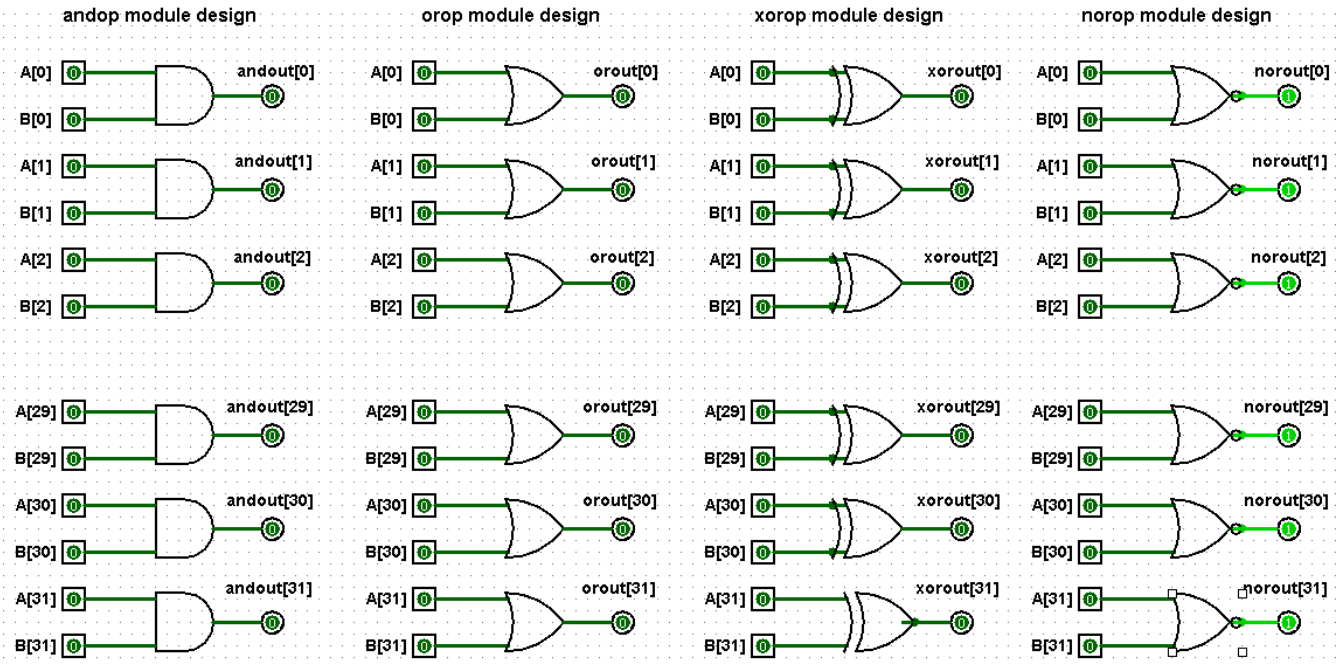




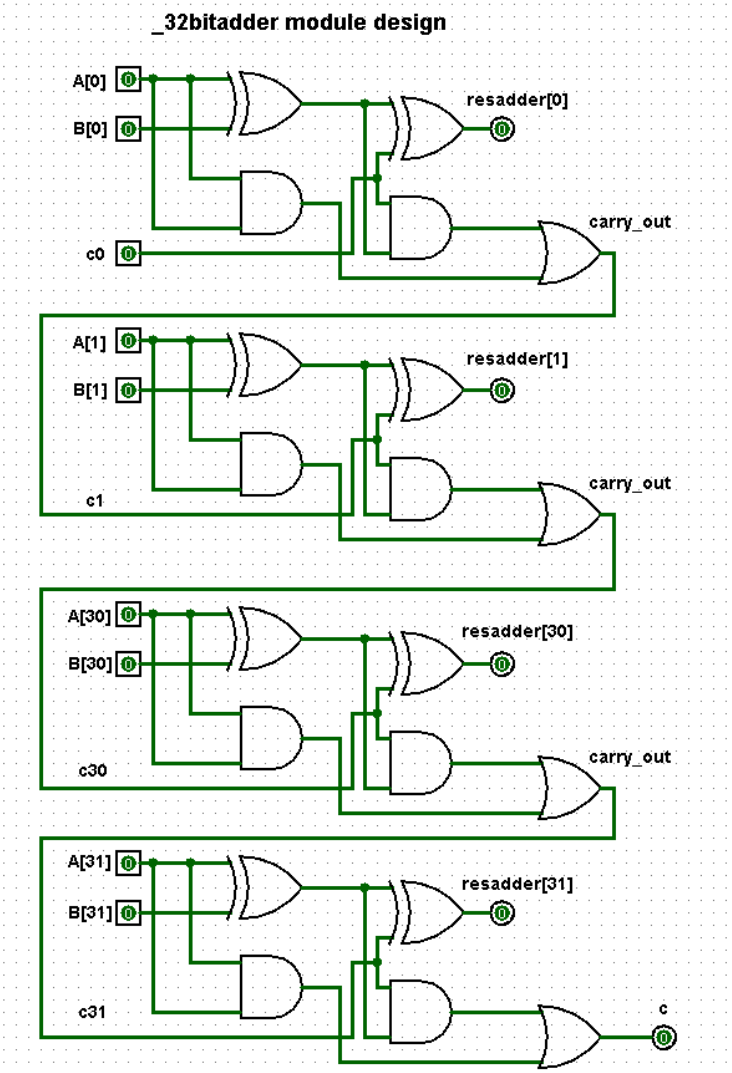
2- andop modülü : Bu modül 2 adet 32 bitlik input alıyor (A,B), sırasıyla A[0]B[0],A[1]B[1]…A[31][B31] şeklinde and işlemlerini tek tek uygulayarak 32 bitlik output değişkenine atıyor.

3- orop modülü : Bu modül de andop modülü gibi 2 adet 32 bitlik input alıyor (A,B), sırasıyla A[0]B[0],A[1]B[1]…A[31][B31] şeklinde or işlemlerini tek tek uygulayarak 32 bitlik output değişkenine atıyor.

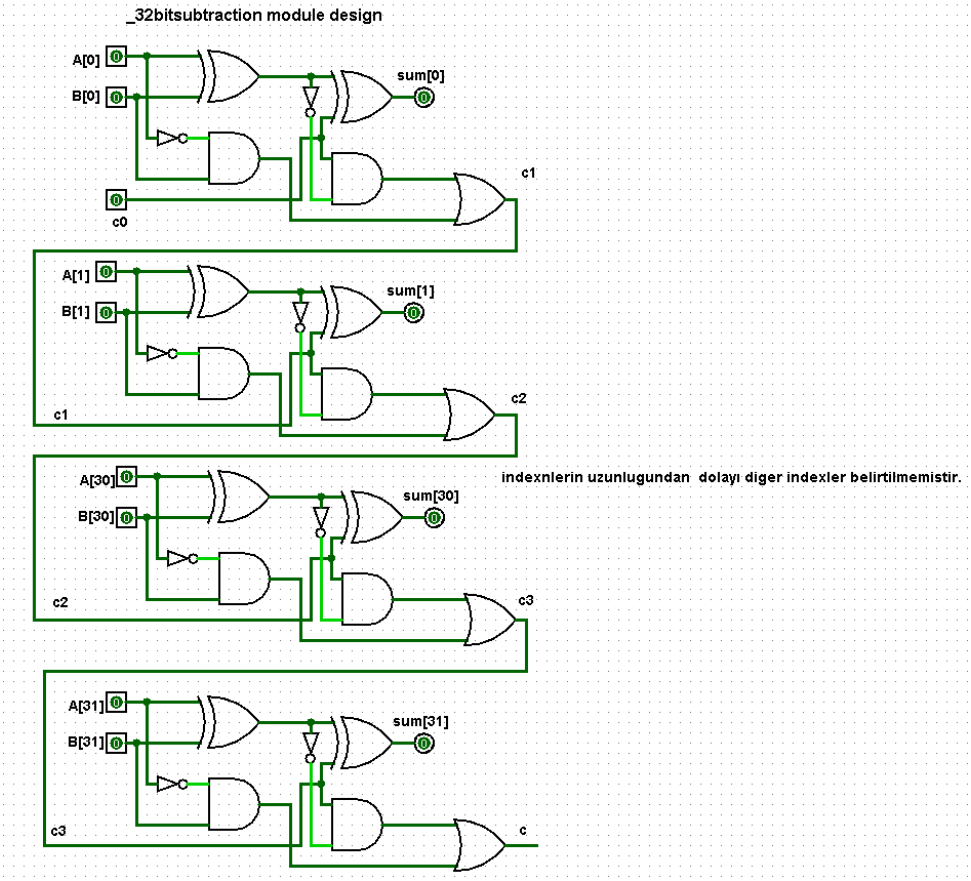
4- xorop modülü : Bu modül de andop ve orop modülü gibi 2 adet 32 bitlik input alıyor (A,B), sırasıyla A[0]B[0],A[1]B[1]…A[31][B31] şeklinde xor işlemlerini tek tek uygulayarak 32 bitlik output değişkenine atıyor



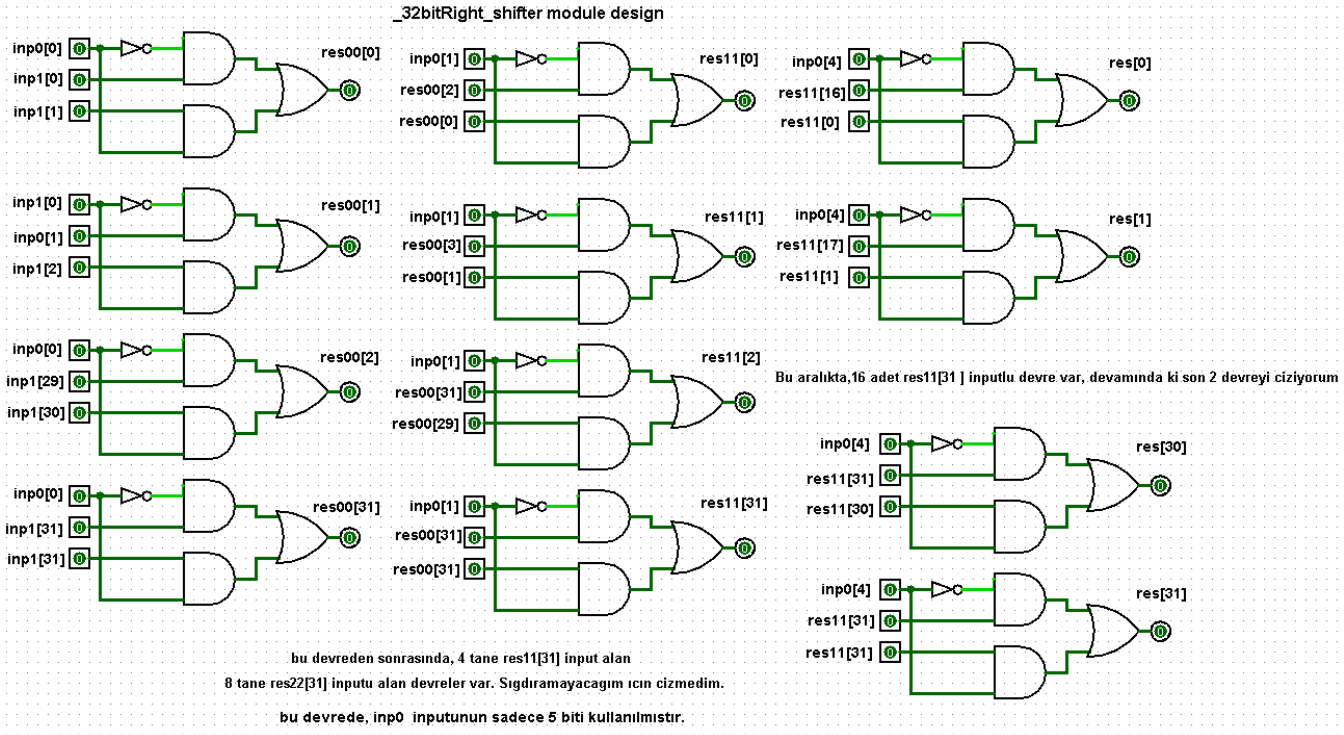
5- _32bitadder modülü : Bu modül 32 bitlik 2 input, 1 adet 1 bitlik carry_in ve 1 adetta carry_out inputu alıyor. Bu modül içerisinde **fulll_adder** modülü 32 defa çağırılarak sonuç output değişkenine atanıyor.



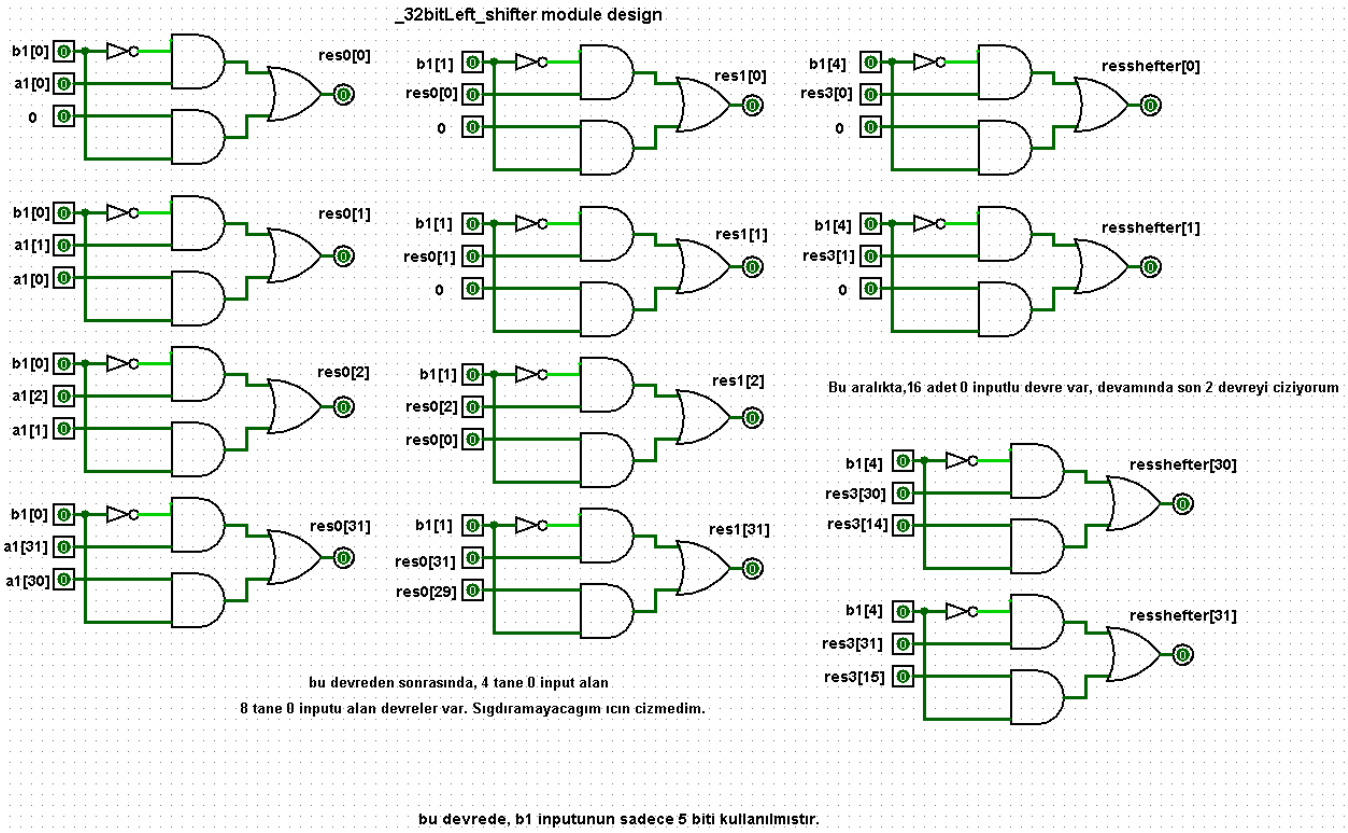
5- _32bitsubtraction modülü : Bu modülde adder modülünde olduğu gibi 2 adet 32 bitlik, 2 adet 1 bitlik inputlar alıyor. Bu 1 bitlik inputlar barrow in ve barrow out için kullanılıyor. Bu modüle içinde fullsubtraction modülü 32 defa çağırılıyor.



6- _32bitRight_shifter modülü : Bu modül sadece 32 bitlik inp0 ve inp1 inputlarını alıyor. İnp0 inputunun ilk 5 biti kullanılarak inp1 inputu right shift ediliyor. Shift etmek için _2x1mux modülü 160 defa çağırılıyor. Bunun bu kadar fazla olmasının nedeni, 31 bit kaydırma yapılabilmek ihtimalidir. _2x1mux modülü 3 adet 1 er bitlik input alıyor. Bu modülün ilk parametresi olarak inp1[0][1]…[31],ikinci parametre olarak ise inp0[0][1]…[31] ve inp0 ın ilk 5 biti gönderiliyor. Ancak, inputların yerleri , shift edilebilmesi için değiştiriliyor. Aşağıda ki logic devrede de görülebileceği gibi, shifter edilmek istendiği kadar inp1[31] biti , diğer muxların sondan başlanarak girişlerine gönderiliyor. Çünkü sağ shift etmemiz istenmektedirbu yüzden Most significant biti ne ise , kaydırdıkça o değerin eklenmesi gerekiyor._2x1 muxlarından elde edilen outputlar, bu modülün outputuna atanarak sonuç elde edilmiş oluyor.

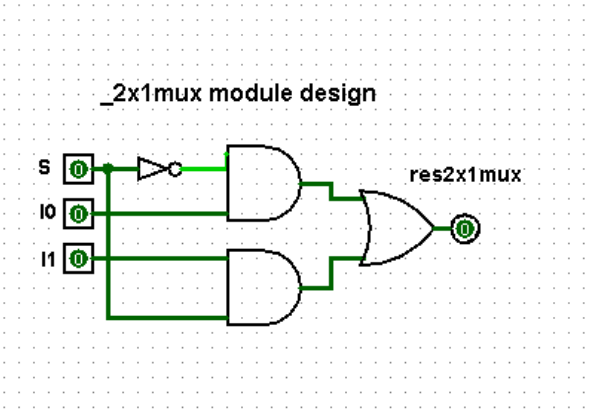


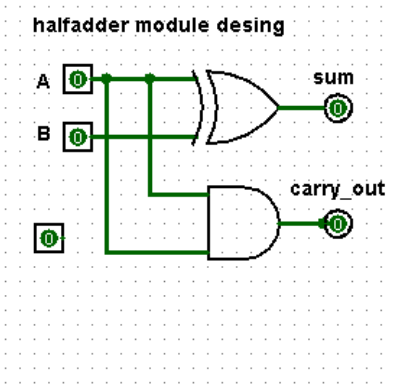
7- _32bitLeft_shifter modülü : Bu modülde 2 32 bitlik a ve b inputlarını alıyor.b inputunun ilk 5 biti left shift etmek için kullanılıyor. Bir önceki right shifterdan farklı olarak, left shift ederken, kaydırdığımız sayı yerine 0 inputu gönderiyoruz ve bu sefer 0. Bitten başlayarak bu durumu gerçekleştiriyoruz. Önceki modülde hatırlanacağı gibi sondan başlamıştık. Aşağıda logic diyagramda modülün iç yapısı anlatılmaktadır.

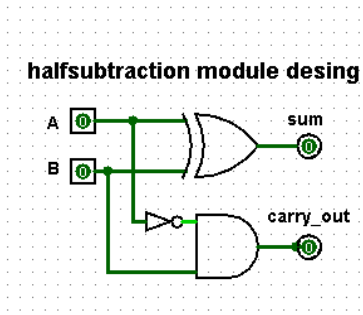


8- norop modülü : Bu modül de andop ve diğer 3 modül gibi modülü gibi 2 adet 32 bitlik input alıyor (A,B), sırasıyla A[0]B[0],A[1]B[1]…A[31][B31] şeklinde nor işlemlerini tek tek uygulayarak 32 bitlik output değişkenine atıyor. (modül tasarımı gatelerin toplu olarak gösterildiği ekran görüntüsünde mevcut).

DİĞER MODÜLLERİN LOGİC TASARIMLARI







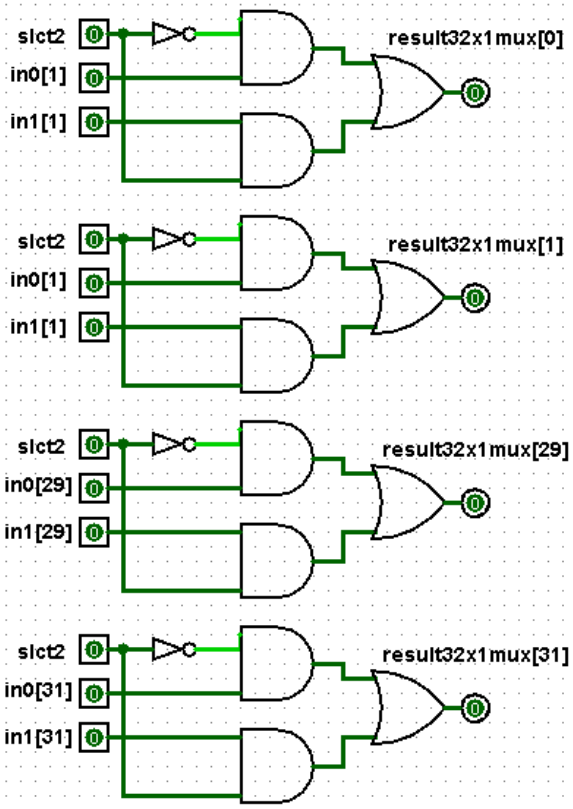
_8x1mux module design

logic işlemlerinin sonuçları 8x1 muxa input olarak 8 tane geliyor

8x1 mux icinde, 32x1 lik mux modulu cagırılıyor ve select bitlerine göre hangi sonuc outputa verilcek seciliyor

_32bitmux module design

32 defa 2x1 mux çagrılıyor.



MODELSİM OUTPUTLARI VE SONUCLARI

AND

A =11111111111111111111111111111111 **B**=11111111111111111111111111111111 **S**= 000

result-->11111111111111111111111111111111

A =10101010101010101010101010101010 **B**=01010101010101010101010101010101 **S**= 000

result-->00000000000000000000000000000000

A =10100010010000011111111111111111 **B**=10100010010000011111111111111111 **S**= 000

result-->10100010010000011111111111111111

OR

A =11111111111111111111111111111111 **B**=00000000000000000000000000000000 **S**= 001

result-->11111111111111111111111111111111

A =10101010101010101010101010101010 **B**=01010101010101010101010101010101 **S**= 001

result-->11111111111111111111111111111111

A =00111110111100011100011101011100 **B**=11000001000001100011100010100011 **S**= 001

result-->11111111111101111111111111111111

ADDER (+)

A =11111111111111111111111111111000 **B**=00000000000000000000000000000110 **S**= 010

result-->11111111111111111111111111111110

A =10000000000000000000000000000000 **B**=01000000000000001000000000000001 **S**= 010

result-->11000000000000001000000000000001

A =01010101010101010101010101010101 **B**=01010101010101010101010101010101 **S**= 010

result-->10101010101010101010101010101010

XOR

A =10101010101010101010101010101010 B=01010101010101010101010101010101 **S**= 011

result-->11111111111111111111111111111111

A =10101010101010101010101010101010 **B**=00000000000000000000000000000000 **S**= 011

result-->10101010101010101010101010101010

A =11111111111111111111111111111111 **B**=00000000000000000000000000000000 **S**= 011

result-->11111111111111111111111111111111

SUBRACT (-)

A =10101010101010101010101010101010 **B**=10101010101010101010101010101010 **S**= 100

result-->11111111111111111111111111111111

A =00000000000000000000000010000000 **B**=00000000000000000000000001010101 **S**= 100

result-->00000000000000000000000000101010

A =01111111111111111111111111111111 **B**=01010101010101010101010101010101 **S**= 100

result-->00101010101010101010101010101001

>> RİGT SHİFTER

A =10000000000000000000000000000000 **B**=00000000000000000000000000000011 **S**= 101

result-->11110000000000000000000000000000

A =10000000000000001111000000000000 **B**=00000000000000000000000000001100 **S**= 101

result-->11111111111110000000000000001111

A =10001110000000000000000000000000 **B**=00000000000000000000000000011111 **S**= 101

result-->11111111111111111111111111111111

<< LEFT SHİFTER

A =00000000000000000000000000000001 **B**=00000000000000000000000000111111 **S**= 110

result-->10000000000000000000000000000000

A =00000000000000000000000000011111 **B**=00000000000000000000000000000101 **S**= 110

result-->00000000000000000000001111100000

A =00000000000000000011111110000000 **B**=00000000000000000000000000001100 **S**= 110

result-->00000011111110000000000000000000

NOR

A =11111111111110001111111111111111 **B**=00000000000000000000000000000000 **S**= 111

result-->00000000000001110000000000000000

A =10101010101010101010101010101010 **B**=01010101010101010101010101010101 **S**= 111

result-->00000000000000000000000000000000

A =11001100110011001101100110011000 **B**=00000000000000000000000000000000 **S**= 111

result-->00110011001100110010011001100111