

T.C.
GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ

XML İLE VLSİ TASARIMI

121342

Ahmet SAVAŞ
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ

TEZ DANIŞMANI
Yrd.Doç.Dr. Fatih KOÇAN

GEBZE

2002

121342

Ahmet SAVAS 'ın tez çalışması, G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 22.07.2002 tarih ve 2002/124 sayılı kararıyla oluşturulan jüri tarafından Bilgi sayı Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRI

ÜYE
(Tez Danışmanı)

: Yrd. Doç. Dr. Fatih Kocan
Fatih Kocan

ÜYE

: Yrd. Doç. Dr. Hasan Sadıkoğlu
H. Sadıkoğlu

ÜYE

: Yrd. Doç. Dr. F. B. Doğan Sevilgen
Sevilgen

ONAY

G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 12.09.2002 tarih ve 2002/130 sayılı kararı.

İmza / Mühür

ÖZET

Bu çalışma, VLSI devrelerinin web teknolojileriyle tasarlanması üzerinedir. Çalışmada VLSI bilgilerinin değişimi ve işlenmesi için XML teknolojisine dayalı bir standart oluşturulmaya çalışılmıştır. Aynı zamanda devrelerin web üzerinde rahat ve hızlı bir çizimi, görülebilmesi amaçlanmıştır. Web üzerinden yayınlanması için VML ve Javascript'den faydalanılmıştır.

Çalışmalar sonunda VLSI bilgilerinin ortak bir standarta getirilmesi amacına yönelik olarak XML teknolojisinin çok uygun olduğu görülmüştür. Ayrıca devre çizimleri için ayrıntılı ve herbiri ayrı bir kurulum gerektiren programların yerine basit bir HTML sayfasının kullanılabileceği gösterilmiştir.



SUMMARY

This work focused on to design VLSI circuits with web technologies. In this work, studied to creating standarts on XML based for interchanging and processing VLSI information. At same time, purposed to easy drawing and displaying circuits on the web. Used VML and Javascript to publish on the web.

At result of studies, seems that XML techonologi is very useful to standartize VLSI information. Instead of programs which hard to install and has very details simple HTML page can use.



TEŞEKKÜR

Günümüzde bilgi en önemli güç kaynaklarından birisi olmuştur. Bunun için bilgiyi en hızlı, verimli ve kullanışlı bir şekilde elde etmek günümüz teknolojilerinin ana konusu haline gelmiştir.

Bu amaçla Very Large Scale Integration (VLSI) devre tasarımlarında yeni bir açılım yakalamak ve gelişen son teknolojiyle birlikte bu tasarımın daha az maliyetle ve verimli bir şekilde yapılmasını sağlamak amacıyla bu çalışma başlatılmıştır.

Çalışmalar sonunda VLSI bilgilerinin ortak bir standarta getirilmesi amacına yönelik olarak XML teknolojisinin çok uygun olduğu görülmüştür. Ayrıca devre çizimleri için ayrıntılı ve herbiri ayrı bir kurulum gerektiren programların yerine basit bir HTML sayfasının kullanılabileceği gösterilmiştir.

Bana bu konuda çalışma olanağı veren danışmanım Sayın Yrd.Doç.Dr. Fatih KOÇAN'a (G.Y.T.E Bilg. Müh.), yardımlarını gördüğüm Sayın Yrd.Doç.Dr. İbrahim SOĞUKPINAR'a (G.Y.T.E. Bilg. Müh.) teşekkürlerimi sunarım.

İÇİNDEKİLER DİZİNİ

	Sayfa
ÖZET	iv
SUMMARY	v
TEŞEKKÜR	vi
İÇİNDEKİLER DİZİNİ	vii
SİMGELER VE KISALTMALAR DİZİNİ	ix
ŞEKİLLER DİZİNİ	x
1.GİRİŞ	1
1.1. Çalışmanın Kapsamı	1
1.2. Konuyla İlgili Daha Önceden Yapılmış Çalışmalar	2
2. UYGULAMADA KULLANILAN ALTYAPI	3
2.1. XML (Extensible Markup Language)	3
2.1.1. Tanımı	3
2.1.2. Tarihçesi	8
2.1.3. Kullanım Alanları	9
2.2. VML (Vector Markup Language)	10
2.2.1. Tanımı	10
2.2.2. Neden VML?	11
2.3. Javascript	11
3. YAPILAN UYGULAMA	12
3.1. XML Bilgi Altyapısının Oluşturulması	12
3.2. Webde Yayınlama	19
3.3. Simülasyon	21
3.4. XML Formatındaki İki Devrenin Karşılaştırılması	24
4. SONUÇLAR / ÖNERİLER	27
KAYNAKLAR	28

ÖZGEÇMİŞ**EKLER**

EK-1 Uygulama Kaynak Kodu – 1 (tez.htm)

EK-2 Uygulama Kaynak Kodu – 2 (tez-cizim.htm)

Ek-3 Örnek Netlist

Ek-4 Örnek XML Dökümanı

EK – 5 Örnek Çıktılar

EK-6 XML Uygulama Alanları



SİMGELER VE KISALTMALAR DİZİNİ

VLSI:	Very Large Scale Integration
XML:	Extensible Markup Language
VML :	Vector Markup Language
HTML:	Hyper Text Markup Language
IE5.0:	Microsoft Internet Explorer Version 5.0
JS:	Java Script
SGML :	Standard Generalised Markup Language
HRML:	Hospital Record Markup Language
DTD:	Document Type Definition
CSS:	Cascading Style Sheet
XSL:	Extensible Style Language



ŞEKİLLER DİZİNİ

Şekil	Sayfa
2.1. XML Namespaces (İsim Havuzları)	7
2.2. HTML ve VML'in Görüntülenmesi	10
3.1. Logic Kapıların Değişik Gösterimi	14
3.2. Not Kapısının Şematik Gösterimi	15



1. GİRİŞ

Günümüzde bilgi en önemli güç kaynaklarından birisi olmuştur. Bunun için bilgiyi en hızlı, verimli ve kullanışlı bir şekilde elde etmek günümüz teknolojilerinin ana konusu haline gelmiştir.

Bu amaçla Very Large Scale Integration (VLSI) devre tasarımlarında yeni bir açılım yakalamak ve gelişen son teknolojiyle birlikte bu tasarımın daha az maliyetle ve verimli bir şekilde yapılmasını sağlamak amacıyla bu çalışma başlatılmıştır.

Çalışmada VLSI tasarımı, gate-level olarak ele alınmıştır. Tasarım aşamasındaki bilgilerin tutulmasında bir standartlaştırma çalışması yapılarak verinin dolaşımı, bölümler arasındaki iletimi daha kolay ve standart bir hale getirilmeye çalışılmıştır.

1.1. Çalışmanın Kapsamı

Bu çalışmada VLSI tasarımının sadece bir modülü baz alınmıştır. Bu modül, lojik kapılar cinsinden tanımlanmış ve işlemler buna göre yapılmıştır. Herhangi bir formattaki modülü oluşturan kapı bilgileri önce belirlenen bir netlist formatına dönüştürülür. Daha sonra bu netlistten modülü oluşturan kapılar, aralarındaki ilişkiler ve kapıların birtakım özellikleri belirlenen formattaki Extensible Markup Language (XML) dökümanına dönüştürülür.

Daha sonra ise, bu XML tabanlı bilgiler çeşitli şekillerde işlenerek kapıların bir web sayfası üzerinde grafiksel çizimi yapılır.

En son aşama olarak ise verilen giriş değerlerine bağlı olarak çizilen kapılar üzerinde gate-level, zero-delay bir simülasyon çalışması yapılmaktadır.

Kullanılacak araçlar, standart bir internet kullanıcısının sahip olduğu araçlar olarak belirlenmiştir. Çalışmanın tamamının yapılabilmesi için, XML ve Vector Markup Language (VML) destekli bir web tarayıcısı yeterli olmaktadır. Günümüzde ki popüler web tarayıcılarının bir çoğu (Internet Explorer (IE5.0) ve Netscape Navigator vb.) bu özelliklere sahiptir.

1.2. Konuyla İlgili Daha Önceden Yapılmış Çalışmalar

Yapılan araştırmalara göre VLSI devrelerinin bilgi tabanı XML olarak ve VML kullanılarak web üzerinde gösterilmesi ile ilgili çalışmalar çok fazla değildir. Hatta bu çalışma bu konudaki ilklerden birisidir. Konuyla ilgili olarak yapılan çalışmalar çok azdır¹. Ayrıca bir konferansta konu başlığı olarak ele alınmıştır².

¹ <http://www.cs.ncl.ac.uk/modules/1998-99/projects/offered/>

² http://www.cs.ait.ac.th/research_report.shtml#sr

2. UYGULAMADA KULLANILAN ALTYAPI

Bir önceki bölümdede belirtildiği üzere uygulamanın her türlü platformda çalışması ve minimum araçlarla yapılması öngörülmüştür. Bu amaçlar doğrultusunda bilgi altyapısı olarak XML kullanılmış, verilerin kullanıcıya sunulması amacıyla ise Hyper Text Markup Language (HTML)³den faydalanılmıştır. HTML'de bilgiler işlenirken VML⁴ ve JavaScript⁵ (JS) kullanılmıştır.

Sırasıyla bu teknolojiler hakkında bilgi edinilmesi amacıyla kısa açıklamalar verilecektir.

2.1. XML (Extensible Markup Language)

2.1.1. Tanımı

Bir biçimleme dili (Markup Language), 'şeyleri' tanımlayan kurallar setidir. Bu şeyler, bir elektronik devre tasarımı, bir tıp tarihi, bir telefon defteri, bir nükleer santral veya hatta bir dil olabilir. Bu tanımlama genellikle bir elektronik döküman içinde yer alır. Verinin mahiyetini açıklayan verilerden ve etiketlerden (tag) oluşur.⁶

Günümüze kadar gelen ve hala mevcudiyetini sürdüren biçimleme dili Standard Generalised Markup Language – Standart Genelleştirilmiş Biçimleme Dili (SGML)⁷'dir. HTML ise web üzerindeki en ünlü biçimleme dilidir.

³ <http://www.w3.org/MarkUp/>

⁴ <http://www.w3.org/TR/MOTE-VML>

⁵ <http://developer.netscape.com/docs/manuals/communicator/jsguide4/index.htm>

⁶ <http://www.w3.org/XML/>

⁷ <http://www.w3.org/MarkUp/>

Basittir ama genişletilebilirlik eksikleri vardır ve içerikten çok görüntüleme üzerine yoğunlaşmıştır.

XML, web üzerindeki yapısal döküman ve dataların gösterimi için evrensel bir formattır.

XML, SGML'in oldukça basit bir alt kümesidir. XML, SGML ve HTML arasındaki iletişimi ve birlikteliği kolay bir şekilde sağlamak üzere tasarlanmıştır. Yapısal olarak XML belgeleri, SGML belgeleri tarafından doğrulanır.

XML, XML dökümanı adıyla bilinen veri objeleri sınıfını tanımlar ve kısmen onları işleyecek edecek olan bilgisayar programlarının davranışlarını tanımlar.

XML, veri yapılandırmasını text formatında tasarlamak için bir kurallar bütünüdür. XML, bir programlama dili değildir. XML, veri üretmeyi, okumayı ve verinin tek olmasının sağlanmasını bilgisayarlar için kolaylaştırır. XML dil tasarımında genel yapılan eksikliklerden kaçınır, o genişletilebilir, platform bağımsızdır, uluslararası dilleri ve yerelliği destekler. XML tam olarak Unicode⁸ uyumludur.

XML, SGML gibi kendi başına bir dildir, diğer bir biçimleme dillerini açıklayan biçimleme dilidir. Bunun anlamı XML ihtiyaçlarımıza uygun olarak başka biçimleme dillerini tanımlamak için kullanabiliriz. Mesela VLSI kayıtlarını kayıtlarını tanımlamak için bir dil oluşturmak istersek, XML'i VLSI Record Markup Language (VRML)'i tanımlamak için kullanabiliriz. Yani XML'i genişleterek VRML'i oluştururuz.

⁸ <http://www.unicode.org/>

Örnek olarak lojik bir kapının adını tanımlamak için şu şekilde bir yapı kullanılabilir.

Örnek:

```
<gate_name>G1</gate_name>
```

Burada <gate_name> etikettir, ve "G1" o etiketin belirttiği veridir.

Verinin gösterilmesi problemi verinin içeriğinin tanımlanmasından tamamen farklı bir olaydır. Eğer ortak bir dil üzerinde uzlaşma sağlanırsa verini değişimi daha basit ve güvenli olacaktır. VRML ile oluşturulan dökümanlar farklı formatlarda ve aygıtlar üzerinde farklı şekillerle gösterilebilir. Bir web browser ve mobil telefonda farklı şekillerde ihtiyaçlara uygun olarak gösterilebilir. Verinin sunulması, uygulamaların etiketleri nasıl gösterileceğini belirlenmesiyle ilgilidir.

Kendi oluşturduğumuz VRML'i Document Type Definition – Döküman Tip Tanımlaması (DTD) içinde de tutabiliriz. DTD, öğelerin geçerliliğini kontrol eder. Yani bir nevi klasik veritabanlarındaki alanların özellikleri gibi. Mesela integer olarak tanımlanan bir alana string değer girilememesi gibi. Dökümanları gösterimi için stylesheet'ler kullanılabilir. Stylesheet'ler, HTML'de kullanılan Cascading Style Sheet (CSS)⁹ler veya daha yeni bir tanımlama olan eXtensible Stylesheet Language (XSL)¹⁰ ile implement edilebilir.

XML ile ilgili başka standartlarda mevcuttur. Xlink, Xpointer veya X-Schema gibi¹¹. XML diğer biçimleme dillerini tanımlar ve SGML'den daha basittir.

⁹ <http://www.w3.org/Style/CSS/>

¹⁰ <http://www.w3.org/Style/XSL/>

¹¹ <http://www.w3.org/XML/>

Aynı zamanda HTML'den üstündür, çünkü XML, verinin içeriğinin ve yapısını tanımlamaya izin verir, oysa HTML sadece verinin nasıl görüneceği ile ilgilidir.

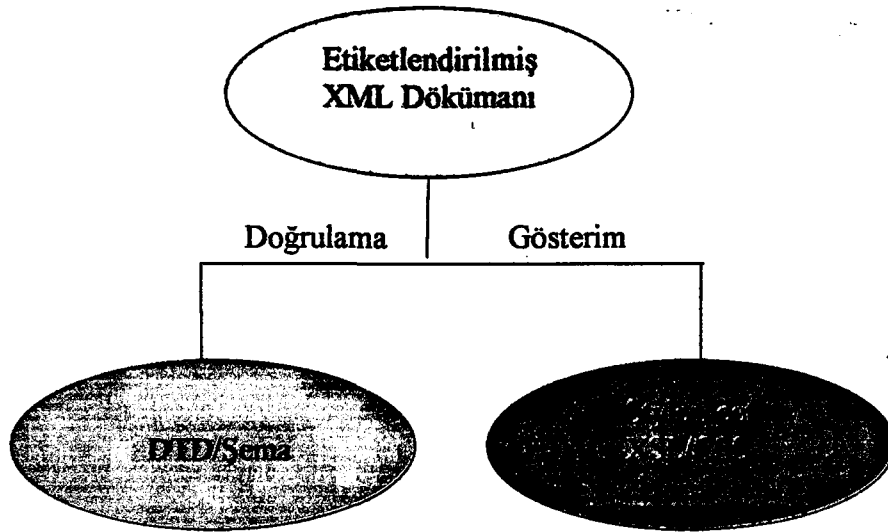
HTML'de olduğu gibi XML'de etiketleri ve özellikleri kullanır. Ama HTML'de her bir etiket ve onun özellikleri önceden belirlenmesine ve etiketlerin ne yapması gerektiği önceden belirlenmesine karşın XML de etiketler ve özellikleri sadece o xml'i yazan tarafından bilinir ve her bir uygulamada farklı davranış sergileyebilir. Mesela <p> etiketi HTML'de bir paragrafı belirtirken XML içinde bir personel'i veya bir parametreyi gösterebilir.

Programlar, adres defterleri, muhasebesel bilgiler ve diğer yapılandırılmış bilgileri üreterek bu bilgileri disk üzerinde binary veya text formatında saklanmasını sağlarlar. Text formatının avantajı, bilgileri üreten programdan bağımsız olarak bu bilgilerin herhangi bir text editor yardımıyla okunabilmesidir. HTML ve XML text formatında saklanır. Fakat hiç kimse bunları text olarak okumaz sadece gerektiğinde okunur. XML'in kuralları HTML'e göre daha katıdır.

Çok geniş uygulama alanına sahiptir. Hayatın her alanında kullanılabilir.

XML 1.0 etiketleri ve özellikleri tanımlayan kurallardır. XML 1.0'ın yanında "XML ailesi" sıkça kullanılan işlerin yapılması için gittikçe büyüyen modüller setine sahiptir. Xlink, bir XML dosyasına web adresleri eklemenin standart yolunu tanımlar. Xpointer ve Xfragments bir XML dökümanını parçalamak üzere geliştirilmiştir. Xpointer biraz URL'e benzer fakat farklı olarak bir web adresini değil XML dökümanı içindeki bir veriyi gösterir. CSS, XML'i HTML olarak görüntülemek için uygulanabilir. XSL daha gelişmiş bir stil dilidir. XSLT tabanlıdır. XSLT etiket ekleme ve özellik ekleme veya değiştirmek veya silmek için kullanılır. DOM, bir programlama dili içinden XML dosyalarını yüklemek ve işlemek için kullanılan fonksiyonlar bütünüdür.

XML Schemas 1 ve 2 geliştiricilerin kendi XML tabanlı format yapılarının tanımlanmasına yardımcı olur. Bunlar gibi birçok modul ve araç kullanılabilir durumdadır ve bir yandan da geliştirilmeye devam etmektedir.



Şekil.2.1. XML Namespaces (İsim Havuzları)

XML birlikte çalışmakla ilgilidir ve herkes bağımsız olarak kendi XML kelimelerini oluşturabilir. Bu durum farklı geliştiricilerin birbirinden farklı varlıkları tanımlamak için belirli kelimeleri seçmesine neden olabilir. Bu karışıklıktan kurtulmak için W3C XML için bir isim havuzu oluşturmuştur.

Mesela bir hamburgerci örneğini ele alalım. Burda 'fiyat' etiketi iki farklı şekilde anlaşılabilir. Çünkü hamburgerin bir müşteri satış fiyatı birde firma alış fiyatı vardır. Bunu önlemek için iki aynı etiket farklı isim havuzlarında farklı şekilde yorumlanmalıdır.

```
<?xml version="1.0"?>
```

```
<hamburgers
```

```
xmlns:purchase="http://fastfood.org/franchise/prices"
```

```
xmlns:sales="http://fastfood.org/customer/prices">
```

```
<hamburger lowfat="dream on">
```

```
<name>CowBurger</name>
```

<description>Greasy and good.</description>

<purchase:price>0.99</price>

<sales:price>2.99</price>

</hamburger>

</hamburgers>

2.1.2. Tarihçesi

XML, 1986'da ISO8879 olarak tanıtılan SGML'in bir alt kümesidir. Sanayi, telekomünikasyon vb. endüstrilerde kullanılmıştır. Daha sonra 1996 yılındaki SGML konferansında W3C'ye bağlı bir çalışma grubu tarafından XML yayınlandı. XML 1.0, 1998 yılının şubat ayında 32 sayfalık bir tavsiye olarak yayınlandı.¹²

XML, SGML ve HTML'e halef olarak ortaya çıkarıldı, SGML ve HTML alanlarında başarılı idiler fakat bazı yönlerden eksiklikleri vardı.

SGML, verileri etiketlemek için 80'lerden itibaren kullanılan uluslararası bir standarttır. SGML son derece güçlü ve esnektir. SGML'den, XML gibi birçok biçimleme dili ortaya çıkarılabilir.

Buna rağmen SGML bir miktar karmaşıktır, özellikle web üzerinde hergün kullanılması. Sadece bu değil, bir miktarda pahalıdır. Bir kelime işlemciye SGML yeteneği kazandırmak fiyatı ikiye veya üçe katlayabilir. Sonuç olarak ticari browserlar SGML sevimli kılmıştır.

Diğer yandan HTML bedava, basit ve çok yaygın desteği vardır. HTML orjinal olarak 1990'larda, SGML'in basit bir versiyonun herkes tarafından

¹² <http://www.w3c.org/XML/>

kullanılabilmesi için üretildi. Bundan sonra HTML söndürülmesi imkansız bir yangın gibi her tarafa yayıldı.

1996'da SGML'in gücüne sahip ve HTML'in basitliğinde bir biçimleme dilinin nasıl oluşturulabileceğine dair tartışmalar başladı. W3C (World Wide Web Consortium), Sun'dan Jon Bosak'ında aralarında bulunduğu bir grup SGML gurusunu destekleme kararı aldı.

Aslında Bosak ve takımı, Java takımı C++'a ne yaptıysa SGML'e aynısı yaptı. Sonuçta, SGML'in kullanılmayan, bazı bölümleri kırpıldı. Geriye biçimleme makinesi XML kaldı. XML özellikleri (çoğunluğu Tim Bray ve C.M. Sperberg-McQueen tarafından yazıldı) sadece 26 sayfaydı. Oysa SGML özellikleri 500 sayfadan fazlaydı. Yinede yararlı bazı işler hala SGML ile yapılabilir, aynı zamanda XML ile de yapılabilir.

Sonuç olarak 1998'de W3C'de XML 1.0 spesifikasyonlarını yayınladı ve böylece yeni bir dil doğmuş oldu.

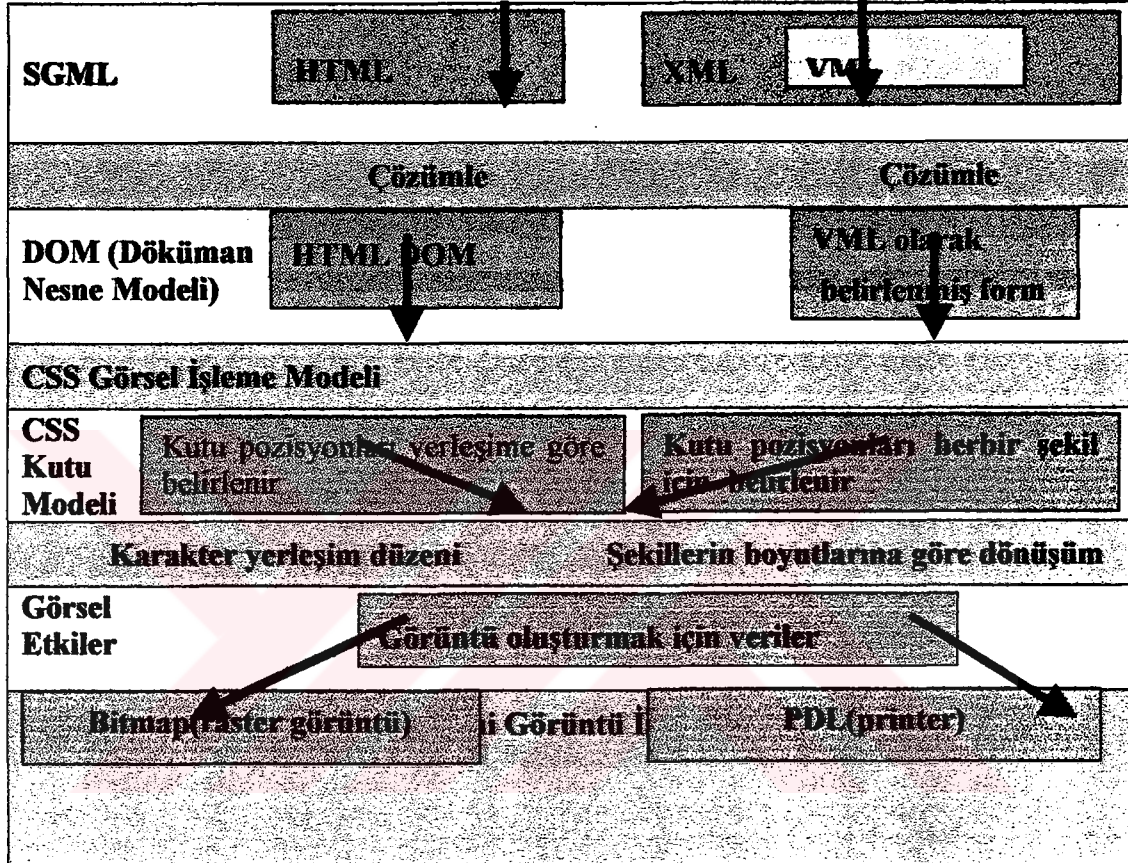
2.1.3. Kullanım Alanları

XML hayatın her alanına uygulanabilir.¹³ Aklımıza gelen her türlü alanda XML kullanabiliriz. Çünkü XML, bir programlama dili değil, bir veri biçimleme dilidir.

¹³ Konuyla ilgili olarak Ek-6'da ayrıntılı bilgi mevcuttur.

2.2. VML (Vector Markup Language)

2.2.1. Tanımı



Şekil.2.2. HTML ve VML'in Görüntülenmesi

Vector biçimleme dili olarak adlandırılan VML, birtakım komutlarla HTML sayfaları üzerinde grafik ve şekiller çizmeye yarayan, XML tabanlı bir biçimleme dilidir¹⁴.

VML'de kullanıcılar hazır şablonları kullanabilecekleri gibi kendi şablonlarını da oluşturup kullanabilirler.

¹⁴ www.w3.org/TR/NOTE-VML

2.2.2. Neden VML?

VML tercih edilmesinin nedeni, oluşturulacak şekilleri birkaç satırlık kodla oluşturmak ve bu şekilleri alışlagelen herhangi bir resim dosyasından almamaktır. Eğer her bir şekli bir .gif veya .jpeg formatlı bir dosyadan almaya çalışılması durumunda oluşacak HTML sayfasının yüklenmesi çok uzun sürecektir.

2.3. Javascript

HTML'de verileri işlemek için kullanılabilecek araçlar çok fazla değildir. VBScript¹⁵ ve Javascript kullanılabilir. Daha yaygın olduğu için Javascript tercih edilmiştir.

¹⁵ <http://msdn.microsoft.com/scripting/>

3. YAPILAN UYGULAMA

Bu çalışmada temel olarak elektronik devre çizimleri için bir veri alışverişi formatı belirleyerek yine XML'in bir altkümesi olan VML kullanarak bu devrelerin webde yayınlanmasını sağlayacak bir yapı oluşturmaya çalışılmıştır.

Kapılar ve aralarındaki ilişkiler netlist'ten alınıp belirlenen XML formatına dönüştürülür.

Sonra oluşturulan bu XML, Javascript yardımıyla işlenerek elektronik devre webde yayınlanır. Daha sonra bu devrenin simülasyonu yapılır.

3.1. XML Bilgi Altyapısının Oluşturulması

VLSI tasarımının sadece gate level kısmı XML'e genişletilmiştir. Özetle girişler, çıkışlar ve ara kapıların bilgileri mevcuttur. Bu bilgiler çok farklı şekillerde olabilir. Bunun için önce verilen bilgiler belirlenen standart bir formata dönüştürülecek ondan sonra XML hali oluşturulacaktır.

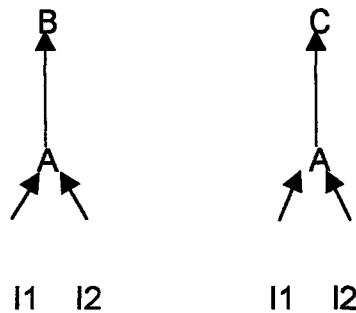
Bu bilgilerin XML haline dönüştürülüp standart bir formata sahip olması demek, XML'i destekleyen herhangi bir uygulama tarafından kolayca işlenebilir olması anlamına gelir. Aksi takdirde her uygulama kendi formatını tanımlayacak ve bilgilerin değişimi aşamasında, bu bilgilerin okunması ve işlenmesi kaynakların israfına neden olacaktır. Günümüzde az kaynakla en verimli şekilde çalışmak ve minimum hata yapmak üretimin temelini oluşturmaktadır. Bilgilerin tek bir formatta standartlaşması bilgi transferi ve sonrasında meydana gelebilecek hataları da minimize edecektir.

Bu işlemler yapılırken kullanıcının minimum bir konfigürasyonla işlemlerini yapabilmesi ve sonuçları görebilmesi amaçlanmıştır. Bu nedenle günümüz popüler web tarayıcılarından herhangi birine sahip olan herkes bu işlemleri kendi bilgisayarını aracılığıyla herhangi bir web sunucuya ihtiyaç duymadan gerçekleştirebilmektedir. Bunun için tarayıcının, XML, VML ve Javascript'i desteklemesi yeterlidir. Zaten birçok tarayıcı sayılan bu özellikleri desteklemektedir. Örneğin IE5.0 ile bahsedilen işlemler gerçekleştirilebilir.

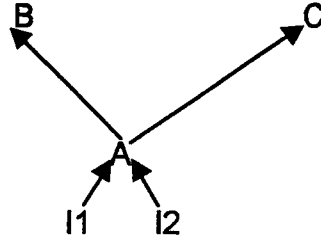
VLSI tasarımında kullanılan kapı girdilerinin XML formatına dönüşmesi demek, XML tabanlı yeni bir biçimleme dilinin ortaya çıkması demektir.

Bilindiği üzere lojik kapılar bir veya daha fazla girişe ve bir çıkışa sahip mantıksal birimlerdir. Aynı zamanda bir kapının çıkışı başka bir kapı veya kapıların girişi olabilir. Burada modüler bir yapı mevcuttur. Aynı kapı farklı kapılar tarafından ortak giriş olarak kullanılabilir. Burada kapılar arasındaki ilişkileri gösteren bilginin tekrarlayan bir biçimde değil aynı bilgiyi tekrar kullanabilir bir şekilde olması gerekir. Yani A kapısının çıkışı, B ve C kapılarının girişleri ise ve A kapısı da başka kapılardan meydana geliyorsa A kapısını B ve C kapılarının altında ayrı ayrı tanımlamak yerine A kapısını gösteren bir işaretin olması yeterli olmak zorundadır. A kapısı I1 ve I2 girişlerinden oluşsun. Bu yapıyı iki şekilde göstermek mümkündür.

a.



b.



Şekil.3.1. Logic Kapıların Değişik Gösterimi

Şekil 3.1.a gösterimi esnekliği yok eder ve modifikasyonların çok daha zor olmasına yol açar. Mesela A kapısına bir üçüncü girişin eklenmesi demek her iki kapıya erişip ordaki A kapısı tanımlarının güncelleştirilmesini gerektirir. Oysa şekil 3.1.b gösteriminde her ikiside A kapısına bir link bulunduğu için A kapısı üzerinde yapılacak tüm modifikasyonlar ek hiçbir zahmete katlanmadan devreye direk yansiyacaktır. İşte XML bu esnekliği çok kolay bir biçimde sağlar. XML, kapılar arasındaki ilişkilerin daha okunabilir olmasını, daha kolay işlenebilmesini ve minimum düzeyde hata yapılmasını sağlar.

Temel olarak bir modul girişlerden (inputs), çıkışlardan (outputs) ve ara kapılardan (gates) oluşur.

Verilen Netlist'ten XML formatı aşağıdaki yapıda tasarlanmıştır.

VLSI

Modul Bilgileri – Modul İsmi açıklama, tarih vs..

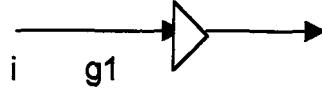
Elements – Modulu oluşturan giriş ve çıkış elemanları

Def_outputs – Çıkışlar

Def_inputs – girişler

Kapılar – kapıların özellikleri (isim, tip vb) ve bu kapıyı oluşturan kapıların veya girişlerin pointerı

Örnek olarak tek bir girişe sahip bir not kapısının gelişimi aşağıdaki şekildedir.



Şekil 3.2. Not kapısının şematik gösterimi

INPUT(i)

OUTPUT(g1)

$g1 = \text{not}(i)$

Netlist'e Dönüşüm yapılır.

```
module test(i,g1);
output g1;
input i;
not g1(i);
end module;
```

sonrada XML formatına çevrilir.

```
<tez>
<module_name>test</module_name>
<elements>
<element>g1</element>
<element>i</element>
</elements>
<def_outputs>
<output>g1</output>
</def_outputs>
<def_inputs>
```

```
<input>i</input>
</def_inputs>
<output name="g1" gate_name="g1" gate_type="not">
<input>i</input>
</output>
</tez>
```

Burada neden iki kere dönüşüm yapıldığı sorusu akla gelebilir. Burada XML'e dönüşüm için standart bir netlist formatı belirlenmiştir. Bunun için diğer tüm formatlar önce belirlenen bu formata çevrilir. Ondan sonra ise XML formatına dönüştürme işi yapılır. Diğer yol da tercih edilebilirdi.

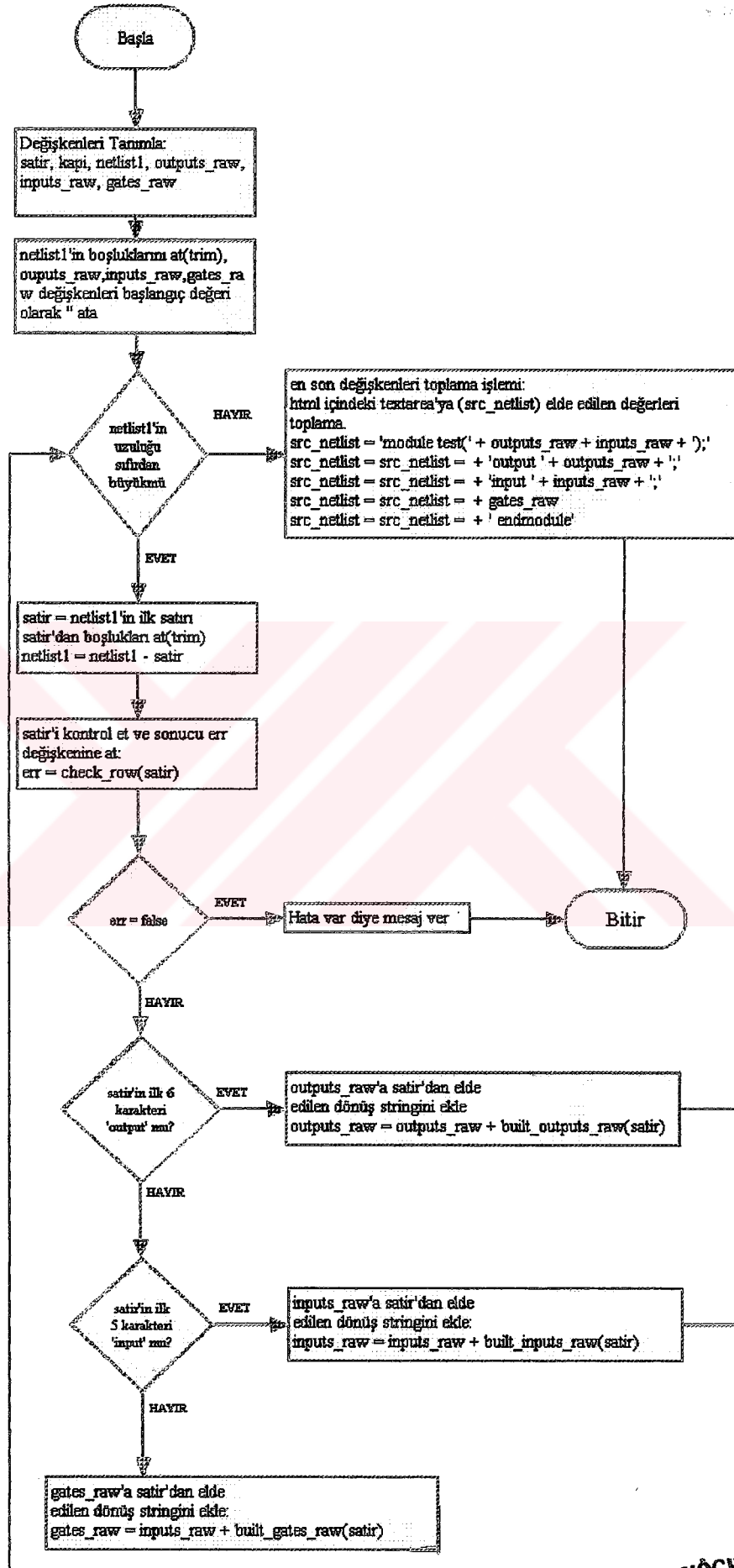
Bu çalışmada netlisti XML'e çevirmek için iki tane fonksiyon türetilmiştir.

`built_from_raw(text data)`

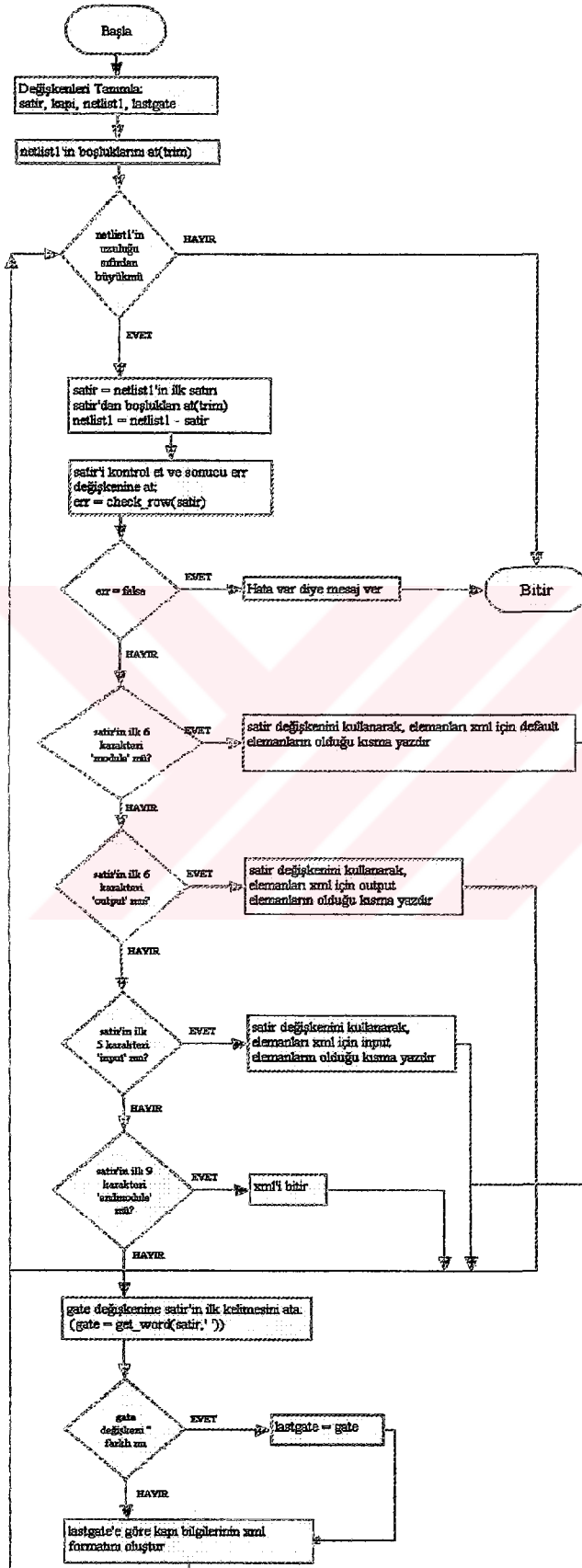
`built_xml(netlist data)`

`built_from_raw(text data)` fonksiyonu yukarda bahsedilen ilk dönüşümü yapmak için kullanılır. Bu fonksiyona parameter olarak kapılar ve aralarındaki mantıksal ilişkiyi gösteren yazı tabanlı bilgi verilir. `built_xml(netlist data)` fonksiyonu ise bu dönüşümden sonar asıl XML formatındaki bilgiyi üretmek için kullanılır.

İlk fonksiyon verilen veriyi satır satır okuyarak ara formu oluşturur. Fonksiyona parametre olarak kaynak netlist verilir.



bu ara form oluşturulduktan sonra, oluşan text yine satır satır okunarak built_xml(netlist) fonksiyonu vasıtasıyla XML formatına dönüştürülür.



3.2.Webde Yayınlama

VLSI devrelerin webde yayınlanması demek devrenin grafiksel olarak bir web sayfası üzerinde gösterilmesi demektir. Burada yapılan devrenin gate level olarak lojik bakımından kapıların ve aralarındaki ilişkilerin bir web sayfasından gösterilmesidir.

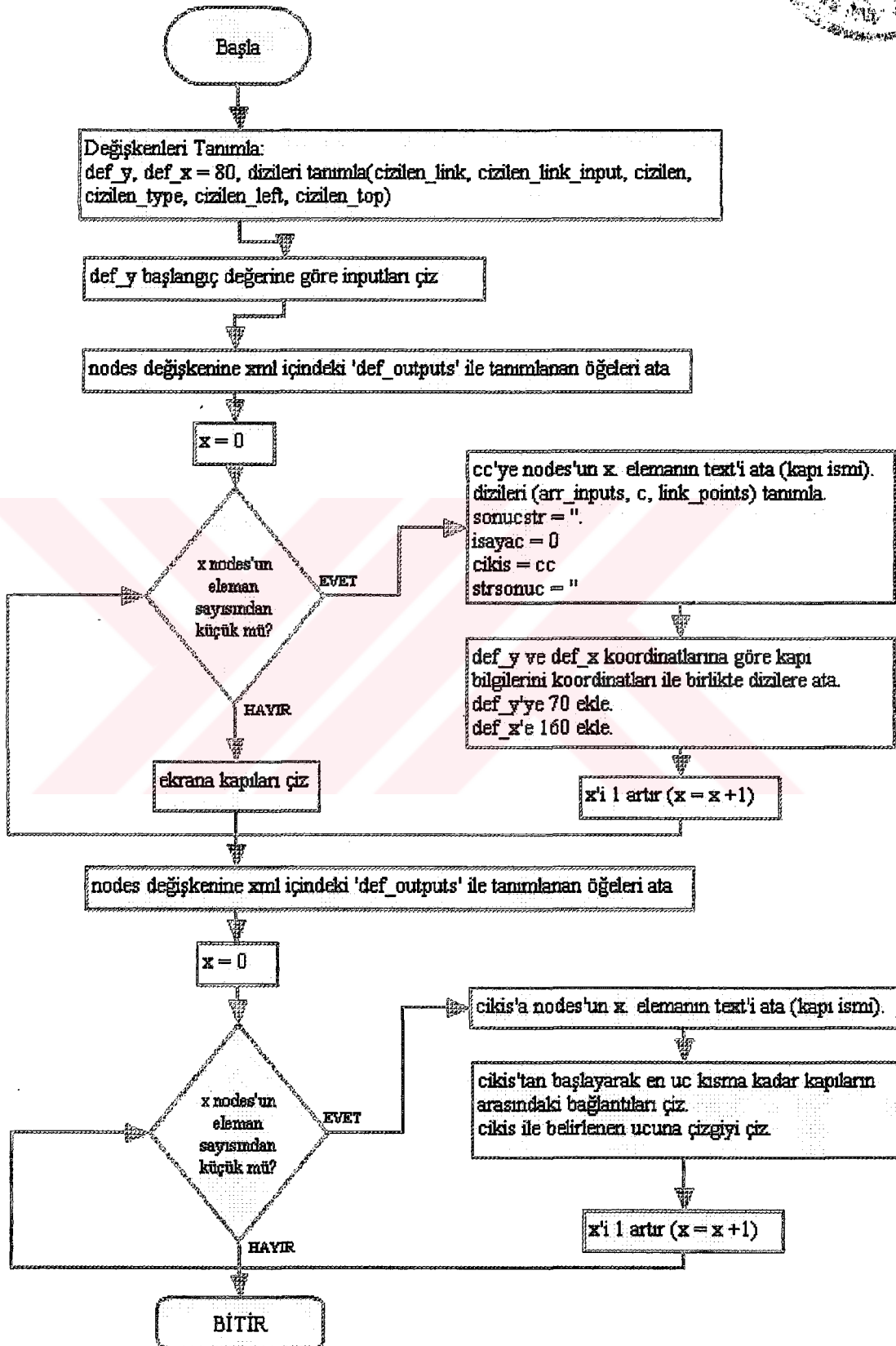
Bunun için yine Javascript dili kullanılmıştır. Grafiksel österimleri yapmak için ise bölüm 2.2'de anlatılan VML kullanılmıştır. Tabii ki yine bilgilerin için temel olarak önceden hazırlanan dökümanı esas alınmıştır.

İlkönce bütün kapı tiplerinin grafiksel gösterimi prototip olarak hazırlanmıştır. VML'in özelliği olarak hazırlanan her bir prototip'e bir ID verilmiştir. Daha sonra sadece bu ID'ler kullanılarak şekillerin ekran üzerine yansıtılması sağlanacaktır. Modüler bir yapı oluşmuştur.

Bu grafiksel gösterim oluşturulurken ilkönce ilk girişler ekrana çizilmiş ve kapı çizimlerinin başlayacağı koordinat belirlenmiştir (def_y).

Daha sonra XML'den bilgiler okunarak kapılar ve bunlarla ilgili uygun koordinatlar bir diziye aktarılmıştır. Bir sonraki adım olarak bu diziden bilgiler okunarak kapılar ekrana çizilmiştir. Sonraki adımda ise kapılar arasındaki bağlantılar çizilmiş ve en son aşama olarak da belirtilen çıkışların ucu çizilmiştir.

Bu işlemleri gerçekleştiren ana fonksiyonun aşağıdadır.



`draw_inputs()` : fonksiyon ile ilk seviyede bütün girişler ekrana çizilir.

`draw_output1(cikis, def_x, def_y)` : Bu rekursif bir fonksiyondur. Verilen kapıdan başlayarak altındaki bütün kapıları ilk girişe kadar kontrol ederek kapıları ve koordinatlarını bir diziye kaydeder. Bu esneklik sayesinde istenirse devre ortada bir yere kadar da çizdirilebilir.

`draw_gates()` : Bu fonksiyon `draw_output1()` fonksiyonu tarafından üretilen dizileri kullanarak kapıları ekrana çizdirir.

`draw_link(cikis)` : Bu fonksiyon da rekursif bir fonksiyondur. Verilen çıkıştan başlayarak altındaki bütün kapıları ilk girişlere kadar tarar ve bu kapılar arasında ki bağlantıları çizer.

`draw_output (cikis)` : Bu fonksiyon verilen çıkışın ucuna en son çıkış çizgisini çizer.

3.3. Simülasyon

Bu bölümde verilen girişlere bağlı olarak çıkışların hesaplanması ve görsel olarak kullanıcıya sunulması kısmı yapılmıştır.

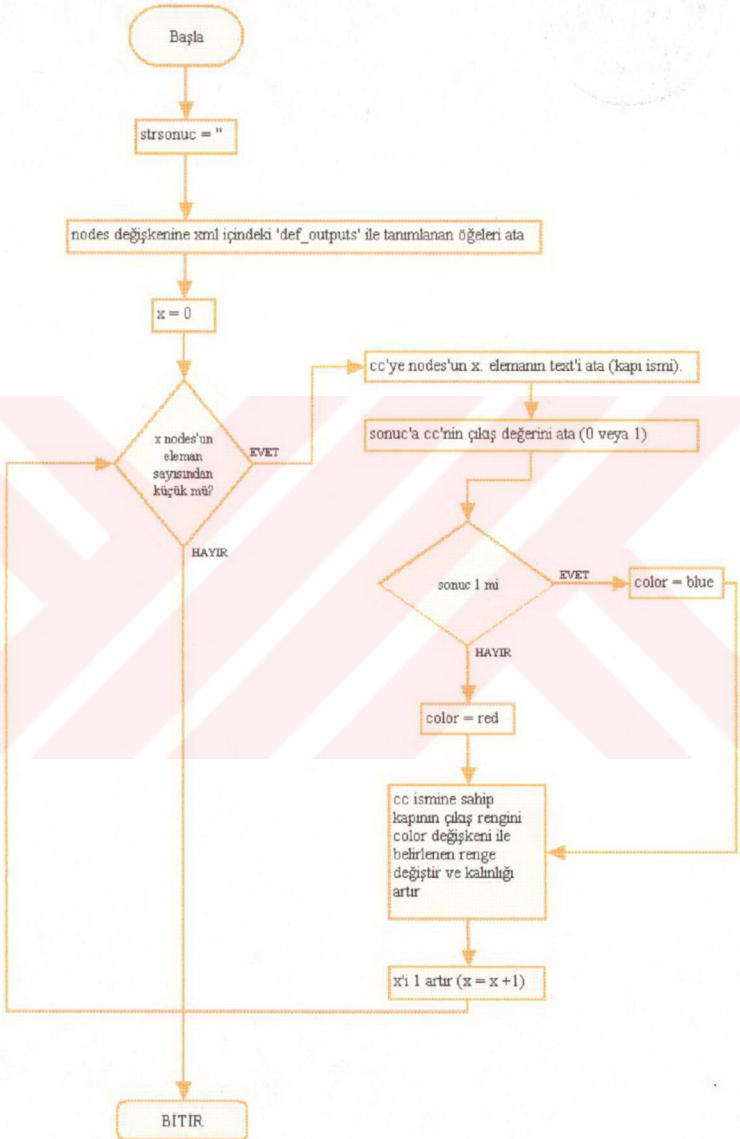
Similasyon alogritması olarak gate-level zero-delay algoritması esas alınmıştır. Yani sadece giriş değerleri verilecek zaman bağlı olmaksızın çıkışlar hesaplanacak ve ekrana yansıtılacaktır.

Kapıların alacağı değerlerin hesaplanması ve görsel olarak kullanıcıya sunulması işlemi aynı fonksiyon içinde yapılmıştır.

Kapılar arasındaki her bir bağlantı ayrı bir ID'ye sahiptir. Bu ID'ler `draw_link()` fonksiyonu tarafından belirlenmiştir. Mesela G1'in çıkışı G2'nin girişi olsun. Bu aradaki bağlantının ID'si 'G2->G1' olacaktır. Similasyon işlemi yapılırken sadece belirlenen değere göre bu bağlantının renginin değiştirilmesi sözkonusudur. Renkler şu şekilde belirlenmiştir. Eğer değer 0 ise kırmızı, 1 ise mavi olacaktır.

Bu işlemler `evulate_outputs()` fonksiyonu ile yapılır. Fonksiyonun genel algoritması aşağıdaki şekildedir.





3.4. XML Formatındaki İki Devrenin Karşılaştırılması

XML formatına dönüştürülen iki devrenin birbirine eşdeğer olup olmadığını karşılaştırmak çok kolaydır. Devrenin girişlerinin sırasının değişmesi veya devredeki kapı isimlerinin değişmesi devrelerin eşdeğer olup olmadıklarını anlamak için bir engel teşkil etmeyecektir. İki devrenin eşdeğer olup olmadığını anlamak için aşağıdaki adımlardan oluşan bir yöntem takip edilmiştir.

- Herbir çıkış için devreyi mantıksal olarak ifade eden bir satır oluşturulur.
- Bütün çıkışlar için bu tekrarlanır.
- Daha sonra diğer XML formatındaki devre için yukardaki adımlar tekrarlanır.
- Bir sonraki adımda oluşturulan bu diziler birbiriyle karşılaştırılır. Eğer diziler birebir eşleşirse devreler eşit demektir, aksi halde devreler eşit değildir.

Örnek :

$$g3 = \text{or}(g1, g2)$$

$$g1 = \text{and}(a, b)$$

$$g2 = \text{and}(c, d)$$

$$g4 = \text{nand}(e, g2)$$

a, b, c, d, e girişleri göstermektedir.

Aynı devreyi ifade eden başka bir liste de şu şekilde verilmiş olsun.

$$f1 = \text{and}(e, f)$$

$$f3 = \text{or}(f1, f2)$$

$$f2 = \text{and}(g, h)$$

$$f4 = \text{nand}(e, f2)$$

Bu iki devre yazım ve isimlendirme olarak birbirinden tamamen farklı görünse bile yaptıkları işlev bakımından eşdeğerdir. Bunu test etmek için aşağıdaki fonksiyondan faydalanılır. Aşağıdaki fonksiyon devrelerin şu şekilde mantıksal yapılarını oluşturur.

`or(and(i,i),and(i,i))`

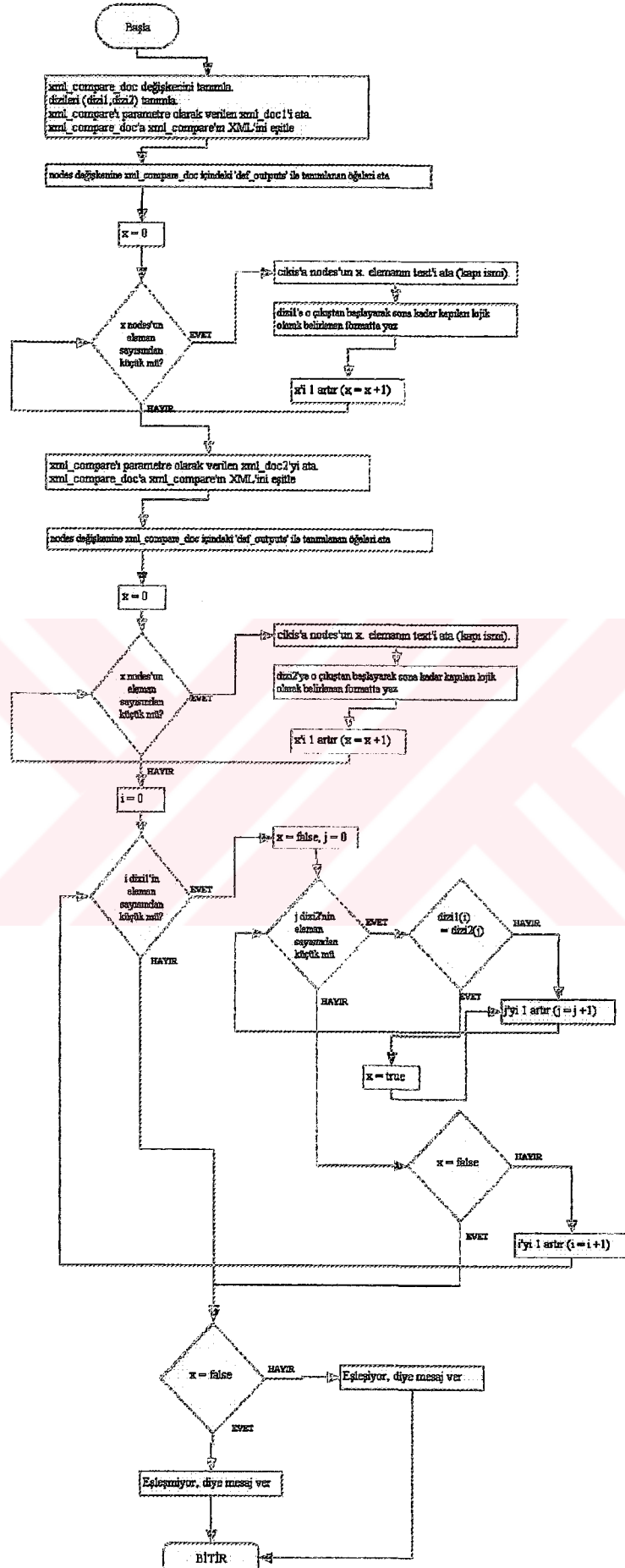
`nand(i,and(i,i))`

ve yine aynı şekilde ikinci devrenin de mantıksal yapısı çıkartılır.

`or(and(i,i),and(i,i))`

`nand(i,and(i,i))`

ve bu iki satır birbirine denk olduğundan devreler eşdeğerdir sonucuna ulaşılır. Fonsksiyonun alogirtması aşağıdaki şekildedir.



4. SONUÇLAR / ÖNERİLER

Bu çalışmayla, web teknolojilerinin VLSI tasarımına uygun olduğu görülmüştür. Kurulumu ve kullanması ve elde edilen sonuçların kişiler arasında dolaşımının daha zor olduğu programlar yerine basit bir HTML sayfasının yeterli olduğu görülmüştür. Ayrıca XML teknolojisinin bu bilgilerin değişimi ve işlenmesi için ne kadar elverişli bir teknoloji olduğu ortaya çıkmıştır.

Bu çalışmada sadece modul seviyesinde çalışma yapılmıştır. Bu projenin VLSI tasarımının tüm aşamalarını kapsayacak şekilde genişletilmesi faydalı olacaktır.

Bu çalışmada sadece istemci bilgisayarda çalışacak şekilde sistem tasarlanmıştır. Bu çalışma sunucu bilgisayarlar tarafında çalışacak uygulamalarla (ASP, PHP vb.) geliştirilebilir.

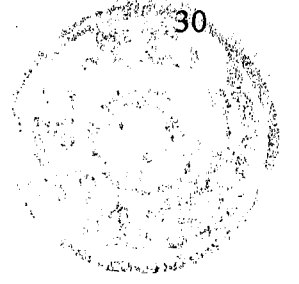
KAYNAKLAR

1. <http://www.carebridge.org/tech/xml-brief.html>
2. <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnxml/html/xmlguide.asp>
3. <http://www.w3.org/XML/1999/XML-in-10-points.html>
4. <http://www.w3.org/TR/MOTE-VML>
5. Adım Adım XML, Arkadaş Yayınevi, Ankara 2001
6. [Koelmans, 1998] Özel konuşmalar. Albert Koelmans, CS, New Castle Univ.
7. Tchumatchenko, V., Vassileva, T., Batanov, D., and Bojkov, D.,

ÖZGEÇMİŞ

Ahmet SAVAŞ 1978 yılında Osmancık/ÇORUM'da doğdu. İlköğrenimini Osmancık'ta, orta ve lise öğrenimini Çorum'da tamamladı. 1994 yılında girdiği İstanbul Üniversitesi Mühendislik Fakültesi Bilgisayar Bilimleri Mühendisliği bölümünü 1998 yılında Bilgisayar Bilimleri Mühendisi olarak bitirdi. 1999 yılında Gebze Yüksek Teknoloji Enstitüsü, Bilgisayar Mühendisliği Bölümüne girdi. Bu tarihten itibaren özel sektörde çeşitli görevler aldı. Halen özel bir şirkette bilgi işlem müdürlüğü görevini sürdürmektedir.





EKLER

EK-1 Uygulama Kaynak Kodu – 1 (tez.htm)

```
<HTML>

<HEAD>

<STYLE>

    a {font-family: Tahoma;}

    h3 {margin-bottom: 0pt;color: Red}

    h4 {margin-top: 0pt; margin-bottom: 0pt;}

    hr {margin-top: 0pt; margin-bottom: 0pt;}

</STYLE>

<TITLE>Tez Çalışması</TITLE>

</HEAD>

<BODY>

<h3>Tez Çalışması</h3>

<hr>

<xml id=xml_tez src=""></xml>

<div id=xml_alani style='background-color:Pink; color:White;'></div>

<textarea name=src_netlist_raw cols=30 rows=15 wrap="off">

INPUT(G1gat)

INPUT(G2gat)

INPUT(G3gat)

INPUT(G6gat)

INPUT(G7gat)

OUTPUT(G22gat)

OUTPUT(G23gat)

G10gat = nand(G1gat, G3gat)

G11gat = nand(G3gat, G6gat)

G16gat = nand(G2gat, G11gat)
```

```
G19gat = nand(G11gat, G7gat)
```

```
G22gat = nand(G10gat, G16gat)
```

```
G23gat = nand(G16gat, G19gat)
```

```
</textarea>
```

```
<textarea cols="25" rows="15" name="src_netlist" wrap="off">
```

```
module mux(f,a,b,sel);
```

```
output f;
```

```
input a,b,sel;
```

```
and g1(f1,a,nsel);
```

```
g2(f2,b,sel);
```

```
or g3(f,f1,f2);
```

```
not g4(nsel,sel);
```

```
endmodule
```

```
</textarea>
```

```
<textarea name=src_xml cols=25 rows=15 wrap="off">
```

```
</textarea>
```

```
<hr>
```

```
<input type=button value="Netlist Translate"
```

```
onclick="built_from_raw(document.all('src_netlist_raw').value)">
```

```
<input type=button value="XML Oluştur"
```

```
onclick="built_xml(document.all('src_netlist').value)">
```

```
<script language=JavaScript>
```

```
line_x = 0
```

```
arr_inputs = new Array()
```

```
isayac = 0
```

```
c = new Array()
```

```
sonucstr="
```

*****RAW'dan Olusturma*****

```

function built_from_raw(netlist)
{
    var satir, kapi, netlist1, outputs_raw, inputs_raw, gates_raw

    netlist1 = trim(netlist)

    outputs_raw = ""
    inputs_raw = ""
    gates_raw = ""

    while (netlist1.length > 0)
    {
        satir = get_row_raw(netlist1)
        satir = trim(satir)

        netlist1 = netlist1.substring(satir.length+2,netlist1.length)

        err = check_row_raw(satir)

        if (!err) {alert('Hata var');return;}

        else if (satir.substring(0,6) == 'OUTPUT') outputs_raw = outputs_raw +
        built_output_element_raw(trim(satir.substring(7,satir.length)))

        else if (satir.substring(0,5) == 'INPUT') inputs_raw = inputs_raw +
        built_input_element_raw(trim(satir.substring(6,satir.length)))

        else
        {
            gates_raw = gates_raw + built_gates_raw(satir)
        }
    }

    src_netlist.value = ""

    src_netlist.value = 'module test(' + outputs_raw + inputs_raw + ');' + '\n'
    src_netlist.value = src_netlist.value + 'output ' + outputs_raw + ';' + '\n'
    src_netlist.value = src_netlist.value + 'input ' + inputs_raw + ';' + '\n'
    src_netlist.value = src_netlist.value + gates_raw
    src_netlist.value = src_netlist.value + 'endmodule';
}

```

```
function get_row_raw(netlist)
```

```
{
```

```
  var i
```

```
  row = "
```

```
  for (i=0;i<netlist.length;i++)
```

```
  {
```

```
    if (netlist.substring(i,i+1) != ')') row = row + netlist.substring(i,i+1)
```

```
    else return row + ')'
```

```
  }
```

```
  return row
```

```
}
```

```
function check_row_raw(pstr)
```

```
{
```

```
  var i
```

```
  if (pstr.substring(0,3) == 'OUT' || pstr.substring(0,3) == 'INP') return true
```

```
  //parameter kontrol
```

```
  sol = 0
```

```
  sag = 0
```

```
  sonuc = false
```

```
  for (i=0;i<pstr.length;i++)
```

```
  {
```

```
    if (pstr.substring(i,i+1) == '(') {sonuc = true; sol++;}
```

```
    else if (pstr.substring(i,i+1) == ')') && sonuc {sonuc = true; sag++;}
```

```
  }
```

```
  if (sol !=1 || sag !=1) sonuc = false
```

```
  return sonuc
```

```
}
```

```

function built_output_element_raw(pstr)
{
    ss = get_word(pstr,')') + ','
    return ss
}

```

```

function built_input_element_raw(pstr)
{
    ss = get_word(pstr,')') + ','
    return ss
}

```

```

function built_gates_raw(pstr)
{
    var gate_name
    gate_name = get_word(pstr, '=')
    gate_name = trim(gate_name)
    pstr = pstr.substring(gate_name.length + 3, pstr.length)
    gate_type = get_word(trim(pstr), '(')
    gate_type = trim(gate_type)

    pstr = pstr.substring(gate_type.length+1, pstr.length)

    pstr = trim_space(pstr)
    ss = gate_type + '' + gate_name + '(' + gate_name + ',' + pstr + '\n'
    return ss
}

```

```

function trim_space(pstr)
{

```

```

var i
son = ""

for (i=0;i<pstr.length;i++)
{
    if (pstr.substring(i,i+1) != ' ')
    {
        son = son + pstr.substring(i,i+1);
    }
}

return son
}

//*****RAV'dan Oluşturma*****

//*****PL Oluşturma*****

function built_xml(netlist)
{
    var satir, kapi, netlist1, lastgate
    netlist1 = trim(netlist)
    while (netlist1.length > 0)
    {
        satir = get_row(netlist1)
        satir = trim(satir)
        netlist1 = netlist1.substring(satir.length+3,netlist1.length)
        err = check_row(satir)
        if (!err) {alert('Hata var');return;}
        else if (satir.substring(0,6) == 'module')
            built_default_element(trim(satir.substring(7,satir.length)))
        else if (satir.substring(0,6) == 'output')
            built_output_element(trim(satir.substring(7,satir.length)))
    }
}

```

```

    else if (satir.substring(0,5) == 'input')
built_input_element(trim(satir.substring(6,satir.length)))

    else if (satir.substring(0,9) == 'endmodule') finish_xml()

    else
    {
        gate = get_word(satir, ' ')
        if (gate != " ") lastgate = gate
        built_gates(satir.substring(gate.length+1,satir.length), lastgate)
    }
}
}
}

```

```

function get_row(netlist)
{
    var i
    row = ""
    for (i=0;i<netlist.length;i++)
    {
        if (netlist.substring(i,i+1) != ';') row = row + netlist.substring(i,i+1)
        else return row
    }
    return row
}

```

```

function check_row(pstr)
{
    var i

    if (pstr.substring(0,3) == 'out' || pstr.substring(0,3) == 'inp' || pstr.substring(0,3) == 'end')
return true

```

//parantez kontrol

sol = 0

```
sag = 0
```

```
sonuc = false
```

```
for (i=0;i<pstr.length;i++)
```

```
{
```

```
    if (pstr.substring(i,i+1) == '(') {sonuc = true; sol++;}
```

```
    else if (pstr.substring(i,i+1) == ')' && sonuc) {sonuc = true; sag++;}
```

```
}
```

```
if (sol !=1 || sag !=1) sonuc = false
```

```
return sonuc
```

```
}
```

```
function built_default_element(pstr)
```

```
{
```

```
    var modul_name
```

```
    modul_name = get_word(trim(pstr), '(')
```

```
    src_xml.value = '<module_name>' + modul_name + '</module_name>\n'
```

```
    src_xml.value = src_xml.value+ "<elements>\n"
```

```
    pstr = pstr.substring(modul_name.length+1, pstr.length-1)
```

```
while (pstr.length>0)
```

```
{
```

```
    element = get_word(pstr,',')
```

```
    pstr = pstr.substring(element.length+1,pstr.length)
```

```
    src_xml.value = src_xml.value + " <element>" + element + "</element>\n"
```

```
}
```

```
src_xml.value = src_xml.value + "</elements>\n"
```

```
}
```

```
function built_output_element(pstr)
```

```
{
```

```
src_xml.value = src_xml.value+ "<def_outputs>\n"
```

```
pstr = trim(pstr)
```

```
while (pstr.length>0)
```

```
{
```

```
    element = get_word(pstr,',')
```

```
    pstr = pstr.substring(element.length+1,pstr.length)
```

```
    src_xml.value = src_xml.value + " <output>" + element + "</output>\n"
```

```
}
```

```
src_xml.value = src_xml.value + "</def_outputs>\n"
```

```
}
```

```
function built_input_element(pstr)
```

```
{
```

```
    src_xml.value = src_xml.value+ "<def_inputs>\n"
```

```
    pstr = trim(pstr)
```

```
while (pstr.length>0)
```

```
{
```

```
    element = get_word(pstr,',')
```

```
    pstr = pstr.substring(element.length+1,pstr.length)
```

```
    src_xml.value = src_xml.value + " <input>" + element + "</input>\n"
```

```
    arr_inputs[isayac] = element
```

```
    isayac++
```

```
}
```

```
src_xml.value = src_xml.value + "</def_inputs>\n"
```

```
}
```

```
function built_gates(pstr,gate_type)
```

```
{
```

```
    var gate_name
```

```
gate_name = get_word(trim(pstr), '(')
```

```
pstr = pstr.substring(gate_name.length+1, pstr.length-1)
```

```
element = get_word(pstr, ',')
```

```
pstr = pstr.substring(element.length+1, pstr.length)
```

```
src_xml.value = src_xml.value + '<output name="' + element + '" gate_name="' +  
trim(gate_name) + '" gate_type="' + gate_type + '">\n'
```

```
while (pstr.length>0)
```

```
{
```

```
    element = get_word(pstr, ',')
```

```
    pstr = pstr.substring(element.length+1, pstr.length)
```

```
    var1 = check_inputs(element)
```

```
    if (var1) src_xml.value = src_xml.value + " <input>" + element + "</input>\n"
```

```
    else src_xml.value = src_xml.value + " <output>" + element + "</output>\n"
```

```
}
```

```
src_xml.value = src_xml.value + "</output>\n"
```

```
}
```

```
function check_inputs(elem)
```

```
{
```

```
    var i
```

```
    for (i=0; i<arr_inputs.length; i++)
```

```
        if (arr_inputs[i] == elem) return true
```

```
    return false
```

```
}
```

```
function finish_xml()
```

```
{
```

```
    src_xml.value = '<tez>\n' + src_xml.value + "</tez>\n"
```

```
}
```

```

function get_word(pstr, ayrac)
{
    var i
    word = ""
    for (i=0;i<pstr.length;i++)
    {
        if (pstr.substring(i,i+1) != ayrac) word = word + pstr.substring(i,i+1)
        else return word
    }
    return word
}

```

```

function trim(pstr)
{
    var i
    son = ""

    for (i=0;i<pstr.length;i++)
    {
        if (pstr.substring(i,i+1) != ' ') {son = pstr.substring(i,pstr.length);break}
    }

    for (i=pstr.length-1;i>0;i--)
    {
        if (pstr.substring(i,i+1) != ' ') {son = pstr.substring(0,i+1);break}
    }
    return son
}

```

```

</script>

```

```

</BODY>

```

```

</HTML>

```

EK-2 Uygulama kaynak Kodu – 2 (tez-cizim.htm)

```

<html xmlns:v="urn:schemas-microsoft-com:vml"
xmlns:o="urn:schemas-microsoft-com:office:office"
xmlns="http://www.w3.org/TR/REC-html40">

<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<STYLE>
v\:* {behavior:url(#default#VML);}
o\:* {behavior:url(#default#VML);}
.shape {behavior:url(#default#VML);}

a {font-family: Tahoma;}
h3 {margin-bottom: 0pt;color: Pink}
h4 {margin-top: 0pt; margin-bottom: 0pt;}
hr {margin-top: 0pt; margin-bottom: 0pt;}
.a {font-family: Tahoma; font-size:10pt; font-weight:bold}
.b {font-family: Tahoma; font-size:10pt}

</STYLE>
<TITLE></TITLE>
</HEAD>
<BODY>

<h3>Ahmet SAVAŞ - XML Tabanlı VML ile VLSI Devre Tasarımı (Gate Level) - ©2002</h3>
<hr>

<xml id=xml_tez src="tez_output.xml"></xml>
<div id=xml_alani style='background-color:#c0c0c0; color:White;'></div>

<v:shapetype id="Input" coordorigin="0,0" coordsize="100, 100"

```

```

    path="at 30,30, 50,50, 20,10,20,10">
    <v:stroke joinstyle="miter"/>
    <v:path gradientshapeok="t" textboxrect="-100 -50 500 200"/>
</v:shapetype>

<v:shapetype id="not_gate" coordorigin="0,0" coordsize="100, 100"
    path="m0,0 l 0,0,50,50 l 50,50,0,100 0xe">
    <v:stroke joinstyle="miter"/>
    <v:path gradientshapeok="t" textboxrect="-100 -50 500 200"/>
</v:shapetype>

<v:shapetype id="buf_gate" coordorigin="0,0" coordsize="100, 100"
    path="m0,0 l 0,0,50,50 l 50,50,0,100 0xe">
    <v:stroke joinstyle="miter"/>
    <v:path gradientshapeok="t" textboxrect="-100 -50 500 200"/>
</v:shapetype>

<v:shapetype id="and_gate" coordorigin="0, 0" coordsize="100, 100"
    path="m0,0 l 0,100,50,100 at 0,0,100,100,50,100,90,-90, 0xe">
    <v:stroke joinstyle="miter"/>
    <v:path gradientshapeok="t" textboxrect="-10 -10 500 100"/>
</v:shapetype>

<v:shapetype id="nand_gate" coordorigin="0, 0" coordsize="100, 100"
    path="m0,0 l 0,100,50,100 at 0,0,100,100,50,100,90,-90, 0xe">
    <v:stroke joinstyle="miter"/>
    <v:path gradientshapeok="t" textboxrect="-10 -10 500 100"/>
</v:shapetype>

<v:shapetype id="or_gate" coordorigin="0, 0" coordsize="100, 100"
    path="m 0,0 wr 0,0,25,100,12,0,12,100 wr -60,0,100,100,12,0,12,100">

```

```

<v:stroke joinstyle="miter"/>
<v:path gradientshapeok="t" textboxrect="-10 -10 500 100"/>
</v:shapetype>

<v:shapetype id="nor_gate" coordorigin="0, 0" coordsize="100, 100"
  path="m 0,0 wr 0,0,25,100,12,0,12,100 wr -60,0,100,100,12,0,12,100">
  <v:stroke joinstyle="miter"/>
  <v:path gradientshapeok="t" textboxrect="-10 -10 500 100"/>
</v:shapetype>

```

```

<script language="JavaScript">

```

```

xml_doc = xml_tez.XMLDocument
root = xml_doc.documentElement
create_inputs()
gate_value = new Array()
gate_value_name = new Array()

```

```

//renklendirme Color 1 Blue

```

```

function general(def_y)
{
  def_y = def_y
  def_x = 80
  cizilen_link = new Array()
  cizilen_link_input = new Array()
  cizilen = new Array()
  cizilen_type = new Array()
  cizilen_left = new Array()
  cizilen_top = new Array()

```

```

draw_inputs(def_y)
nodes = xml_doc.getElementsByTagName('def_outputs')
nodes = nodes(0)
for (x=0;x<nodes.childNodes.length;x++)
{
    cc = nodes.childNodes(x).text
    arr_inputs = new Array()
    isayac = 0
    c = new Array()
    linkpoints = new Array()
    sonucstr="

    cikis = cc
    strsonuc = "

    draw_output1(cikis, def_x, def_y) // + x*50)
    def_y = def_y + 70
    def_x = def_x + 160
}
draw_gates()

/*****Aradaki linkleri çizmek için*****/
nodes = xml_doc.getElementsByTagName('def_outputs')
nodes = nodes(0)
for (x=0;x<nodes.childNodes.length;x++)
{
    cikis = nodes.childNodes(x).text
    draw_link(cikis)
    draw_link_output(cikis)
}
}

```

```

function mod(divisee,base)
{
    return Math.round(divisee - (Math.floor(divisee/base)*base));
}

function create_inputs()
{
    document.write('<table border=0>')
    document.write('<tr class=a>')
    document.write('<td>Inputs</td>')
    document.write('<td>Outputs</td>')
    document.write('</tr>')
    document.write('<tr>')
    document.write('<td valign=top>')

    document.write('<table border=1>')
    nodes = xml_doc.getElementsByTagName('def_inputs')
    nodes = nodes(0)
    z = -1 //ilk kapının başlangıcı y koordinatı için
    for (i=0;i<nodes.childNodes.length;i++)
    {
        x = mod(i,6)
        if (x == 0)
        {
            document.write('<tr>')
            z++
        }
        cc = nodes.childNodes(i).text
        document.write('<td class=b><font color=Blue><b>' + cc +
            '</b></font><select id=n' + cc + ' name=n' + cc +
            '><option>0</option><option>1</option></select></td>')
    }
}

```

```

        .if (x == 0) document.write('</tr>')
    }
    document.write('</table>')
    document.write('</td>')

    document.write('<td valign=top>')
    document.write('<table border=1>')
    nodes = xml_doc.getElementsByTagName('def_outputs')
    nodes = nodes(0)

    if (z < nodes.childNodes.length) z = nodes.childNodes.length - 1

    for (i=0;i<nodes.childNodes.length;i++)
    {
        document.write('<tr>')
        cc = nodes.childNodes(i).text
        document.write('<td class=b><font color=red><b>' + cc +
            ' => </b></font><input type=text size=3 class=b id=n' + cc +
            ' name=n' + cc + '></td>')

        document.write('</tr>')
    }

    document.write('</table>')
    document.write('</td>')
    document.write('</table>')

    z = 200 + (z - 1) * 30

    document.write('<input type=button value="Draw Circuit" onclick="general(' + z + ');
this.disabled=true; document.all(\b_animate\').disabled=false">')

    document.write('<input type=button name="b_animate" value="Animate(Zero Delay)"
onclick="evulate_outputs()" disabled=true>')

```

```

document.write('<input type=button value="Create XML"
onclick="javascript:window.location.href=\'tez.htm\'">')

document.write('<input type=button value="Show XML"
onclick="javascript:window.location.href=\'tez_output.xml\'">')
}

```

```

function evulate_outputs()
{
    strsonuc = "
    nodes = xml_doc.getElementsByTagName('def_outputs')
    nodes = nodes(0)
    for (x=0;x<nodes.childNodes.length;x++)
    {
        cc = nodes.childNodes(x).text
        sonuc = evulate_output(cc)
        document.all('n' + cc).value = sonuc

        if (sonuc == '1') color = 'blue'
        else scolor = 'red'

        document.all('l_' + cc).strokecolor=scolor
        document.all('l_' + cc).strokeweight=2

        gate_value[gate_value.length] = sonuc
        gate_value_name[gate_value.length-1] = cc
    }
}

```

```

function draw_inputs(y)
{
    nodes = xml_doc.getElementsByTagName('def_inputs')
    nodes = nodes(0)

```

```

for (i=0;i<nodes.childNodes.length;i++)
{
    cc = nodes.childNodes(i).text
    //alert(cc)

    xml_alani.innerHTML = xml_alani.innerHTML + '<v:shape id="' + cc + '" type="input" ' +
        'style=\'position: absolute;left: 0; top: 190; width: 30; height: 30\' fillcolor="#f46">' +
        '<v:fill type="gradient" color2="fill lighten(0)" ' +
        ' method="linear sigma" angle="-270" focus="100%">' +
        '<v:textbox>' +
        '<font color=Blue><b>' + cc + '</b></font>' +
        '</v:textbox>' +
        '</v:shape>'

    document.all(cc).style.top = y
    m = cizilen.length
    cizilen[m] = cc
    cizilen_type[m] = 'input'
    cizilen_left[m] = 0
    cizilen_top[m] = y
    y = y + 30
}
}

```

```

function trim(pstr)
{
    var i
    son = "

    for (i=0;i<pstr.length;i++)
    {
        if (pstr.substring(i,i+1) != ' ') {son = pstr.substring(i,pstr.length);break}
    }
}

```

```
for (i=pstr.length-1;i>0;i--)
{
    if (pstr.substring(i,i+1) != ' ') {son = pstr.substring(0,i+1);break}
}
return son
}
```

```
function ciz(gname, gate_type, left, top)
```

```
{
    var k, m, j
    if (document.all(gname) == null)
    {
        m = cizilen.length
        cizilen[m] = gname
        cizilen_type[m] = gate_type
        cizilen_left[m] = left
        cizilen_top[m] = top
    }
```

```
for (k=0;k<m+1;k++)
```

```
{
    if (cizilen_left[k] == left && cizilen_top[k] == top)
    {
        for (j=0;j<m+1;j++)
            if (cizilen_left[j] >= left && cizilen[j] != gname && cizilen_top[j] == top) cizilen_left[j] =
cizilen_left[j] + 80
        break
    }
}
}
```

```

function draw_gates()
{
    var k
    for (k=0;k<cizilen.length;k++)
    {
        if (cizilen_type[k] == 'input') continue //input'ları tekrar çizmeyecek

        xml_alani.innerHTML = xml_alani.innerHTML + '<v:shape id="' + cizilen[k] + '" type="#'+
        cizilen_type[k] + '_gate' style="position: absolute; left:250; top:250; width:30; height:30"
        fillcolor="#ff6">' +

            '<v:fill type="gradient" color2="fill lighten(0)" +
            ' method="linear sigma" angle="-135" focus="100%">' +

            '<v:textbox><font color=red>' + cizilen[k] + '</font></v:textbox>' + '</v:shape>'

        if (cizilen_type[k] == 'buf') document.all(cizilen[k]).style.left = cizilen_left[k] + 5
        else document.all(cizilen[k]).style.left = cizilen_left[k]
        document.all(cizilen[k]).style.top = cizilen_top[k]
    }
}

function ciz_link(x1,y1,x2,y2,fark,lname)
{
    xlines = create_path(x1,y1,x2,y2,fark,lname)
    xml_alani.innerHTML = xml_alani.innerHTML + xlines
}

function draw_input(input_name,gate_name,fark)
{
    var tmp
    x1 = get_left(gate_name)
    y1 = get_top(gate_name)
    x1 = to_number(x1)
    y1 = to_number(y1-45)
    tmp = node_bul(gate_name)

```

```
tmp = tmp.getAttribute("gate_type")
if (tmp == "and" || tmp == "nand")
{
    x1 = x1 - 10
}
else if (tmp == "not")
{
    x1 = x1 - 10
}

x2 = get_left(input_name)
y2 = get_top(input_name)
x2 = to_number(x2)
y2 = to_number(y2-40)

if (x1 > x2) xlines = create_path(x2,y2,x1,y1,fark)
else xlines = create_path(x1,y1,x2,y2,fark)
xml_alani.innerHTML = xml_alani.innerHTML + xlines
}
```

```
function get_left(node_name)
{
    var left
    for (k=0;k<cizilen.length;k++)
        if (cizilen[k] == node_name) return cizilen_left[k]
}
```

```
function get_top(node_name)
{
    var top, k
```

```

for (k=0;k<cizilen.length;k++)
  if (cizilen[k] == node_name) return cizilen_top[k]
}

```

```

function kaydir(x)
{
  var left, nleft
  for (i=0;i<cizilen.length;i++)
  {
    cizilen_left[i] = cizilen_left[i] + x
  }
}

```

```

function kaydir_top(x)
{
  var left, nleft
  for (i=0;i<cizilen.length;i++)
  {
    cizilen_top[i] = cizilen_top[i] + x
  }
}

```

```

function to_number(x)
{
  sonuc = 0
  eval('sonuc = ' + x)
  return sonuc;
}

```

```

function draw_output1(node_str, left, top)

```

```

{
    var cc, i, node_local, pleft, ptop
    pleft = left
    ptop = top
    if (cizildimi(node_str)) return //Eger daha önce çizilmiŝe bir daha çizmeyecek
    node_local = node_bul(node_str)
    gate_type=node_local.getAttribute("gate_type")
    ciz(node_str, gate_type, left, top)
    node_list = node_local.getElementsByTagName("output")
    for (i=0;i<node_local.childNodes.length;i++)
    {
        cc = node_local.childNodes(i)
        if (cc.nodeName == 'output')
        {
            c[c.length] = cc.text
            draw_output1(cc.text, pleft, ptop)
            ptop = ptop + 70
        }
    }
}

```

```

function evulate_output(node_str)
{
    var cc, i, node_local, sonuc
    node_local = node_bul(node_str)
    gate_type=node_local.getAttribute("gate_type")

    cc = node_local.childNodes(0)
    if (cc.nodeName == 'output')
    {
        sonuc = evulate_output(cc.text)
    }
}

```

```

}

else sonuc = get_gate_value(cc.text)

if (sonuc == 0) document.all(node_str + '->' + cc.text).strokecolor = 'red'
else document.all(node_str + '->' + cc.text).strokecolor = 'blue'

if (node_local.childNodes.length == 1)
{
    if (gate_type == 'not')
    {
        if (sonuc == 0) sonuc = 1
        else sonuc = 0
    }
    else sonuc = sonuc
}

for (i=1;i<node_local.childNodes.length;i++)
{
    cc = node_local.childNodes(i)
    if (cc.nodeName == 'output')
    {
        s = evaluate_output(cc.text)

        sonuc = output_gate(sonuc, s, gate_type)

        if (s == '0') document.all(node_str + '->' + cc.text).strokecolor = 'red'
        else document.all(node_str + '->' + cc.text).strokecolor = 'blue'
    }
    else
    {
        s = get_gate_value(cc.text)

        sonuc = output_gate(sonuc, s, gate_type)

        if (s == '0') document.all(node_str + '->' + cc.text).strokecolor = 'red'
        else document.all(node_str + '->' + cc.text).strokecolor = 'blue'
    }
}

```

```

    }
}

z = gate_value.length
gate_value[z] = sonuc
gate_value_name[z] = node_str
return sonuc
}

function get_gate_value(node_str)
{
    return document.all('n' + node_str).options.selectedIndex;
}

```

```

function output_gate(arg1, arg2, opr)
{
    var sonuc1
    if (opr == 'and') sonuc1 = (arg1 && arg2)
    else if (opr == 'nand') sonuc1 = !(arg1 && arg2)
    else if (opr == 'or') sonuc1 = arg1 || arg2
    else if (opr == 'nor') sonuc1 = !(arg1 || arg2)
    else if (opr == 'not') sonuc1 = !arg1
    else if (opr == 'buf') sonuc1 = arg1

    if (sonuc1) sonuc1 = 1
    else sonuc1 = 0
    return sonuc1
}

```

```

function draw_input_links(node_str, left, top, fark)
{
    var cc, i, node_local, pleft, ptop, tmp

```

```

pleft = left
ptop = top
if (cizildimi_link_input(node_str)) return
node_local = node_bul(node_str)
cizilen_link_input[cizilen_link_input.length] = node_str
for (j=0;j<node_local.childNodes.length;j++)
{
    fark = fark + 1
    if (node_local.childNodes(j).nodeName == 'input')
        draw_input(node_local.childNodes(j).text,node_str,fark)
}

node_list = node_local.getElementsByTagName("output")
for (i=0;i<node_local.childNodes.length;i++)
{
    cc = node_local.childNodes(i)
    if (cc.nodeName == 'output')
    {
        c[c.length] = cc.text
        fark = fark + 1
        draw_input_links(cc.text, pleft, ptop, 25)
        ptop = ptop + 40
    }
}
}

```

```

function draw_link(node_str)
{
    var cc, i, node_local, x1, y1, tmp, type, fark
    x1 = get_left(node_str)

```

```

y1 = get_top(node_str)
if (cizildimi_link(node_str)) return
cizilen_link[cizilen_link.length] = node_str
node_local = node_bul(node_str)
type = node_local.getAttribute("gate_type")
if (type == "and" || type == "nand")
{
  x1 = x1-6
  y1 = y1
}
else if (type == "or" || type == "nor")
{
  x1 = x1
  y1 = y1
}
else if (type == "not" || type == "buf")
{
  x1 = x1
  y1 = y1
}

fark = 23
for (i=0;i<node_local.childNodes.length;i++)
{
  cc = node_local.childNodes(i)
  l_name = node_str + '->' + cc.text
  if (cc.nodeName == 'output')
  {
    y1 = to_number(to_number(y1-4) + 2)
    fark = fark - 3
    x2 = get_left(cc.text)

```

```

y2 = get_top(cc.text)
tmp = node_bul(cc.text)
type = tmp.getAttribute("gate_type")
if (type == "or" || type == "nor") ciz_link(x1-5,y1-35,to_number(x2)+20,y2-35,fark,l_name)
else if (type == "and" || type == "nand") ciz_link(x1-5,y1-35,to_number(x2)+20,y2-
35,fark,l_name)
else if (type == "not" || type == "buf") ciz_link(x1-5,y1-35,to_number(x2)+6,y2-
35,fark,l_name)
//alert(node_str)
draw_link(cc.text)
}
else
{
y1 = to_number(to_number(y1-4) + 2)
fark = fark - 3
x2 = get_left(cc.text)
y2 = get_top(cc.text)
tmp = node_bul(cc.text)
if (type == "or" || type == "nor") ciz_link(x1-5,y1-35,to_number(x2),y2-35,fark,l_name)
else if (type == "and" || type == "nand") ciz_link(x1-5,y1-35,to_number(x2),y2-
35,fark,l_name)
else if (type == "not") ciz_link(x1-5,y1-35,to_number(x2)+6,y2-35,fark,l_name)
}
}
}

function draw_link_output(node_str)
{
x1 = get_left(node_str)
y1 = get_top(node_str)
y1 = y1 - 38
x1 = to_number(x1) + 20

```

```
x2 = x1 + 50
```

```
xml_alani.innerHTML = xml_alani.innerHTML + '<v:line id=l_' + node_str + ' strokcolor="red"
from=' + x1 + ',' + y1 + ' to=' + x2 + ',' + y1 + '"></v:line>'
```

```
}
```

```
function get_color(node_str)
```

```
{
```

```
var k
```

```
for (k=0;k<gate_value.length;k++)
```

```
{
```

```
if (gate_value_name[k] == node_str)
```

```
{
```

```
if (gate_value[k] == '1') return 'red'
```

```
else return 'blue'
```

```
}
```

```
}
```

```
/*Eğer dinamik input demektir.
```

```
if (get_gate_value(node_str) == '1') return 'red'
```

```
else return 'blue'
```

```
}
```

```
function node_bul(node_name)
```

```
{
```

```
for (f=0;f<xml_doc.documentElement.childNodes.length;f++)
```

```
{
```

```
ss = xml_doc.documentElement.childNodes(f).getAttribute("name")
```

```
if (ss == node_name) return(xml_doc.documentElement.childNodes(f))
```

```
}
```

```
}
```

```
function cizildimi(node_name)
```

```
{
```

```
for (k=0;k<cizilen.length;k++)  
    if (cizilen[k] == node_name) return true;  
return false;  
}
```

```
function cizildimi_link(node_name)  
{  
    for (k=0;k<cizilen_link.length;k++)  
        if (cizilen_link[k] == node_name) return true;  
    return false;  
}
```

```
function cizildimi_link_input(node_name)  
{  
    for (k=0;k<cizilen_link_input.length;k++)  
        if (cizilen_link_input[k] == node_name) return true;  
    return false;  
}
```

```
function vami(node_name)  
{  
    document.write('<font color=Blue><br>vami() c:' + c.toString())  
    document.write('<br>vami() node_name:' + node_name)  
    document.write('<br>vami() c length:' + c.length + '</font>')  
  
    for (k=0;k<c.length;k++)  
        if (c[k] == node_name) return true;  
    return false;  
}
```

```
//*****PATH OLUŞTURMA*****
```

```
function create_path(x1,y1,x2,y2,fark,lname)
```

```
{
```

```
var xx1,yy1,xx2,yy2, snc
```

```
xx1 = x1
```

```
xx2 = x2
```

```
yy1 = y1
```

```
yy2 = y2
```

```
snc = ""
```

```
if (yy1 == yy2) snc = create_direct_path_x(xx1, xx2, yy1, fark, lname)
```

```
if (yy1 < yy2) snc = create_path_down(xx1, xx2, yy1, yy2, fark, lname)
```

```
if (yy1 > yy2) snc = create_path_up(xx1, xx2, yy1, yy2, fark, lname)
```

```
return snc
```

```
return snc
```

```
}
```

```
function create_direct_path_x(x1,x2,y1,fark, lname)
```

```
{
```

```
var xx1,xx2,yy1,snc
```

```
xx1 = x1
```

```
xx2 = x2
```

```
yy1 = y1
```

```
if (x1 > x2)
```

```
{
```

```
xx1 = x2
```

```
xx2 = x1
```

```
}
```

```
snc = '<v:polyline id="' + lname + '" points="' + xx1 + ',' + yy1 + ' ' + (xx1+fark) + ',' + yy1 + ' '
```

```
snc = snc + ' ' + (xx1+fark) + ',' + yy1 + ' ' + (xx1+fark) + ',' + (yy1-fark) + ' '
```

```
snc = snc + ' ' + (xx1+fark) + ',' + (yy1-fark) + ' ' + (xx2-fark+5) + ',' + (yy1-fark) + ' '
```

```
snc = snc + ' ' + (xx2-fark+5) + ',' + (yy1-fark) + ' ' + (xx2-fark+5) + ',' + (yy1) + ' '
```

```
snc = snc + '' + (xx2-fark+5) + ',' + (yy1) + '' + (xx2) + ',' + (yy1) + '></v:polyline>'
```

```
return snc
```

```
}
```

```
function create_path_down(x1,x2,y1,y2,fark,lname)
```

```
{
```

```
    //alert('down')
```

```
    var xx1,xx2,yy1,yy2,snc
```

```
    xx1 = x1
```

```
    xx2 = x2
```

```
    yy1 = y1
```

```
    yy2 = y2
```

```
    if (xx1 < xx2)
```

```
    {
```

```
        snc = '<v:polyline id="' + lname + '" filled="false" points="' + xx1 + ',' + yy1 + '' + (xx1-fark) + ',' + yy1 + ''
```

```
        snc = snc + (xx1-fark) + ',' + (yy1+fark) + ''
```

```
        snc = snc + (xx2+fark+5) + ',' + (yy1+fark) + ''
```

```
        snc = snc + (xx2+fark+5) + ',' + (yy2) + ''
```

```
        snc = snc + (xx2) + ',' + (yy2) + '"></v:polyline>'
```

```
    }
```

```
    else
```

```
    {
```

```
        snc = '<v:polyline id="' + lname + '" filled="false" points="' + xx1 + ',' + yy1 + '' + (xx1-fark) + ',' + yy1 + ''
```

```
        snc = snc + (xx1-fark) + ',' + (yy1+fark) + ''
```

```
        snc = snc + (xx2+fark+5) + ',' + (yy1+fark) + ''
```

```
        snc = snc + (xx2+fark+5) + ',' + (yy2) + ''
```

```
        snc = snc + (xx2) + ',' + (yy2) + '"></v:polyline>'
```

```
    }
```

```

return snc
}

function create_path_up(x1,x2,y1,y2,fark,lname)
{
var xx1,xx2,yy1,yy2,snc

xx1 = x1
xx2 = x2
yy1 = y1
yy2 = y2

if (xx1 < xx2)
{
snc = '<v:polyline id="' + lname + '" filled="false" points="' + xx1 + ',' + yy1 + ' ' + (xx1-fark) +
',' + yy1 + ' '

snc = snc + (xx1-fark) + ',' + (yy1-fark) + ' '
snc = snc + (xx2+fark+5) + ',' + (yy1-fark) + ' '
snc = snc + (xx2+fark+5) + ',' + (yy2) + ' '
snc = snc + (xx2) + ',' + (yy2) + '"></v:polyline>'
}
else
{
snc = '<v:polyline id="' + lname + '" filled="false" points="' + xx1 + ',' + yy1 + ' ' + (xx1-fark) +
',' + yy1 + ' '

snc = snc + (xx1-fark) + ',' + (yy1-fark) + ' '
snc = snc + (xx2+fark+5) + ',' + (yy1-fark) + ' '
snc = snc + (xx2+fark+5) + ',' + (yy2) + ' '
snc = snc + (xx2) + ',' + (yy2) + '"></v:polyline>'
}

return snc
}

```

//***DEVRE KAŞILAŞTIRMASI *******

```
function compare(xml_doc1, xml_doc2)
{
    var xml_compare_doc
    dizi1 = new Array()
    dizi2 = new Array()

    xml_compare.src = xml_doc1
    xml_compare_doc = xml_compare.XMLDocument
    nodes = xml_compare_doc.getElementsByTagName('def_outputs')
    nodes = nodes(0)
    for (x=0;x<nodes.childNodes.length;x++)
    {
        cikis = nodes.childNodes(x).text
        dizi1[dizi1.length] = create_dizi(cikis)
    }

    xml_compare.src = xml_doc2
    xml_compare_doc = xml_compare.XMLDocument
    nodes = xml_compare_doc.getElementsByTagName('def_outputs')
    nodes = nodes(0)
    for (x=0;x<nodes.childNodes.length;x++)
    {
        cikis = nodes.childNodes(x).text
        dizi2[dizi2.length] = create_dizi(cikis)
    }

    for (i=0;i<dizi1.length;i++)
    {
        x = false
```

```

        for (j=0;j<dizi2.length;j++)
        {
            if (dizi1[i] == dizi2[j])
            {
                x = true
                break
            }
        }
        //Eğer hiç eşleşmezse ...
        if (!x) break
    }

    if (!x)
        alert('Eşleşmiyor...')
    else
        alert('Eşleşiyor...')
}

function create_dizi(node_str)
{
    var cc, i, node_local, sonuc
    node_local = node_bul_compare(node_str)
    gate_type=node_local.getAttribute("gate_type")

    sonuc = gate_type + '('
    cc = node_local.childNodes(0)
    if (cc.nodeName == 'output')
    {
        sonuc = sonuc + create_dizi(cc.text)
    }
    else sonuc = sonuc + ']'
}

```

```
for (i=1;i<node_local.childNodes.length;i++)
```

```
{
```

```
    cc = node_local.childNodes(i)
```

```
    if (cc.nodeName == 'output')
```

```
    {
```

```
        s = ',' + create_dizi(cc.text)
```

```
        sonuc = sonuc + s
```

```
    }
```

```
    else
```

```
    {
```

```
        s = ','
```

```
        sonuc = sonuc + s
```

```
    }
```

```
}
```

```
sonuc = sonuc + ')
```

```
return sonuc
```

```
}
```

```
function node_bul_compare(node_name)
```

```
{
```

```
    for (f=0;f<xml_compare.documentElement.childNodes.length;f++)
```

```
    {
```

```
        ss = xml_compare.documentElement.childNodes(f).getAttribute("name")
```

```
        if (ss == node_name) return(xml_compare.documentElement.childNodes(f))
```

```
    }
```

```
}
```

```
</script>
```

```
</BODY>
```

```
</HTML>
```

Ek-3 Örnek Netlist

```
module test(G22gat,G23gat,G1gat,G2gat,G3gat,G6gat,G7gat,);  
output G22gat,G23gat,;  
input G1gat,G2gat,G3gat,G6gat,G7gat,;  
nand G10gat(G10gat,G1gat,G3gat);  
nand G11gat(G11gat,G3gat,G6gat);  
nand G16gat(G16gat,G2gat,G11gat);  
nand G19gat(G19gat,G11gat,G7gat);  
nand G22gat(G22gat,G10gat,G16gat);  
nand G23gat(G23gat,G16gat,G19gat);  
endmodule
```

Ek-4 Örnek XML Dökümanı

```
<tez>
<module_name>test</module_name>
<elements>
  <element>G22gat</element>
  <element>G23gat</element>
  <element>G1gat</element>
  <element>G2gat</element>
  <element>G3gat</element>
  <element>G6gat</element>
  <element>G7gat</element>
</elements>
<def_outputs>
  <output>G22gat</output>
  <output>G23gat</output>
</def_outputs>
<def_inputs>
  <input>G1gat</input>
  <input>G2gat</input>
  <input>G3gat</input>
  <input>G6gat</input>
  <input>G7gat</input>
</def_inputs>
<output name="G10gat" gate_name="G10gat" gate_type="nand">
  <input>G1gat</input>
  <input>G3gat</input>
```

</output>

<output name="G11gat" gate_name="G11gat" gate_type="nand">

<input>G3gat</input>

<input>G6gat</input>

</output>

<output name="G16gat" gate_name="G16gat" gate_type="nand">

<input>G2gat</input>

<output>G11gat</output>

</output>

<output name="G19gat" gate_name="G19gat" gate_type="nand">

<output>G11gat</output>

<input>G7gat</input>

</output>

<output name="G22gat" gate_name="G22gat" gate_type="nand">

<output>G10gat</output>

<output>G16gat</output>

</output>

<output name="G23gat" gate_name="G23gat" gate_type="nand">

<output>G16gat</output>

<output>G19gat</output>

</output>

</tez>

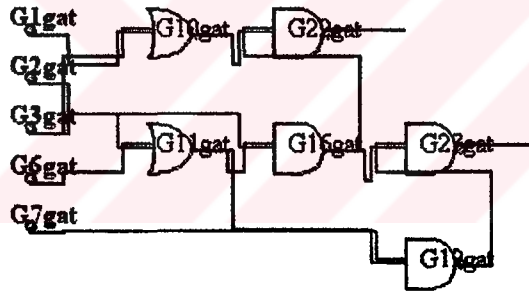
EK – 5 Örnek Çıktılar

Ahmet SAYGAS - XML Tabanlı VHDL ile VLSI Devre Tasarımı (Gate Level) - ©2002

Inputs					Outputs	
G1gat=	G2gat=	G3gat=	G6gat=	G7gat=	G22gat =>	
G1gat=	G2gat=	G3gat=	G6gat=	G7gat=	G23gat =>	
Draw Circuit		Animate(Zero Delay)		Create XML	Show XML	

Ahmet SAYGAS - XML Tabanlı VHDL ile VLSI Devre Tasarımı (Gate Level) - ©2002

Inputs					Outputs	
G1gat=	G2gat=	G3gat=	G6gat=	G7gat=	G22gat =>	
G1gat=	G2gat=	G3gat=	G6gat=	G7gat=	G23gat =>	
Draw Circuit		Animate(Zero Delay)		Create XML	Show XML	



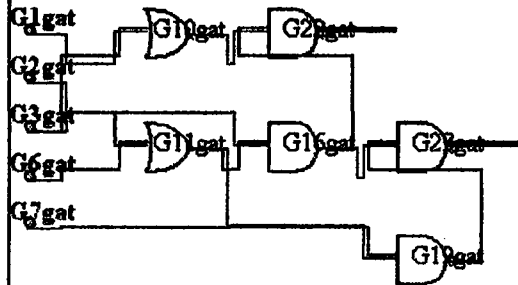
Inputs

G1gat=1 G2gat=0 G3gat=0 G6gat=0 G7gat=1

Outputs

G22gat => 0
G23gat => 1

Draw Circuit Animals(Zero Delay) Create XML Show XML



EK-6 XML Uygulama Alanları

Örnek olması açısından burada birkaç konu başlığı verilmiştir. Ayrıntılı bilgi için http://www.xml.org/xml/industry_industrysectors.jsp adresine başvurunuz.

XML and Electronic Design Automation (EDA)

"<http://www.oasis-open.org/cover/xmlAndEDA.html>"

Robotic Markup Language (RoboML)

"<http://www.oasis-open.org/cover/roboML.html>"

Description Logics Markup Language (DLML)

"<http://www.oasis-open.org/cover/dlml.html>"

Manuscript Access through Standards for Electronic Records (MASTER)

"<http://www.oasis-open.org/cover/master.html>"

Electronic Metadata for Endangered Languages Data (EMELD)

"<http://www.oasis-open.org/cover/emeld.html>"

Metadata Encoding and Transmission Standard (METS)

"<http://www.oasis-open.org/cover/mets.html>"

Web Modeling Language (WebML)

"<http://www.oasis-open.org/cover/webML.html>"

Platform for Internet Content Selection (PICS)

"<http://www.oasis-open.org/cover/pics.html>"

Extensible Graph Markup and Modeling Language (XGMML)

"<http://www.oasis-open.org/cover/xgmml.html>"



CellML

"<http://www.oasis-open.org/cover/cellML.html>"

Bank Internet Payment System (BIPS)

"<http://www.oasis-open.org/cover/#bips>"

