Emre İŞSEVER
Gömülü Sistem Mühendisi

04.12.25

HDL Coder ile FPGA Tasarımında Model Tabanlı Yaklaşım

FİGES
applied physics & mathematics

MathWorks®
Authorized Reseller

Agenda

01 **FPGA Teknolojisi**

FPGA Nedir? Avantajları Nedir? Nerede Kullanılır?

02 **Temel Sayısal Tasarım Altyapısı**

Lisans Düzeyi Temel Sayısal Tasarım Uygulamaları ve Vivado Kurulumu

03 **FPGA Programlama Metodları**

Uygulamalı geleneksel FPGA Programlama Metodları

04 **Model Tabanlı Yaklaşım ile FPGA Programlama**

Akademik hayatta MBD ile FPGA Programlamanın Yeri

05 **Alana Özgü Uygulamalar**

Soyutlanma Seviyesi Yüksek Tasarımlar ve Uygulanması

06 **FPGA Doğrulama**

FPGA Doğrulamada MATLAB Araçlarının Yeri

07 **Q & A**

FPGA Nedir?

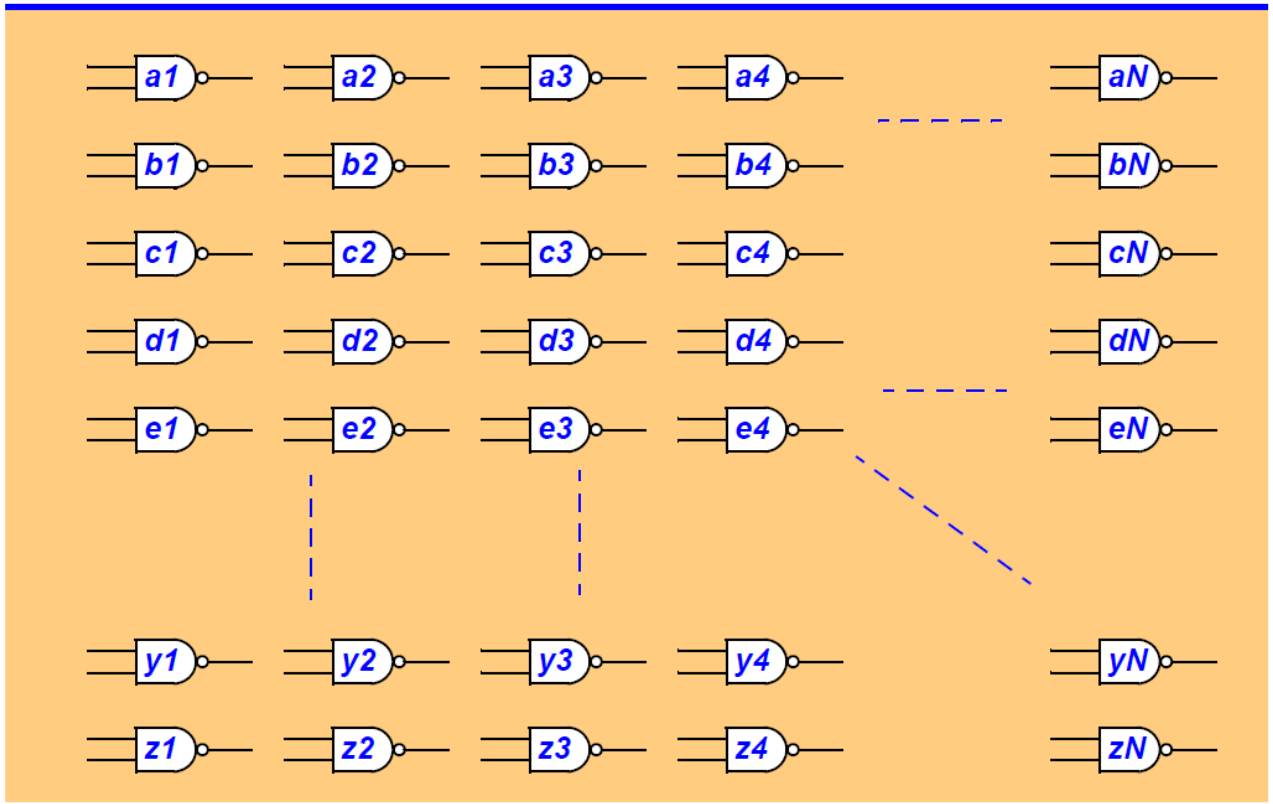
***F**ield **P**rogrammable **G**ate **A**rray (FPGA),* davranışını özelleştirmek için programlayabileceğiniz dijital mantık devreleri içeren bir elektronik aygıttır.

- ***"Field Programmable"*** – Donanım, fabrika dışında (=sahada 'Field') programlanabilir olduğu için sahada programlanabilir ismini alır.

- ***"Gate Array"*** – 2 Boyutlu Mantık kapı dizisi.

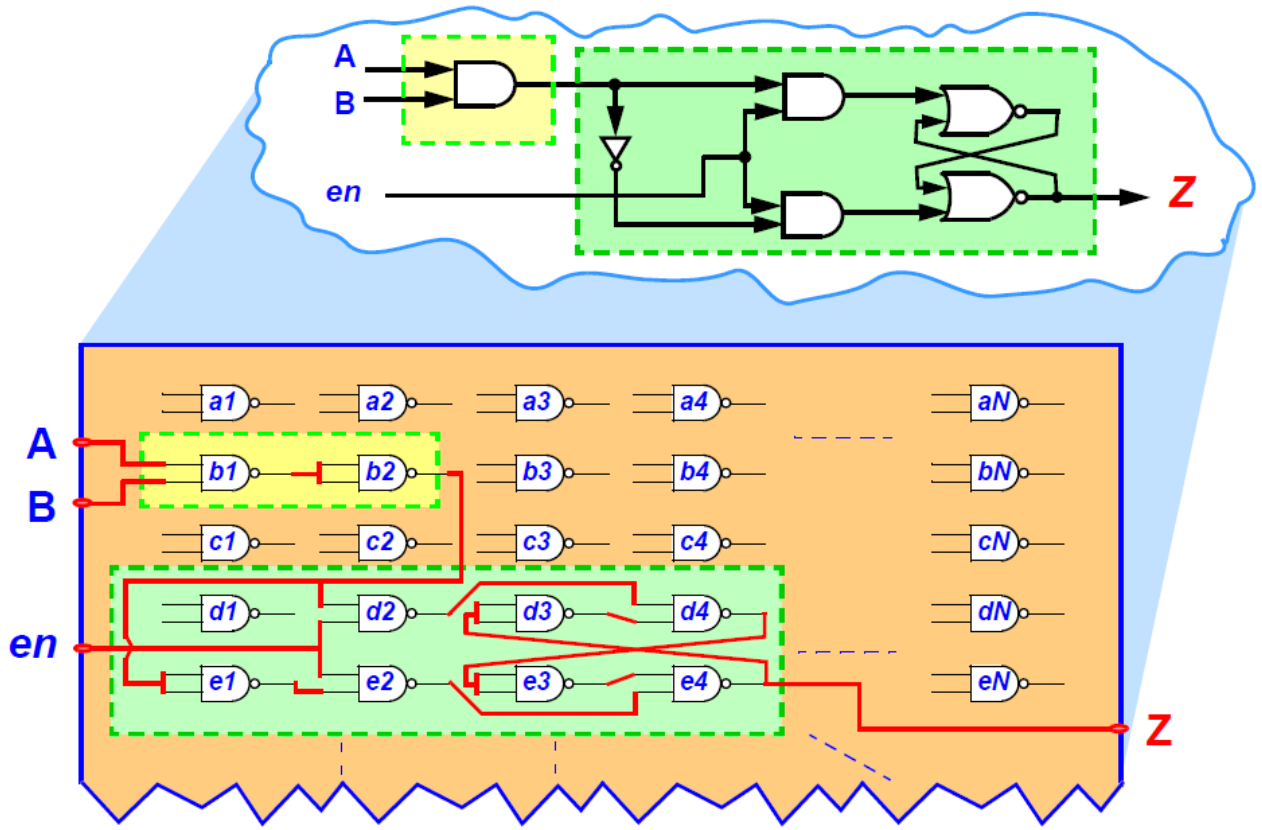
Gate Array – I

Önceleri kapı dizisi *(ing. Gate Arrays)* basitçe bir dizi NAND kapısına deniyordu.

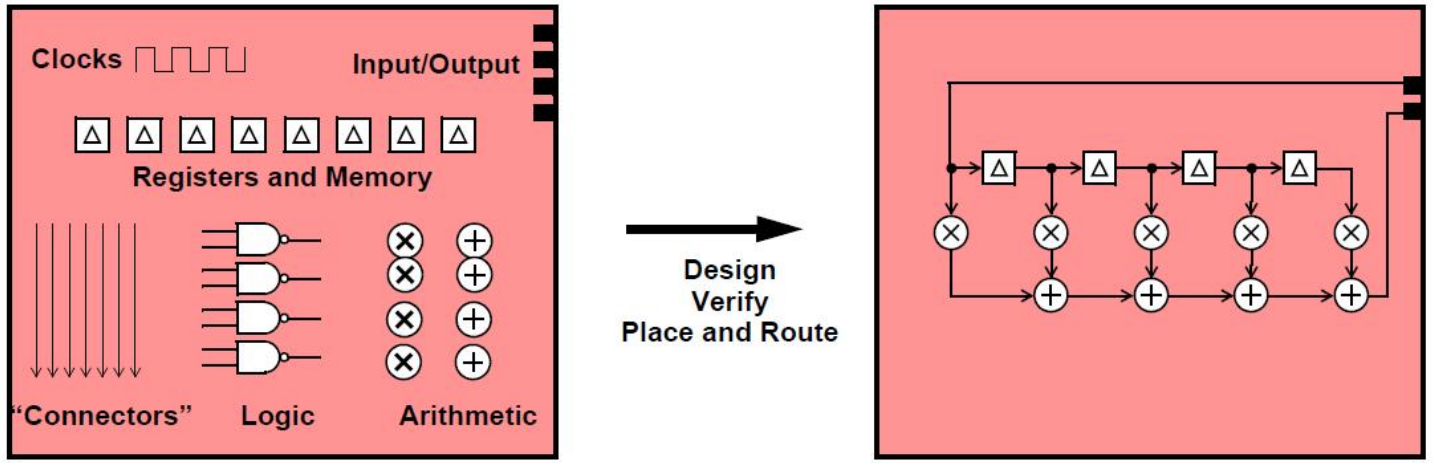


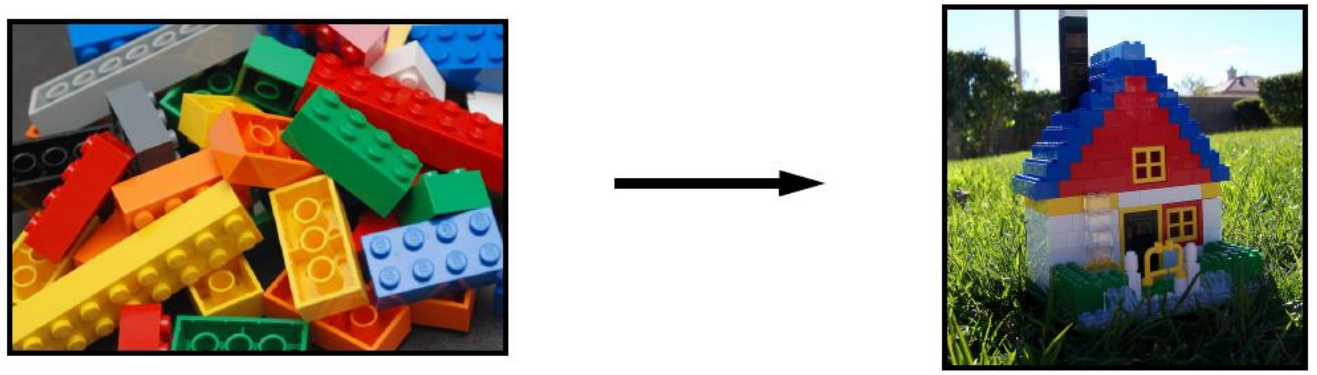
Gate Array – II

Örneğin burada ***AND Kapısı*** ve ***D-Latch*** yapısı NAND kapıları kullanılarak tasarlanmıştır.



FPGA : Bir Kutu Dijital Mantık Devresi





Nerede Kullanılır?

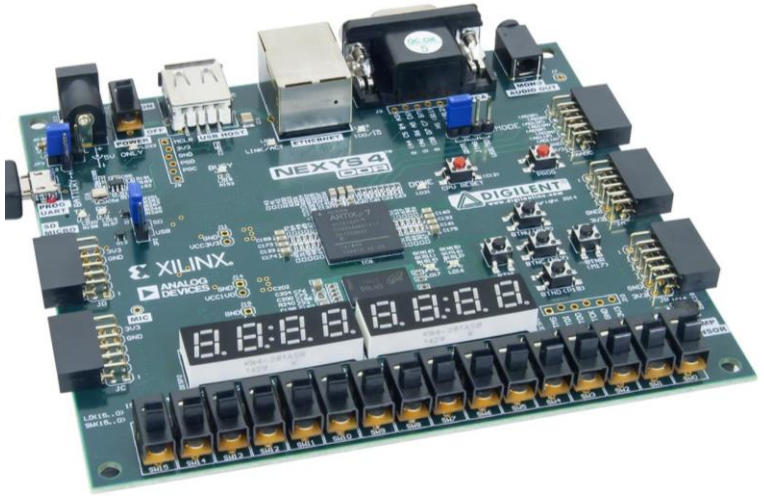
- **Paralel Hesaplama**
  - Dijital Sinyal İşleme
  - Dijital Haberleşme (ör. SDR)
  - Yüksek hesaplama gücü gerektiren algoritmaların gerçeklenmesi.
- **Düşük Gecikme**
  - Motor Kontrol, Güç Dönüştürücüler , Servo Sistemler
- **Deterministik Zamanlama**
  - Radar , Elektronik Harp , Güvenlik Kritik Sistemler
- **Yeniden Programlanabilirlik**
  - Yeni algoritma denemeleri
  - Çip Tasarım Prototipleri
  - Yüksek Lisans/Doktora Projeleri

FPGA Programlama / Tasarlama

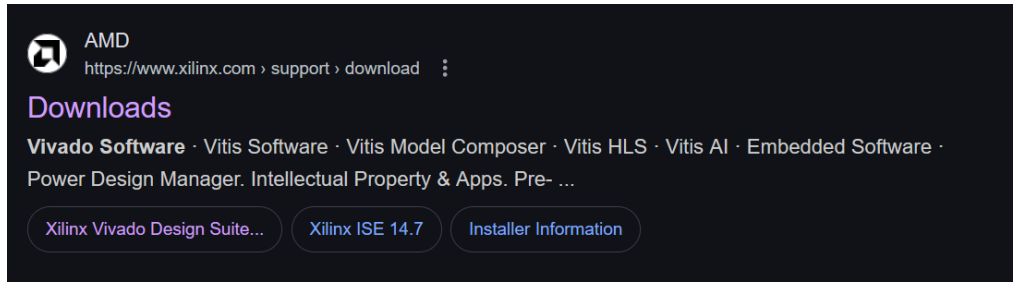
Hardware Description Language (HDL)

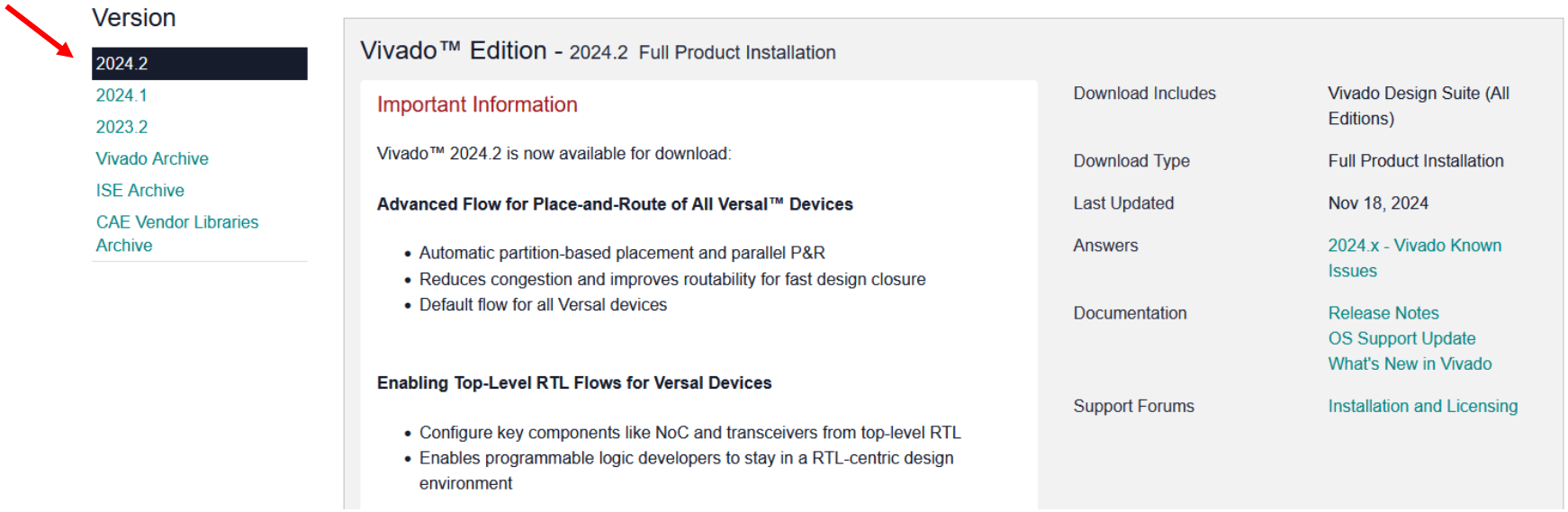
VHDL (.vhd)
Verilog (.v)
SystemVerilog (.sv)

- AMD Vivado
- Intel Quartus
- Microchip Libero



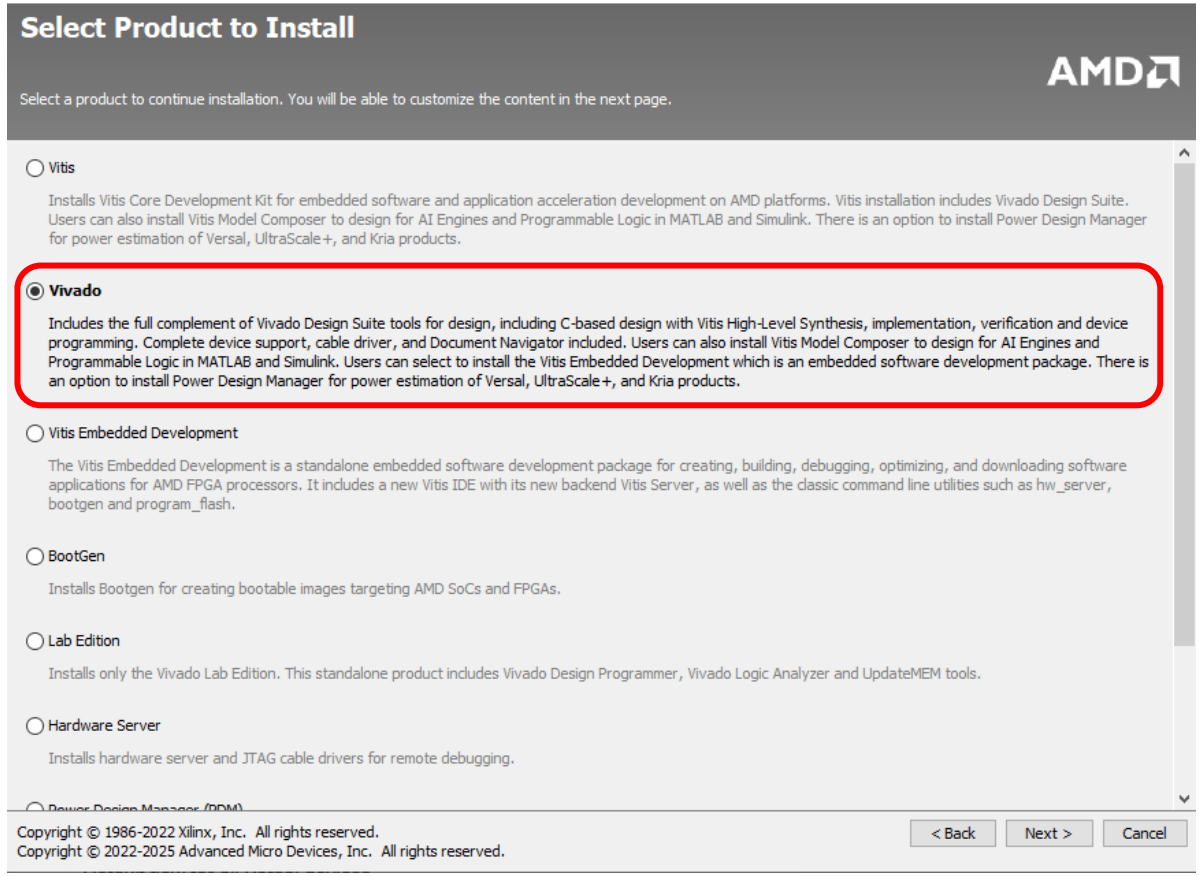
AMD (Xilinx) Vivado Kurulumu

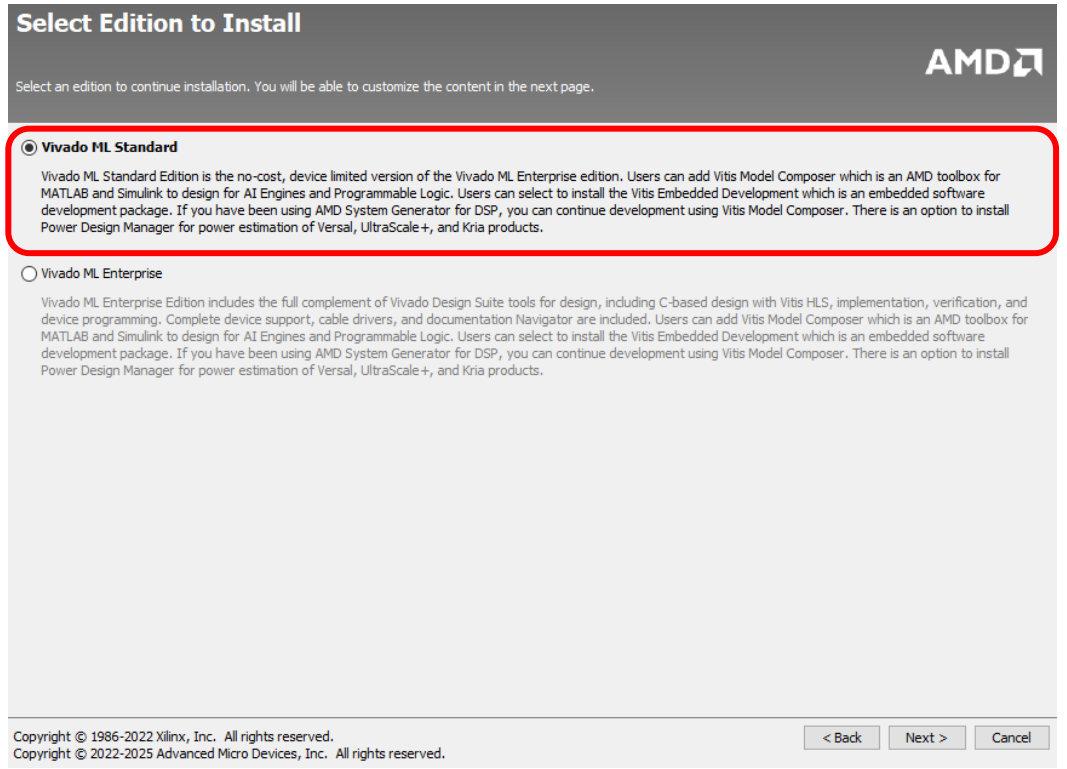




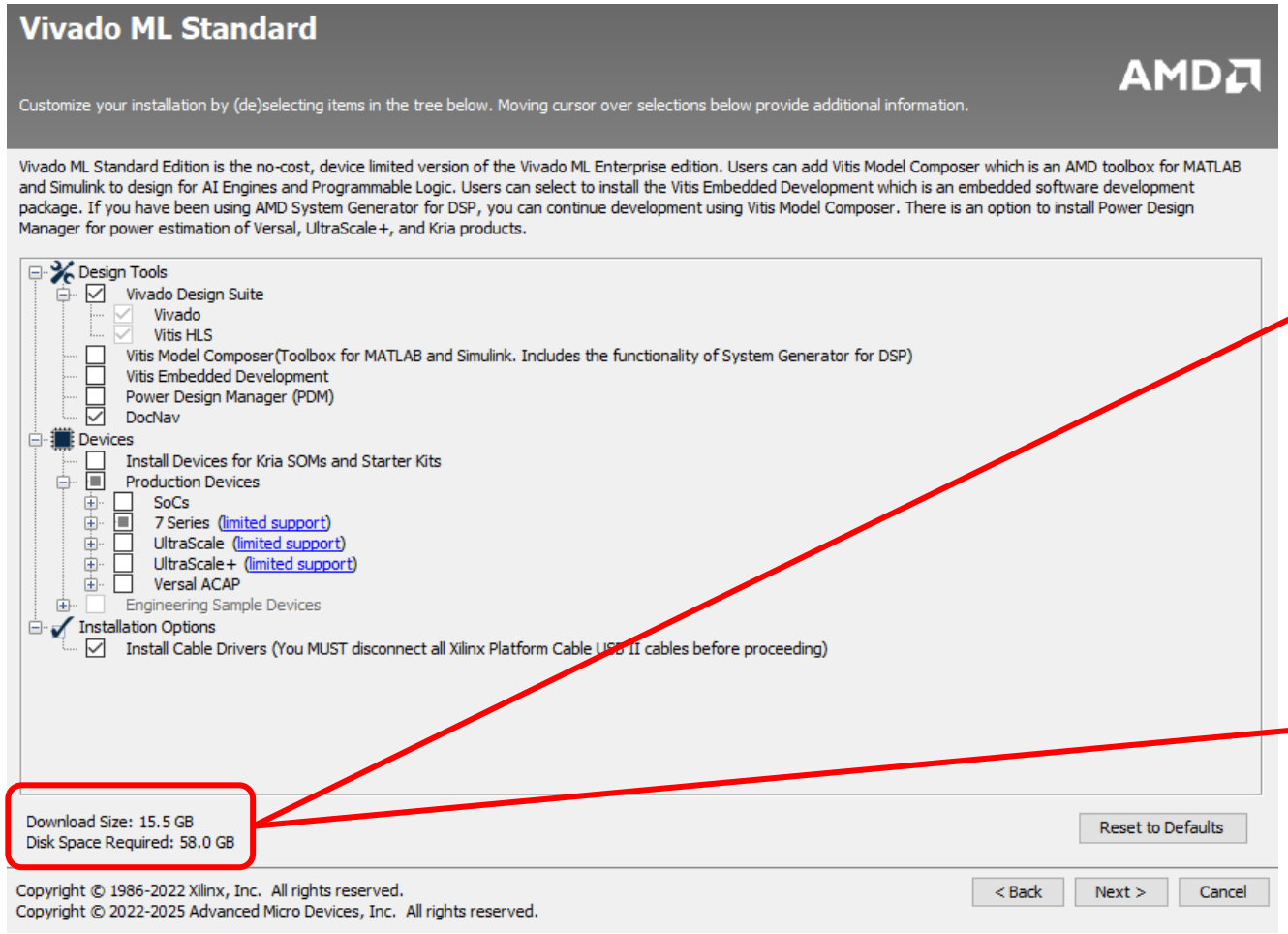
FIGES | MathWorks Authorized Reseller

AMD (Xilinx) Vivado Kurulumu





AMD (Xilinx) Vivado Kurulumu



Download Size: 15.5 GB
Disk Space Required: 58.0 GB

FİGES | MathWorks Authorized Reseller

Geleneksel Tasarım Metodolojileri

Yapısal (Kapı Seviyesi) Modelleme

Veri Akışı (Fonksiyonel) Modelleme

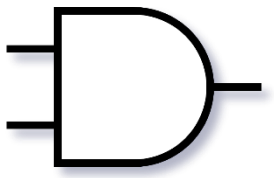
Davranışsal Modelleme

Soyutlama
(Abstraction)

Mantık Kapıları

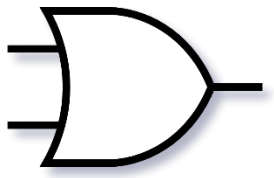
VE Kapısı (AND Gate)

A	B	Çıkış (Q)
0	0	0
0	1	0
1	0	0
1	1	1



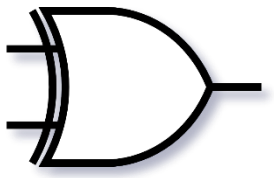
VEYA Kapısı (OR Gate)

A	B	Çıkış (Q)
0	0	0
0	1	1
1	0	1
1	1	1



ÖZEL-VEYA Kapısı (XOR Gate)

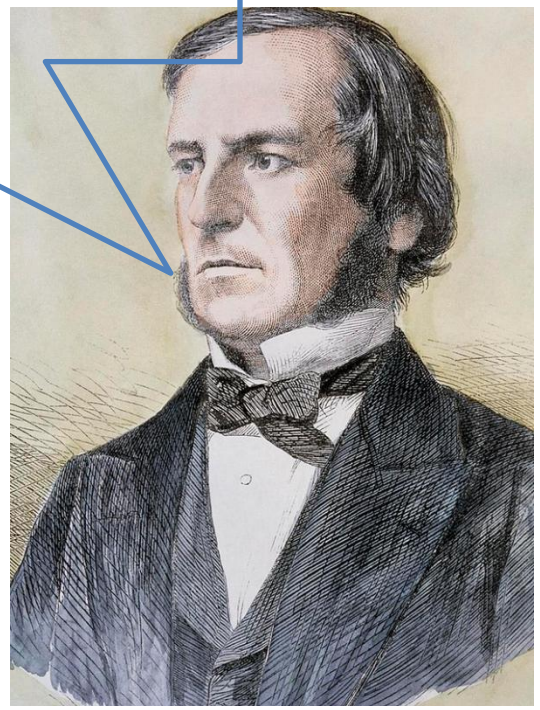
A	B	Çıkış (Q)
0	0	0
0	1	1
1	0	1
1	1	1



ÖRNEK : Geleneksel Tasarım Metodolojileri

A	B	C	Çıkış (Q)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

$$\text{Çıkış (Q)} = A'B'C + A'BC + AB'C' + ABC'$$



George Boole

ÖRNEK : Mantıksal ifadelerin sadeleştirilmesi

$$A'B'C + A'BC + AB'C' + ABC'$$

$$= A'C(B'+B)+AB'C'+ABC'$$

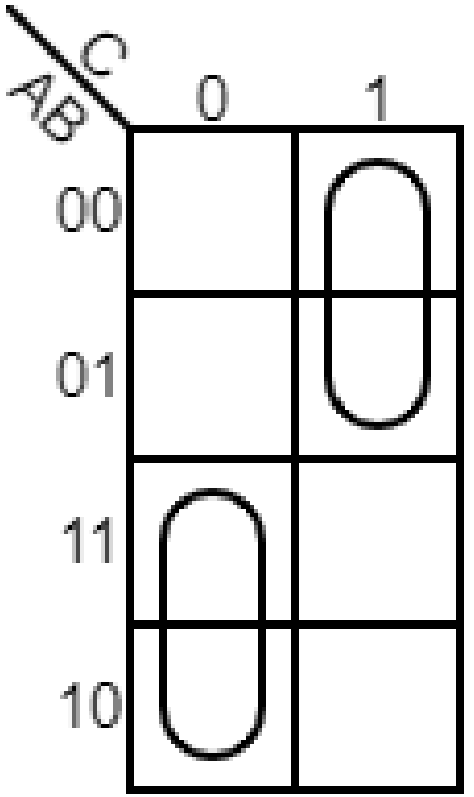
$$= A'C1+AB'C'+ABC'$$

$$= A'C+AB'C'+ABC'$$

$$= A'C+AC'(B'+B)$$

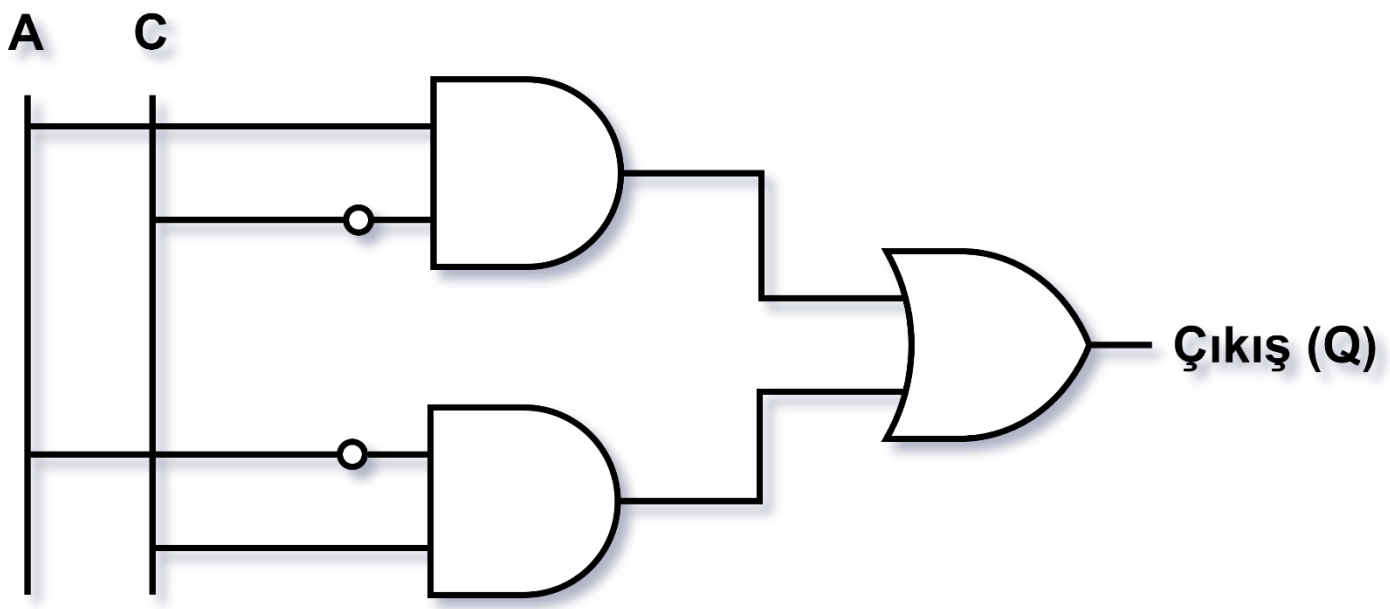
$$= A'C+AC'1$$

$$= A'C+AC'$$



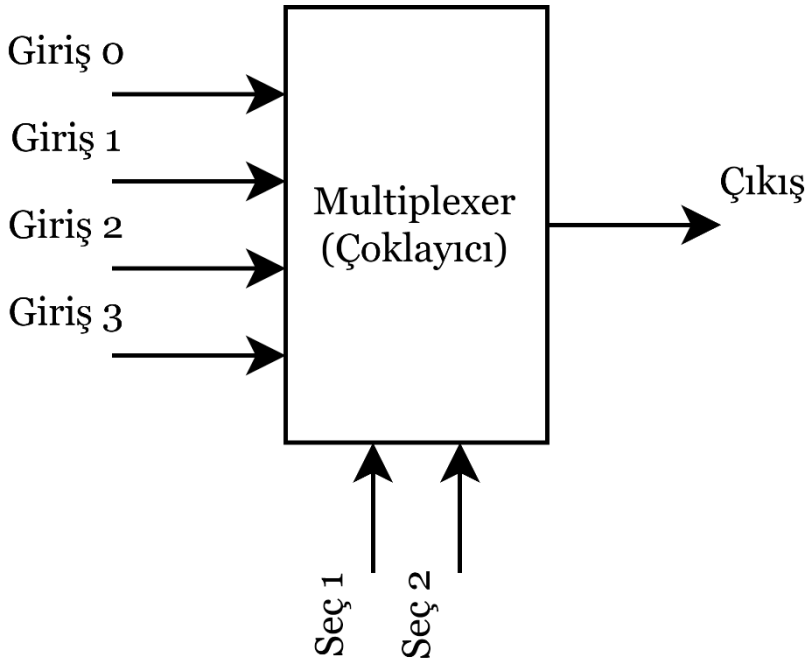
$$= A'C+AC'$$

ÖRNEK : Mantıksal ifadelerin sadeleştirilmesi



ÖRNEK : Geleneksel Tasarım Metodolojileri

- Özel bir giriş hattının seçilmesi bir grup seçme hattıyla kontrol edilir. 2^n giriş hattı ve hangi girişin seçileceğini belirleyen bit kombinasyonlarını oluşturan n adet seçme hattı vardır.



Seç1	Seç2	Çıkış
0	0	Giriş0
0	1	Giriş1
1	0	Giriş2
1	1	Giriş3

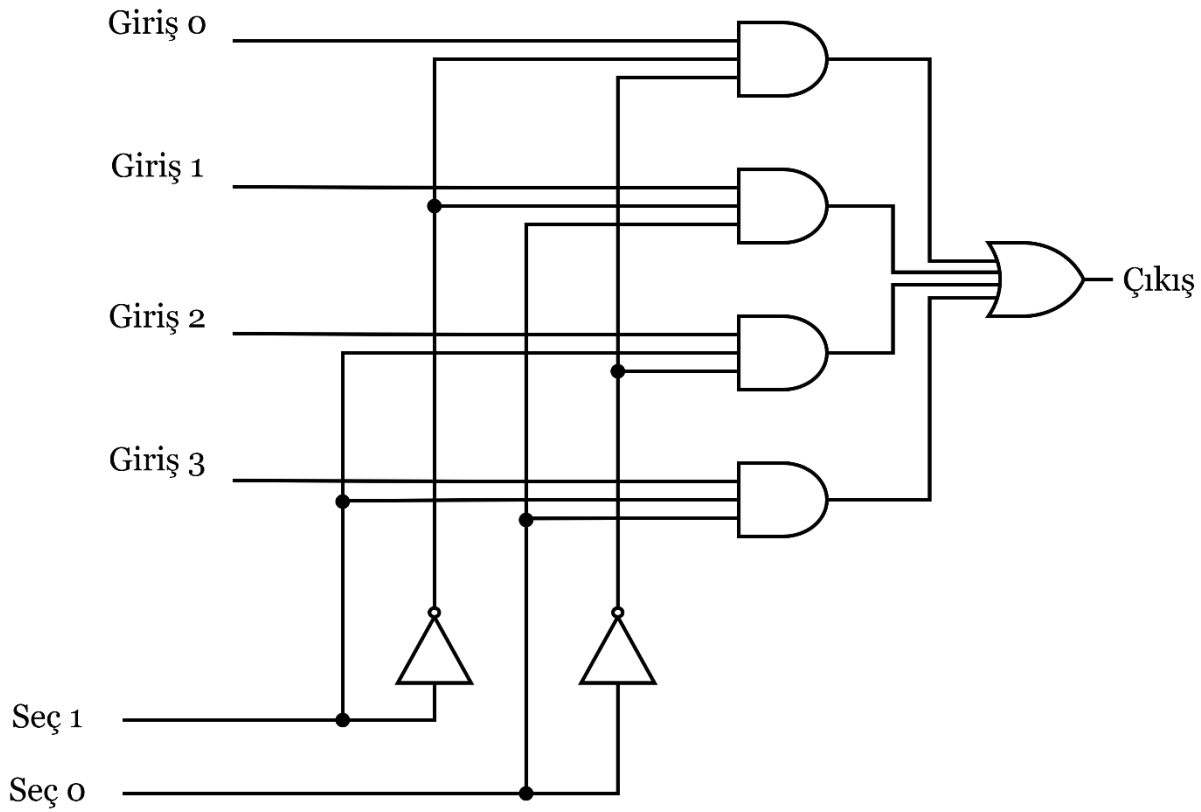
ÖRNEK : Geleneksel Tasarım Metodolojileri

S1	S0	Ç
0	0	Giriş0
0	1	Giriş1
1	0	Giriş2
1	1	Giriş3

Seç1	Seç0	Giriş0	Giriş1	Giriş2	Giriş3	Çıkış
0	0	0	x	X	X	0
0	0	1	x	X	X	1
0	1	X	0	X	X	0
0	1	X	1	X	X	1
1	0	X	X	0	X	0
1	0	X	X	1	X	1
1	1	X	X	X	0	0
1	1	X	X	X	1	1

$$Çıkış = S0'.S1'.G0 + S0.S1'.G1 + S0'.S1.G2 + S0.S1.G3$$

ÖRNEK : Geleneksel Tasarım Metodolojileri



Yapısal Modelleme (Structural Modeling)

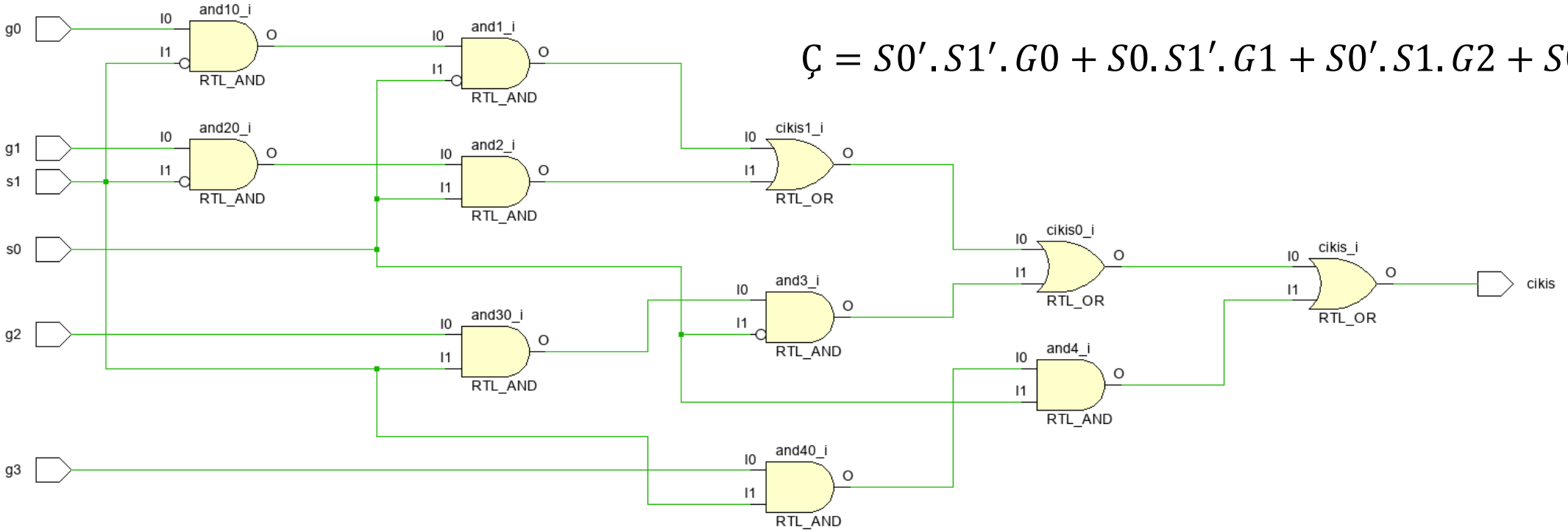
- Bu yöntemde kullanılacak her eleman bir yapı olarak tanımlanmış olmalıdır. Bu sebeple bu tanımlama yöntemine kapı seviyesinde modelleme (Gate-Level Modelling) de denmektedir.

```
1  `timescale 1ns / 1ps
2
3  module Structural(g0, g1, g2, g3, s0, s1, cikis);
4
5      input s0, s1;
6      input g0, g1, g2, g3;
7      output cikis;
8
9      wire not_s0, not_s1;
10     wire and1, and2, and3, and4;
11
12     not (not_s0, s0);
13     not (not_s1, s1);
14     and (and1, g0, not_s1, not_s0);
15     and (and2, g1, not_s1, s0);
16     and (and3, g2, s1, not_s0);
17     or  (cikis, and1, and2, and3, and4);
18     and (and4, g3, s1, s0);
19 endmodule
20
```

RTL Tasarım Çıktıları (Yapısal Modelleme)

- ***Register Transistor Level*** ifadesinin kısaltılmasıdır.
- Girişler ile Çıkışlar arasındaki ilişkiyi dijital sinyallerin akışı ve bu sinyaller üzerinde gerçekleştirilen mantıksal işlemleri modelleyen bir tasarım soyutlamasıdır.

$$Ç = S0'.S1'.G0 + S0.S1'.G1 + S0'.S1.G2 + S0.S1.G3$$



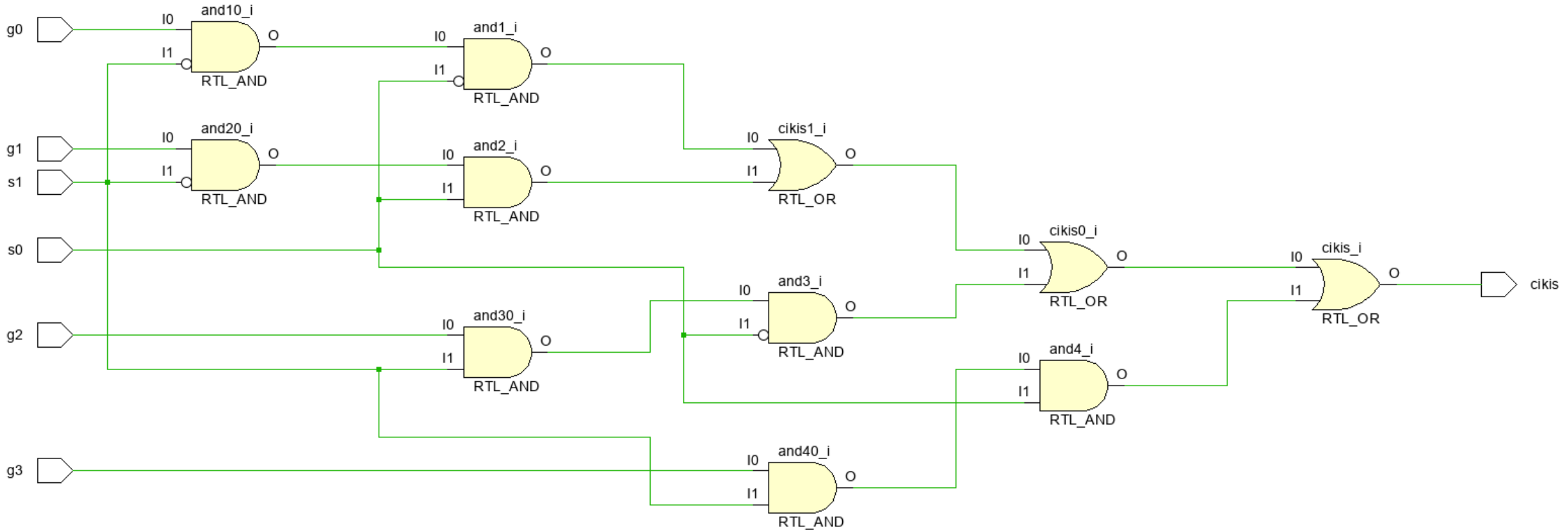
Veri Akışı Modelleme (Dataflow Modeling)

- Bu metotta giriş ve çıkış portları arasındaki ilişki bir fonksiyon olarak ifade edilir. Bu sebepten dolayı bu yönteme fonksiyonel modelleme (Functional Modelling) de denir.

```
1  `timescale 1ns / 1ps
2
3  module Dataflow(g0, g1, g2, g3, s0, s1, cikis);
4
5      input s0, s1;
6      input g0, g1, g2, g3;
7      output cikis;
8
9      assign
    cikis = (~s0 & ~s1 & g0) | (~s0 & s1 & g1) | (s0 & ~s1 & g2) | (s0 & s1 & g
   3);
10
11 endmodule
```

RTL Tasarım Çıktıları (Veri Akışı Modelleme)

$$Ç = S0'.S1'.G0 + S0.S1'.G1 + S0'.S1.G2 + S0.S1.G3$$



Davranışsal Modelleme (Behaviroal Modelling)

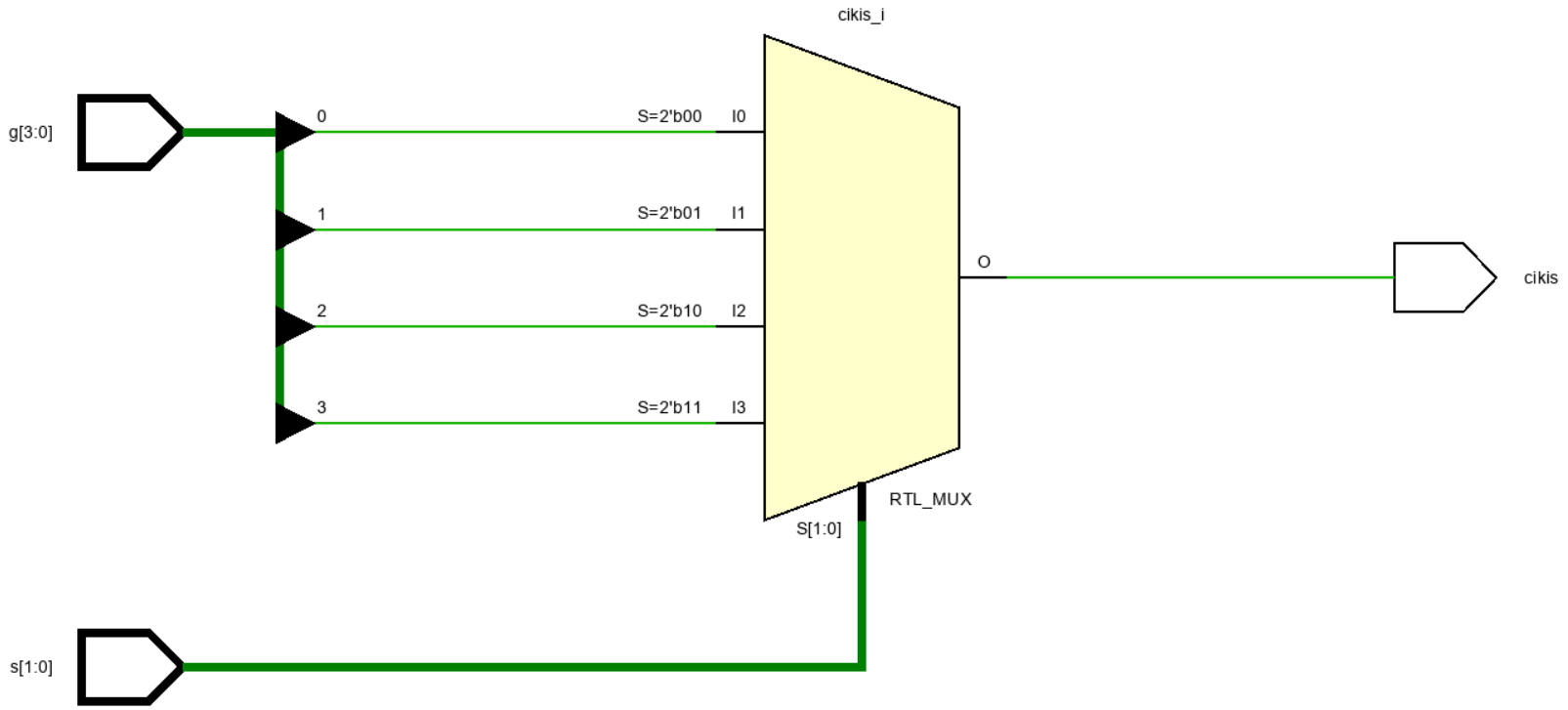
- Tasarlanması istenen sayısal sistem davranışı ile ifade edilir. Yani, koşullu ve özyinemeli ifadelere karşılık gelen HDL anahtar kelimeleri kullanılabilir.

```
1  `timescale 1ns / 1ps
2
3  module Behavioral(g,s,cikis);
4
5      input [1:0] s;
6      input [3:0] g;
7      output reg cikis;
8
9      always @(s,g,cikis) begin
10         case (s)
11             2'b00 : cikis <= g[0];
12             2'b01 : cikis <= g[1];
13             2'b10 : cikis <= g[2];
14             2'b11 : cikis <= g[3];
15         endcase
16     end
17
18 endmodule
```

Klasik programlama dillerindeki döngü yapısına benzer.

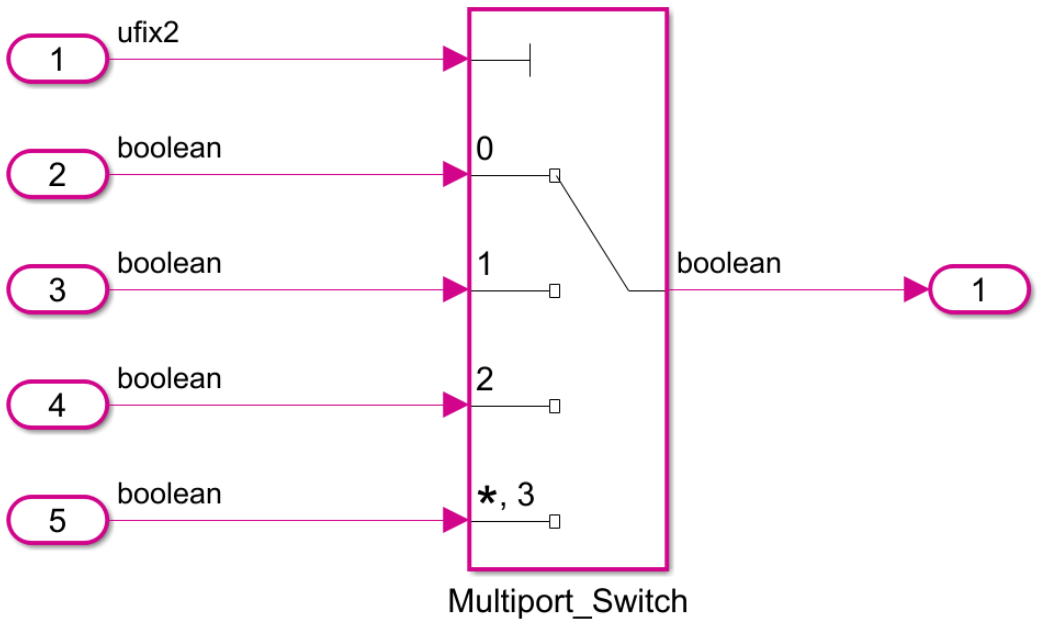
Klasik programlama dillerindeki şart yapıları gibi.

RTL Tasarım Çıktıları (Behaviroal Modelling)



Model Tabanlı Tasarım ile MUX Tasarımı

- 4 Girişli MUX Tasarımı

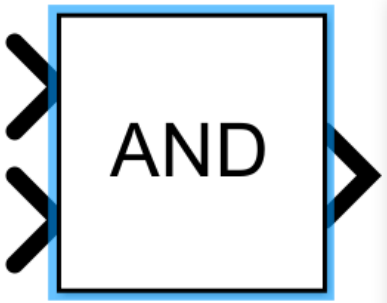



```
38  input   [1:0] In1;  // ufix2
39  input   In2;
40  input   In3;
41  input   In4;
42  input   In5;
43  output  Out1;
44
45
46  wire Multiport_Switch_out1;
47
48
49  assign Multiport_Switch_out1 = (In1 == 2'b00 ? In2 :
50              (In1 == 2'b01 ? In3 :
51              (In1 == 2'b10 ? In4 :
52              In5)));
53
54  assign Out1 = Multiport_Switch_out1;
```

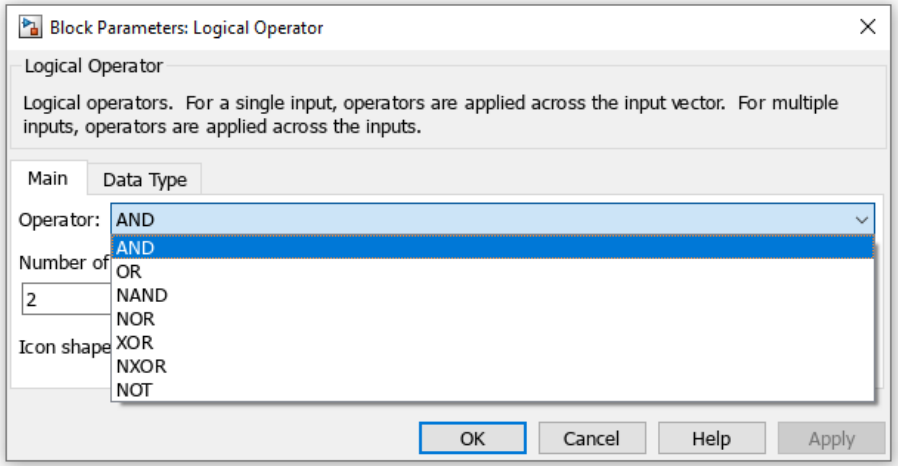
Lisans Seviyesi Sayısal Tasarım Kazanımları

- **Sayı Sistemleri**
- **Mantık Kapıları**
- **Boolean Cebri**
- **Combinational Devreler**
  - Mux
  - Encoder
  - Decoder
  - Comparators
  - Adder
- **Sequential Devreler**
  - Counter
  - Shift Register
  - FSM
    - Mealy Moore
  - Sequence Detector
  - UART
- **Hafıza Birimi Tasarımı**
  - Register
  - RAM
  - FIFO

Model Tabanlı Mantık Tasarımı



Logical Operator

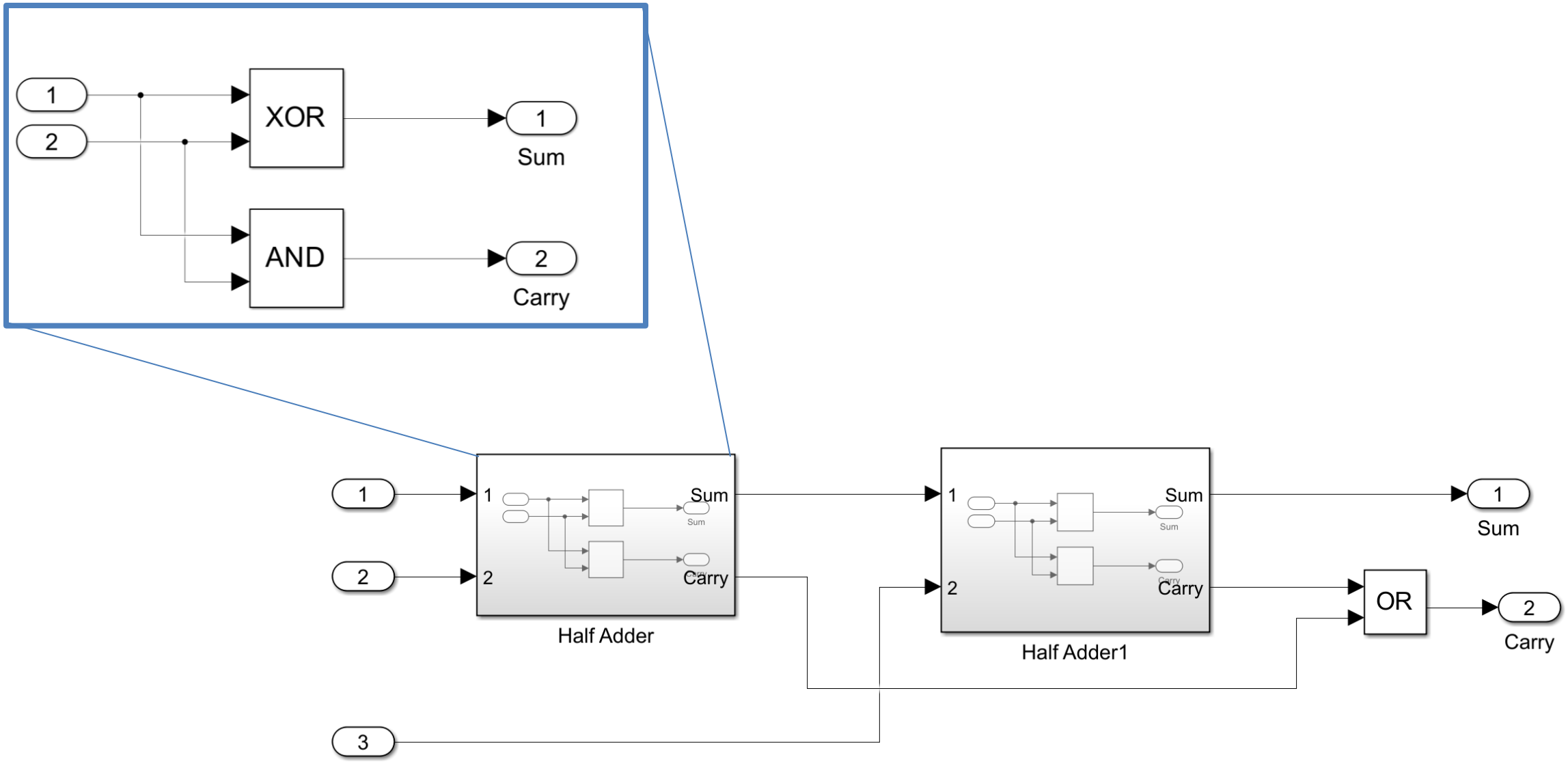


Logical Operator

Relational Operator

Bitwise Operator

Model Tabanlı Full Adder Tasarımı

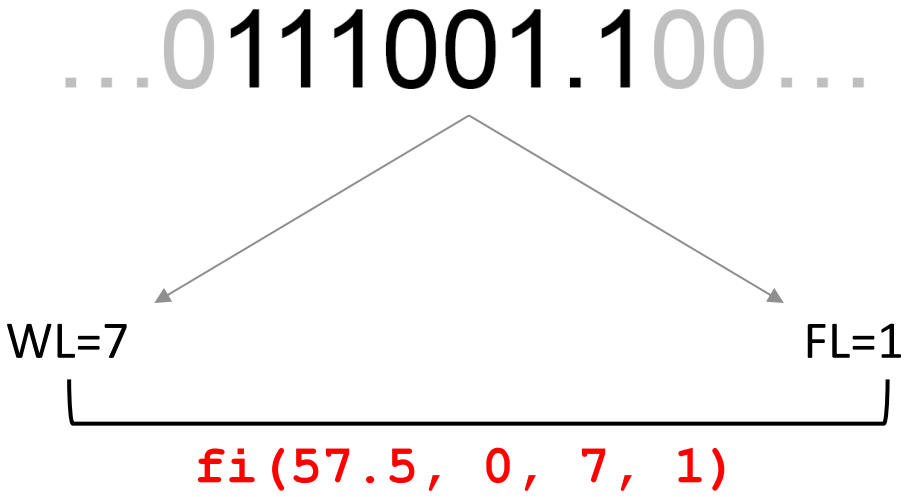


İkili Sayı Sistemi

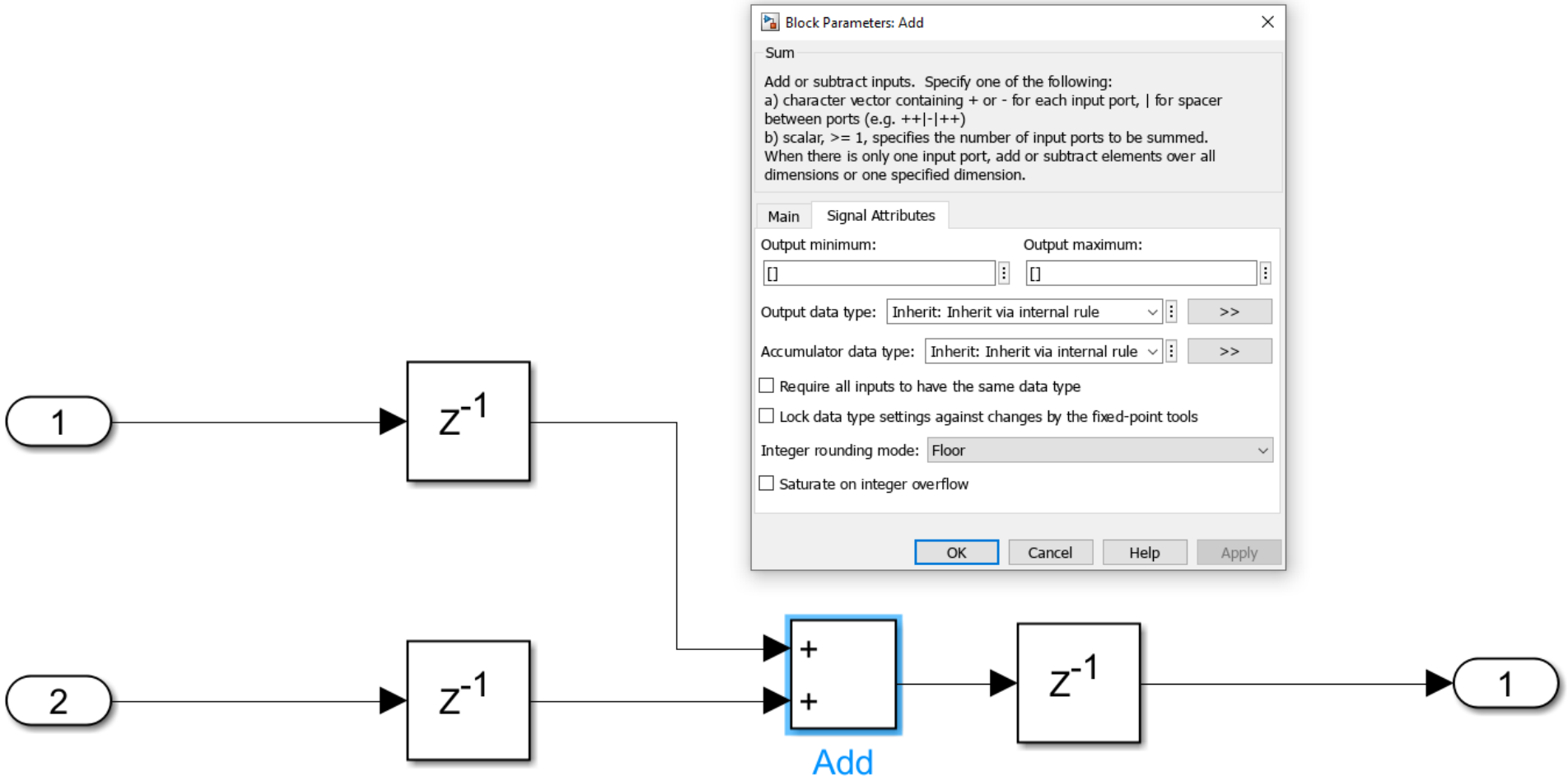
- Create fixed-point objects using the `fi` function:

```
fi(value, signedness, wordLength, fractionLength)
```

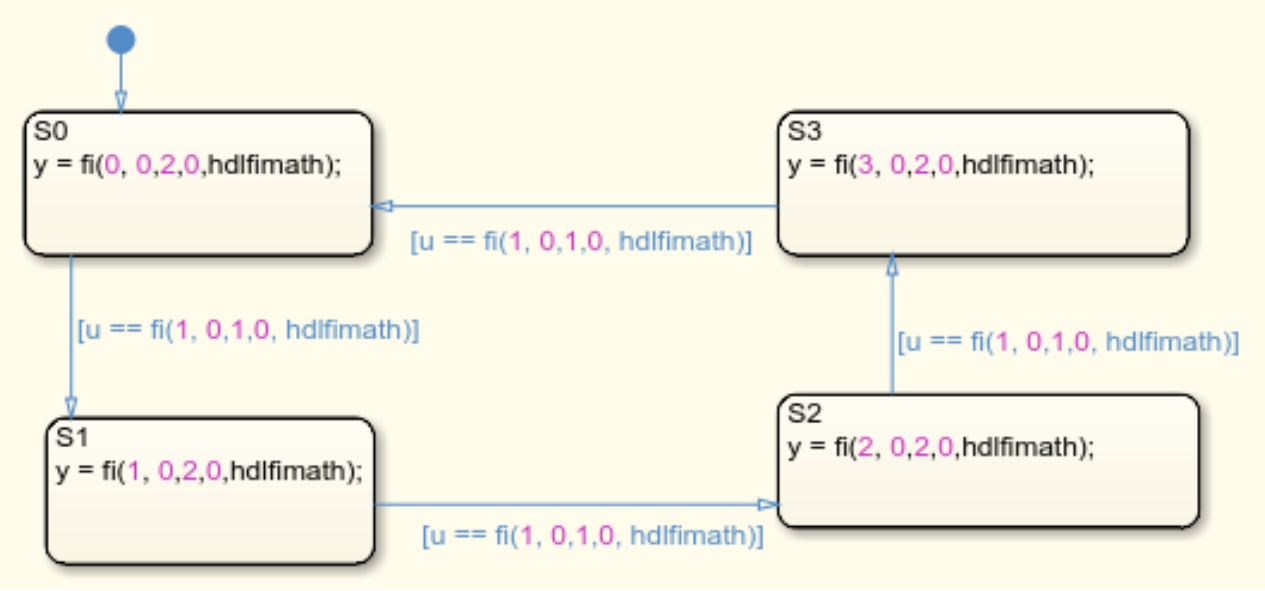
- Example



Model Tabanlı Full Adder Tasarımı



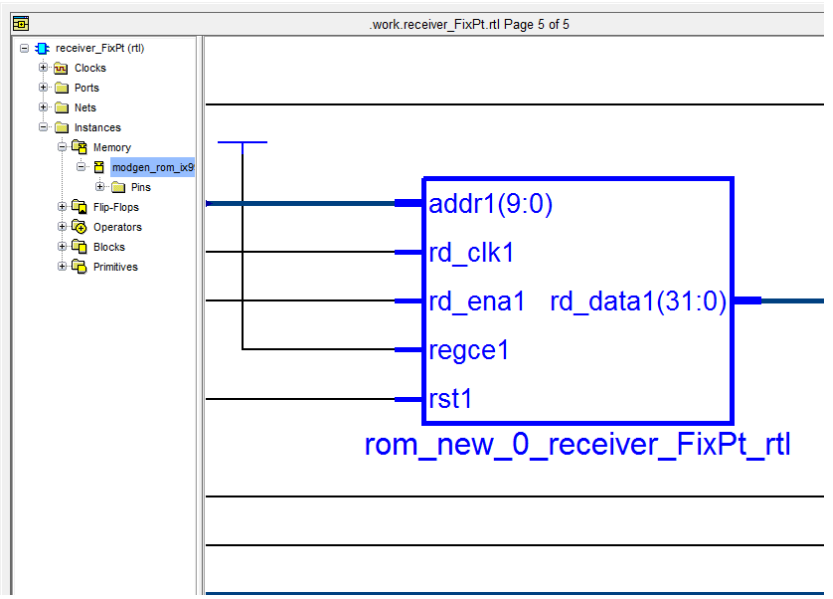
Ardışıl (Sequential) Devreler için Stateflow

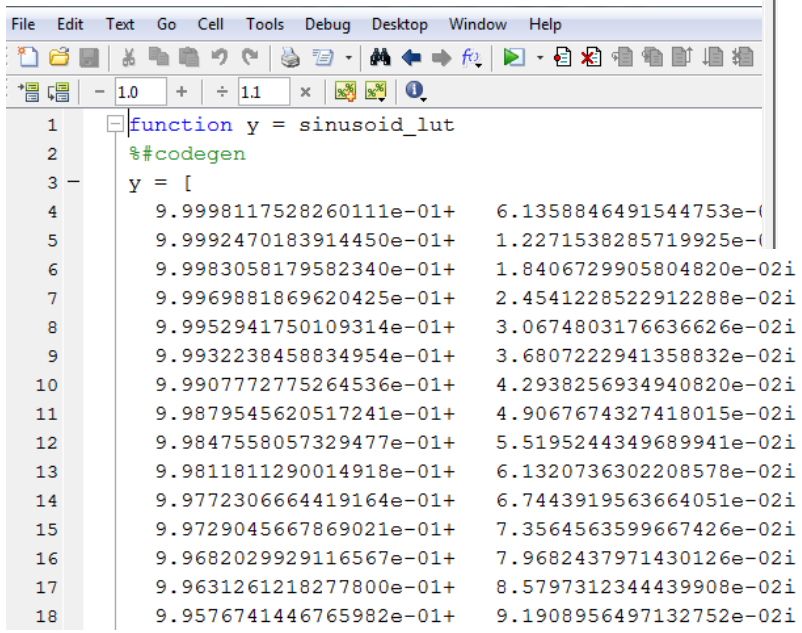



```
always @(posedge clk or posedge reset)
  begin : Mealy_Chart_1_process
    if (reset == 1'b1) begin
      is_Mealy_Chart <= is_Mealy_Chart_IN_S0;
    end
    else begin
      if (enb) begin
        is_Mealy_Chart <= is_Mealy_Chart_next;
      end
    end
  end

always @(is_Mealy_Chart, u) begin
  is_Mealy_Chart_next = is_Mealy_Chart;
  y_1 = 2'b00;
  case ( is_Mealy_Chart)
    is_Mealy_Chart_IN_S0 :
      begin
        if (u == 8'sb00000001) begin
          y_1 = 2'b00;
          is_Mealy_Chart_next = is_Mealy_Chart_IN_S1;
        end
      end
    is_Mealy_Chart_IN_S1 :
      begin
        if (u == 8'sb00000001) begin
          y_1 = 2'b01;
          is_Mealy_Chart_next = is_Mealy_Chart_IN_S2;
        end
      end
    is_Mealy_Chart_IN_S2 :
      begin
        if (u == 8'sb00000001) begin
          y_1 = 2'b10;
          is_Mealy_Chart_next = is_Mealy_Chart_IN_S3;
        end
      end
    default :
      begin
        if (u == 8'sb00000001) begin
          y_1 = 2'b11;
          is_Mealy_Chart_next = is_Mealy_Chart_IN_S0;
        end
      end
  endcase
end

assign y = y_1;
```

Donanımda Veri Tipleri

- Arrays can be represented as:
  - Wires
  - Flip-Flops
  - RAM
  - ROM

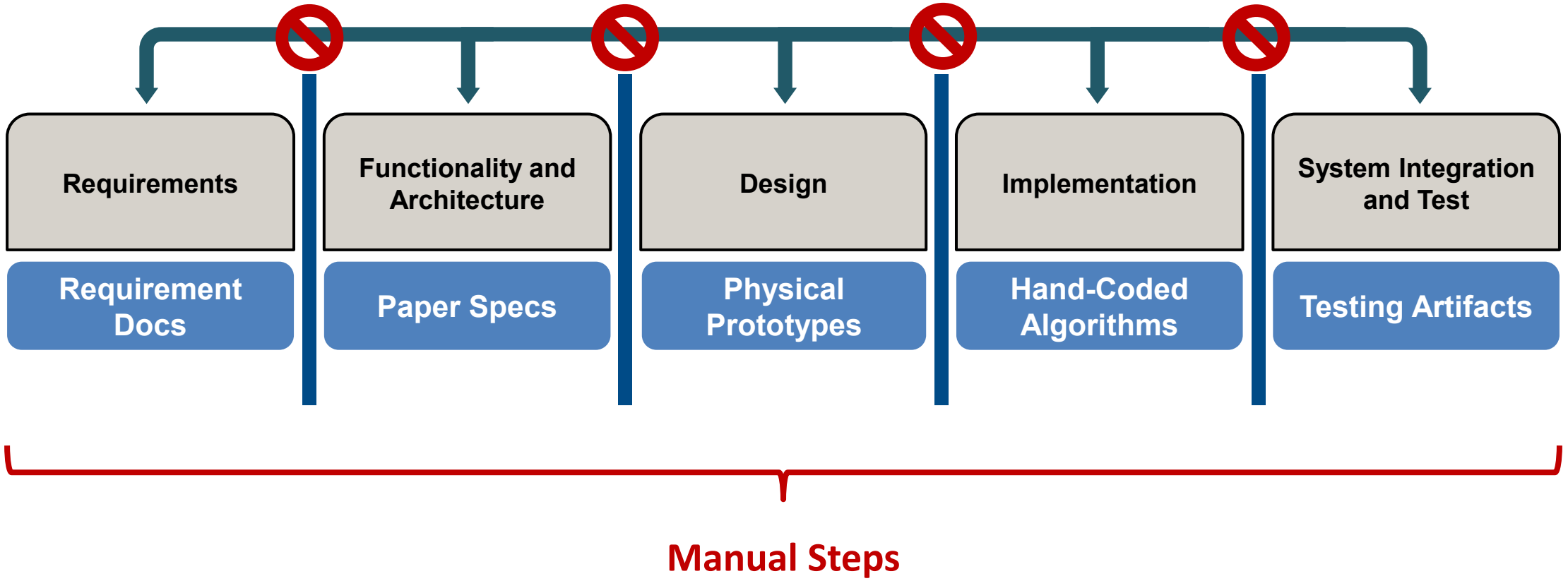




Hardware knowledge

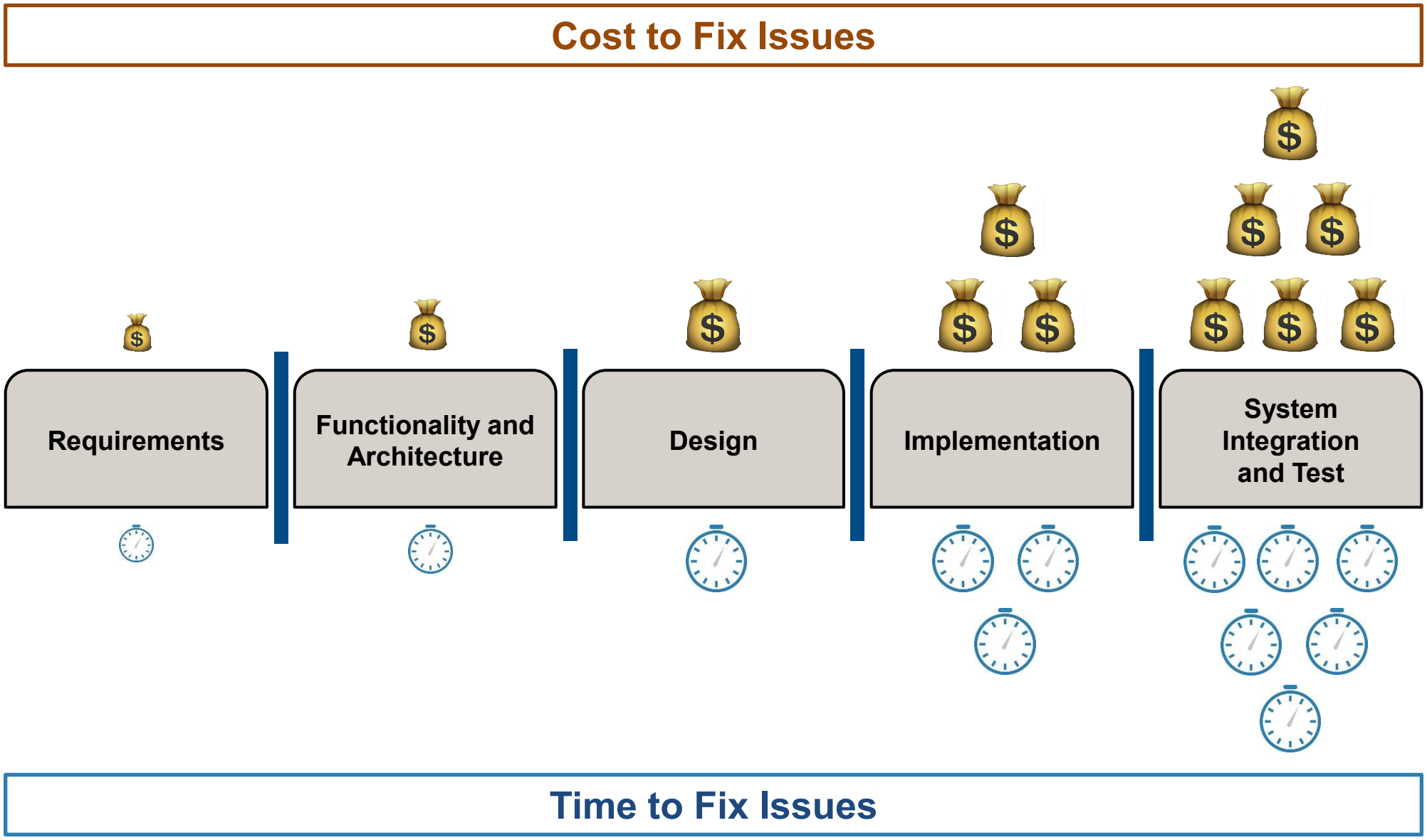
Think about how arrays are used in the design to get the best results

Traditional Development Process

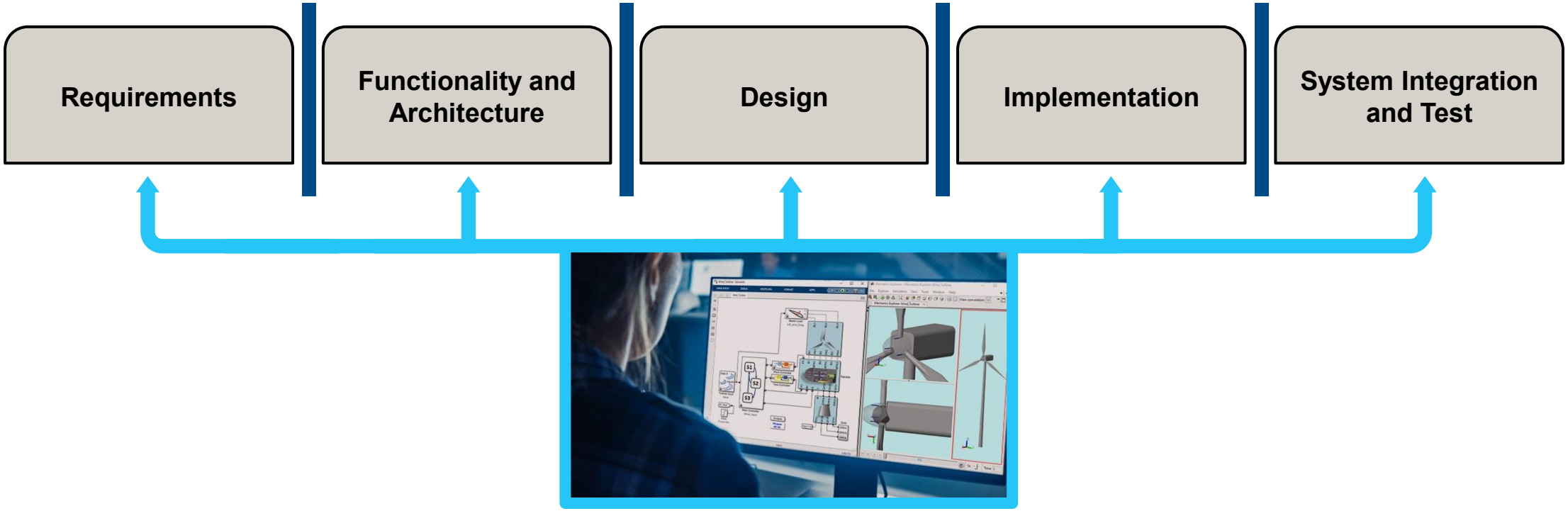


Manual steps introduce **errors** and **slow down** the development process

Issues found late in the process are more **costly** and **time-consuming** to fix



Model-Based Design

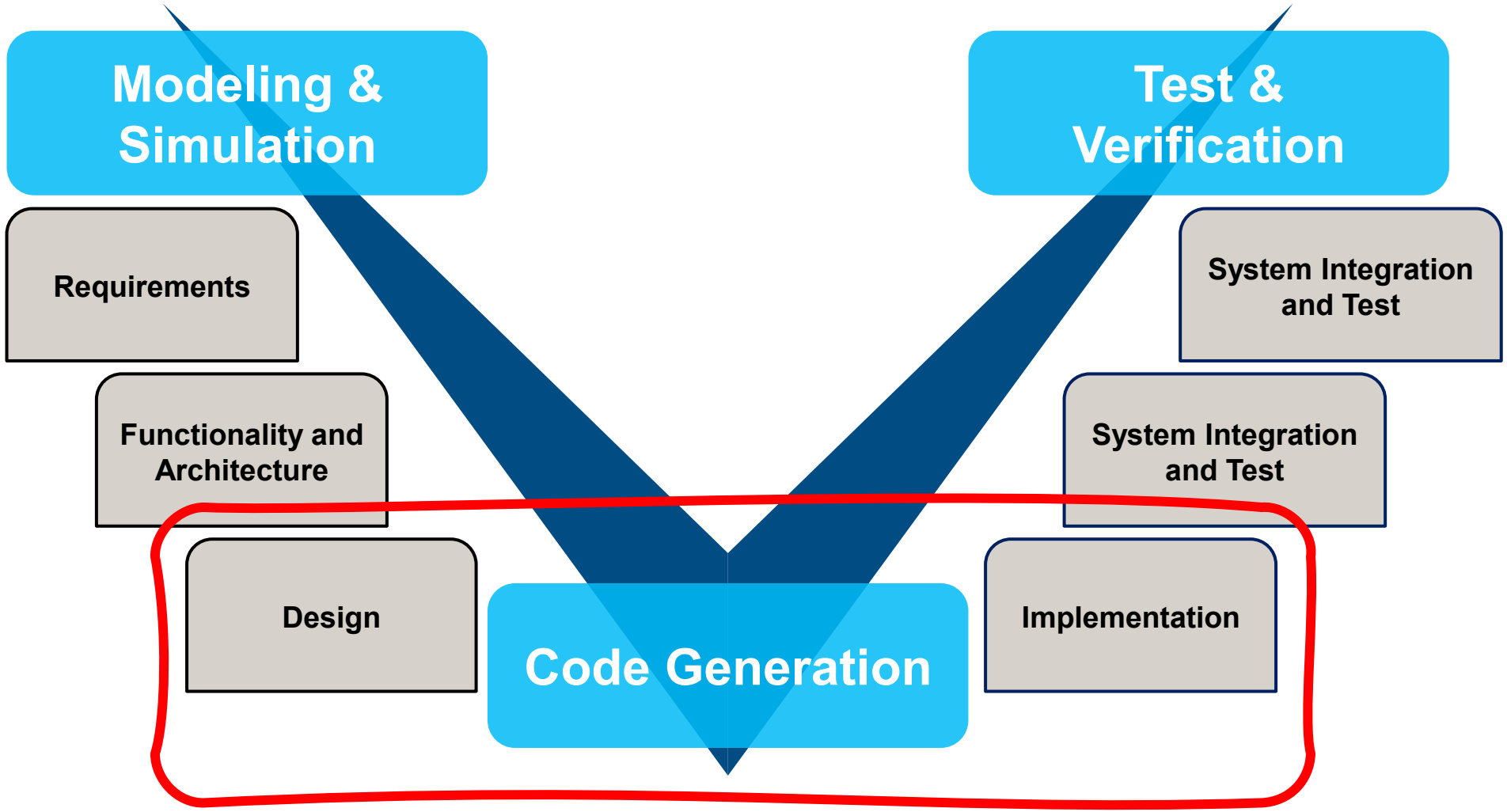


Models are at the **center** of your development process

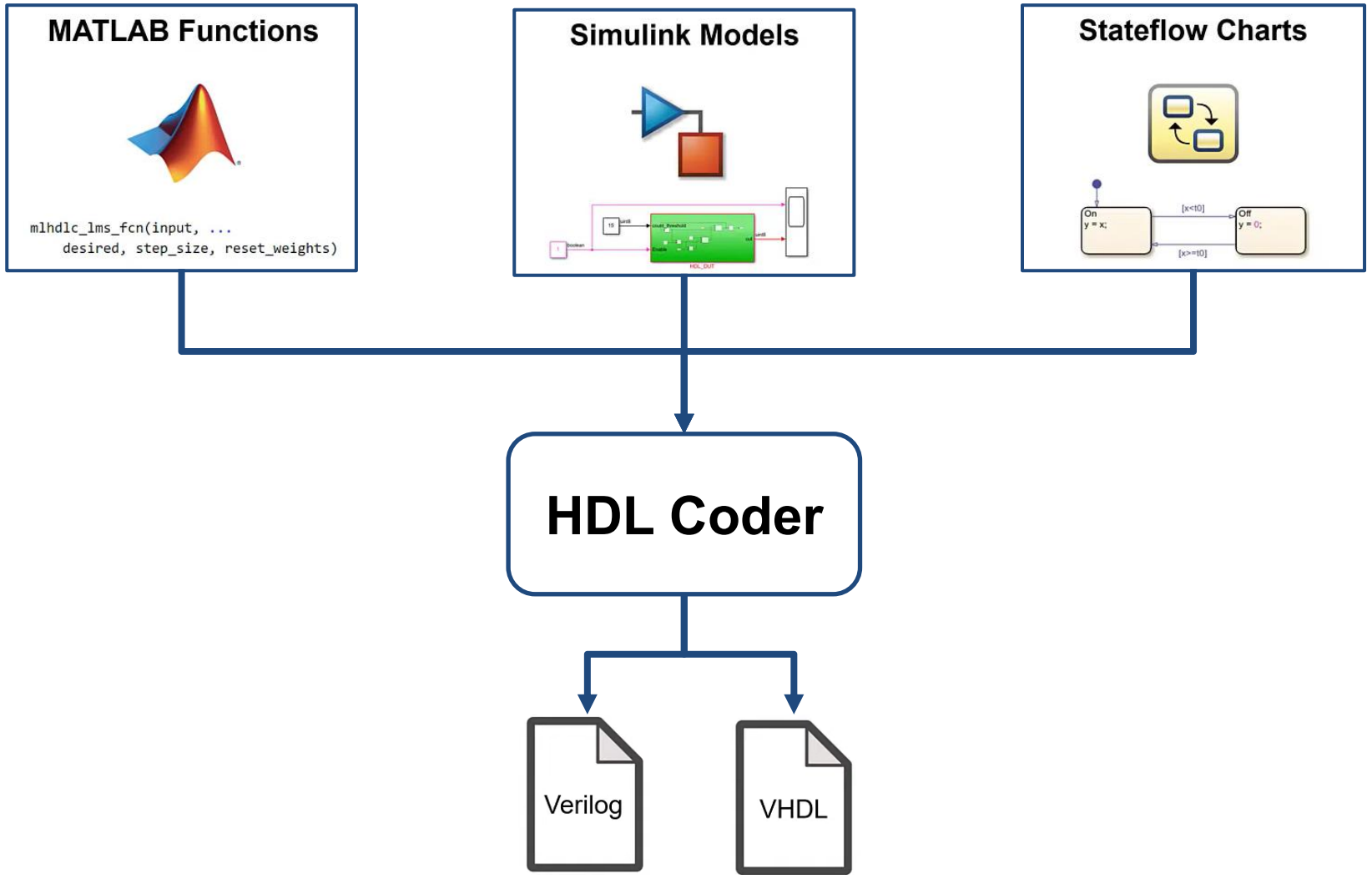
Requirements **capture** and artifact **traceability** throughout the process

V Model

Three key pieces to Model-Based Design

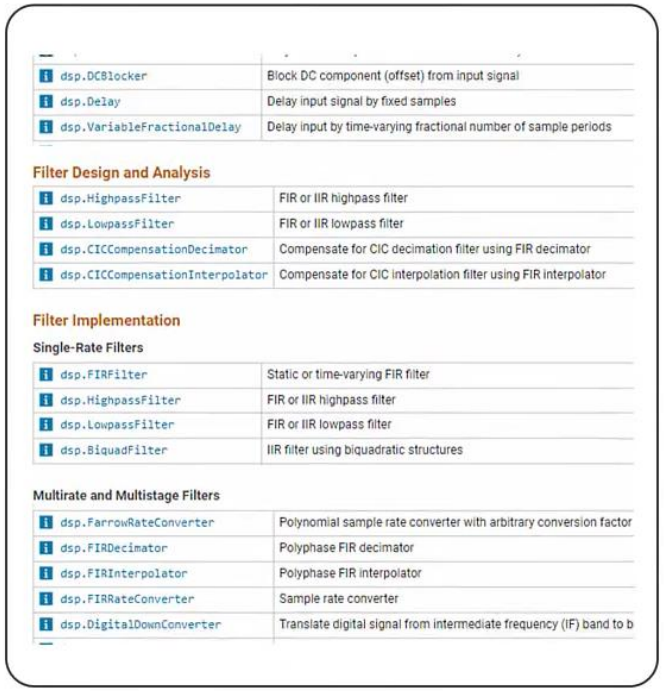


HDL Coder Özet



HDL Coder Uyumluluğu

200+ MATLAB Functions



dsp.DCBlocker	Block DC component (offset) from input signal
dsp.Delay	Delay input signal by fixed samples
dsp.VariableFractionalDelay	Delay input by time-varying fractional number of sample periods

Filter Design and Analysis

dsp.HighpassFilter	FIR or IIR highpass filter
dsp.LowpassFilter	FIR or IIR lowpass filter
dsp.CICCompensationDecimator	Compensate for CIC decimation filter using FIR decimator
dsp.CICCompensationInterpolator	Compensate for CIC interpolation filter using FIR interpolator

Filter Implementation

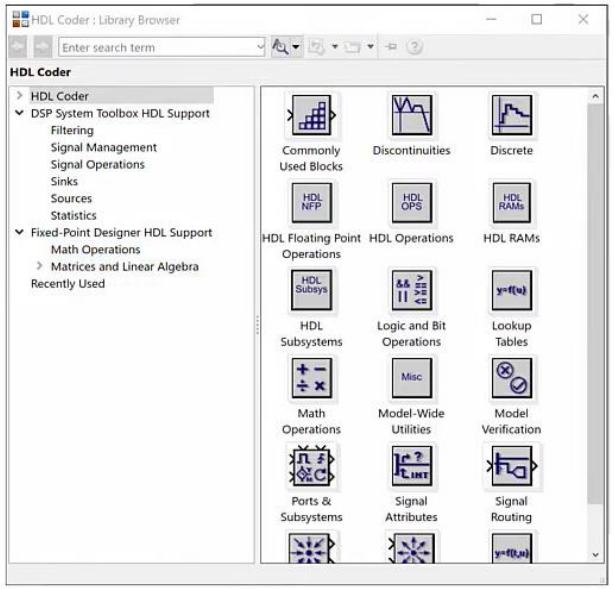
Single-Rate Filters

dsp.FIRFilter	Static or time-varying FIR filter
dsp.HighpassFilter	FIR or IIR highpass filter
dsp.LowpassFilter	FIR or IIR lowpass filter
dsp.BiquadFilter	IIR filter using biquadratic structures

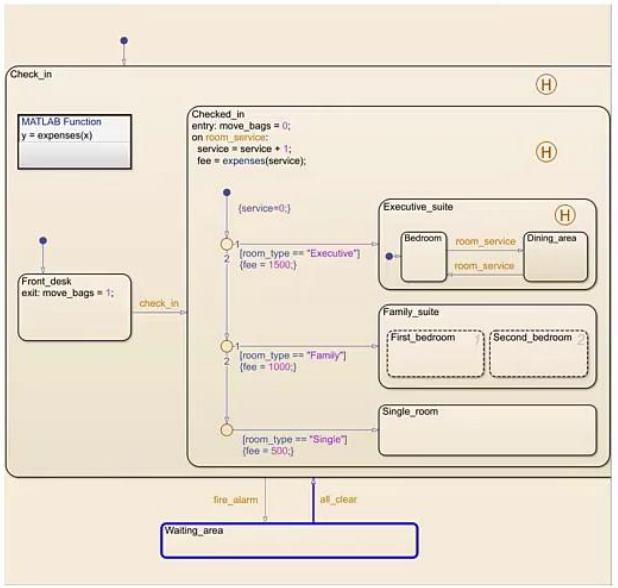
Multirate and Multistage Filters

dsp.FarrowRateConverter	Polynomial sample rate converter with arbitrary conversion factor
dsp.FIRDecimator	Polyphase FIR decimator
dsp.FIRInterpolator	Polyphase FIR interpolator
dsp.FIRRateConverter	Sample rate converter
dsp.DigitalDownConverter	Translate digital signal from intermediate frequency (IF) band to b

300+ Simulink Blocks

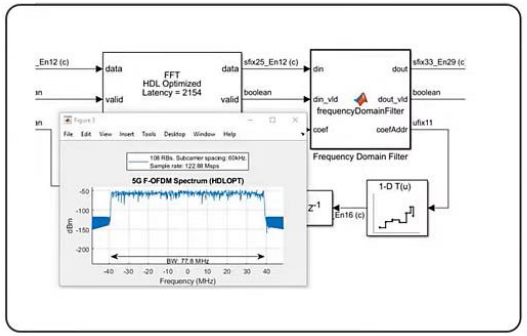


Stateflow Charts



Model Tabanlı Full Adder Tasarımı

Wireless Communications



DSP

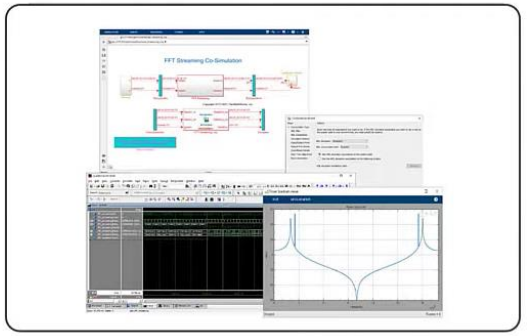
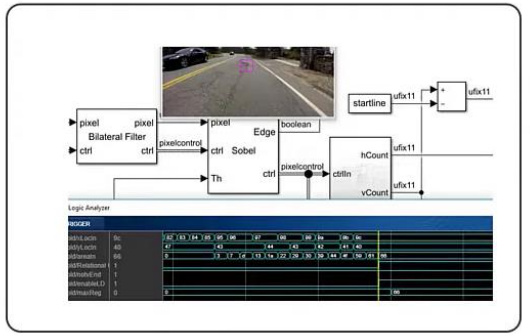
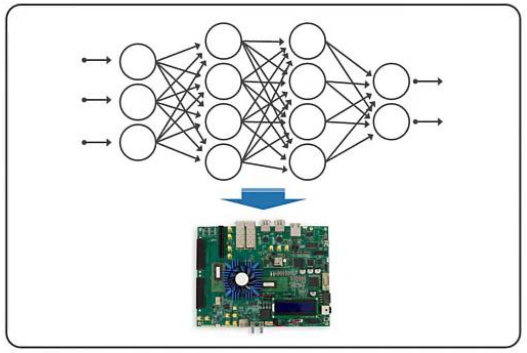


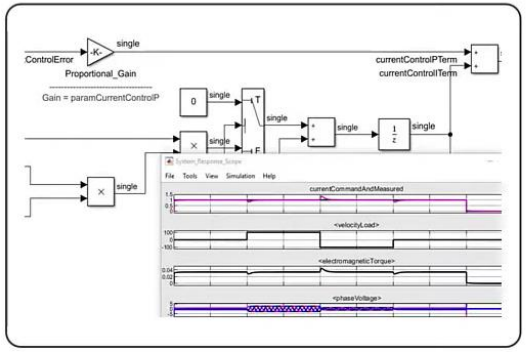
Image and Video Processing



Deep Learning Networks

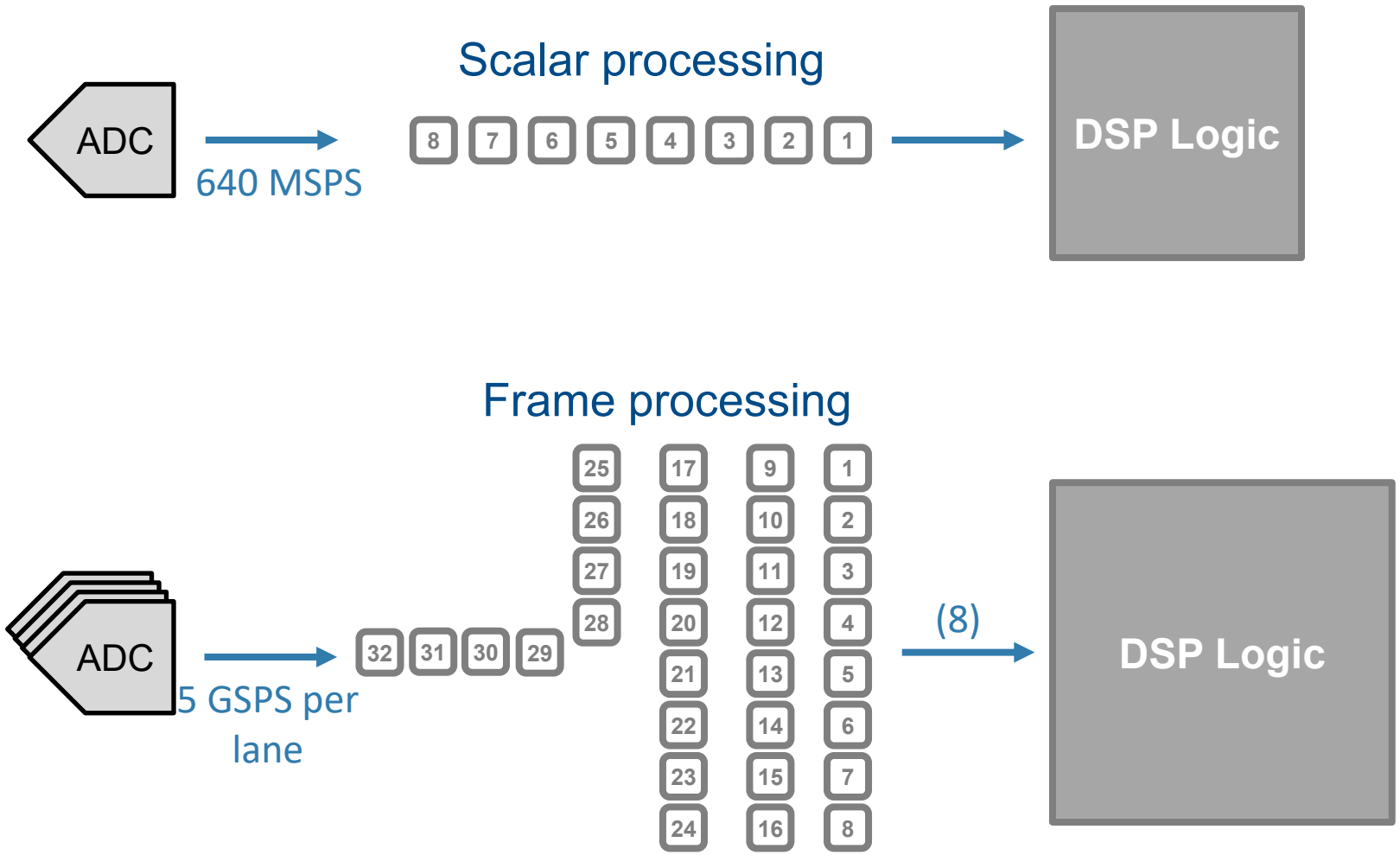


Motor and Power Control

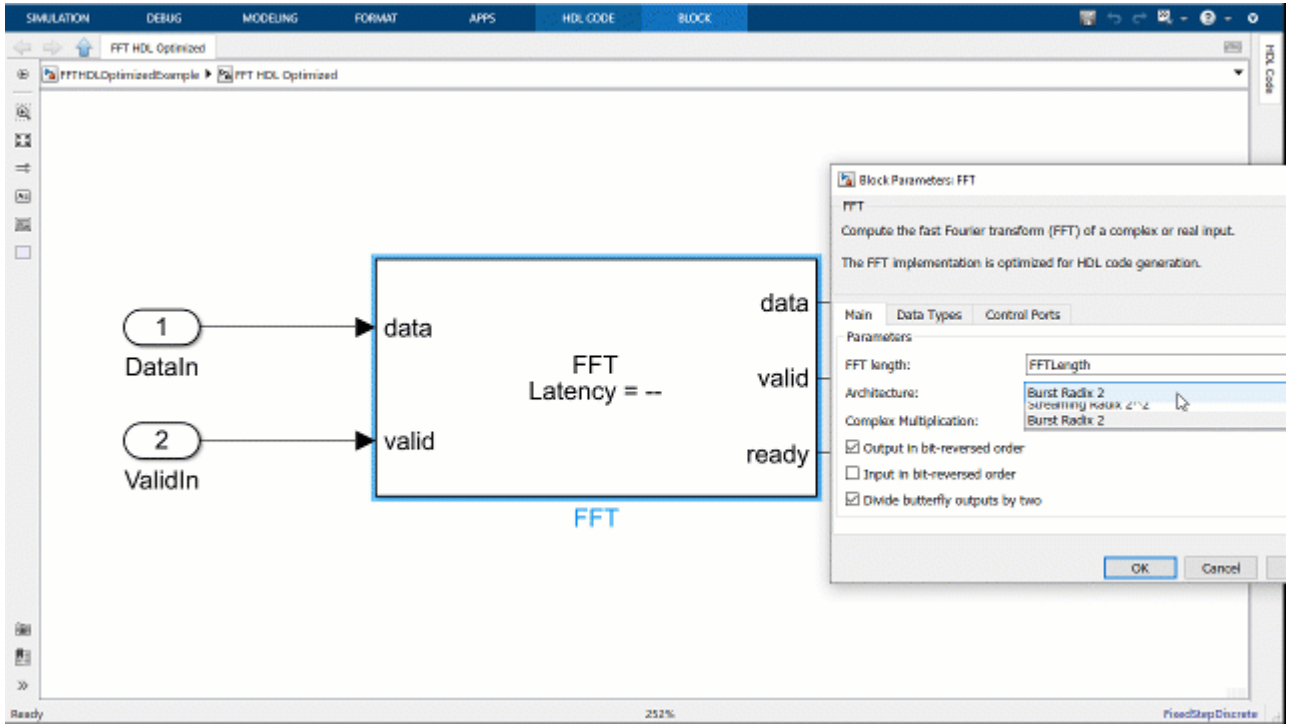


Frame-Based Processing

Parallel implementations for gigasample-per-second throughput



FFT Implementation Exploration



Minimize resources`

- Burst Radix 2
- Latency = 2721 cycles
- Fmax = 672 MHz

Resource summary

Resource	Usage	Available	Utilization (%)
CLB LUTs	793	274080	0.29
CLB Registers	1229	548160	0.22
DSPs	4	2520	0.16
Block RAM Tile	3	912	0.33

Low latency

- Streaming Radix 2^2
- Latency = 599 cycles
- Fmax = 646 MHz

Resource summary

Resource	Usage	Available	Utilization (%)
CLB LUTs	3161	274080	1.15
CLB Registers	4917	548160	0.90
DSPs	16	2520	0.63
Block RAM Tile	1	912	0.11

High throughput

- Streaming Radix 2^2
- Latency = 139 cycles
- Fmax = 469 MHz
- Vector input size = 8

3.75 GSPS

Resource summary

Resource	Usage	Available	Utilization (%)
CLB LUTs	15968	274080	5.83
CLB Registers	28026	548160	5.11
DSPs	108	2520	4.29
Block RAM Tile	1.500000e+00	912	0.16

Implementation metrics are post-synthesis, targeting Xilinx Zynq® UltraScale+ speed grade -2

Desteklenen Fonksiyonlar ve Bloklar

- You can generate efficient code for a subset of **MATLAB Built-in Functions** and **Toolbox Functions** that you can call from MATLAB code. [Function List]

hdl.interpn

R2025b

Find interpolated value based on n-dimensional input data

Since R2024b

collapse all in page

Syntax

```
Vq = hdl.interpn(X1,X2,...,Xn,V,Xq1,Xq2,...,Xqn)
Vq = hdl.interpn(X1,X2,...,Xn,V,Xq1,Xq2,...,Xqn,interpolation,extrapolation)
```

Description

`Vq = hdl.interpn(X1,X2,...,Xn,V,Xq1,Xq2,...,Xqn)` returns the interpolated values of a function of `n` variables at a specific query point using linear interpolation and linear extrapolation. `X1,X2,...,Xn` contain the coordinates of the sample points in each dimension. `V` contains the corresponding function values at each sample point. `Xq1,Xq2,...,Xqn` contain the coordinates of the query point. example

`Vq = hdl.interpn(X1,X2,...,Xn,V,Xq1,Xq2,...,Xqn,interpolation,extrapolation)` also specifies `interpolation` and `extrapolation` approximation methods. example

Extended Capabilities

collapse all

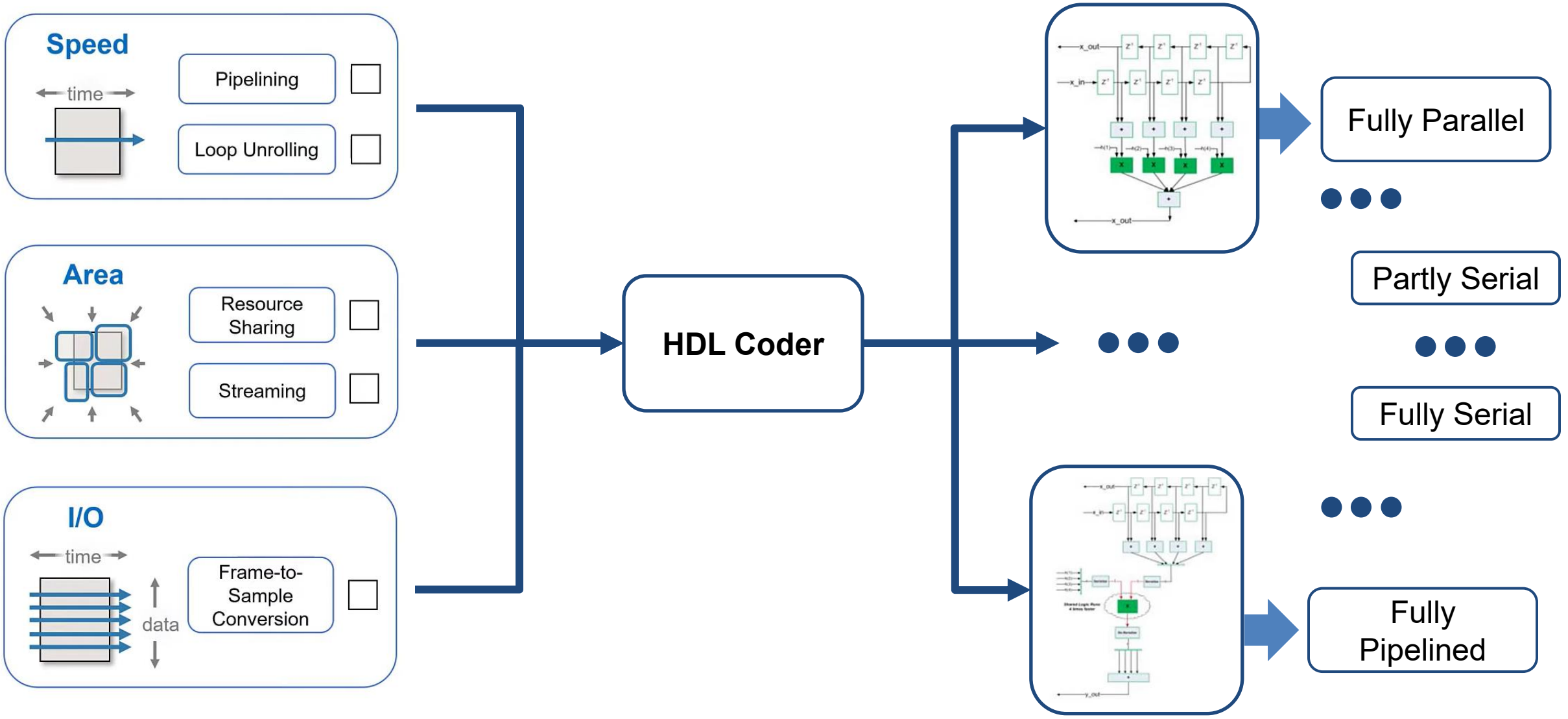
HDL Code Generation
Generate VHDL, Verilog and SystemVerilog code for FPGA and ASIC designs using HDL Coder™.

The approximation methods supported for code generation are `"linear"` for `interpolation`, and `"linear"` and `"nearest"` for `extrapolation`.

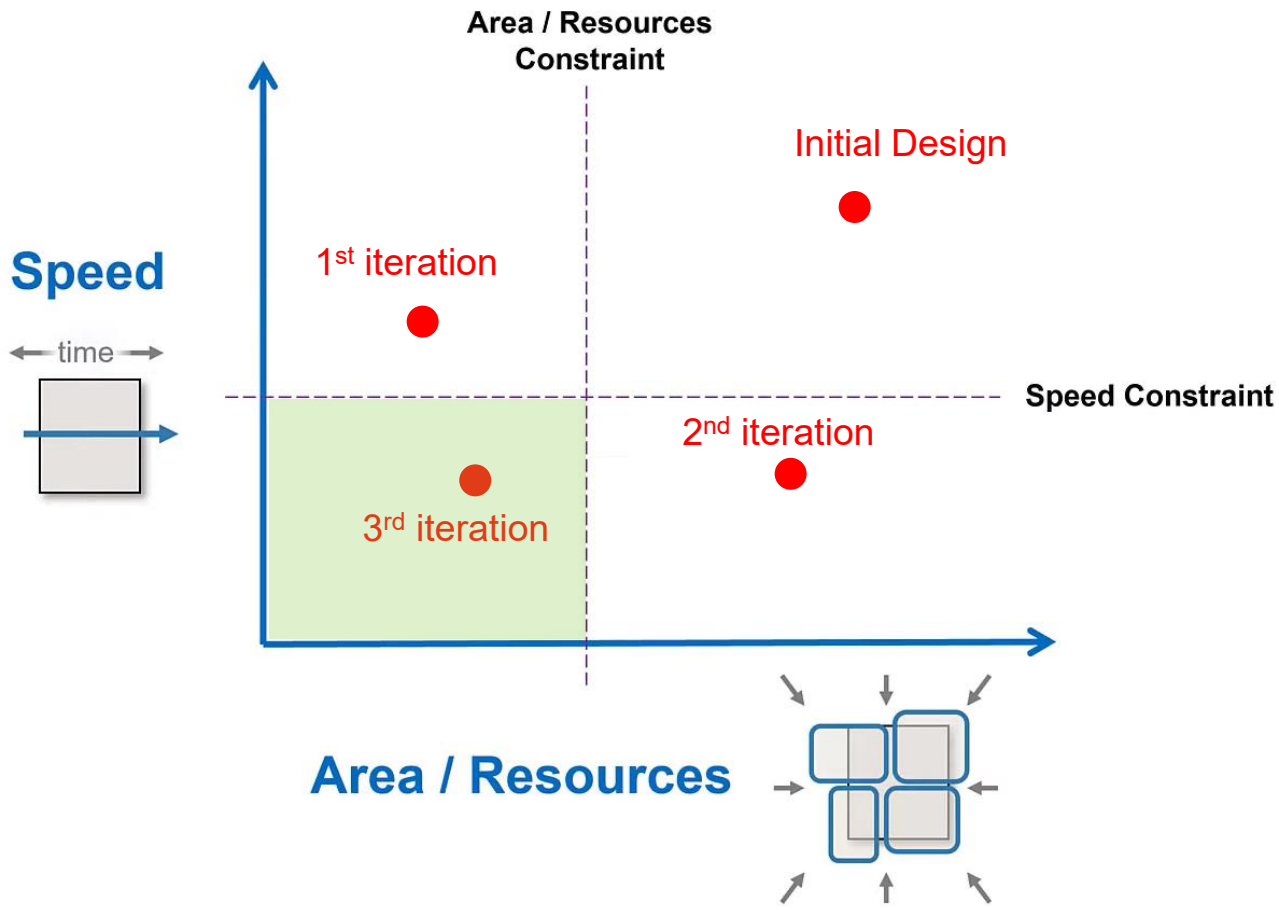
For code generation, the `hdl.interpn` function does not support:

- `Fixed-point` data type
- Multiple query points

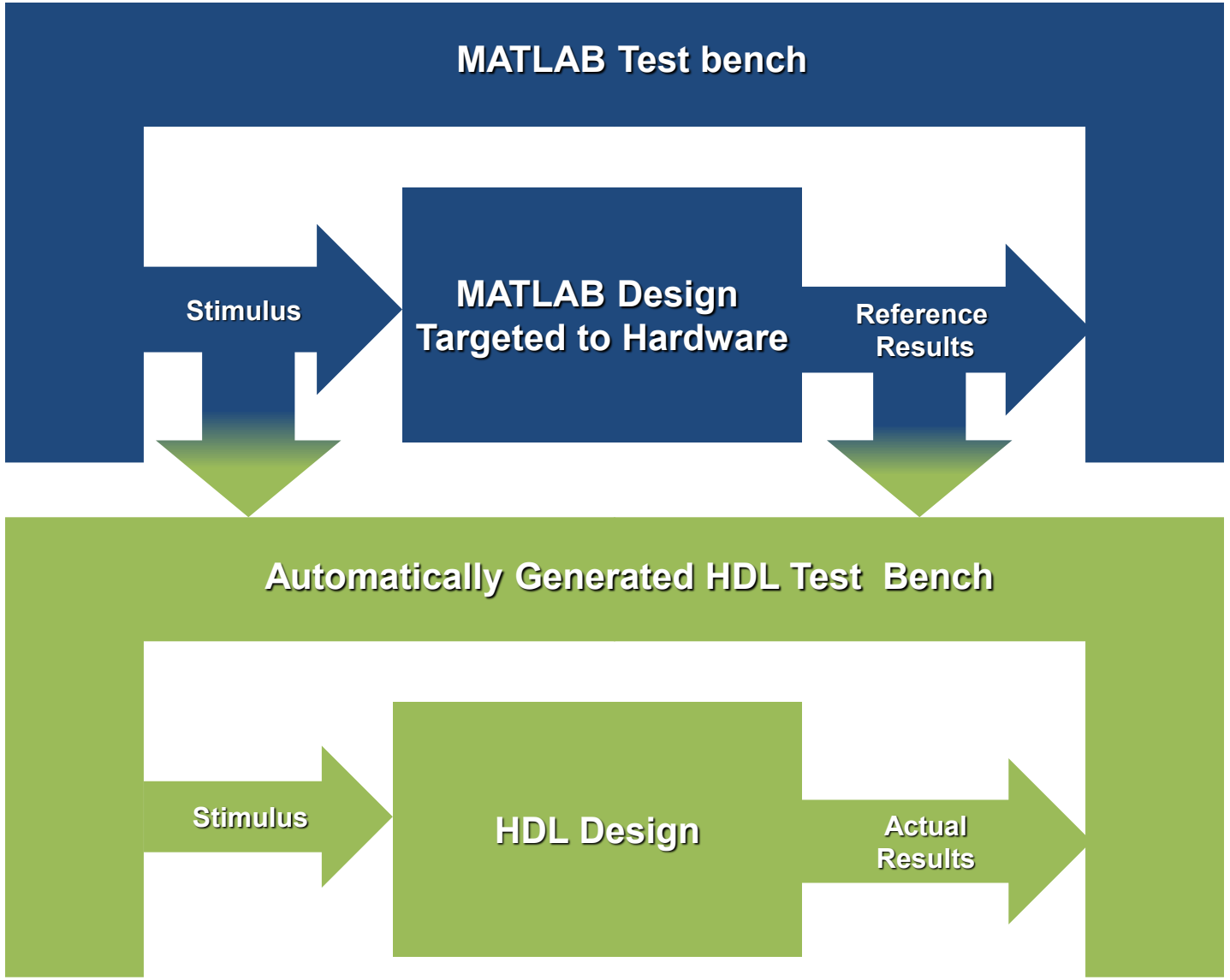
Otomatik Optimizasyon Seçenekleri



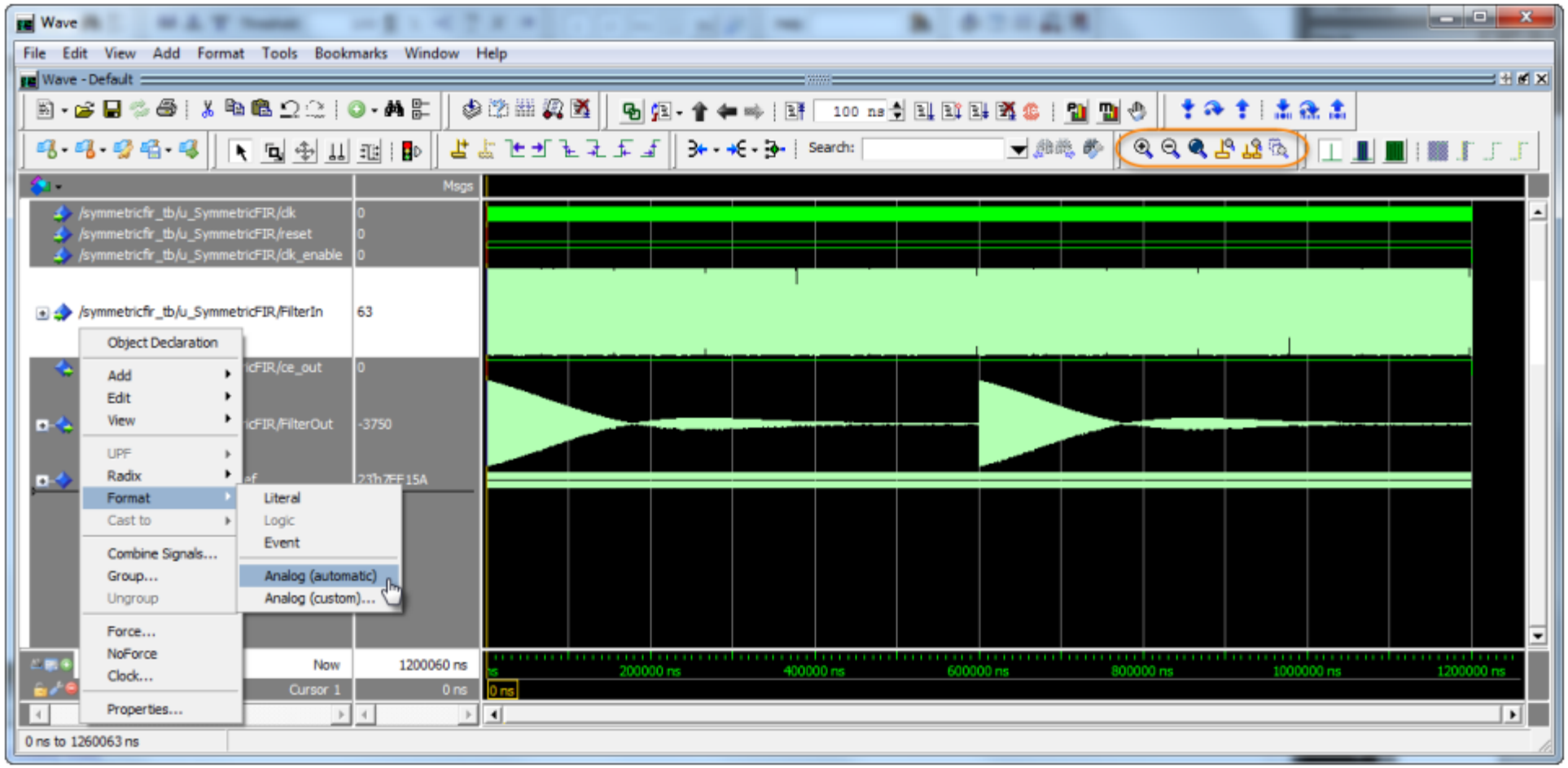
İteratif Tasarım Optimizasyonu



Testbench Tabanlı Doğrulama

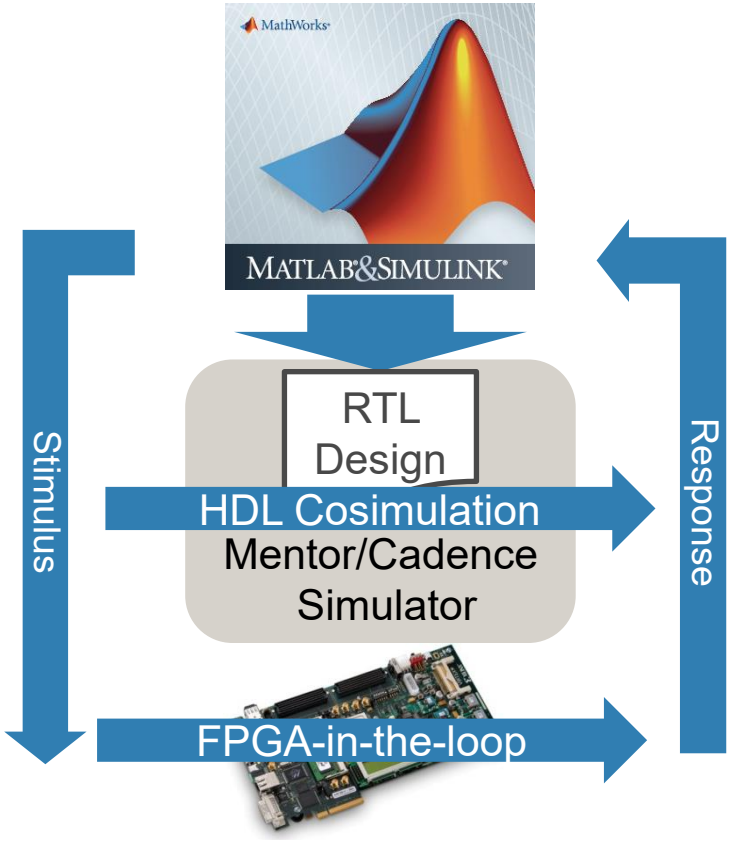


Testbench Tabanlı Doğrulama



HDL Verifier

- **Cosimulate** RTL back-to-back with the model to debug before FPGA deployment
- Simulate **FPGA-in-the-loop** with your MATLAB/Simulink tests



FİGES
applied physics & mathematics

MathWorks®
Authorized Reseller

TEŞEKKÜRLER