ELE519/ BIL569

Ödev#2 (Teslim Tarihi 22/06/19 saat 23:59)

"Half of the engineering is to meet the deadlines"

Önemli Not: Kodlarınızı ve cevaplarınızı tek bir pdf dosyası içinde

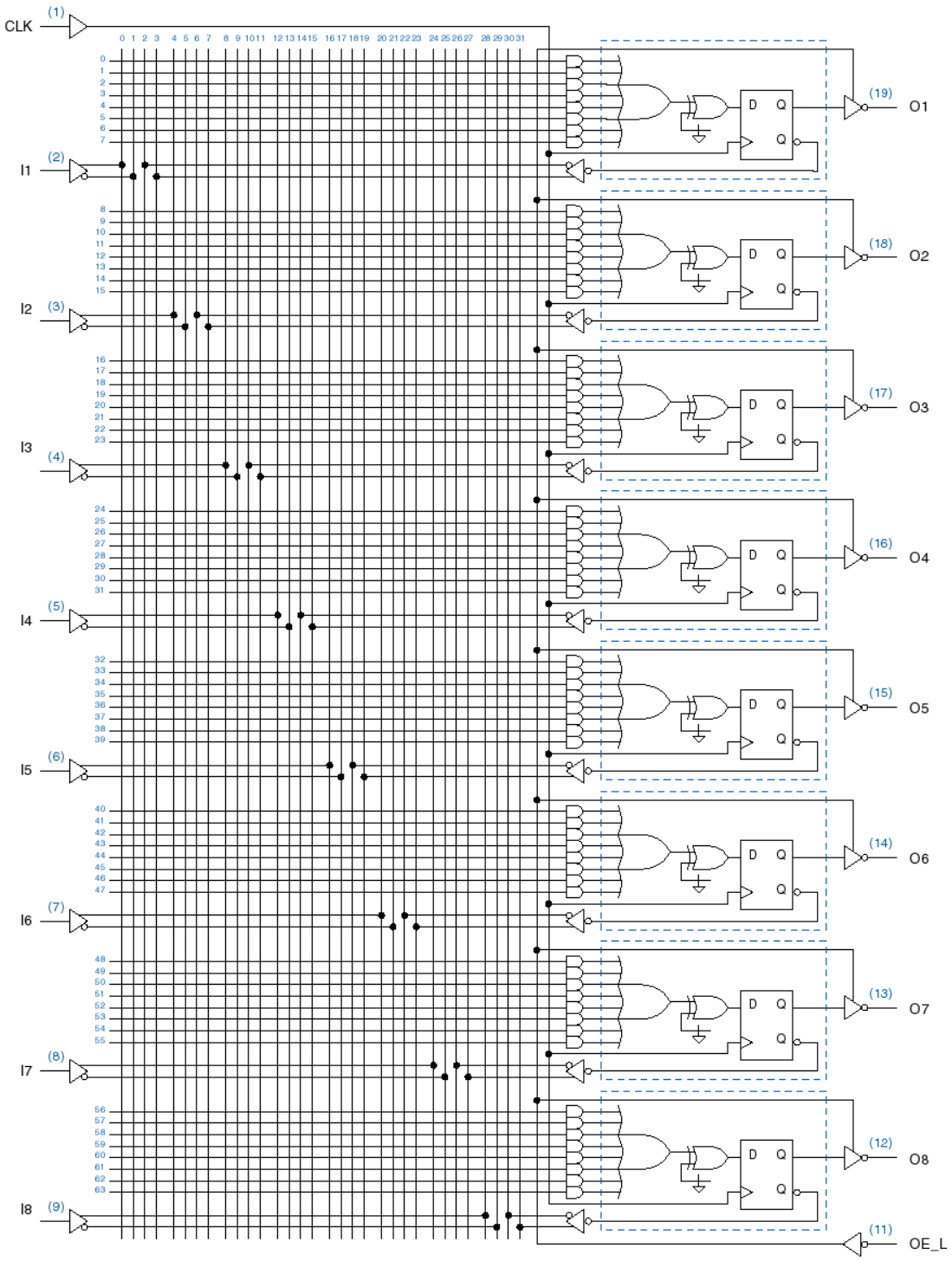
https://www.dropbox.com/request/aE3w0Zcai3bHAACoHiBl

adresine yükleyiniz. Elle yazdığınız kısımların fotoğraflarını çekerek dökümanın içine koyabilirsiniz.

Soru 1 (20 Puan)

Aşağıdaki şekilde **GAL16V8** Programmable Logic Device (PLD)'nin iç yapısı görülmektedir. Bu PLD entegresinn **I1,I2,I3** girişlerini ve **O1,O2** çıkışlarını kullanarak bir full adder devresi gerçeklemek istediğimizi düşünelim. I1,I2 giriş; I3 elde giriş(CIN) işareti, O1: toplam(SUM), O2: elde çıkış (COUT).

a) Full Adder fonksiyonunu **AND**, **OR, NOT** kapılarını kullanarak yazınız (5 puan)
b) Bu fonksiyonu gerçeklemek için **GAL16V8** üzerinde hangi bağlantıların yapılması gerektiğini nokta koyarak belirtiniz. (10 puan).
c) Bu bağlantıları programlamak için gerekli sigorta yapısını çiziniz (5 puan)



Onemli not:

1. Derste belirtildiği üzere PAL çizimlerinde basitlik olması açısından AND kapısı girişleri tek bir çizgi olarak gösterilir. Aynı hatta bağlı işaretler AND'lenmiş kabul edilir.

2. Çıkışlarda INVERTER olduğunu unutmayın!- 5 puan

Soru 2 (10 Puan)

Spartan 6 FPGA'ler üzerinde bulunan LUT6 elemanları, 6 girişli, 2 çıkışlıdır. Full Adder fonksiyonunu girişler AIN, BIN, CIN, çıkışlar SUM, COUT olacak şekilde LUT6 kullanarak gerçeklemek istediğimizi düşünelim.

a) Girişleri hangi pinlere uygulayacağınızı, çıkışları hangi pinlerden alacağınızı belirtiniz? A6 pininin değeri ne olacaktır? (2 puan)
b) LUT5 elemanlarının içeriğini her adrese bir bit gelecek şekilde tablo halinde yazınız? (8 puan)

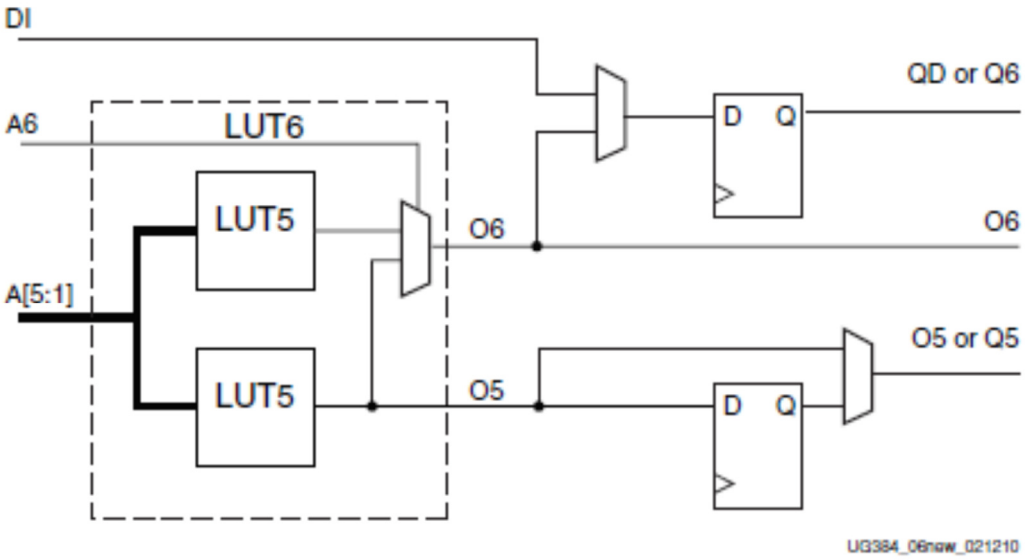


Figure 6: **LUT6**

Soru 3 (70 Puan)

a) Full-Adder devresini derste gösterildiği şekilde VHDL kullanarak gerçekleyiniz. (**FA.vhd, HA.vhd**) **(5 puan)**

b) Gerçeklemesini yaptığınız Full-Adder devresini kullanarak **GENERATE** statement yardımıyla iki tane 128 bitlik sayıyı toplayan bir adder fonksiyonunu VHDL kullanarak gerçekleyiniz. Gerçeklemesini yapacağınız adder devresinin girişleri **A_IN, B_IN (128 bit),** çıkışları **SUM_OUT (128 bit), CARRY_OUT (1 bit)** olacaktır. (**ADDER_128.vhd**) **(10 puan)**

c) 128 bit'lik Full-Adder devresini, **CARRY-PROPAGATE ADDER** yöntemi ile gerçekleyiniz. Multiplexer'ları gerçeklemek amacıyla **WITH-SELECT** statement kullanınız. **(10 puan) (ADDER_CARRY_LOOKAHEAD.vhd)**

d) 128 bit'lik Full-Adder devresini, **CARRY-PROPAGATE ADDER** yöntemi ile Virtex-7 serisi FPGA'ların CARRY4 makrosunu kullanarak gerçekleyiniz. CARRY4 makrosunun nasıl instantiate edileceği ile bilgiyi aşağıdaki linkte bulabilirsiniz:

http://www.xilinx.com/support/documentation/sw_manuals/xilinx2012_2/ug953-vivado-7series-libraries.pdf **(15 puan)** (**ADDER_CARRY_PROPAGATE_MACRO.vhd**)

e) **IEEE_STD_LOGIC_ARITH.ALL ve IEEE_STD_LOGIC_UNSIGNED.ALL** paketleri '**+**' operatörünün toplama yaparken kullanılmasına imkan vermektedir.
Bu paketlerin vhdl kodları aşagıdaki klasörde yer alır. İnceleyebilirsiniz.

<Drive>:\<Xilinx kurulum klasörü>\Vivado\<2015.4>\data\vhdl\src\ieee\distributable

128 bitlik adder devresini **+** operatörünü kullanarak, girişleri **INTEGER'a** çevirerek gerçekleştiriniz. **(15 puan)** (**ADDER_PLUS_SIGN.vhd**)

(İki integer toplanacak, daha sonra STD_LOGIC'e geri çevirilecek.)

Değişkenleri **STD_LOGIC**'den **INTEGER**'a çevirmek, **INTEGER**'dan STD_**LOGIC**'e dödürmek için aşağıdaki fonksiyonlara ihtiyacınız olacaktır. Bunlar yukarıdaki kütüphanelerde tanımlıdır.

function CONV_STD_LOGIC_VECTOR(ARG: STD_ULOGIC; SIZE: INTEGER) return STD_LOGIC_VECTOR

function CONV_INTEGER(ARG: STD_LOGIC_VECTOR) return INTEGER

Önemli Not: 128 bit'lik integer yoktur!

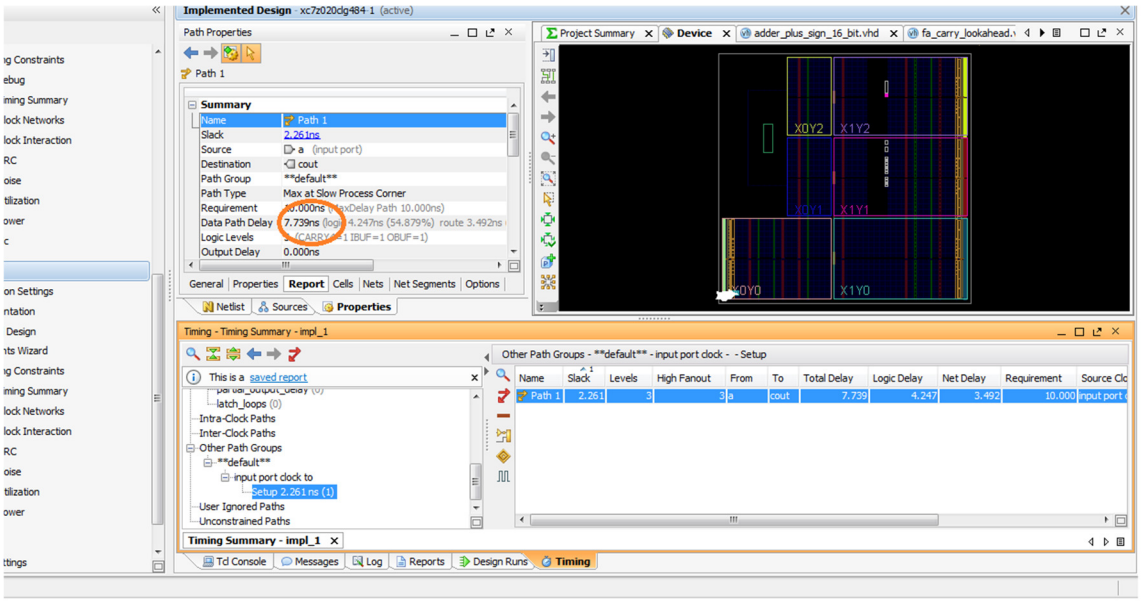
f) Bir .xdc dosyası oluşturarak maximum carry gecikmesini bulmamızı sağlayan timing constaint'i aşağıdaki satırları ekleyerek oluşturunuz.

set_max_delay -from [get_ports a_in] -to [get_ports carry_out] 10.000
set_max_delay -from [get_ports b_in] -to [get_ports carry_out] 10.000

Bu .xdc dosyasını constaint dosyası olarak ekleyerek **(Add or create constraints)** b,d,e şıklarında yazdığınız kodları sentezleyiniz. Vivado-> Implemantation altındaki **Report Timing Summary**'i çalıştırarak b,d,e şıklarında gerçeklediğiniz devreler için giriş portları ile carry_out arasında ne kadar gecikme olduğunu bulunuz. Bunları tablo halinde yazınız. **(10 puan)** Hangi ADDER gerçeklemesi daha hızlı neden? **(5 puan)**

Gecikme bilgisi köşede bulunan **Timing Summary** ekranında **Other Path Groups -> Input Port Clock to** kısmı açıldığında **Path Properties** ekranında görülüyor.

Örnek olarak A_IN portu ile CARRY_OUT geçikmesi için alınan görüntü aşağıda görülmekte.



İYİ ÇALIŞMALAR