Karadeniz Teknik Üniversitesi/Bilgisayar Mühendisliği Bölümü

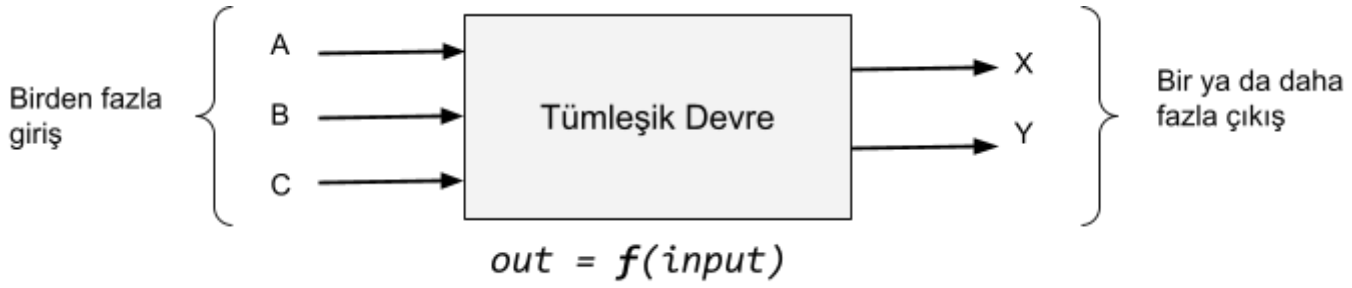
Sayısal Tasarım Laboratuvarı

VHDL ile Kombinasyonel Devre Tasarımı

Sayısal Tasarım uygulamalarında tasarlanan entegre devreler kombinasyonel (tümleşik) ve ardışıl olmak üzere iki türe ayrılır. Bu deneyde temel olarak kombinasyonel devrelerin VHDL dili ile tasarımı ve simulasyonu ele alınacaktır.

Kombinasyonel devrelerin çıkışları, herhangi bir zamanda, yalnızca mevcut girişlerin mantıksal işlevlerine bağlı olarak değişmektedir. Bir başka deyişle kombinasyonel devrelerde çıkış her zaman girişlerin kombinasyonuna bağlıdır. Bu tür devrelerde geri besleme (ve doğal olarak bellek) yoktur ve bu devrelerinin girişlerine uygulanan sinyallerde herhangi bir değişiklik olması durumunda, çıkışta hemen bir değişiklik gözlenmektedir.

Kombinasyonel mantık devreleri, daha karmaşık anahtarlama devreleri üretmek için "birleştirilen" veya birbirine bağlanan temel NAND, NOR veya NOT kapılarından oluşur. Bu kapılar, kombinasyonel devrelerinin yapı taşlarıdır. Bir kombinasyonel devre örneği olarak girişinde bulunan ikili kod verilerini, çıkışında farklı bir dizilimde ve sayıda çıkış hattına dönüştüren bir kod çözücü verilebilir. Kombinasyonel devreler çok basit veya çok karmaşık olabilir ve herhangi bir kombinasyonel devre yalnızca NAND ve NOR kapıları kullanarak gerçekleştirilebilir.



Şekil: Kombinasyonel devrelerin blok şeması

Günümüzde kombinasyonel devreler üç sınıfa ayrılmakta ve her sınıfta farklı devreler tasarlanarak kullanılmaktadır:

1. Aritmetik-Mantık Devreleri
    a. Temel düzeyde NAND, NOR, NOT kapıları ve bu kapıların farklı kombinasyonları
    b. Toplayıcılar (half-adder, full-adder)
    c. Karşılaştırıcılar
    d. Programlanabilir Mantık Devreleri (PLD's)
        i. Programlanabilir ROM (PROM)
        ii. Programlanabilir Dizi Mantığı (PAL) (programlanabilir AND + sabit bağlantılı OR)
        iii. Programlanabilir Mantık Dizisi (PLA) (programlanabilir AND + programlanabilir OR)
2. Veri İletim Devreleri (Data Transmitters)
    a. Multiplexer (çoktan bire): örn; 4 to 1 mux devrede 4 giriş, 1 çıkış ve 2 seçme ucu bulunur
    b. Demultipler (birden çoğa): örn; 1 to 4 demux devrede 1 giriş, 4 çıkış ve 2 seçme ucu bulunur.
    c. Encoder (kodlayıcı): 8:3 encoder devresinde, 8 giriş, 3 çıkış bulunur
    d. Decoder (kod çözücü): 3:8 decoder devresinde 3 giriş, 8 çıkış bulunur
3. Kodlayıcılar (Data Converters)
    a. Binary kodlayıcılar
    b. BCD kodlayıcılar
    c. Gray kod üreticiler
    d. Hata bulma devreleri

KODLAMA ve HATA BULMA TEKNİKLERİ

Giriş

Kodlama, elektronik dünyasında çok sık kullanılan, hatta vazgeçilmesi mümkün olmayan bir kavramdır. En basit anlamda kodlama, eldeki verinin (Sayısal veya Analog olabilir) başka birimde gösterilişi olarak tanımlanabilir. Özellikle sayısal sistemlerde ağırlıklı olarak kullanılan birçok kodlama tekniği mevcuttur. Kodlama teknikleri; güvenli veri iletimi, veri sıkıştırma, hata kontrolü ve giderimi, bellek işlemlerinde

verimlilik artırma, aritmetik işlemlerde kolaylık sağlama, süratli veri işleme gibi birçok amaç için kullanılmaktadır.

Amaca bağlı olarak sayılar için konumsal kodlama, BCD (ikili kodlanmış onlu) kodlar kullanılırken, ölçmelerde hatayı azaltmak için gray kodu, bilgisayarlarda hatayı bulmak ve azaltmak için eşitlik (parity) ve hamming kodlamaları kullanılmaktadır. Verilen bir veriye ilave bitler katarak bu verilerin saklanmasında veya iletiminde ortaya çıkabilecek hatalar algılanıp otomatik olarak giderilebilir. Bu çeşit kodlama hata yapma ihtimalinin yüksek olduğu iletişim sistemlerinde ve gürültülü ortamda çalışan sanayi tesislerinde yaygın olarak kullanılmaktadır. Hata bilgiyi temsil eden 0-1 dizisindeki bazı bitlerin değer değiştirmesi olarak yorumlanmalıdır. Hamming kodu hata bulma ve giderme amacıyla kullanılırken, eşitlik biti tekniği yalnız hata algılama için kullanılabilir.

Yalnızca sayısal karakterlerin (0, 1, …., 9) kodlanmasıyla ortaya çıkan kodlama yöntemine 'Sayısal Kodlama', hem alfabetik hem de sayısal karakterlerin kodlanmasını içeren kodlama yöntemine ise 'Alfasayısal Kodlama' denir.

Sayısal Kodlama

Yalnızca sayısal karakterlerin kullanıldığı sayısal kodlama sistemlerinin uygulama alanının çok geniş olması nedeniyle, çok sayıda sayısal kodlama yöntemleri kullanılmaktadır. Sayısal kodlama yöntemlerine aşağıdaki örnekler verilebilir:

- İkili Kodlanmış Onlu (Binary Coded Decimal-BCD)
- Ağırlıklı İkili Kodlama
- 3 Fazlalı Kod
- Bitişik Kodlama
- Gray Kodu
- Eşitlik Kodu
- Hamming Kodlama

BCD (Binary Coded Decimal)

Bir bilgisayarda, girilen sayılar üzerinde az sayıda aritmetik işlem yapılıp sonuç kullanıcıya onlu biçimde gösterilecekse, o zaman onludan-saf ikiliye ve saf ikiliden-onluya yapılacak dönüşümlerden kaçınmak için, başka bir deyişle bilgisayarı bu dönüşümlerden kurtarmak amacıyla BCD gösterim kullanılır. Böyle bir çalışma bilgisayarda işlem hızını artırmakla beraber ilave aritmetik işlemcilerin (yani ilave donanımın) kullanımını gerektirir.

Onlu sistemdeki herhangi bir sayının, her bir basamağının ikili sayı sistemdeki karşılığının dört bit ile ifade edildiği kodlamaya, **"İkili Kodlanmış Onlu-BCD Kodu"** denir.

Onlu bir sayıyı BCD kodlamak yazmak istersek, her bir basamağını 4 bitlik ikili sayı şeklinde yazmamız gerekir. Elde edilen gruplar bir araya getirildiğinde BCD kod üretilmiş olur.

Örnek 1. $(369)_{10}$ sayısını BCD kodu ile kodlayalım.

Her bir basamaktaki sayının ikili karşılığı 4 bitle ifade edilirse;

3 6 9 → 0011 0110 1001 sayıları bulunur.

Sonuçta; $(369)_{10}$ = (001101101001) BCD eşitliği elde edilir.

Onlu sayı sisteminde 0,1,2,…,8,9 olmak üzere on tane rakam vardır. Bu yüzden her onlu rakam için en az 4 bitlik bir ikili sayı gerekir. 4 bit ile 2^4=16 farklı kod oluşturulabildiği halde BCD rakamlar bunlardan yalnız 10'unu kullanacaklardır. Bundan dolayı artıklı kodlama da denmektedir.

BCD sayılar üzerinde işlem yapmak zor olduğundan günümüz bilgisayarlarında sayıları göstermek için bu kodlama kullanılmamaktadır.

Onlu Sayı	Kod Sözcüğü		Onlu Sayı	Kod Sözcüğü	
0	0000	Doğal İkili Kodlamaya göre geçerli BCD haneler	10	1010	Doğal İkili Kodlamaya göre geçersiz BCD haneler
1	0001		11	1011	
2	0010		12	1100	
3	0011		13	1101	
4	0100		14	1110	
5	0101		15	1111	
6	0110				
7	0111				
8	1000				
9	1001				

Ağırlıklı İkili Kodlama

Sayıların ağırlıklı kodlama kullanılarak 2 tabanında gösterilmesidir. Ağırlıklı kodlamada genel mantık her bir bite birer ağırlık değeri verilmesidir. Örneğin, ağırlıklı ikili kodlamada her bir bite ağırlıklar 2'nin katları şeklinde verilir.

Örneğin: 11001 = 1.24 + 1.23 + 0.22 + 0.21 + 1.20 = 25

Bilgisayarlarda sayıları göstermek için ağırlıklı ikili kodlama kullanılır.

3 Fazlalı Kod

3 Fazlalı Kod, bazı aritmetik hesaplamalarda işlem kolaylığı sağlaması sebebiyle BCD kod yerine kullanılmaktadır. Herhangi bir BCD kod 3 Fazlalı Kod'a dönüştürülürken, onlu sayıdaki karşılığı olan ikili sayıya 3 eklenmektedir. Dolayısıyla bu kodlama yöntemine, **'3 Fazlalı Kod'** denmektedir. Bu kodda da yine sayılar, BCD kodunda olduğu gibi dört bitlik ikili sayılar şeklinde ifade edilir.

En önemli avantajı; hesaplamada ve hata düzeltmede kolaylık sağlamasıdır ancak tümleyenini almadaki güçlükler nedeniyle son zamanlarda pek kullanılmamaktadır.

Sayı	Kod Sözcüğü	Sayı	Kod Sözcüğü
0	0011	9	1100
1	0100	8	1011
2	0101	7	1010
3	0110	6	1001
4	0111	5	1000

Bitişik Kodlama

Birbirini izleyen sayılara denk düşen ikili kod sözcükleri arasındaki uzaklık (Hamming'e göre) "1" ise, söz konusu sözcükler "Bitişik" bir kodlama oluşturur. Örneğin 0'dan 3'e kadar olan sayıları kodlamada ise 00, 01, 11, 10 sözcükleri kullanılırsa, bitişik bir kodlama yapılmış olur. Çünkü birbirini izleyen her sözcük arasındaki fark 1'dir. Kod sözcüklerinin "birincisi" ile "sonuncusu" arasındaki uzaklık yine "1" olduğundan, kodlama aynı zamanda "çevrimli" olmuş olur.

Gray Kodu

Gray kodu, minimum değişim sayıda gösteren kod sınıfı içerisinde yer almaktadır. Kodlamada bir sayıdan diğerine geçişte yalnızca bir bit konum değiştirmektedir.

Gray kodlu sayılarda basamak değeri mevcut olmadığından, bu kodlama yöntemi aritmetik işlemlerin mevcut olduğu yerlerde kullanılmamaktadır.

Gray kodlu sayıların dezavantajı; aritmetik işlemleri yapabilmek için ikili sayı sistemine dönüştürülmesinin zorunlu olmasıdır.

Kodlanacak Sayılar	Doğal İkili Kodlama	Gray Kodlama
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Bu kodlamanın yapısı gereği, bir sözcükten diğerine geçişte bitlerden yalnızca biri değer değiştirdiğinden oluşacak hata en fazla bu sözcüklerin farkı kadar olur. Bu nedenle ölçme tekniğinde sıkça kullanılır.

	00	**01**	**11**	**10**
00	0	1	2	3
01	7	6	5	4
11	8	9	10	11
10	15	14	13	12

Karnaugh diyagramı kullanılarak Gray kodun elde edilmesi

Eşitlik(Parity) Kodu

İkili sistemde ifade edilen herhangi bir bilginin iletilmesi dijital dünyada sıkça karşılaşılan bir olaydır. Bilginin iletilmesi esnasında, bazı dış etkenlerden ötürü gürültü oluşumu ile karşı tarafa iletilen bilginin bozulması da sıkça görülmektedir. Oluşan bu hataların tespit edilmesi ve eğer mümkünse düzeltmesi sayısal sistemlerin temel özelliklerindendir.

Bir bitlik hataların tespit edilmesinde en yaygın kullanılan yöntem **"eşitlik kodu"** yöntemidir. Yöntemde, hataların algılanmasını sağlamak amacıyla BCD kodlu sayının başına ya da sonuna **"eşitlik biti"**

eklenmektedir. Eşitlik biti, kodlanan verideki 1 ya da 0'ların tek ya da çift sayıda olduğunu ifade eder. Eşitlik biti yöntemi çift eşitlik (even parity) ve tek eşitlik (odd parity) olmak üzere iki farklı şekilde kullanılabilmektedir..

Çift eşitlikte; eşitlik biti, kodlanacak verideki 1'lerin toplam sayısı (eşitlik biti dâhil) çift olacak şekilde seçilmektedir. Kodlanacak sayıdaki 1'lerin sayısı tek ise, eşitlik biti olarak '1' eklenir. Çift olması durumunda ise, eşitlik biti olarak '0' eklenir.

Tek eşitlikte; tek fark, kodlanacak verideki 1'lerin toplam sayısı tek olacak şekilde seçilir.

Aşağıda tek eşitliğe göre oluşturulan tabloda çeşitli durumlar gösterilmiştir. Altı çizili olanlar hatalı olarak iletilen bitleri parantez içindekiler ise veriye eklenen eşitlik bitini göstermektedir.

Gönderilen	Alınan	Açıklama
1 0 1 1 0 1 1 0 (0)	1 0 <u>0</u> 1 0 1 1 0 (0)	Çift eşitlik, hata algılandı
1 0 1 1 0 1 1 0 (0)	1 <u>1 0</u> 1 0 1 1 0 (0)	Tek eşitlik, hata algılanamadı
1 0 0 1 0 1 1 0 (1)	1 0 0 1 0 1 1 0 (<u>0</u>)	Çift eşitlik, hata algılandı
1 0 0 1 0 1 1 0 (1)	<u>0</u> 0 0 1 0 1 1 <u>1</u> (1)	Tek eşitlik, hata algılanamadı
↑ Eşitlik Biti		

Seri veri iletiminde ve manyetik bilgi saklama işlemlerinde eşitlik yöntemi yaygın olarak kullanılır.

Hamming Kodlama

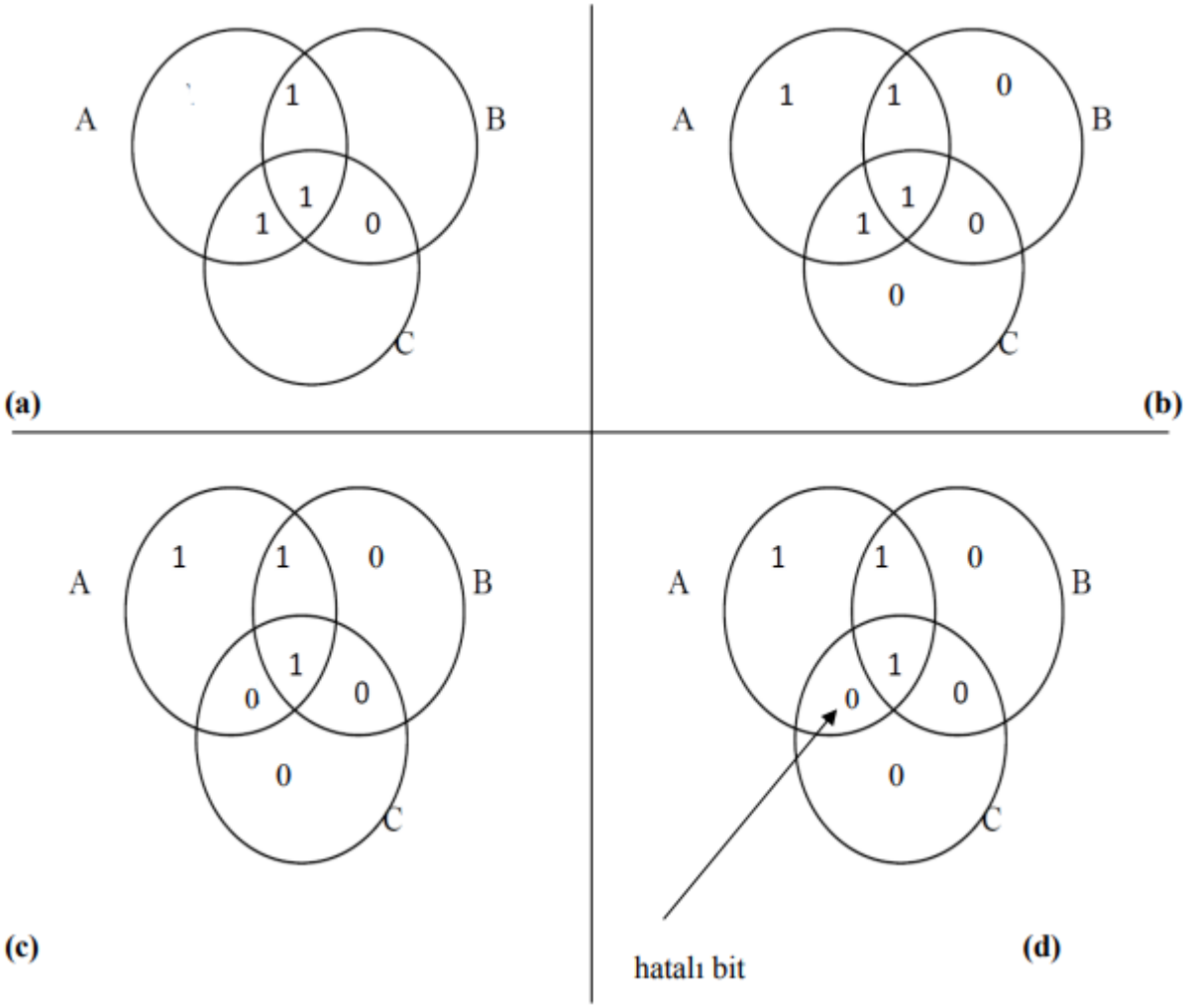
Hamming Kodlama kullanılarak hata araştırıldığında aşağıdaki üç sonuçtan biri elde edilir.

1) Hata algılanılmadı
2) Hata algılandı ve bu hatayı düzeltmek mümkündür.
3) Hata algılanır ama hatayı düzeltmek imkansızdır. Bu durum sadece rapor edilir.

Hamming kodlama her veri kelimesi üzerinde çok sayıda eşitlik denetimi gerçekleştirilerek yapılır. Eşitlik denetimine ilişkin ilave bitler veri kelimesi ile birlikte gönderilir.

Hamming Kodunu anlayabilmek için aşağıdaki Venn diyagramları kullanılsın. Kesişen üç dairenin olması durumunda yedi bölüm oluşur. İçteki(kesişim) bölümlere 4 veri biti atanır.(Şekil 1.a) Geri kalan bölümler eşitlik bitleri ile doldurulur. Her eşitlik biti, kendi dairesindeki 1 değerli bitlerin sayısı çift olacak şekilde seçilir. Veriyi Hamming kodlayabilmek için 3 düzeltme bitini A,B,C kümelerine bakarak tayin edebiliriz. A kümesinde tek sayıda "1" olduğu için çift eşitlik kavramından giderek, düzeltme biti olarak "1" koyulmasına karar verilir. Yine aynı şekilde B kümesinde çift sayıda "1"olduğu için düzeltme biti "0" olmalıdır. Düzeltme bitinin belirlenmesi basit bir XOR işlemidir. (Şekil 1.b)

Hamming kodlu sözcükte bozulma olduğu varsayılsın (Şekil 1.c). Buna göre; A ve C kümesinde tek sayıda "1" vardır. B kümesi ise normaldir. A ve C kümelerindeki "1"lerin sayısı çift yapılırsa hata giderilmiş olacaktır. Bu da Şekil 1.d 'deki işaret edilen bit "1" yapılarak gerçekleştirilir.



Şekil 1. Hamming Hata Düzeltme

Bu kodlamayı bir örnekle inceleyelim;

a0, a1, a2, a3 biçiminde 4 bitlik verimiz olsun. Bu veriye a4, a5, a6 düzeltme bitleri eklenerek 7 bitlik hamming kodlu sözcük oluşturulsun.

Düzeltme bitleri, veri bitlerinden giderek eşitlik hesabıyla şöyle bulunur;

$a4 = a0 \oplus a1 \oplus a2$

$a5 = a1 \oplus a2 \oplus a3$

$a6 = a2 \oplus a3 \oplus a0$

Bu işlem sonunda **veri bitleri;** a0, a1, a2, a3, **düzeltme bitleri;** a4, a5, a6 olmuş olur.	Hamming kodlu sözcük oluşturulmuş olur.

Hamming kodlu sözcük, başka bir ortama iletildikten ya da saklandıktan sonra yeniden okunduğunda elde edilen 7 bitlik dizi $a'_0, a'_1, a'_2, a'_3, a'_4, a'_5, a'_6$ olsun.

Alıcı tarafta düzeltme bitlerinin değeri yeniden hesaplanır.

$$a''_4 = a'_0 \oplus a'_1 \oplus a'_2$$
$$a''_5 = a'_1 \oplus a'_2 \oplus a'_3$$
$$a''_6 = a'_2 \oplus a'_3 \oplus a'_0$$

Hesaplanan a''_4, a''_5, a''_6 ile alınan ya da okunan a'_4, a'_5, a'_6 karşılaştırılır.

$$S_4 = a'_4 \oplus a''_4$$
$$S_5 = a'_5 \oplus a''_5$$
$$S_6 = a'_6 \oplus a''_6$$

Bu hesap sonunda S_4, S_5, S_6 sıfır değeri alırsa hata yoktur. Bu yöntemle hata algılandığı gibi hangi bitte hata olduğu algılanıp düzeltilir.

Aşağıdaki tabloda her S' ye ilişkin satırda, o S terimi ile ilgili değişkenlerin karşısına "x" işareti koyulmuştur.

	a_0	a_1	a_2	a_3	a_4	a_5	a_6
S_4	X	X	X		x		
S_5		X	X	X		x	
S_6	X		X	X			x

S_4	**S_5**	**S_6**	**Hatalı Bit**
0	0	0	Hatalı Bit Yok
0	0	1	a_6
0	1	0	a_5
0	1	1	a_3
1	0	0	a_4
1	0	1	a_0
1	1	0	a_1
1	1	1	a_2

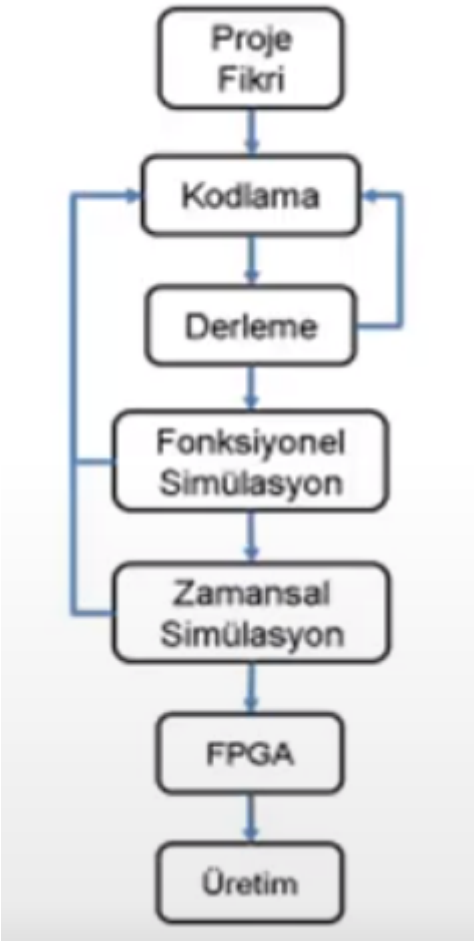
Genel olarak k düzeltici değişken varsa 1 durum hatalı olmama, geri kalan 2k-1 durum ise hatalı olabilecek bitleri göstermeye yarar. Bu örnekte 3 düzeltici değişken var ve ancak bu değişkenle 8 durum kodlanır.(2^3=8). Bu durumlardan biri hatasızlığı geri kalan durumlar ise, 7 bitten hatalı olanını gösterir. Bu açıklanan kod, 'tek hata-düzeltme' olarak (Single Error Correction-SEC) bilinir ve eğer aynı anda ancak bir bitte hata olmuşsa geçerlidir. Bitlerden 2 tanesinde hata oluşmuşsa Hamming kod hatayı algılar ama düzeltemez. (Çift hata algılama: Double Error Detection, DED). Hata düzeltme, ilave bir lojik gerektirirken, belleğin ya da manyetik saklama ortamının güvenilirliğini iyileştirir.

Örneğin; IBM 30XX bilgisayarları ana bellekteki her 64 bitlik data için 8 bitlik bir SEC- DED kodu kullanır. Bu nedenle, ana belleğin gerçek boyu, kullanıcıya görünenden daha büyüktür. VAX bilgisayarlar ise, belleğin her 32 biti için 7 bitlik SEC-DED kullanırlar.

Alfasayısal Kodlama

Bu kodlama yönteminde sayılara ek olarak alfabedeki harfler, noktalama işaretleri ve diğer özel karakterler de kodlanmaktadır. Alfasayısal kodlama, tüm büyük ve küçük harfleri, 7 noktalama işaretini, 0'dan 9'a kadar olan 10 sayıyı ve +, /, #, %, *, vb. özel karakterleri içerir. Yaygın olarak kullanılan iki farklı alfasayısal kodlama yöntemi: ASCII (Amerikan Standart Code For Information Interchance) ve EBCDIC (Extended BCD Interchance Code) kodları. Bu kodlardan ASCII kodu daha yaygın olarak kullanılmaktadır.

VHDL ile Kodlama



Öğrenci deneye gelmeden önce VHDL kodlamanın nasıl yazıldığını anlatan video anlatımları izleyerek deney alanına gelmektedir.

- https://www.youtube.com/watch?v=Z6Ql3Jw2hTU

Temel Kavramlar ve Deneyle İlgili Bazı Notlar

- HDL (Hardware Description Language) - Donanım Tanımlama Dili
- VHDL ve Verilog FPGA programlamada en çok kullanılan HDL dilleridir.
- VHDL → Very High –Speed Integrated Circuit Hardware Description Language
- FPGA → Field Programmable Gate Array (Alanda Programlanabilen Kapı Dizisi)

VHDL iki amaç için kullanılır.

- Sentezleme: FPGA'e yüklenecek kodu oluşturmak için
- Simülasyon: FPGA'e yüklenecek kodun simülasyonunu yapmak için

Behavioral Modelling - Davranışsal Modelleme

- Modelin giriş ve çıkış tepkileri davranışsal olarak tanımlanır.
- Modelin iç yapısı ile ilgilenilmez.
- Yanlızca devrenin işlevi ve fonksiyonelitesi ile ilgilenilir

Structural Modelling - Yapısal Modelleme

- Bir bileşenin daha alt seviyesindeki bileşenlerle arasındaki ilişkileri gösterir.
- Yapısal tasarım, modelin yapısının tasarımcı tarafından yapılandırılması ilkesine dayanır.

Tasarım Adımları

Proje fikri → Kodlama → Derleme → Fonksiyonel Sümulasyon → Zamansal Simulasyon → FPGA → Üretim

Tümleşik devrelerin üretim aşamaları (HDL → Compile → Simulation → FPGA → Product)

VHDL proje bölümleri → Proje fikri → Kodlama → Derleme → Fonksiyonel Sümulasyon → Zamansal Simulasyon → FPGA → Üretim (Akış diyagramı?)

VHDL Donanım Dilinin Yapısı

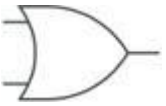
VHDL donanım tanımlama proje dosyaları temelde 4 bölümden oluşur.

a) Başlık bölümü: projenin genel bilgilerinin verildiği açıklama satırlarının bulunduğu kısımdır. Ayrıca VHDL dilinde yorumlar çift tire (--) den sonra yapılır.

b) Kütüphane ve paketlerin eklendiği bölüm:

c) Devre elemanının adı ve portlarının tanımlandığı kısım.

d) Port tanımı yapılan devre elemanının mimarisinin bulunduğu kısım. Aşağıda bir VEYA kapısı için VHDL kodu verilmiştir.

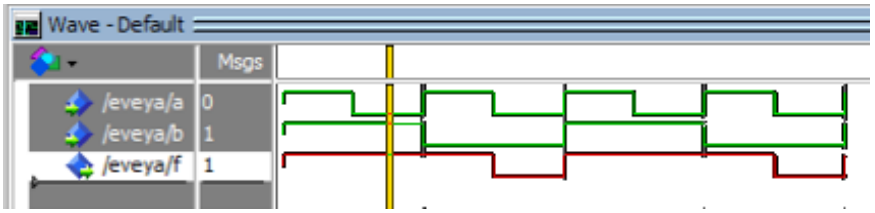


```
----------------------------------------
-- Veya kapisi
-- standart iki girisli Veya kapisi tasarimi
-- Giris(ler): a, b
-- Cikis(lar): f
----------------------------------------
Library IEEE;
Use IEEE.std_logic_1164.all;

Entity eVeya is -- devre elemani adi
Port ( a: in std_logic; -- giris ve cikis portlari b: in
          std_logic;
          f: out std_logic ); End
eVeya;

Architecture Behv of eVeya is
Begin -- a ve b girisleri Veya islemine giriyor
       f <= a or b; -- ve sonuc f cikisina yaziliyor End
Behv;
----------------------------------------
```

VEYA kapısının ModelSim uygulamasında simülasyonunun sonucu Şekil 2'de verilmiştir.



***Şekil 2**- ModelSim uygulamasında Veya kapısının simülasyonu.*

Hamming Kodu

```
library ieee;
use ieee.std_logic_1164.all;

Entity eHam is
port (   A: in std_logic_vector(3 downto 0);   -- 4 bit veri
                B: in std_logic_vector(6 downto 0);   -- 7 bit test verisi
                F: out std_logic_vector(6 downto 0); -- 7 bit  A çıkışı
                E: out std_logic_vector(6 downto 0)
                );
End Entity;

Architecture behv of   eHam is
        signal  C: std_logic_vector(6 downto 0);   -- Code = veri + hamming


Begin
                C(3 downto 0) <= A;                             -- A kopyalandı...
                C(4) <= A(0) xorA(1) xorA(2) ;        -- Hamming code
                C(5) <= A(1) xorA(2) xorA(3) ;
                C(6) <= A(2) xorA(3) xorA(0) ;
                F <= C;                  -- hesaplanan code cikisa aktarildi

                E(2) <=  C(6) xor B(6);         -- hata kodu
                E(1) <=  C(5) xor B(5);
                E(0) <=  C(4) xor B(4);

End behv;
```

Deneyin Yapılışı

- Deney VHDL dilinde kodlanacaktır.
- Orijinal veri girişi için A giriş portunu 4 bit olarak belirlenir
- B test portunu da A portu gibi belirlenir,
- F çıkış sinyalini 7 bit olarak tanımlanır
- Orijinal A portu için hata kodunu hesaplanır
  - a4= A(0) xor A(1) xor A(2);
  - a5= A(1) xor A(2) xor A(3);
  - a6= A(0) xor A(2) xor A(3);
- B portundaki bilgi için hata kodunu hesaplayınız
  - b4= B(0) xor B(1) xor B(2);
  - b5= B(1) xor B(2) xor B(3);
  - b6= B(0) xor B(2) xor B(3);
- Hata kodlarını karşılaştır ve sonucu F portuna yazınız.
- Devrenin kodunu ModelSim uygulamasında yazınız ve simülasyonu yapınız.

Deney Hazırlık Soruları

- Kodlama işlemi nerelerde kullanılır.

- Analog ve dijital işaretin farkları nelerdir?
- Analog işaretten dijital işaret nasıl elde edilir?
- İyi kodlamanın özellikleri nelerdir? Avantaj ve dezavantajlarını araştırınız.
- Eşitlik bitinin hata algılayamadığı durumları söyleyin ve buna rağmen neden güvenilir olduğunu açıklayın.
- Hata algılama yöntemlerini karşılaştırınız.