Digital Design

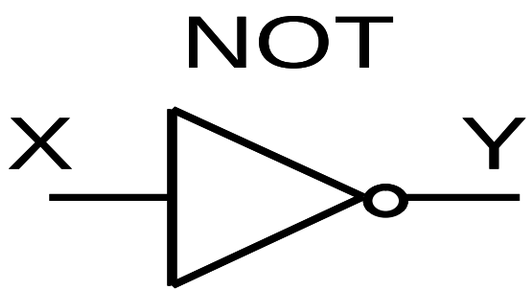
“HDL ”

Dr. Cahit Karakuş, February-2018

Basic Logic Gates
and Basic Digital Design

- **NOT, AND, and OR Gates**
- NAND and NOR Gates
- DeMorgan's Theorem
- Exclusive-OR (XOR) Gate
- Multiple-input Gates

NOT Gate -- Inverter



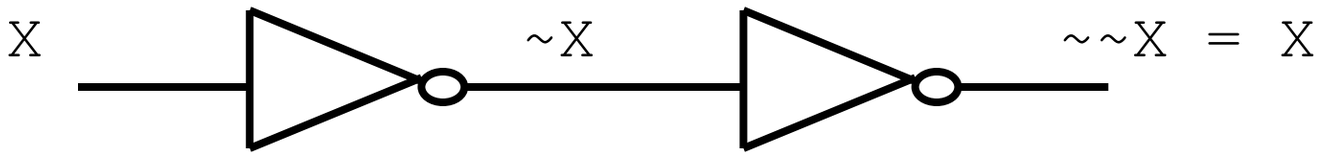
$Y = \sim X$

X	Y
0	1
1	0

NOT

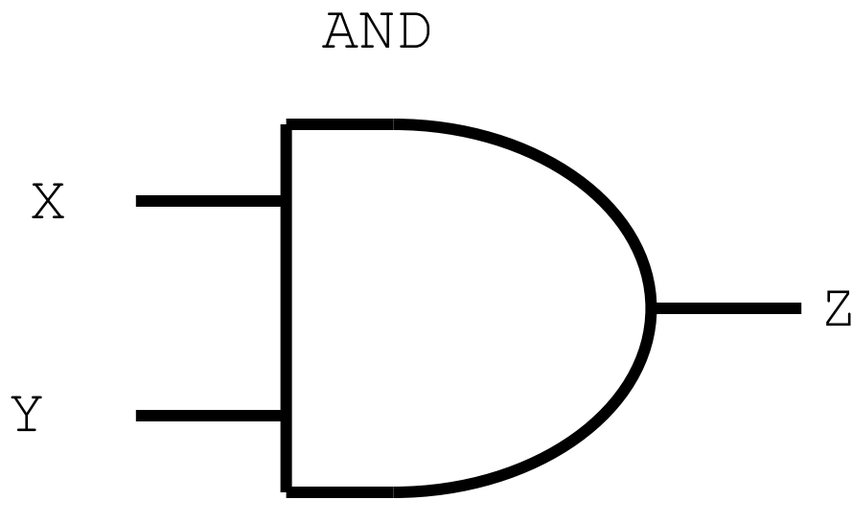
- Y = ~X (Verilog)
- Y = !X (ABEL)
- Y = **not** X (VHDL)
- Y = X'
- Y = X¬
- Y = $\overline{X}$ (textook)
- **not**(Y,X) (Verilog)

NOT



X	~X	~~X
0	1	0
1	0	1

AND Gate



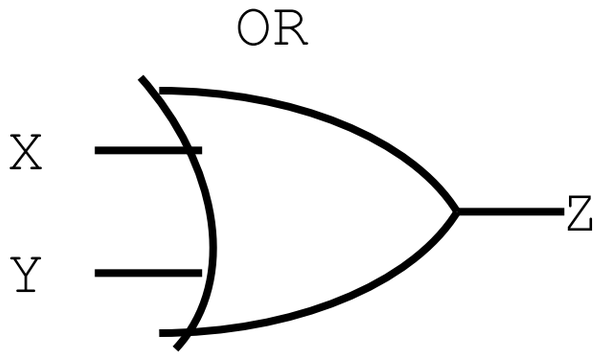
Z = X & Y

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

AND

- X & Y (Verilog and ABEL)
- X **and** Y (VHDL)
- X ∧ Y
- X ∩ Y
- X * Y
- XY (textbook)
- **and**(Z,X,Y) (Verilog)

OR Gate



Z = X | Y

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

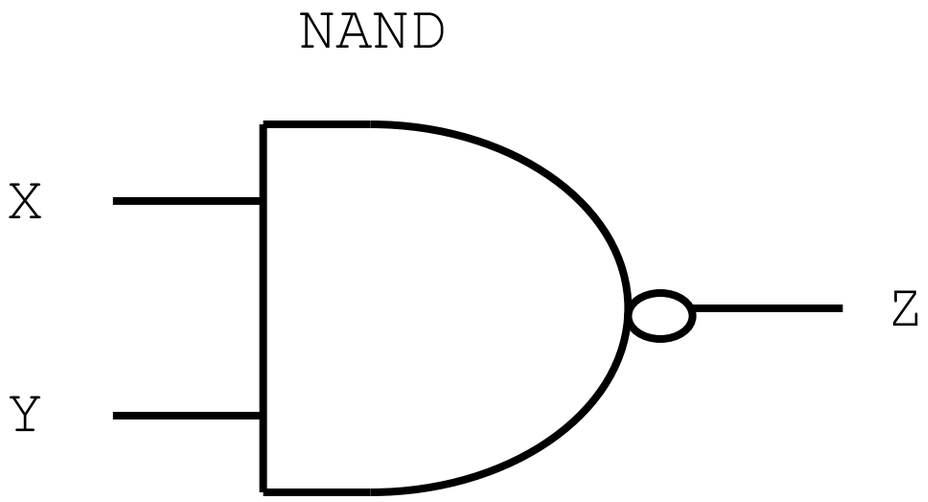
OR

- `X | Y` (Verilog)
- `X # Y` (ABEL)
- `X` **`or`** `Y` (VHDL)
- `X + Y` (textbook)
- `X` **`V`** `Y`
- `X` **`U`** `Y`
- **`or`**`(Z,X,Y)` (Verilog)

Basic Logic Gates
and Basic Digital Design

- NOT, AND, and OR Gates
- **NAND and NOR Gates**
- DeMorgan's Theorem
- Exclusive-OR (XOR) Gate
- Multiple-input Gates

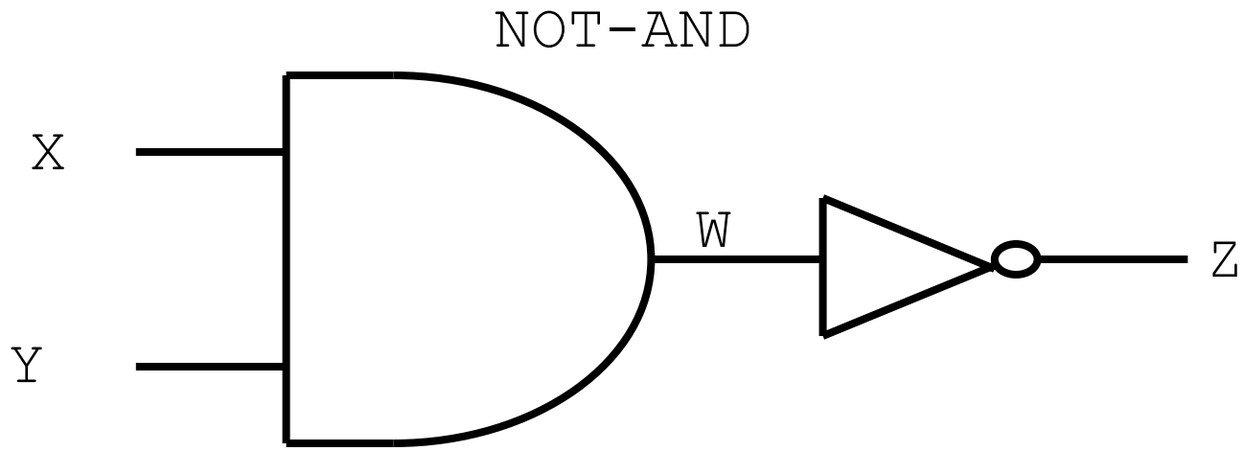
NAND Gate



X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

```
Z = ~(X & Y)
nand(Z,X,Y)
```

NAND Gate

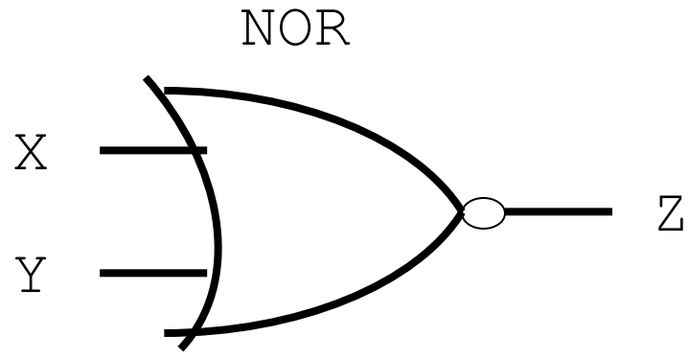


X	Y	W	Z
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

W = X & Y

Z = ~W = ~(X & Y)

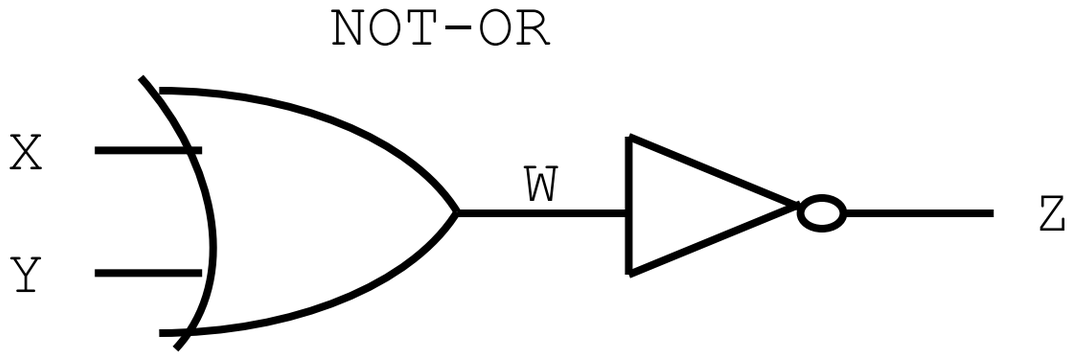
NOR Gate



X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

```
Z = ~(X | Y)
nor(Z,X,Y)
```

NOR Gate



X	Y	W	Z
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

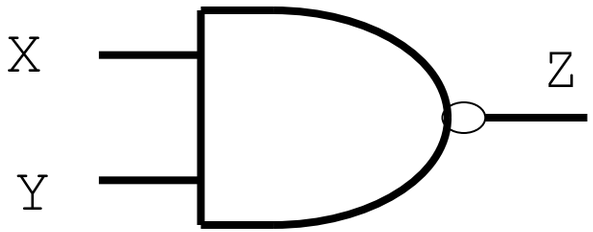
W = X | Y

Z = ~W = ~(X | Y)

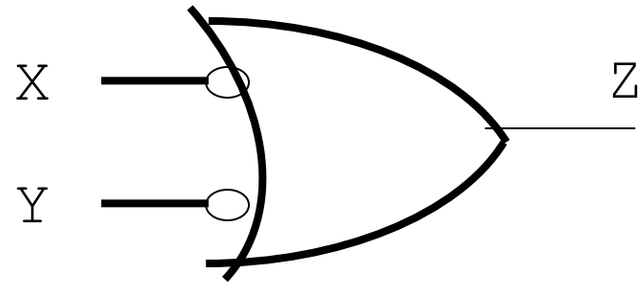
Basic Logic Gates
and Basic Digital Design

- NOT, AND, and OR Gates
- NAND and NOR Gates
- **DeMorgan's Theorem**
- Exclusive-OR (XOR) Gate
- Multiple-input Gates

NAND Gate



=



`Z = ~(X & Y)`

X	Y	W	Z
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

`Z = ~X | ~Y`

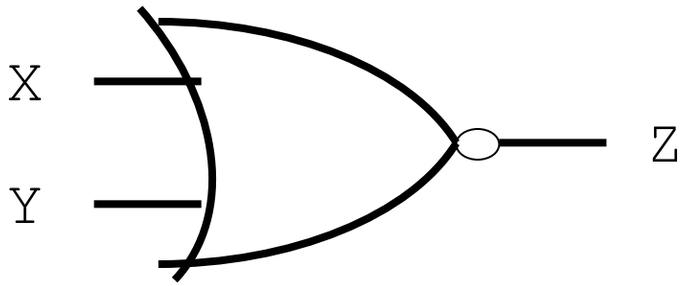
X	Y	~X	~Y	Z
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

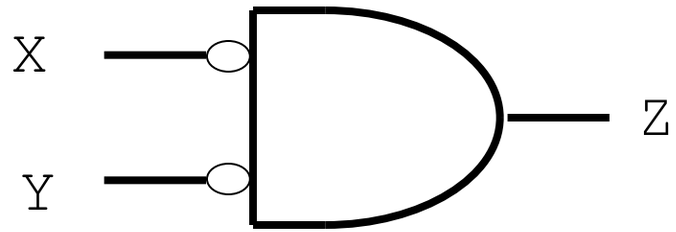
De Morgan's Theorem-1

```
~(X & Y) = ~X | ~Y
```

- NOT all variables
- Change & to | and | to &
- NOT the result

NOR Gate





`Z = ~(X | Y)`

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

`Z = ~X & ~Y`

X	Y	~X	~Y	Z
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

De Morgan's Theorem-2

```
~(X | Y) = ~X & ~Y
```

- NOT all variables
- Change & to | and | to &
- NOT the result

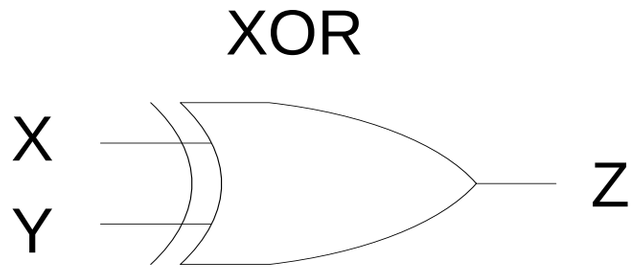
De Morgan's Theorem

- NOT all variables
- Change & to | and | to &
- NOT the result
- ----------
- ~X | ~Y = ~(~~X & ~~Y) = ~(X & Y)
- ~(X & Y) = ~~(~X | ~Y) = ~X | ~Y
- ~X & !Y = ~(~~X | ~~Y) = ~(X | Y)
- ~(X | Y) = ~~(~X & ~Y) = ~X & ~Y

Basic Logic Gates
and Basic Digital Design

- NOT, AND, and OR Gates
- NAND and NOR Gates
- DeMorgan's Theorem
- **Exclusive-OR (XOR) Gate**
- Multiple-input Gates

Exclusive-OR Gate




```
Z = X ^ Y
xor(Z,X,Y)
```

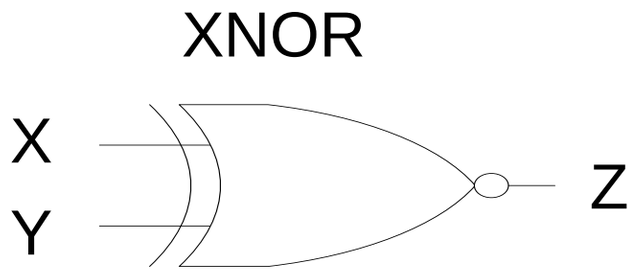
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

XOR

- `X ^ Y` (Verilog)
- `X $ Y` (ABEL)
- `X @ Y`

  g$X \oplus Y$ **(textbook)**
- **`xor`**`(Z,X,Y)` (Verilog)

Exclusive-NOR Gate




```
Z = ~(X ^ Y)
Z = X ~^ Y
xnor(Z,X,Y)
```

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	1

XNOR

- `X ~^ Y` (Verilog)
- `!(X $ Y)` (ABEL)
- `X @ Y`

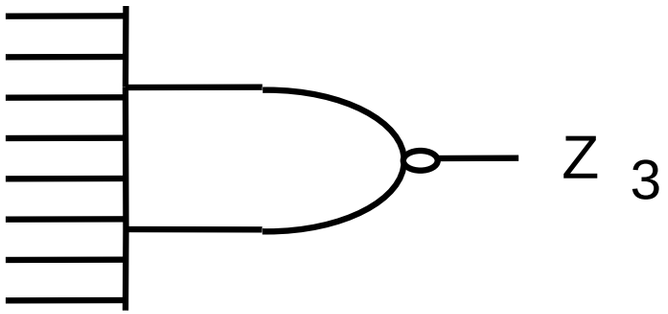
`gX e Y`

- **`xnor`**`(Z,X,Y)` (Verilog)

Basic Logic Gates
and Basic Digital Design

- NOT, AND, and OR Gates
- NAND and NOR Gates
- DeMorgan's Theorem
- Exclusive-OR (XOR) Gate
- **Multiple-input Gates**

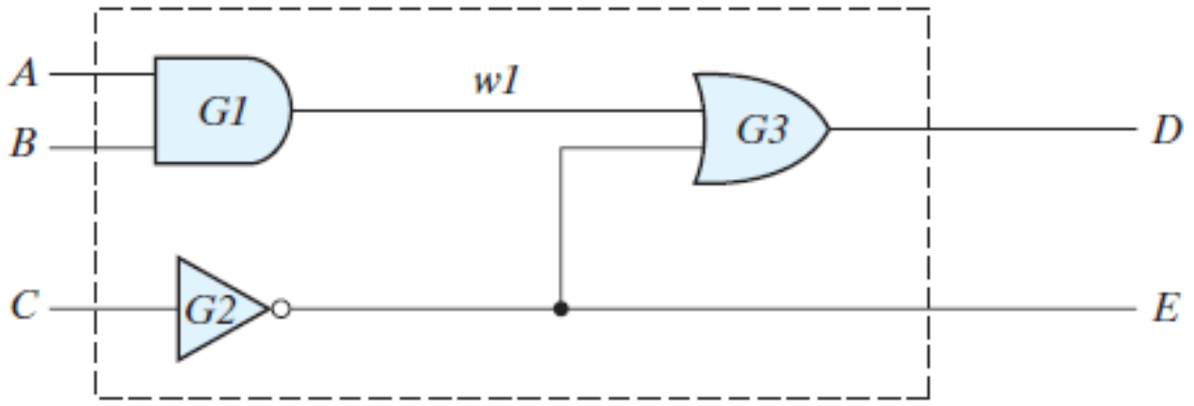
Multiple-input NAND Gate



Output Z_3 is LOW only if all inputs are HIGH

Hardware description language (HDL)

- Bir donanım tanımlama dili (HDL), sayısal sistemlerin donanımını metin biçiminde tanımlayan bilgisayar tabanlı bir dildir.
- C gibi sıradan bir bilgisayar programlama dilini andırır, ancak özellikle donanım yapılarını ve mantık devrelerinin davranışını tanımlamaya yöneliktir.
- Mantık diyagramlarını, doğruluk tablolarını, Boolean ifadelerini ve dijital sistemin davranışının karmaşık soyutlamalarını temsil etmek için kullanılabilir.
- Bir HDL'yi görüntülemenin bir yolu, bir devrenin girişleri olan sinyaller ile devrenin çıktıları olan sinyaller arasındaki ilişkiyi tanımladığını gözlemlemektir.
- Örneğin, bir AND kapısının bir HDL tanımlaması, kapının çıkışının mantık değerinin girdilerinin mantıksal değerleri tarafından nasıl belirlendiğini açıklar.



HDL Example (Combinational Logic Modeled with Primitives)

```
// Verilog model of circuit of Figure 3.35. IEEE 1364–1995 Syntax

module  Simple_Circuit (A, B, C, D, E);
  output        D, E;
  input         A, B, C;
  wire          w1;

  and           G1 (w1, A, B); // Optional gate instance name
  not           G2 (E, C);
  or            G3 (D, w1, E);
endmodule
```

HDL Example describes a circuit that is specified with the following two Boolean expressions:

$$E = A + BC + B'D$$
$$F = B'C + BC'D'$$

The equations specify how the logic values E and F are determined by the values of A, B, C, and D.

HDL Example (Combinational Logic Modeled with Boolean Equations)

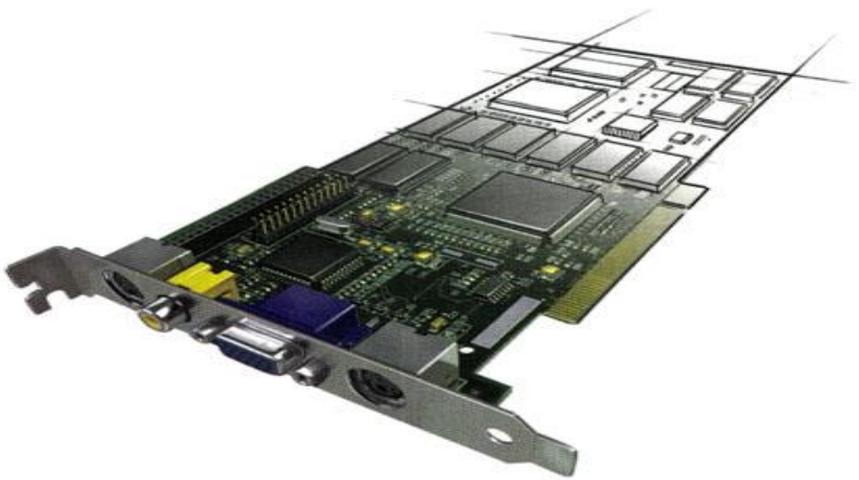
```
// Verilog model: Circuit with Boolean expressions

module Circuit_Boolean_CA (E, F, A, B, C, D);
  output      E, F;
  input       A, B, C, D;

  assign E = A || (B && C) || ((!B) && D);
  assign F = ((!B) && C) || (B && (!C) && (!D));
endmodule
```

HDL Example (User-Defined Primitive)

```
// Verilog model: User-defined Primitive
primitive UDP_02467 (D, A, B, C);
 output D;
 input  A, B, C;
//Truth table for D 5 f (A, B, C) 5 Σ(0, 2, 4, 6, 7);
 table
//    A   B   C   :   D    // Column header comment
      0   0   0   :   1;
      0   0   1   :   0;
      0   1   0   :   1;
      0   1   1   :   0;
      1   0   0   :   1;
      1   0   1   :   0;
      1   1   0   :   1;
      1   1   1   :   1;
 endtable
endprimitive

// Instantiate primitive

// Verilog model: Circuit instantiation of Circuit_UDP_02467

module Circuit_with_UDP_02467 (e, f, a, b, c, d);
 output      e, f;
 input       a, b, c, d

 UDP_02467          (e, a, b, c);
 and                (f, e, d);          // Option gate instance name omitted
endmodule
```

VHDL *(**V**HSIC **H**ardware **D**escription **L**anguage)*

- VHDL bir donanım tanımlama dilidir.
- Donanım, bilgisayar ve uzantısı olan işlevsel tüm teknolojik sistemlere, modüllere denir.



VHDL *(**V**HSIC **H**ardware **D**escription **L**anguage)*

Öncü diller

- **ISP** (Carnegie Mellon University) - 1976
- **KARL** (Kaiserslautern University) - 1977

Yaygın diller

- **Verilog HDL** (Gateway Design Automation – 1985)
- **VHDL** (Very High Speed Integrated Circuit Hardware Description Language – Amerikan Savunma Bakanlığı -1987)

Diğer

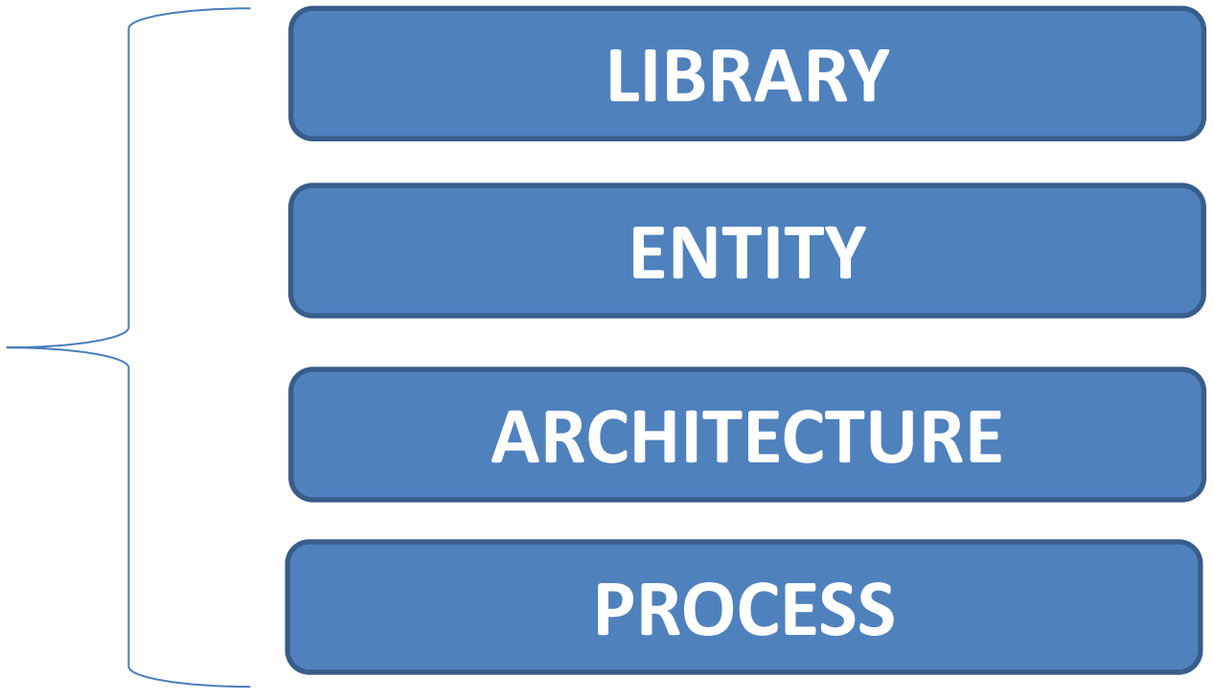
- AHDL (Altera HDL)
- RHDL (Ruby HDL)
- Confluence
- CUPL (Logical Devices Inc.)

VHDL & Verilog HDL

VHDL	Verilog HDL
Daha katı kurallı olan bir dil. Akademik seviye	Öğrenmesi ve kod yazması daha hızlı ve kolay
Karmaşık devrelerde daha az kapı kullanan derleyici yapısı	Karmaşık devrelerde daha fazla kapı kullanan derleyici
Yaygın olarak Avrupa ve Japonya	Yaygın olarak Amerika
ADA programlama dilini baz alan Pascal sınıfında bir dil	C'ye yakın syntax yapısı
ATI – Nokia	NVIDIA - AMD

VHDL Kod Yapısı

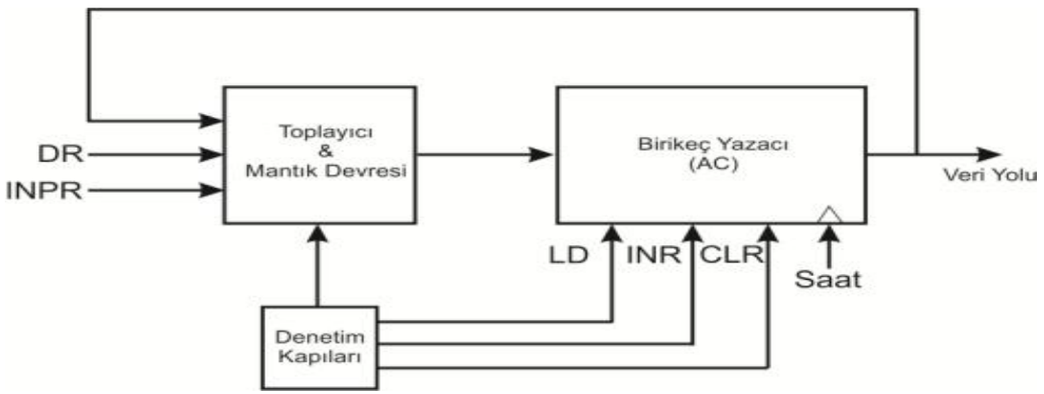
TEMEL VHDL KODU



VHDL Kod Yapısı


```
entity and_gate is
  port(
   A: in bit,
   B: in bit,
   X: out bit);
end entity and_gate;

architecture mimari of and_gate is
  begin
   X <= A and B;
end architecture mimari;
```

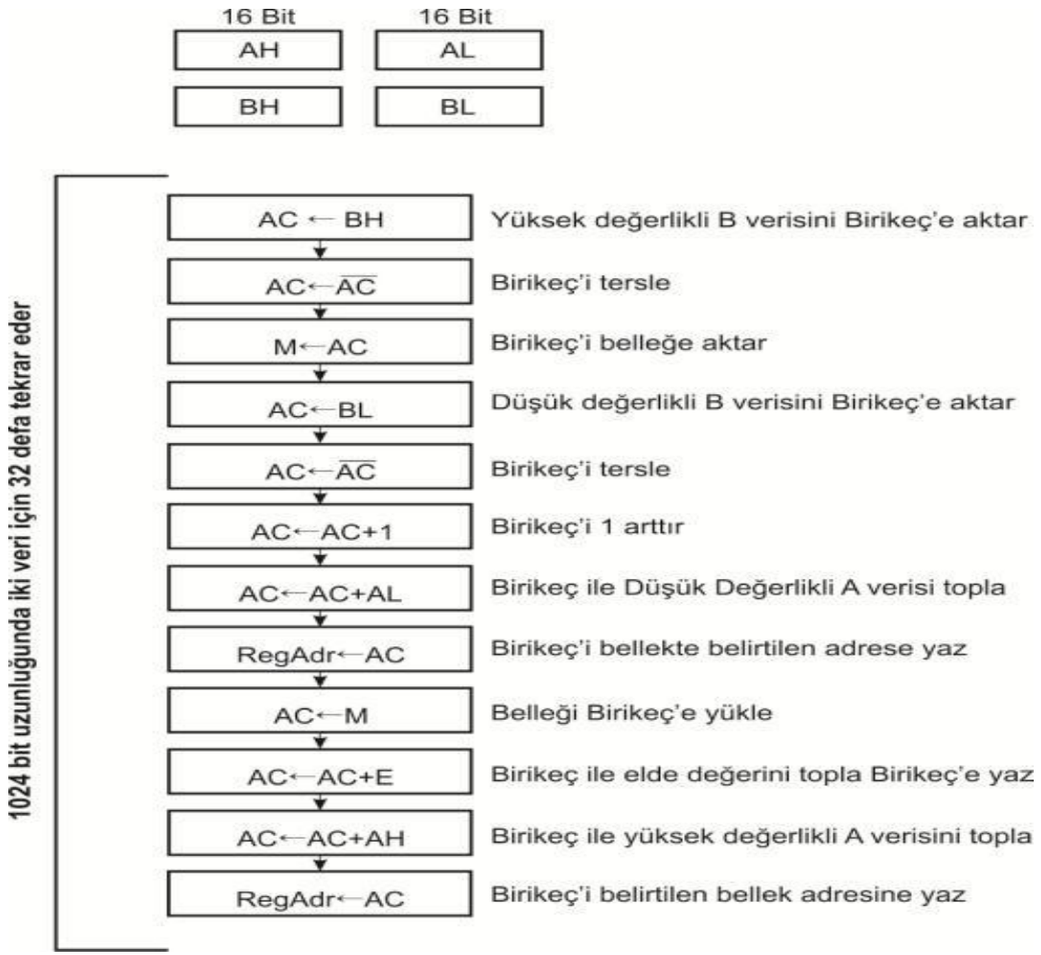
Bilgisayar Aritmetiğinde Çıkartma İşlemi



Birikeç (Akümülatör) Yapısı

- Standart bir bilgisayar işlemcisinde,
  - Birikeç Yazacı (Akümülatör-AC)
  - Toplayıcı ve Mantık devresi
  - Denetim kapılarından
- İşlenecek veri, kaydedicilerden veri yolu ile buraya aktarılarak işlem süreci gerçekleştirilir.

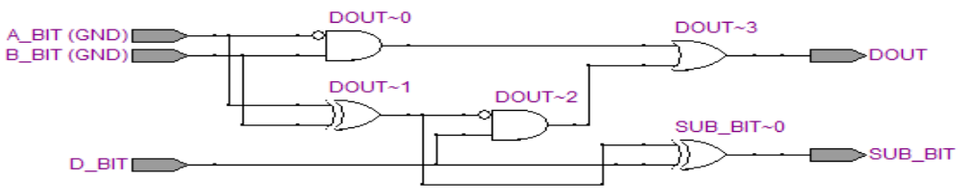
Bilgisayar Aritmetiğinde Çıkartma İşlemi



32 bitlik iki veriye çıkarma işlemi uygulama algoritması

- 32 bit bir işlemciye sahip olan standart bir bilgisayar, yazılımsal olarak birçok adımda çıkarma işlemini gerçekleştirir.
- İşlemcinin veri kapasitesi ile aynı veri boyutu büyüklüğüne sahip iki değerin dahi çıkarma işlemi birçok adımda gerçekleştirilebilmektedir.

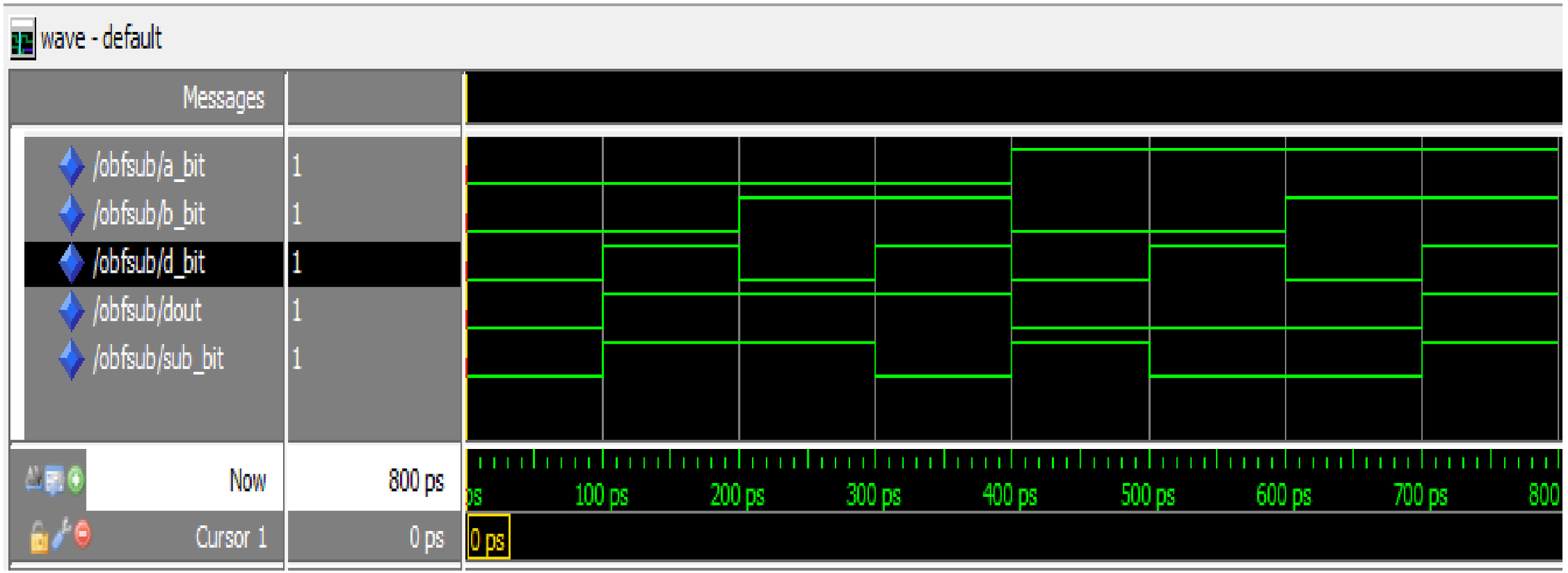
Yüksek Kapasiteli Çıkartma Devresi Tasarımı



VHDL ile Bir Bit Tam Çıkarıcı Devresi

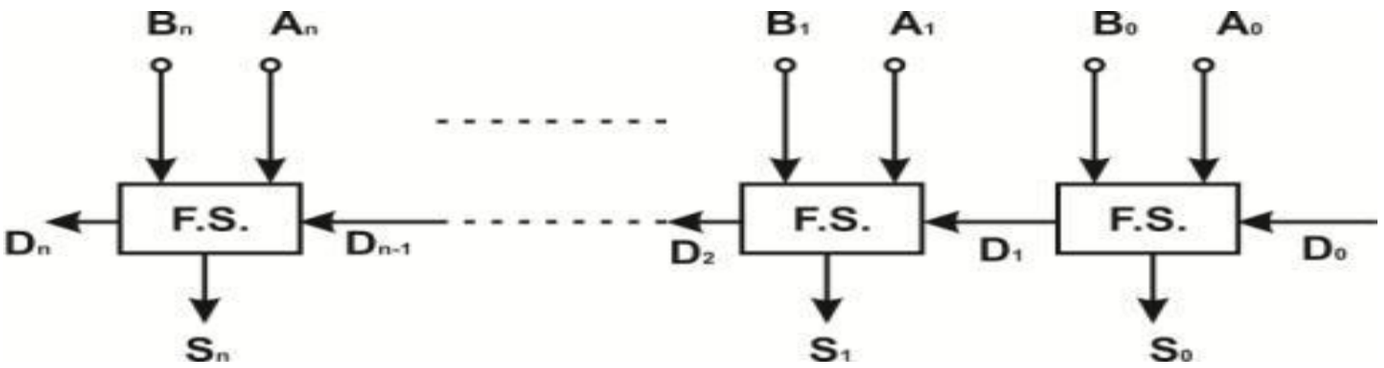
```
01  …
02  …
03  …
04  …
05  entity obfsub is port(
06      A_BIT :in std_logic;
07      B_BIT :in std_logic;
08      D_BIT :in std_logic;
09      DOUT  :out std_logic;
10  SUB_BIT  :out std_logic);
11  End obfsub;
12
13  Architecture struct of obfsub is
14      begin
15
16      process(A_BIT, B_BIT, D_BIT)
17
18      begin
19  SUB_BIT <= A_BIT xor B_BIT xor
20  D_BIT;
21
22  DOUT <= ((not A_BIT) and B_BIT) or
23  ((not (A_BIT xor B_BIT)) and
    D_BIT);
        End process;
24  End struct;
25
26
```

Yüksek Kapasiteli Çıkartma Devresi Tasarımı



Bir Bit Tam Çıkarıcı devresinin ModelSim benzetim sonuçları

Yüksek Kapasiteli Çıkartma Devresi Tasarımı



N bit Çıkarıcı Devresi Blok Şeması

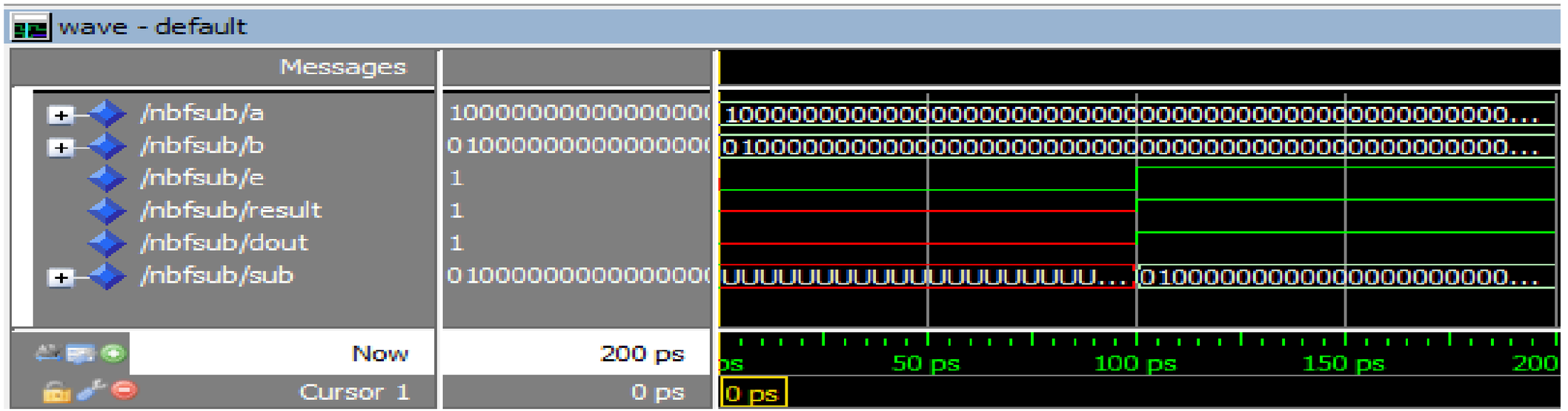
Yüksek Kapasiteli Çıkartma Devresi Tasarımı

```
…   …
…   …
04  entity nbfsub is
05
06  generic (
07  SZ:natural:=63;
08  A:std_logic_vector(63 downto
    0):="10000000000000000000000000000000
    00000000000000000000000000000000";
09  B:std_logic_vector(63 downto
    0):="01000000000000000000000000000000
    00000000000000000000000000000000");
10
11  port (
12  E       :in std_logic;
13  RESULT  :out std_logic;
14  DOUT    :out std_logic);
15  end nbfsub;
16
17  architecture struct of nbfsub is
18  signal SUB_T:std_logic_vector(SZ
    downto 0);
19  signal SUB:std_logic_vector(SZ downto
20  0);
    signal T        :std_logic_vector(SZ+1
21  downto 0);
22
```

```
22
23  component obfsub
24  port(
25   A_BIT,B_BIT,D_BIT: in std_logic;
26   SUB_BIT, DOUT: out std_logic);
27  end component;
28
29  begin
30          T(0) <= '0';
31
32  FS: for i in 0 to SZ generate
33          obfsub_i:       obfsub
34          port map (A(i), B(i), T(i),
35  SUB_T(i), T(i+1));
36  end generate FS;
37
38  process(E)
39  begin
40          if E='1' then
41                  DOUT <= T(SZ);
42                  SUB <= SUB_T;
43                  RESULT<='1';
44          end if;
45  end process;
    end struct;
```

“n” bit Tam Çıkarıcı devresi VHDL kod yapısı

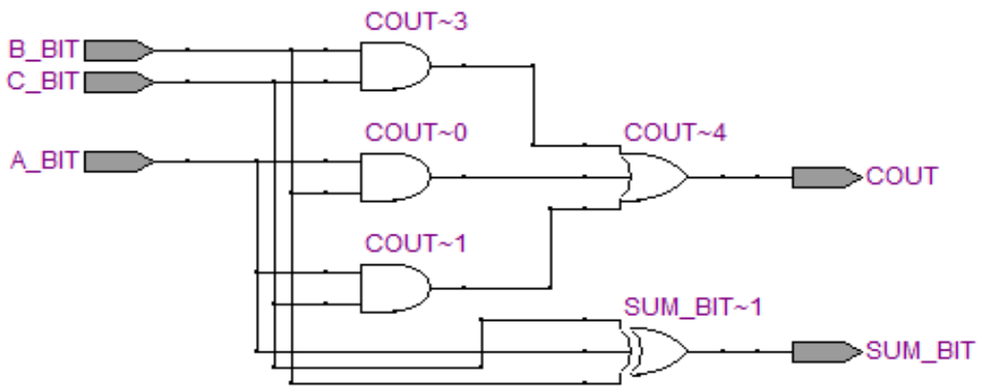
Yüksek Kapasiteli Çıkartma Devresi Tasarımı



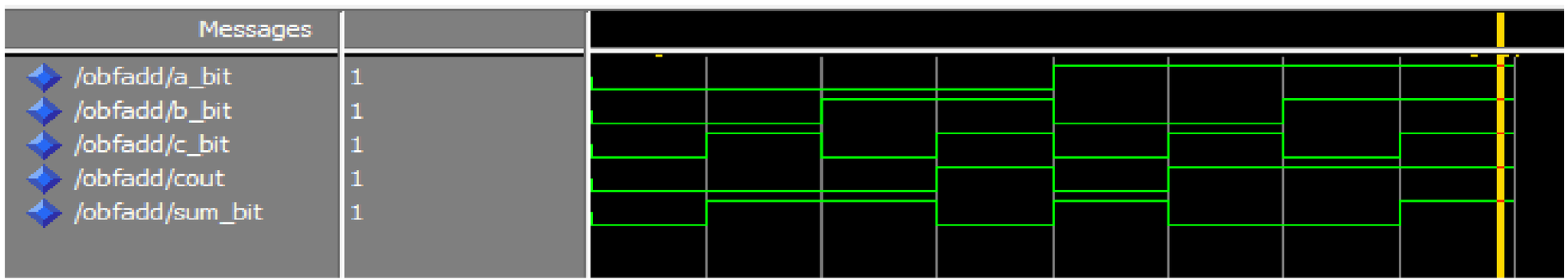
N bit tam Çıkarıcı devresi ModelSim benzetim sonuçları

Yüksek Kapasiteli Toplama Devresi Tasarımı

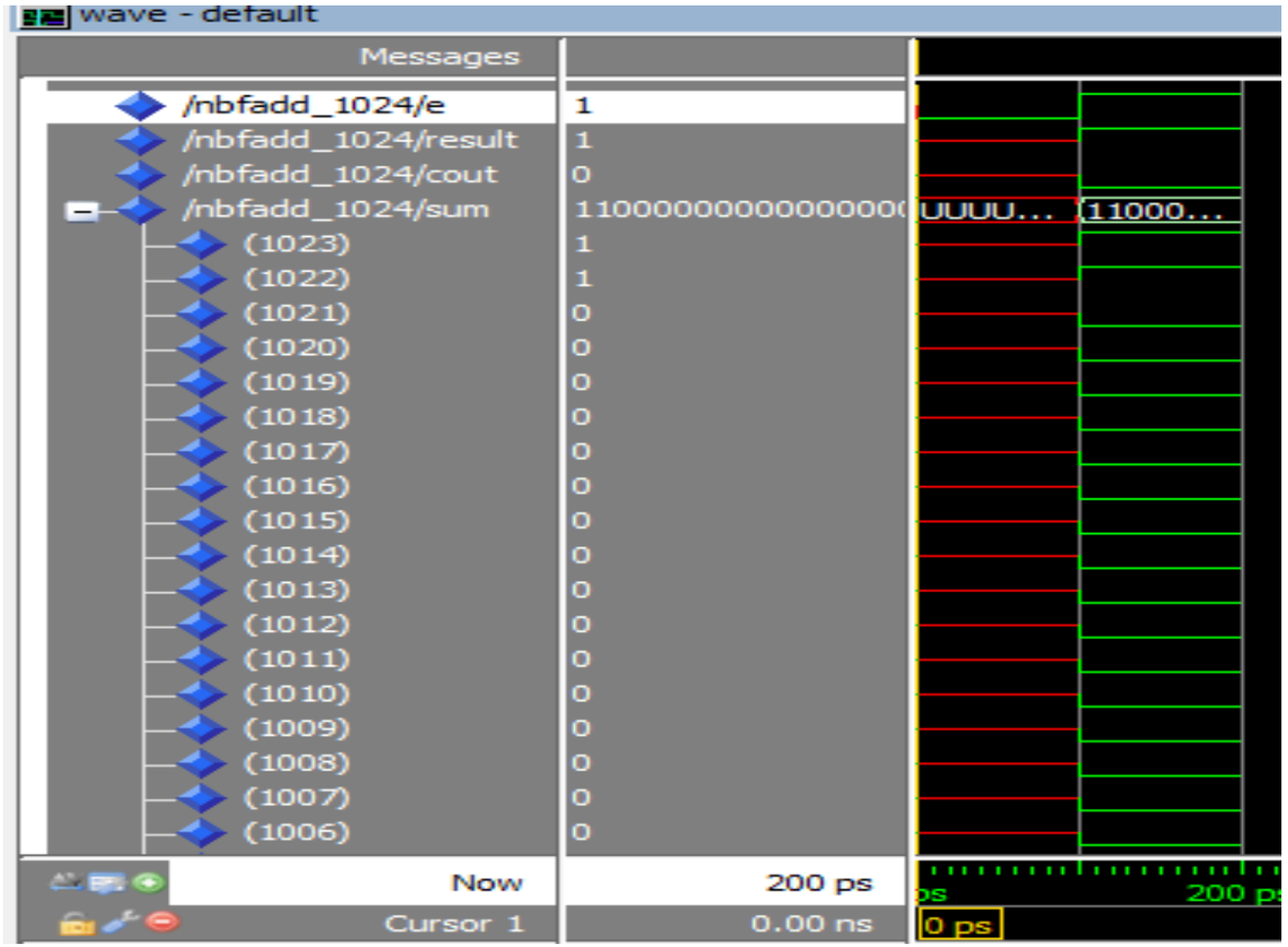
Bir Bit toplayıcı VHDL kodlarının RTL görünüşü



Bir Bit Toplayıcı ModelSIM benzetim sonuçları

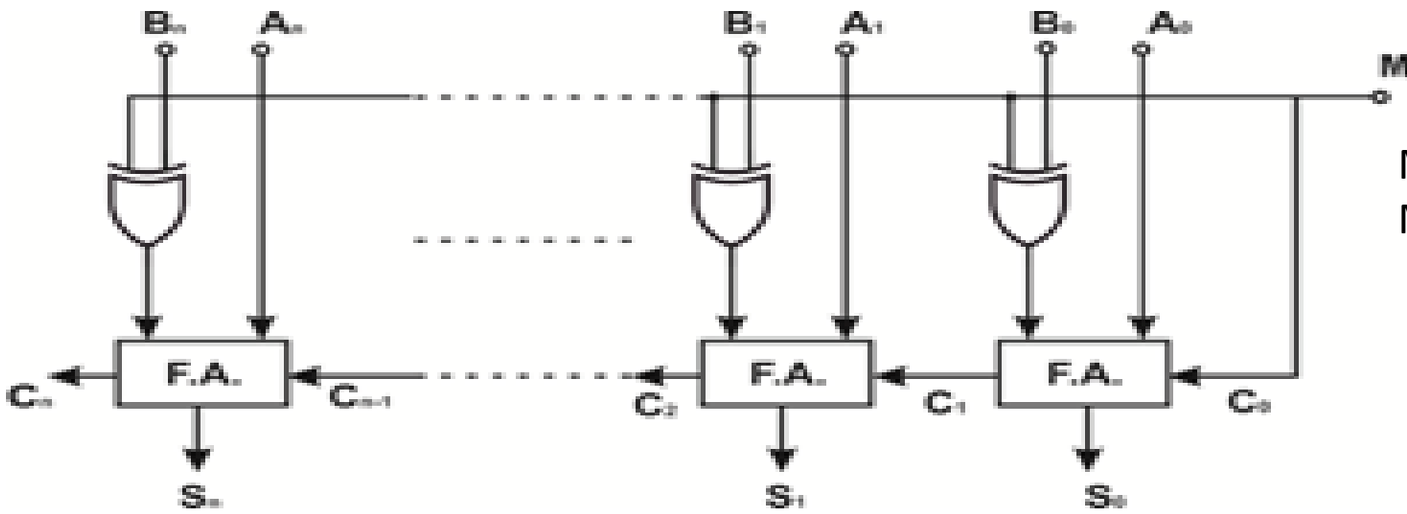


Yüksek Kapasiteli Toplama Devresi Tasarımı



1024 bit toplayıcının ModelSIM benzetim sonuçları

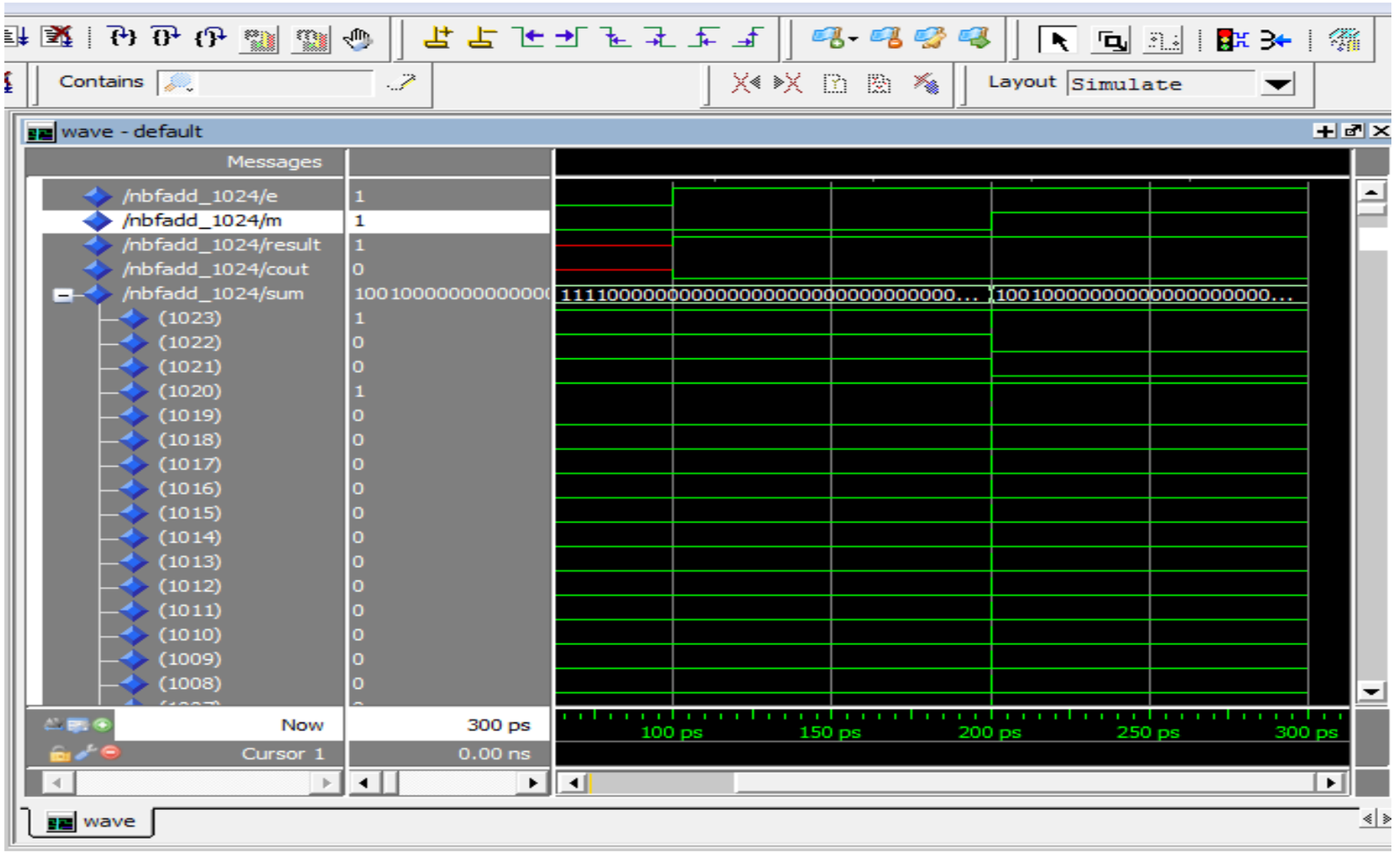
Yüksek Kapasiteli Toplama - Çıkarma Devresi Tasarımı



M=0 Toplayıcı
M=1 Çıkarıcı

Toplayıcı & Çıkarıcı Devrtesi

Yüksek Kapasiteli Toplama - Çıkarma Devresi Tasarımı



1024 bit Toplayıcı & Çıkarıcı Devresi ModelSIM benzetim sonuçları

Sorunlar & Öneriler

- FPGA devrelerinin giriş ve çıkış pin sayısı
  - Öneri:
    - Veri aktarımı için arı bir modül tasarımı
    - Çoklu FPGA kullanımı

Sonuç

- Gerçekleştirilen bu çalışmada, FPGA donanım yapısının esnek ve kolay programlama kabiliyetinden faydalanılarak yüksek kapasiteli çıkarıcı devresi benzetim seviyesinde tasarlanmıştır.
- VHDL ve FPGA kullanarak Yüksek Kapasiteli Aritmetik Ünite tasarlanabilir.
- Yüksek Kapasiteli Aritmetik Ünite, Büyük sayılar ile çalışan bazı şifreleme algoritmaları için donanım altyapısı oluşturmak için kullanılabilir.
- Günümüzde piyasadaki en güçlü PC'den daha güçlü hesaplama kapasitesi sunar.
- Donanım Tasarım Dili kullanılarak çok daha yüksek kapasiteli aritmetik işlemler için aritmetik ünite tasarlanabilir.

Kaynakça

- https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-111-introductory-digital-systems-laboratory-spring-2006/lecture-notes/
- http://web.ee.nchu.edu.tw/~cpfan/FY92b-digital/Chapter-4.ppt
- http://www.cs.nccu.edu.tw/~whliao/ds2003/ds4.ppt
- http://www.just.edu.jo/~tawalbeh/cpe252/slides/CH1_2.ppt
- Lessons In Electric Circuits, Volume IV { Digital By Tony R. Kuphaldt Fourth Edition, last update July 30, 2004.
- Digital Electronics Part I – Combinational and Sequential Logic Dr. I. J. Wassell.
- Digital Design With an Introduction to the Verilog HDL, M. Morris Mano Emeritus Professor of Computer Engineering California State University, Los Angeles; Michael D. Ciletti Emeritus Professor of Electrical and Computer Engineering University of Colorado at Colorado Springs.
- Digital Logic Design Basics, Combinational Circuits, Sequential Circuits, Pu-Jen Cheng.