ddApp-10

Sayısal Tasarım ve FPGA Uygulamaları

Buton Uygulaması

Durum	Tamamlandı ✅
Güncelleme	9 Kasım 2023
Hazırlayan	Emre İŞSEVER
E-Posta	emre.issever@fpgaturkey.com info@beti.com.tr

Amaç

Basys3 FPGA kartının üzerinde bulunan butonları kullanarak harici donanım sinyallerini doğru ve etkin bir şekilde işlemek ve bu işleme sonucunda elde edilen verileri önceden belirlenen kriterlere uygun çıkış sinyallerinde dönüştürmektir.

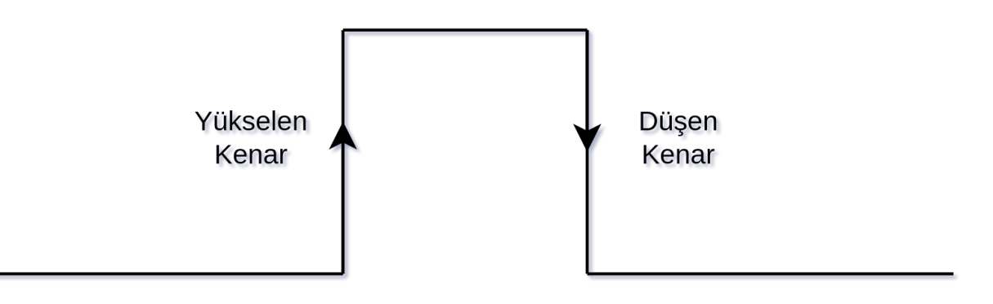
Kazanımlar

• Sinyalin Kenar Tespiti
Sinyallerde meydana gelen değişiklikleri doğru bir şekilde algılamanın önemi vurgulanarak, yükselen ve düşen kenar tespit metodolojileri detaylandırılır. VHDL içerisinde bu tespitlerin nasıl gerçekleştirildiği, pratik örneklerle öğrencilere aktarılır.
• Harici Sinyallerin Senkronizasyonun Önemi:
Bu bölümde, sinyal işlemenin doğru ve güvenilir şekilde gerçekleştirilmesi için sinyallerin clk sinyali ile neden senkronize edilmesi gerektiği üzerinde durulur. clk sinyalinin, sistemdeki bileşenler arasındaki uyumu sağlama ve sinyal gecikmelerini standartlaştırma rolleri örneklerle açıklanır.
• Debouncing Önemi ve Uygulaması
Mekanik butonlardan kaynaklanan istenmeyen gürültülerin (bounce) FPGA sistemlerinde nasıl sorun yaratabileceği ve bu gürültülerin etkin bir şekilde nasıl filtre edileceği ele alınır. Debouncing algoritmasının temelleri ve bu algoritmanın VHDL kodu içinde nasıl uygulanabileceği örneklerle açıklanır.
• Harici Sinyaller ve 'clk' Sinyalleri
Harici sinyallerin (örneğin, buton girişleri) ve clk (saat) sinyallerinin FPGA sistemleri içindeki farklı rolleri ve işlevleri anlatılır. clk sinyallerinin sistemin zamanlamasını nasıl kontrol ettiği ve harici sinyallerin kullanıcı etkileşimini nasıl temsil ettiği üzerinde durulur, böylece öğrenciler bu iki sinyal türü arasındaki temel ayrımları net bir şekilde anlarlar.

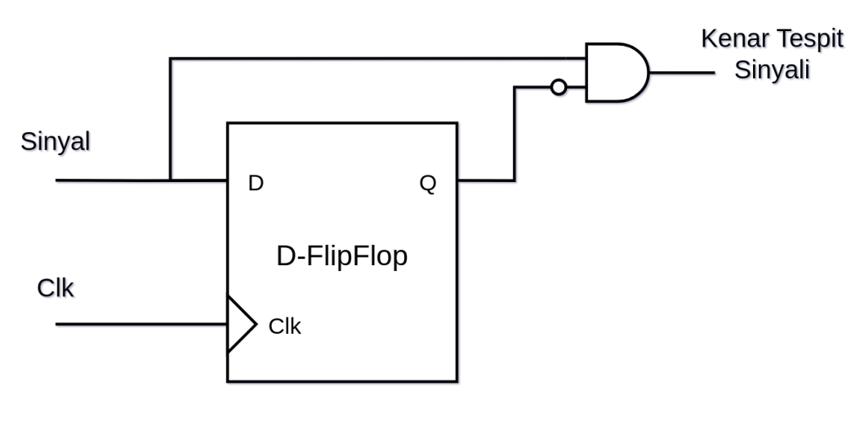
Böylece öğrencilere, FPGA tabanlı sistemlerin temel prensipleri ve VHDL programlama dilinin kullanımı hakkında net, anlaşılır ve kapsamlı bir bilgi sunulmuş olur. Bu kazanımlar sayesinde, öğrenciler sinyal işlemenin ve donanım tasarımının temel prensiplerini pratik bir şekilde öğrenirken, bu bilgileri gerçek dünya uygulamalarına entegre etme becerisi kazanırlar.

Uygulamanın Çalışma Prensibi

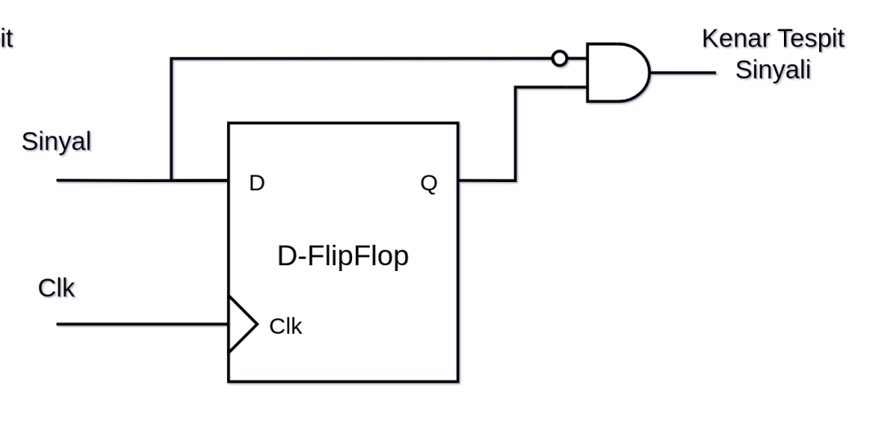
Uygulamamızın temelini, yükselen ve düşen kenar tespiti oluşturur. Yükselen ve düşen kenar mantığı, dijital elektronikte ve özellikle FPGA tabanlı sistemlerde, sinyal durumlarının değişimini algılamada temel bir role sahiptir. Yükselen kenar, bir sinyalin düşük ('0') durumdan yüksek ('1') duruma geçişini; düşen kenar ise yüksek durumdan düşük duruma geçişi temsil eder. Bu kenarlar, sinyal durumlarının zaman içindeki değişimlerini izlemek ve bu değişimlere göre belirli eylemleri tetiklemek için kullanılır.



Dijital sistemlerde bu tür kenar değişimlerini algılamak için genellikle D Flip-Flop, AND ve NOT kapılarından oluşan mantıksal devreler kullanılır. VHDL içinde bu işlevsellik, çoğunlukla rising_edge fonksiyonu ya da sinyalin bir olayı ('event') olduğunu ve belirli bir değere ('1' veya '0') eşit olduğunu kontrol eden ifadeler (örneğin signal'event and signal = '1') ile modellenir. Bu yapılar özellikle clk gibi sürekli değişen ve sabit bir frekansta olan sinyaller için uygundur.



Yükselen Kenar Algılayan Mantık Devresi



Yükselen Kenar Algılayan Mantık Devresi

Dışarıdan gelen donanım sinyalleri (örneğin butonlardan gelen sinyaller) ile FPGA içerisindeki clock (clk) sinyalleri farklı şekillerde işlenmelidir. Bu durumda, sinyalin önceki ve mevcut durumları arasındaki değişim, özel olarak tasarlanmış bir mantık devresi ile modellenebilir. Örneğin, bir butonun durumundaki değişim, butonun önceki ve mevcut değerlerinin karşılaştırılmasıyla tespit edilebilir.

```vhdl
if buton1 = '0' and buton1_prev = '1' then
-- Düşen Kenar Tespit Edildi
end if;

if buton2 = '1' and buton2_prev = '1' then
-- Yükselen Kenar Tespit Edildi
end if;
```

Yukarıda sunulan VHDL kod örneği, kenar tespit mekanizmasının basit bir modellemesini sergiler.

Burada kullanılan "_prev" takısı, ilgili sinyalin bir önceki durumunu temsil eder. Sinyalin geçmişteki değeri '1' iken mevcut değeri '0' olarak algılandığında, bu durum bir düşen kenarın varlığını gösterir. Bu, sinyalin yüksek seviyeden düşük seviyeye geçiş yaptığını ifade eder. Öte yandan, eğer sinyalin geçmişteki değeri '0' ve mevcut değeri '1' ise, bu bir yükselen kenarın tespit edildiğini gösterir, yani sinyalin düşük seviyeden yüksek seviyeye geçtiğini belirtir. Bu mantıksal yaklaşım, sinyal değişimlerinin zaman içindeki evrimini doğru bir şekilde yakalamak için temel bir yöntem sunar ve FPGA tabanlı sistemlerde sinyal işleme için kritik öneme sahiptir.

Gerçek dünyada, özellikle mekanik butonlardan gelen sinyaller, fiziksel titreşimler ve temas esnasında oluşan elektriksel gürültüler (bounce) nedeniyle istenmeyen dalgalanmalar içerebilir. Bu dalgalanmalar, sistemin yanlış sinyaller algılamasına neden olabilir. Debouncing, bu tür istenmeyen gürültüleri filtreleyerek sinyallerin daha temiz ve doğru bir şekilde işlenmesini sağlar.

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity top is
    generic (
        DEBOUNCE_LIMIT : integer := 50000
    );
    Port (
        clk         : in    std_logic;
        btnU        : in    std_logic;
        btnC        : in    std_logic;
        btnD        : in    std_logic;
        led         : out   std_logic_vector (2 downto 0)
    );
end top;

architecture Behavioral of top is
    signal btnU_stable, btnU_prev, btnU_debounced: std_logic := '0';
    signal btnD_stable, btnD_prev, btnD_debounced: std_logic := '0';
    signal btnC_stable, btnC_prev, btnC_debounced: std_logic := '0';
```

Yukarıdaki örnek, bir debouncing algoritmasının uygulamasını göstermektedir. Burada, bir butonun sabit durumu algılandığında sayaç sıfırlanır. Eğer buton durumu değişirse ve belirlenen bir eşik değere (DEBOUNCE_LIMIT) ulaşırsa, butonun stabil olduğu kabul edilir ve ilgili durum güncellenir. Böylece, sinyaller daha temiz bir şekilde işlenir ve yanlış tetiklemelerin önüne geçilir.

Debouncing algoritmasının önemi, onun uygulanmadığı durumlarda daha belirgin hale gelir. Örneğin, debouncing algoritması olmadan doğrudan yükselen ve düşen kenarları tespit etmeye yönelik bir kod, Basys3 kartına entegre edildiğinde, mekanik butonların doğasından kaynaklanan arklar ve istenmeyen sinyal değişimleri rahatlıkla gözlemlenebilir. Bu durum, mekanik butonlardan gelen sinyallerin, fiziksel titreşimler ve temas sırasında oluşan elektriksel gürültüler (bounce) nedeniyle, beklenmeyen sinyal dalgalanmaları içerebileceğini açıkça ortaya koyar. Debouncing algoritması bu tür gürültüleri filtreleyerek, sinyallerin daha temiz ve doğru bir şekilde işlenmesini sağlar. Dolayısıyla, bu algoritmanın uygulanmaması durumunda, butonların basılma ve bırakılma eylemlerinin doğru bir şekilde algılanması güçleşir, bu da sistemde yanlış tetiklemelere ve istenmeyen davranışlara yol açabilir. Bu gözlem, debouncing algoritmasının pratik uygulamalardaki kritik önemini ve FPGA tabanlı sistemlerde sinyal işlemenin doğruluğunu artırma rolünü vurgular.

Kodun Açıklaması

> *Bu doküman içerisindeki kodlar **Vivado 2020.03** üzerinde gerçekleştirilip test edilmiştir.*

Bu VHDL kodu, Basys3 FPGA kartında bulunan butonların durumuna göre LED'leri kontrol eden bir yapılandırmadır. Kod, FPGA üzerinde sinyal işleme ve kontrol mekanizmalarının nasıl kurulacağını ve bu mekanizmaların çeşitli donanım girişleriyle nasıl entegre edileceğini örnekler.

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity top is
    generic (
        DEBOUNCE_LIMIT : integer := 50000
    );
    Port (
        clk         : in    std_logic;
        btnU        : in    std_logic;
        btnC        : in    std_logic;
        btnD        : in    std_logic;
        led         : out   std_logic_vector (2 downto 0)
    );
end top;

architecture Behavioral of top is
    signal btnU_stable, btnU_prev, btnU_debounced: std_logic := '0';
    signal btnD_stable, btnD_prev, btnD_debounced: std_logic := '0';
    signal btnC_stable, btnC_prev, btnC_debounced: std_logic := '0';
```

Projeyi İndir

Kod, IEEE standart kütüphanesini kullanarak, FPGA için gerekli lojik sinyalleri ve veri türlerini tanımlar. Bu kütüphane, FPGA programlamada yaygın olarak kullanılan standart bir araçtır. entity bölümünde, giriş ve çıkış portları tanımlanmıştır; burada clk (saat sinyali), btnU, btnC, btnD (butonlar) girişleri ve led çıkışları yer alır.

Kodun en önemli bölümlerinden biri, butonlardan gelen sinyallerdeki gürültüyü (bounce) filtreleyen debouncing mekanizmasıdır. Bu işlem, her buton için ayrı ayrı gerçekleştirilir. Sinyal stabil olduğunda, yani buton durumu değişmediğinde, ilgili sayaç sıfırlanır. Eğer buton durumu değişirse, belirlenen bir eşik değere (DEBOUNCE_LIMIT) ulaşıncaya kadar sayaç arttırılır. Bu eşik değere ulaşıldığında, butonun stabil olduğu kabul edilir ve ilgili durum güncellenir.

Kenar tespiti, butonların önceki ve mevcut durumları karşılaştırılarak yapılır. Örneğin, btnU için yükselen kenar tespiti (düşükten yükseğe geçiş) yapılırken, btnD için düşen kenar tespiti (yüksekten düşüğe geçiş) yapılır. btnC ise her iki kenarın tespiti için kullanılır.

Her buton, belirlenen kenar tespitine göre bir LED'in durumunu değiştirir. Örneğin, btnU yükselen kenarı tespit edildiğinde led_state(0) güncellenir. Benzer şekilde, btnD düşen kenarı tespit ettiğinde led_state(2), btnC ise herhangi bir kenar değişikliği tespit ettiğinde led_state(1) güncellenir.

Tüm bu işlemler, FPGA'nın clk sinyali ile senkronize bir şekilde gerçekleştirilir. Bu, sistemin zamanlamasının doğru bir şekilde yönetilmesini ve tüm sinyal işleme adımlarının güvenilir ve tutarlı olmasını sağlar.

Bu kod, FPGA tabanlı sistemlerde donanım kontrolünün temel prensiplerini ve VHDL programlama dilinin kullanımını pratik bir şekilde gösterir. Öğrenciler, bu bilgiler sayesinde, sinyal işlemenin ve donanım kontrolünün temel prensiplerini anlama ve bu prensipleri gerçek dünya uygulamalarına entegre etme becerisi kazanırlar.

Deneyin Yapılışı

1. Kişisel bilgisayarınıza Proje Dosyasını indiriniz.
2. İndirdiğiniz **'.zip'** dosyasını, belirteceğiniz bir çalışma klasöre ayıklayınız.
3. Dosyaların içindeki **'.xpr'** Vivado Project File dosyasına çift tıklayarak veya Vivado arayüzündeki üst panelden **'File > Open'** yolunu takip ederek açınız.
4. Projeniz açıldığında proje ekranının sağ üst köşesinde **'write_bitstream_Complate'** ve ✅ sembolünü görmelisiniz.
5. Basys3 kartının üzerindeki güç anahtarını açarak, **'Power'** ledinin yandığına emin olun.
6. Vivado'da sol taraftaki **'Flow Navigator'** penceresinden **'Open Hardware Manager'** altındaki **'Open Target > Auto Connect'** yolunu takip edin. Bu cihazınız ile FPGA kartı arasındaki bağlantıyı sağlayacaktır.
7. Cihazı programlamak için **'Open Hardware Manager'** altındaki **'Program Device'** butonuna tıklayın.

Deney yüklendiğinde gözlenmesi gereken davranışlar aşağıdaki kutucukta paylaşılmıştır.

> ***Böylece, üstteki butona basıldığında ;***
> *En sağdaki LED0 yükselen kenarda tetiklenerek yanacaktır, tekrar basılması durumunda yine yükselen kenarda tetiklenerek sönecektir.*
> ***Merkezdeki butona basıldığında ;***
> *LED1 hem yükselen kenarda hem düşen kenarda tetiklenecektir, dolayısıyla parmağınızı bastığınızda tetiklenecek ve led yanacaktır. Çektiğinizde ise tekrar tetiklenecek ve sönecektir.*
> ***Alttaki Butona Basıldığında ;***
> *LED2 düşen kenarda tetiklenecektir, butona basıldığında değil elinizi butondan çekerken tetiklenecek ve yanacaktır. Tekrar bastığınızda aynı şekilde elinizi çektiğinizde sönecektir.*

Telif Hakkı

Bu doküman, **Beti Elektronik San. ve Tic. Ltd. Şti**'nin fikri mülkiyetine tabidir ve firmanın izni olmadan kopyalanması, çoğaltılması, dağıtılması veya yeniden yayınlanması yasaktır. Kitabın herhangi bir bölümünün veya içeriğinin izinsiz kullanılması, telif hakkı ihlali anlamına gelir ve yasal işlemlere yol açabilir. Dokümanın içeriği veya herhangi bir kısmı kullanılacaksa, yazılı izin alınmalıdır. İzin talepleri için info@beti.com.tr ile iletişime geçiniz. Makul alıntılar yapılabilir, ancak bu alıntılar mutlaka kaynak gösterme ile birlikte sunulmalıdır. Kaynak göstermeden yapılan alıntılar veya içerik kullanımları, yine telif hakkı ihlali olarak kabul edilir.