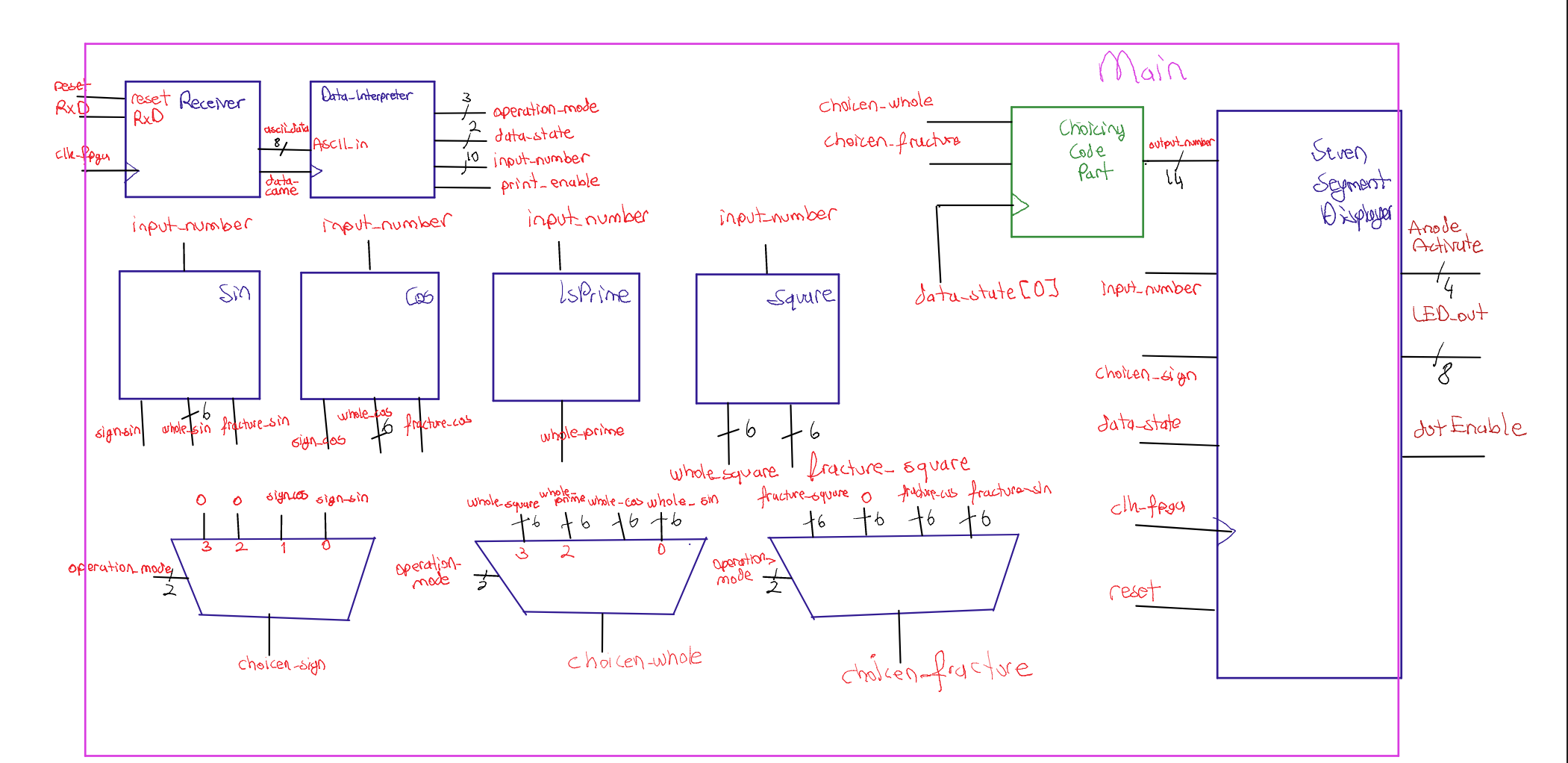
**Main Modülü**

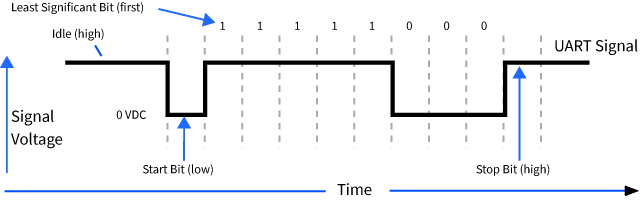
Main modülü projenin en üst modülü olup birer bitlik Reset, RxD, clk\_fpga girişlerine sahiptir. Bunun dışında 7-segment display’i kontrol etmek için kullanılan 4 bitlik Anode\_Activate, 8 bitlik Led\_Out ve 1 bitlik dotEnable çıkışına sahiptir.

Bu modülün giriş ve çıkışları defConstraint.xdc dosyasından alınan dosyada FPGA’nın portlarına bağlanmıştır.

Projede genel manada veri UART protokolü ile alınmaktadır. UART protokolünün uygulandığı Receiver modülü gelen her bir data bittiğinde ascii\_data isimli 8 bitlik veriyi ve data\_came isimli bir sinyali Data\_Interpereter isimli modüle göndermektedir. Data\_Interpreter bu sinyal geldiğinde kendisine gelen verinin geçerli bir veri olduğunu anlayıp bu veriyi yorumlamaktadır. Data\_Interpreter her data\_came sinyalinde gelen verileri yorumlayarak( sayı-işlem, sayı-sayı-işlem, sayı-sayı-sayı-işlem ) durumları yakalandığı zaman bu durumlara göre dışarıya sinyal ve gelen verinin yorumunu belirten bazı çıkışlar vermektedir. Bu çıkışlar işlemin türünü, gelen ascii verinin 10 bitlik bir sayı halindeki çıkışını ve o an verinin tamamlanıp tamamlanmadığını gösteren data\_state isimli bir çıkış vermektedir. Data\_Interpreter’in nasıl çalıştığı ileriki bölümlerde detaylı olarak anlatılmıştır.

Data\_Interpreter modülünden çıkan veri işlemi gerçekleştiren Sin, Cos, IsPrime, Square isimli modüllere aynı anda gönderilmektedir. Data\_Interpreter modülünden çıkan operation\_mode isimli sinyale göre bu işlemlerin sonuçlarından doğru olan kısımları seçilip, tek bir sayı haline getirilerek SevenSegmentDisplayer modülüne gönderilmektedir. Bu seçme işlemi Data\_Interpreter’den çıkan data\_state verisi her 1 olduğunda gerçekleşmektedir.

Aynı zamanda bu data\_state verisi ve Data\_Interpereter’den çıkan input verisi de SevenSegmentDisplayer’a gönderilmektedir. Bu gönderimin sonucunda oluşan çıktılar SevenSegmentDisplayer modülünden çıkarılarak FPGA’de gösterilmektedir. SevenSegmentDisplayer modülü sadece sonucu değil, girilen sayıyı da işlemi belirten sinyal gelene kadar göstermektedir. Aynı zamanda geçerli olmayan bir veri girişi geldiğinde de özel olarak ‘Err’ yazmaktadır. Bu yüzden üstteki girişlerin hepsi içinde kullanılmaktadır. Buna detaylıca SevenSegmentDisplayer kısmında değinilecektir.

**Receiver Modülü**

Receiver, UART protokolüne uygun olarak gelen veriyi sıralı bir şekilde asenkron olarak almaktadır. UART protokolü gereği, veri gelmediği zaman oluşan sinyal 1 olmaktadır. Sinyal 0’a indiği an yukarıdaki şekilde görüldüğü gibi bu başlangıç biti olarak algılanmaktadır. Başlangıç biti geldikten sonraki verileri, o verilerin gelme zamanının tam orta noktasından okuyarak 8 bit okumaktadır. Bu bitleri sayarak son bite gelip gelmediğini anlamaktadır. Bütün bunları yaparken aynı zamanda arka planda şu an veri alıp almadığını da tutan bir durum makinesi olarak çalışmaktır. Bu projede parity bit kullanılmamıştır.

Bu modüle gelen veri ascii şeklinde gelmektedir, eğer 8 bitin tamamı geldiyse data\_came sinyalini dışarıya vermektedir. Bu sinyal ise Data\_Interpreter modülünün clock’u olarak görev yapmaktadır.

**Sin-Cos Modülü**

Sin ve Cos modülünde ilk başta gelen sayının birim çember üzerindeki aralığı bulunmaktadır. Bu aralığa göre işareti atanmakta ve bu açının 0-90 derece aralığındaki karşılığı bulunmaktadır. Bu aralıktan sonra 0-90 arasına karşılık gelen değerler whole ve 7 bitlik fracture(yani ondalıklı kısım olarak) işaretle(sign) beraber dışarıya verilmektedir.

**Data Interpreter**

Amaç: Receiver modülününden gelen ASCII değerlerinin geçerli giriş olup olmadığını kontrol edip Hesaplamalardan önce 7 segment display’de gösterilecek sayının belirlenmesi, hesaplamalarda kullanılacak sayının belirlenmesi.

Diagram

Description automatically generatedText, letter

Description automatically generatedDurum Diyagramı:

Bu işlemlerin yapılmasında kullanılan giriş ASCII\_in ve çıkışlar modeSelect, validCheck, ve numOut.

İlk durumda gelen ascii değerinin 0,1,2,3,4,5,6,7,8,9 sayılarının ascii değeri veya gelen ascii değerinin “s,c,r,q” harflerinden birinin ascii değeri olup olmadığı kontrol edilir. Eğer bu kontrolden geçmezse çıktı olarak numOut = 0, modeSelect = 4, validCheck = 2. validCheck ‘2’ olduğu için veriler geçersizdir ve Main modulündeki hesaplama kısımı hesaplamaları yapmaz. Eğer ilk durumda gelen değer “s,c,r,q” harflerinden birinin ascii değeri ise bu durumda da çıktı olarak numOut = 0, modeSelect = 4, validCheck = 2. validCheck ‘2’ verilir, çünkü “don’t care” değerinin hesaplanmasını istemiyoruz. İlk durumda 0,1,2,3,4,5,6,7,8,9 sayılarının ascii değeri geldigi zaman birler basamağıni tutan “reg” degiskenine bu ascii değerinin tam sayı karşılığı yazılır ve sonraki duruma geçilir.

İkinci durumda yine geçersiz bir giriş olup olmadığı kontrol edildikten sonra; 0,1,2,3,4,5,6,7,8,9 sayılarının ascii değeri geldiyse sonraki duruma geçilir ve bu ascii değerin karşılığı olan tam sayı birler basamağına yazılır, onceden birler basamağında olan sayı onlar basamağına yazılır, “s,c,r,q” harflerinden birinin ascii değeri geldiyse A durumuna geri dönülür ve çıktı olarak A durumunda yazılan sayı verilir, validCheck = 1, ve modeSelect hangi hesaplama komutunun ascii değeri verildiyse buna karşılık gelen sayı olur (s(Sin hesapla ascii değeri) = 2’b00 ,c(Cos hesapla ascii değeri) = 2’b01, q(Asal hesapla ascii değeri) = 2’b10, r(kok hesapla ascii değeri) = 2’b11; validCheck değerinin 1 olması veri girişinin bittigini ifade eder ve hesaplama kısımına geçilir.

Üçüncü durumda yine geçersiz bir giriş olup olmadığı kontrol edildikten sonra; önceki kısımdaki kontroller yine yapılır; sin, cos, asal, kok işlemlerini yaptıracak ascii değerleri gelirse A durumuna dönülür hesaplama yapılır, rakamların ascii değeri girilirse sonraki duruma devam edilir. Bir rakamın ascii değeri geldiyse onlar basamağındaki bulunan değer yüzler basamağına yazılır, birler basamağında bulunan değer onlar basamağına yazılır ve yeni girilen değer birler basamağına yazılır.

Son duruma gelindiyse bu 3 kez sayı girildi demektir. Burada yeniden sayı girilirse bu girilen sayı yeni bir sayının birler basamağı kabul edilir ve “B” durumuna dönülür. Eğer sin, cos gibi bir hesap komutunun ascii değeri gelirse “A” durumuna dönülür ve sayının hesaplaması yapılır.

printEnable çıkışı işlem komutunu belirten ascii değeri gelmeden önce 7 segment display’de data interpreter içindeki sayının gösterilip gösterilmeyeceğini belirten ve sadece debug için kullanılan bir çıkıştır.

**Asal Sayı alma (is\_prime)**

Asal sayı alma lookup table ile yapılır 360’a kadar olan asal sayılar bilinmektedir. Data Interpreter modülünden asal sayı hesapla sinyali gelirse, data interpreter modülünün çıktı olarak verdiği değer bu bilinen asal sayılar ile kıyaslanır. Eğer gönderilen sayı bunlardan biri ise çıktı olarak whole\_prime gönderilir. Burada whole\_prime = 1 olarak belirlenir. Eğer sayı asal değilse whole\_prime = 0 olur.

**Kök Alma (square\_calculator)**

Chart, line chart

Description automatically generated Kök alma işlemi hem lookup table hem de lineer yaklaşımla yapılır. Lineer yaklaşım kök almak için hassasiyetten biraz taviz verilecekse yeterli bir yöntemdir.

Grafikte de görüldüğü gibi ikinin karesinin(4)çevresindeki sayılar için(ör. 3,84) lineer yaklaşım iyi bir yaklaşık değer sunmaktadır. Biz de kök alma işleminde bu sebepten lineer yaklaşım kullandık. Bir sayının kare kökünün lineer yaklaşımının yapılabilmesi için hangi iki sayının karesinin arasında olduğunun belirlenmesi gerekmektedir. Bu sebepten 19’dan 0’a kadar olan sayıların karesi ile Data Interpreter’dan gelen sayı kıyaslanır ve hangi aralıkta olduğu bulunur. Bundan sonra ise bu aralıktaki (üst sınır olan sayı – sayı) ile (sayı – alt sınır olan sayı) değerleri karşılaştırılarak hangi çıkarma işlemi daha küçükse o sınıra göre yaklaşım yapılır. Bu modülün hesapladığı değer 1000 ile çarpılır. Bunun sebebi ise verilog’un tam olmayan sayıları aşağı yuvarlamasıdır. Burda hesaplanan lineer yaklaşım da her zaman tam sayı olmadığı için hesaplanırken sayı 1000 ile çapılır, böylece virgülden sonraki 3 basamak korunmuş olur. Buradan sonra ise tam kısım çıktısı sayının 1000’e bölümüdür, virgülden sonraki kısımın çıktı olarak kaydedilmesi için sayının 1000’e bölümünden kalan sayı 10’a bölünür ((sayı%1000)/10) böylece virgülden sonraki iki basamak da ayrı bir değişken olarak çıktı olarak verilir.

**SevenSegmentDisplayer Modülü**

Bu modül Main modülünde yapılan işlemler sırasında oluşan uygun çıktıları FPGA üzerindeki seven-segment-display üzerinde göstermektedir. Bunu yaparken en basit haliyle gelen sayının her bir basamağını özel bir frekans değerinde çok hızlı bir şekilde her bir parçada göstermektedir. Bu geçiş çok hızlı olduğu için sayı tek seferde tamamı gösterilmiş gibi olur.

Sayıyı gösterebilmek için öncelikle Anode\_Activate isimli çıkışla sayının hangi basamağının hangi çıkış üzerinde gösterileceği seçilir. Bu seçime göre ve gelen sayıya göre LED\_out çıkışından ve dotEnable çıkışından uygun veriler gönderilir.

Modülümüz, verileri gösterirken temel manada üç duruma göre göstermektedir. Bu durumlar Data\_Interpreter’in 2 bitlik data\_state çıkışı ile belirlenmektedir. Bu çıkış, SevenSegmentDisplayer modülüne giriş olarak gönderilerek hangi durumda çalışacağı belirlenir. Bu durumlar Input\_State, Output\_State ve Error\_State olmak üzere üç tanedir. Bunun yanı sıra gelen sayının negatif olup olmadığını belirten 1 bitlik sign girişi de vardır.

Data\_State değişkeni 0 olduğu zaman modülümüz Input\_State durumundadır. Bu durumda Data\_Interpreter modülünden çıkan input\_number isimli veriyi, yani kullanıcı tarafından klavyeden girilen sayı gösterilmektedir. Bu durumdayken normal bir hesap makinesinin göstergesi gibi çalışır.

(Örnek: İlk başta girilen sayı 1 olsun. Gösterilen değer sadece 1 olur. Sonrasında 2 girilirse bu 12 olur, en sonunda 3 girilirse bu sayı 123 olur. Sayının bu değişimi daha önce bahsedildiği gibi Data\_Interpreter modülünde gerçekleşmektedir.)

Data\_State değişkeni 1 olduğu zaman modülümüz Output\_State durumundadır. Bu durumda hesaplama modüllerinden çıkarak Main modülünde seçilerek ortaya çıkan sonuç yani output\_number girişi, işaret dikkate alınarak FPGA’daki seven-segment ekranında gösterilir. İşaret, sadece en anlamlı parçada gösterilir.

(Örnek: -0,11 için sign 1 olması beklenir.)

Data\_State değişkeni 2 olduğu zaman modülümüz Error\_State durumundadır. Bu durumda yanlış bir giriş yapıldığı anlamına gelir. Dolayısıyla, FPGA üzerinde “Err” kodu yazdırılır.

Gelen sayı hangi sayı olursa olsun, binler basamağı için 1000’e bölünerek, 100’ler basamağı için 1000’e modu alınıp 100’e bölünerek, 10’lar basamağı için 100’ bölünüp 10’a modu alınır, birler basamağı için ise direk 10’a modu alınır.

Bunların yanısıra FPGA üzerindeki Seven-Segment’ta doğru şekilde veirlerin gösterilmesi için, gelen clock’un belirli bir miktar yavaşlatılması gerekir. Bu durum, 20 bitlik refresh\_counter isimli bir değişken ile sağlanır. Refresh\_Counter’ın en anlamlı iki biti LED\_Activating\_Counter isimli değişkene aktarılır. Bu değişkene göre hangi Anodun seçilerek hangi basamağın basılacağı seçilir.

Basamağın sayısal değerinin gösterilmesi için LED\_PLACEHOLDER değişkenine göre hangi katotların kapatılıp açılacağı belirlenir. Burada aynı zamanda e,r,- gibi özel karakterler de 10’dan sonraki değerlerde kodlanmıştır.

Seven segment display şu şekilde çalışır, Annode’lar ile 4 tane seven segment displayden hangisinin seçileceğini belirtir. Katodlar ise herhangi bir seven segment display üzerindeki hangi segmentlerin çalışacağını belirtir. Bunu bir örnek üzerinden açıkça görebiliriz. Annode seçimi 4’b0111 olunca sadece en soldaki seven segment seçilir ve buradadaki katotlarında seçimi 7’b1001111 şeklinde olursa en soldaki seven segment displayde 1 rakamı gözükür. Her seven segment’in nokta için de bir katodu vardır. Bu da hepsi için aynı değişkenle kontrol edilir hangi seven segment displayin noktasının yanacağı annode seçimine göre belirlenir. Seven segemnt displaylerde hem annode hem katotlar için bir seçim yapmak istersek onu 0 yaparız seçim yapmayacağımız kısımlar 1 olarak kalır. Önceden örneklendiği gibi 4’b0111 annode seçimi en soldaki seven segment displayi seçer.