

به نام خداوند جان و خرد



دانشگاه صنعتی شریف

دانشکده مهندسی کامپیوتر

گزارش پروژه

# آشکارساز پالس

پروژه درس طراحی سیستم های دیجیتال

امید دلیران - ۴۰۰۱۰۴۹۳۱

رضا حیدری - ۴۰۰۱۰۹۶۱۶

سینا نمازی - ۴۰۰۱۱۰۱۵۴

[https://github.com/omid-d/Pulse\\_Detector](https://github.com/omid-d/Pulse_Detector):github لینک

لینک ویدیو در فایل readme در گیت هاب قرار دارد

۱۴۰۲ تیرماه

## توضیحات پروژه

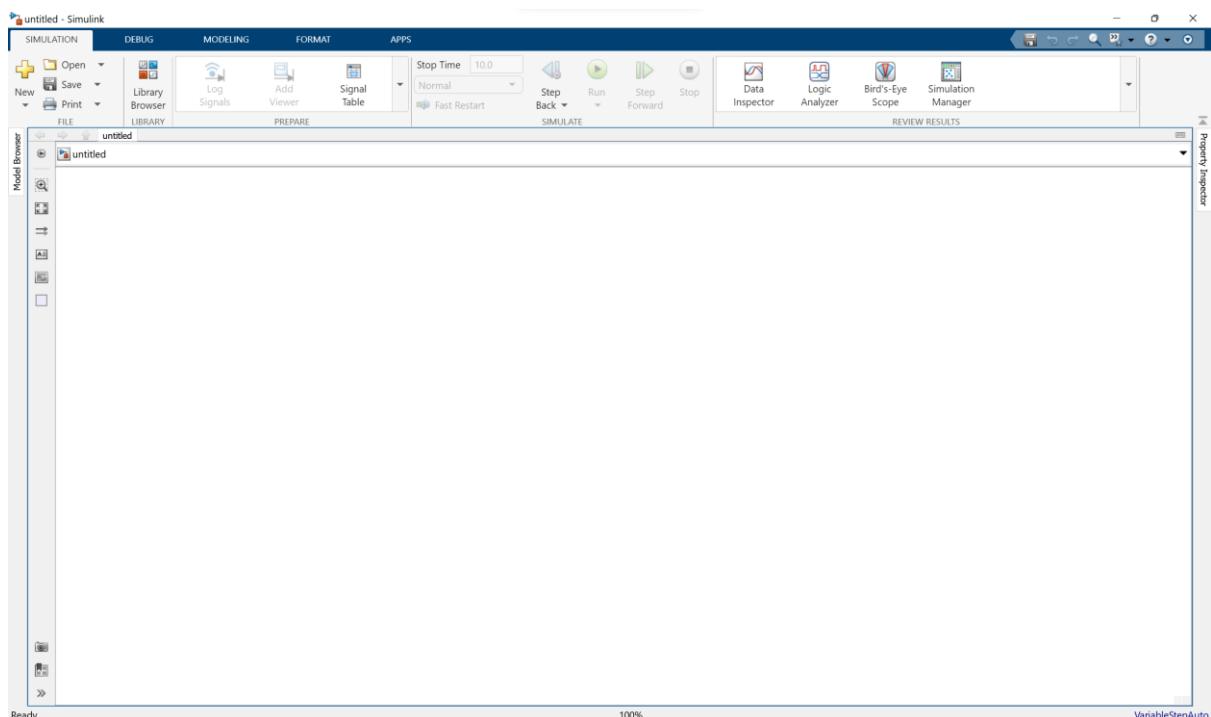
در این پروژه ما قصد داریم ابتدا با MATLAB و Simulink سیستمی برای شناسایی کردن یک پالس در سیگنال بسازیم و سپس با استفاده از HDL coder آن را به کد وریلگ تبدیل کرده و سیمولیت کیم و در نهایت آن را بهینه سازی کرده و توان مصرفی و مصرف منابع را پایین بیاوریم و احتمالاً دقت کار نیز پایین می‌آید.

## ۱ مراحل‌های مربوط به Cache

در ابتداء باید طبق دستورالعمل داده شده سیستم را در سیمولینک پیاده و تست کنیم:

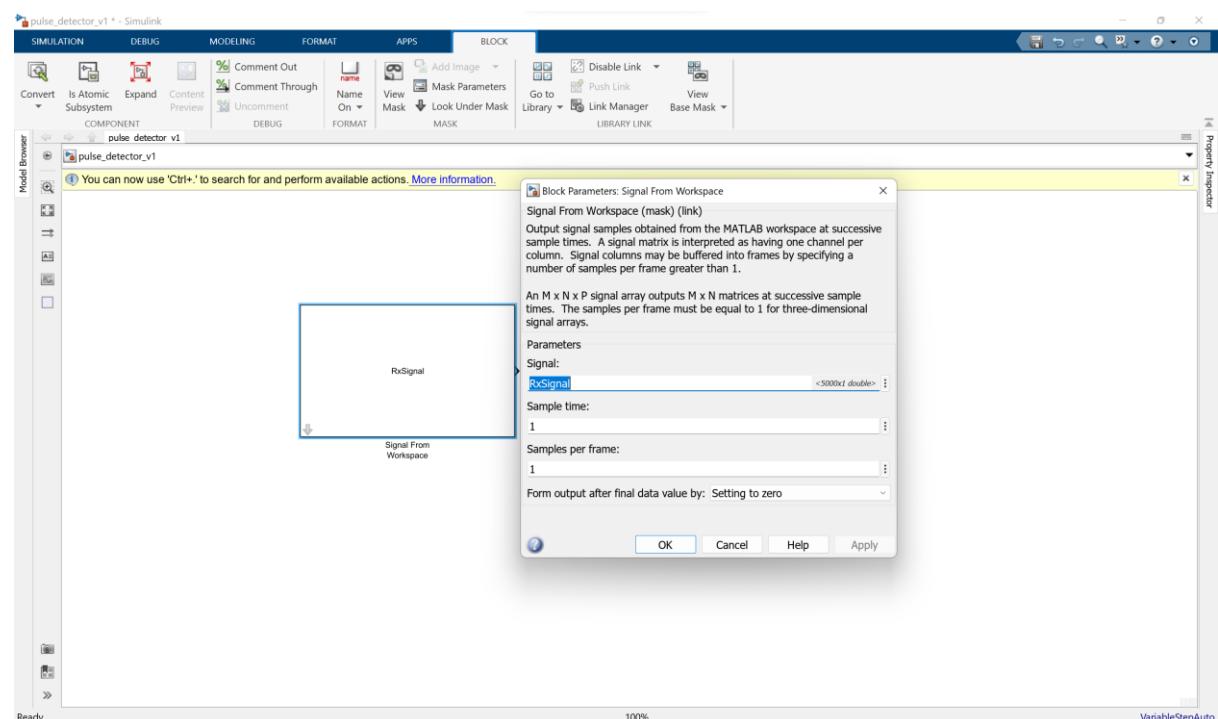
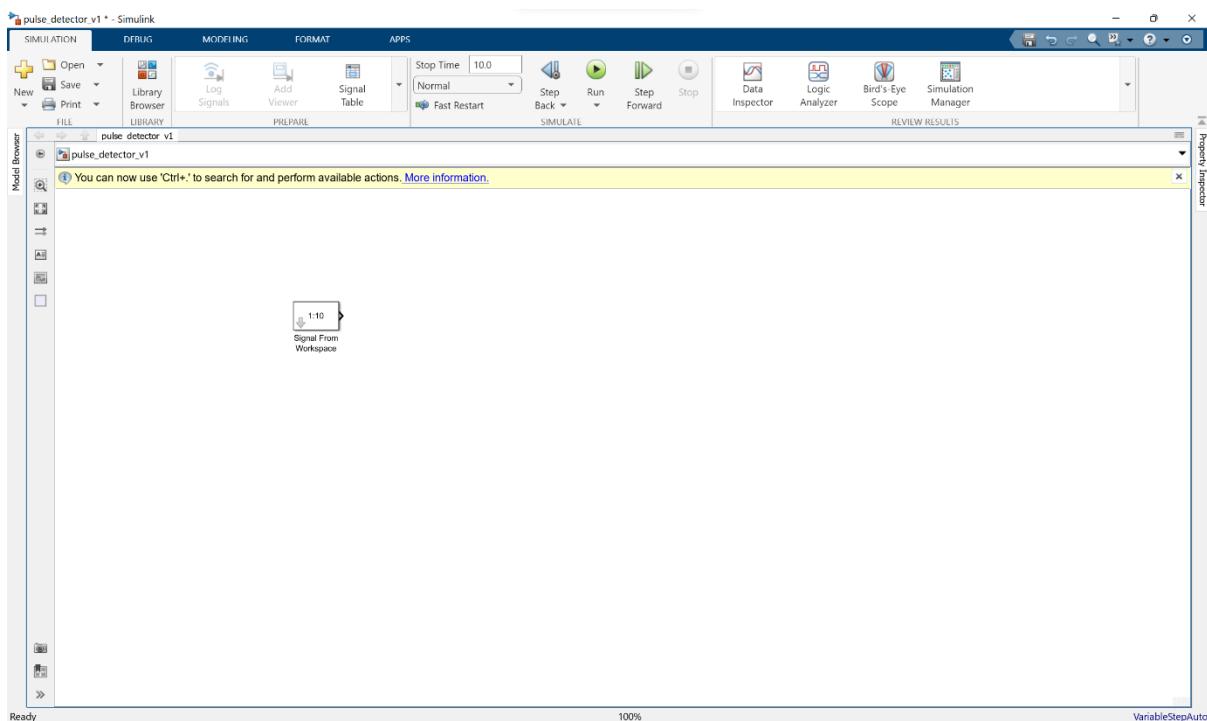
### ۱-۱ ساخت مدل در سیمولینک

ابتدا فایل سیمولینک جدید می‌سازیم و نام گذاری می‌کنیم:



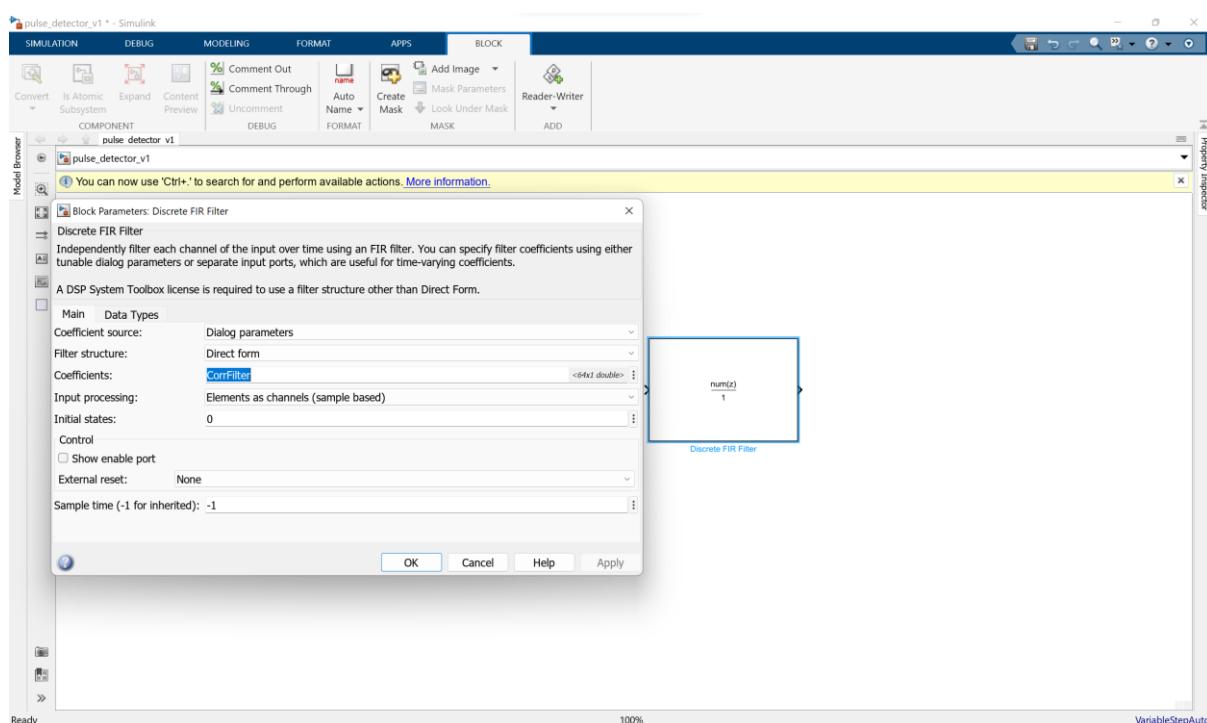
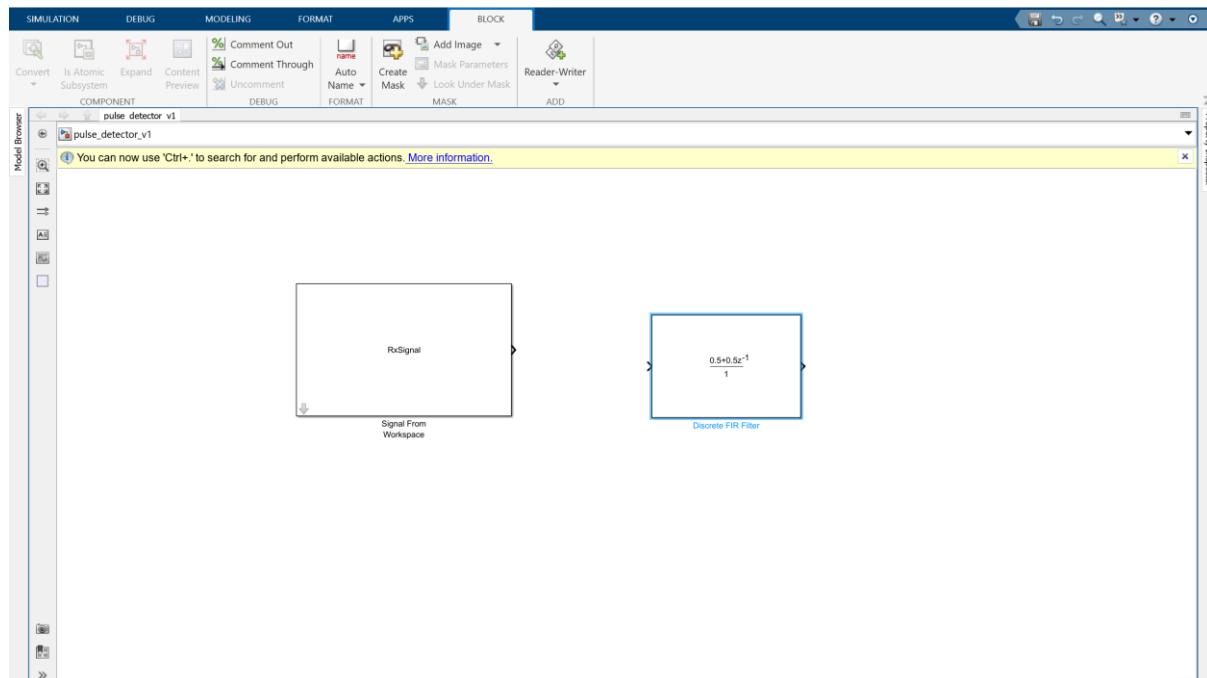
سپس یک بلاک Signal from workspace را برای تولید سیگنال اضافه می‌کنیم و سیگنالش را مشخص می‌کنیم:

## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس



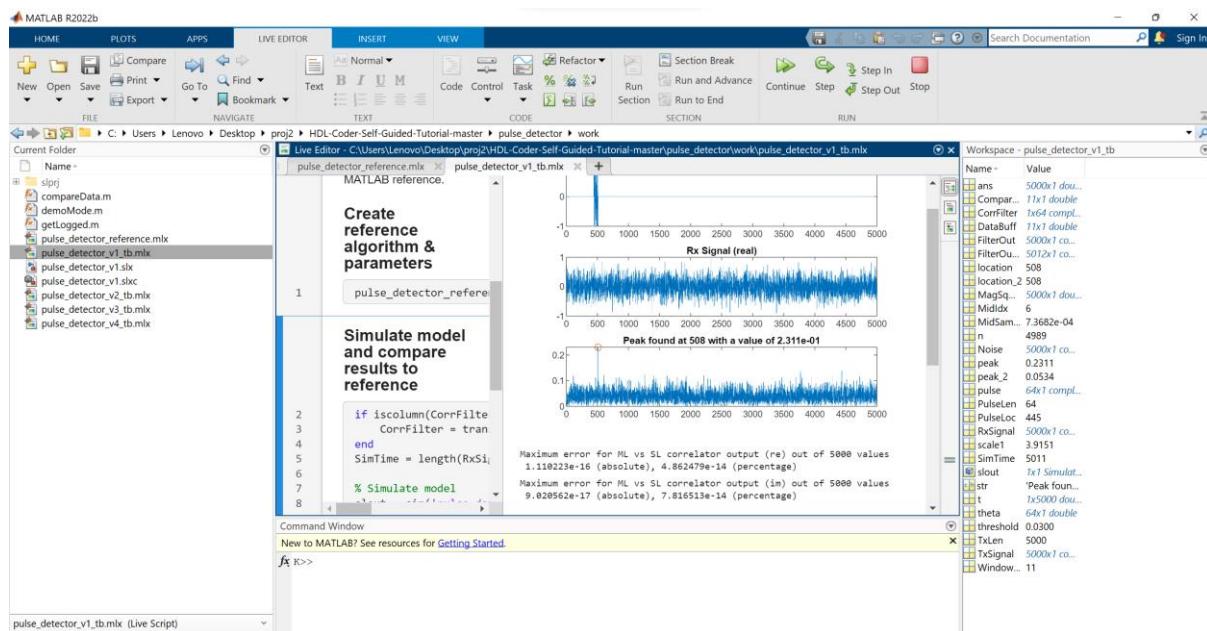
سپس یک بلاک discrete FIR filter و پارامتر آن تنظیم می کنیم:

## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس



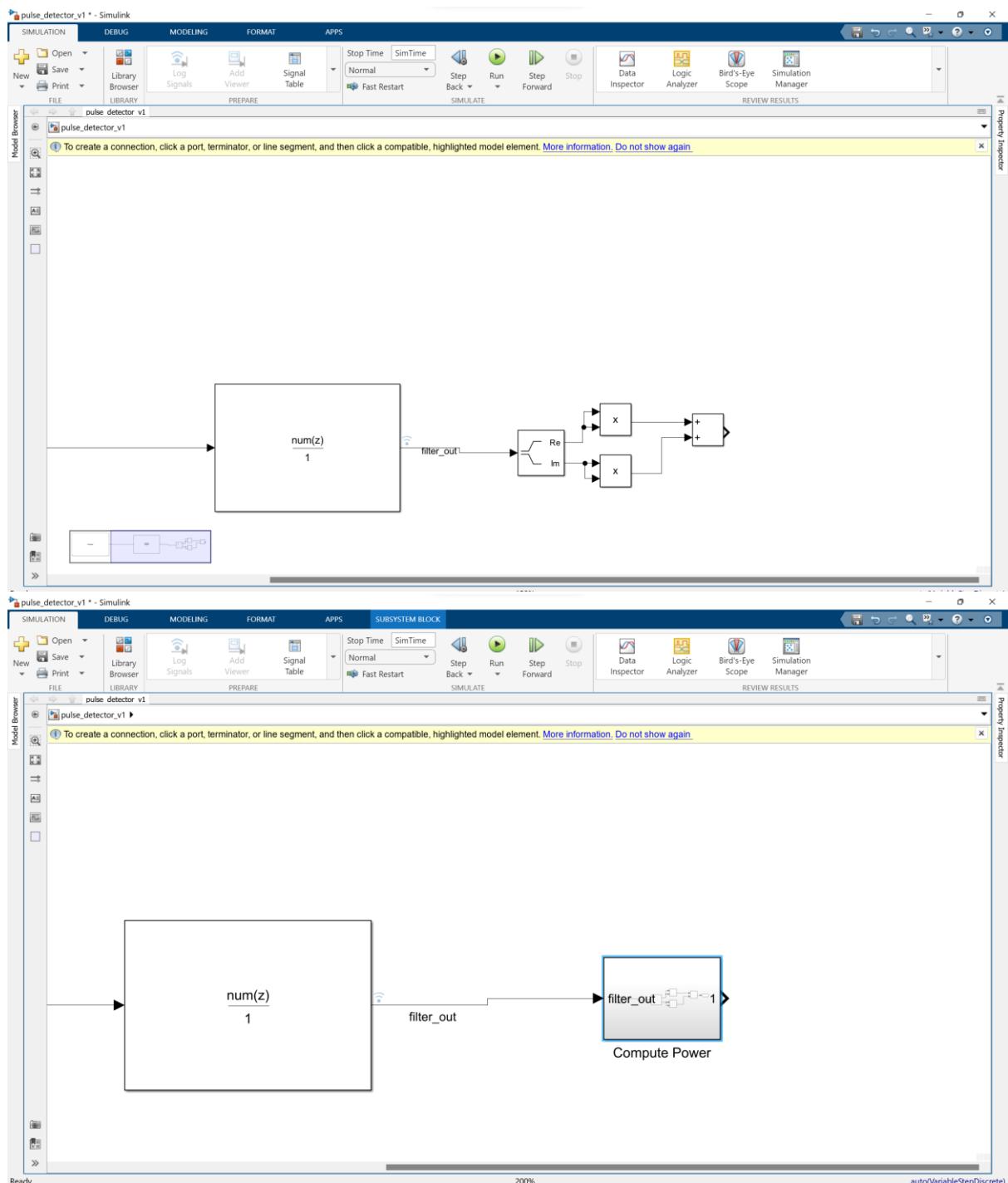
سپس تست بنج pulse\_detector\_v1\_tb.mlx را تا خط ۱۶ ران کرده و نتایج را با مقدار reference مقایسه می کنیم:

## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس

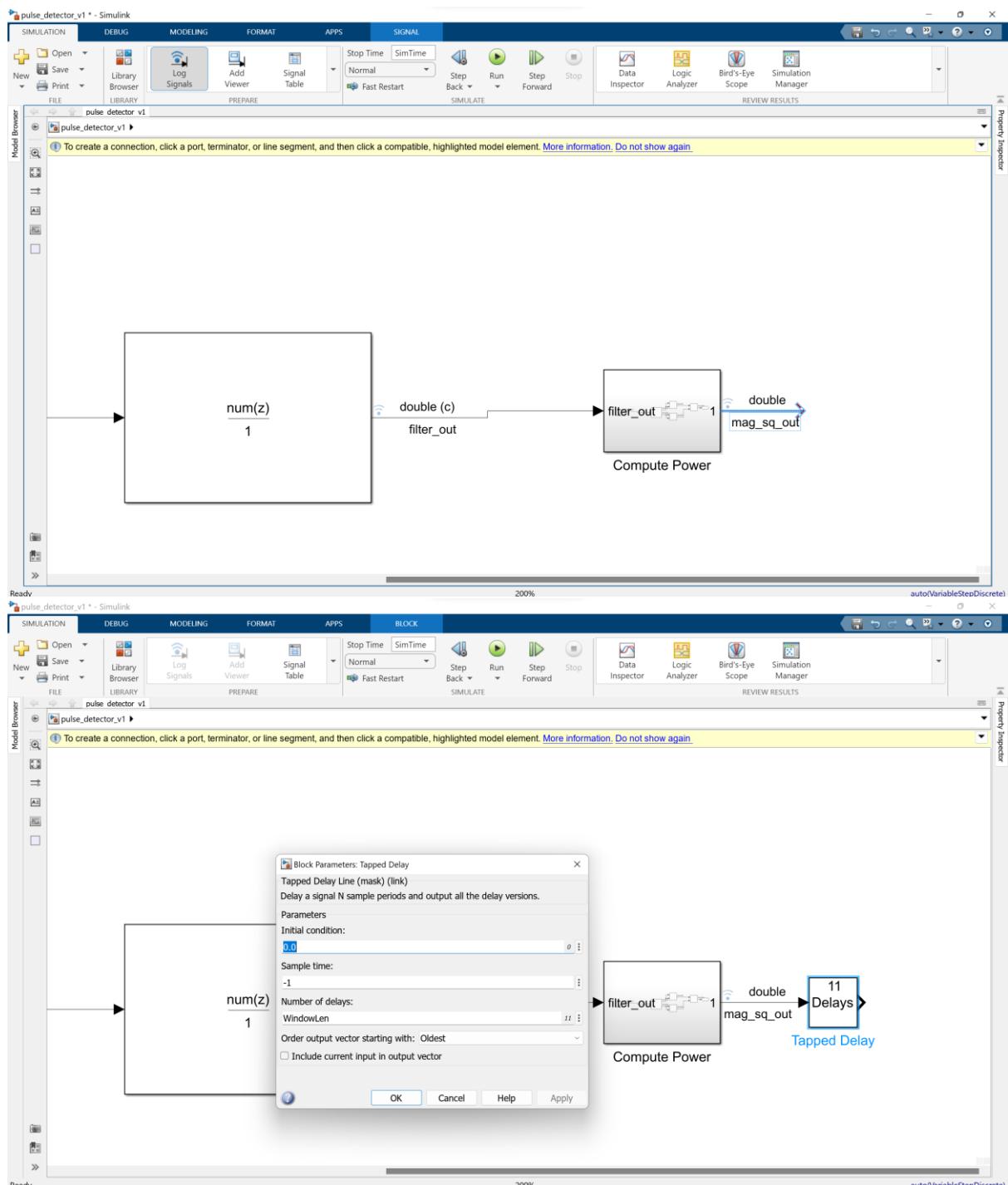


سپس ادامه سیستم را به صورت مراحل زیر می سازیم:

## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس



## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس



## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس

**Simulink Model:**

```

graph LR
    filter_out[filter_out] --> ComputePower[Compute Power]
    ComputePower -- mag_sq_out --> Delays[11 Delays]
    Delays -- u --> fcn[fcn]
    fcn -- y --> Out[ ]

```

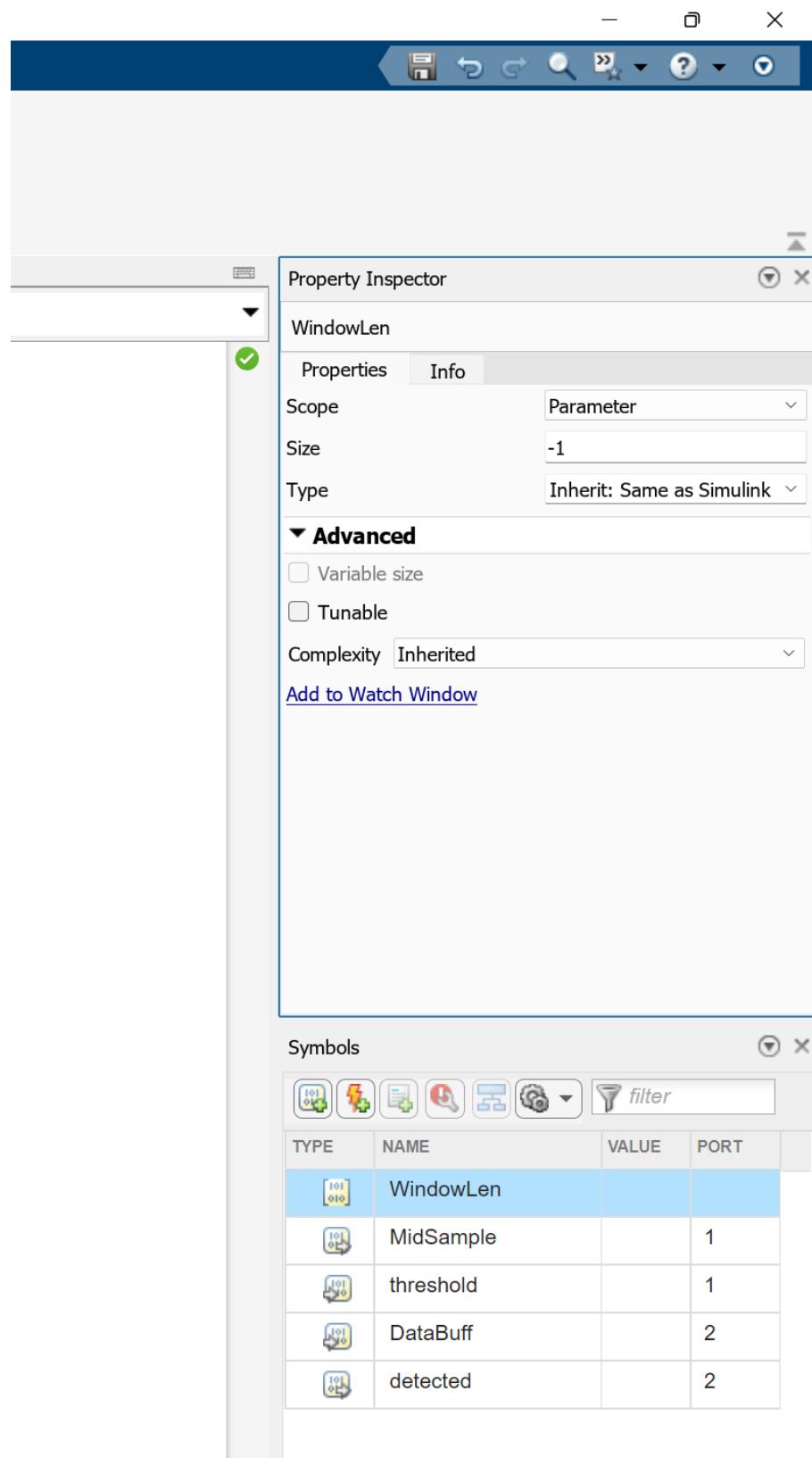
**MATLAB Function Code:**

```

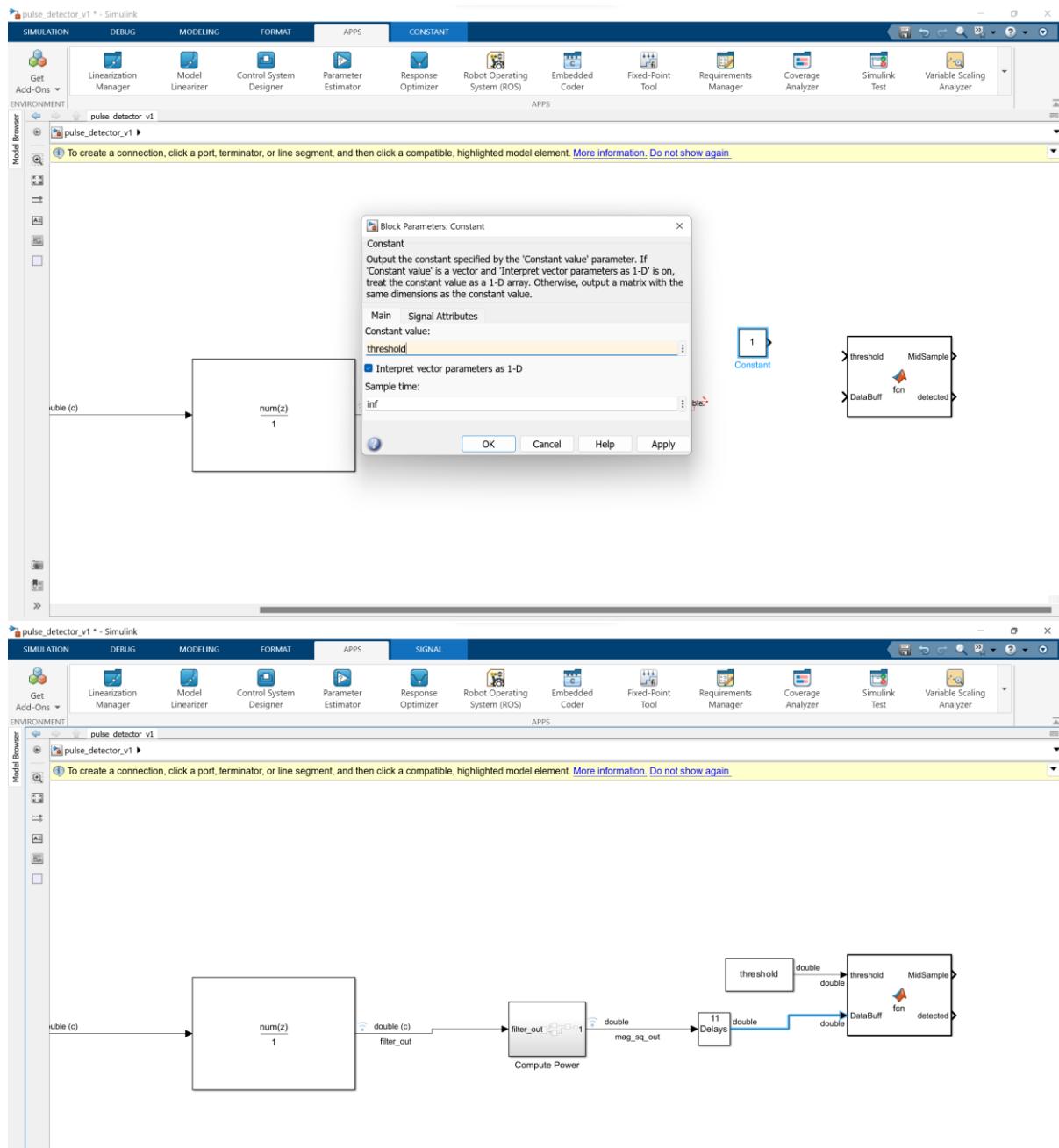
function [MidSample,detected]=fcn(WindowLen,threshold,DataBuff)
    MidIdx = ceil(WindowLen/2);
    MidSample=DataBuff(MidIdx);
    CompareOut=DataBuff-MidSample;

    % if all values in the result are negative and the middle sample is
    % greater than a threshold, it is a local max
    if all(CompareOut <= 0) && (MidSample > threshold)
        detected=1;
    else
        detected=0;
    end

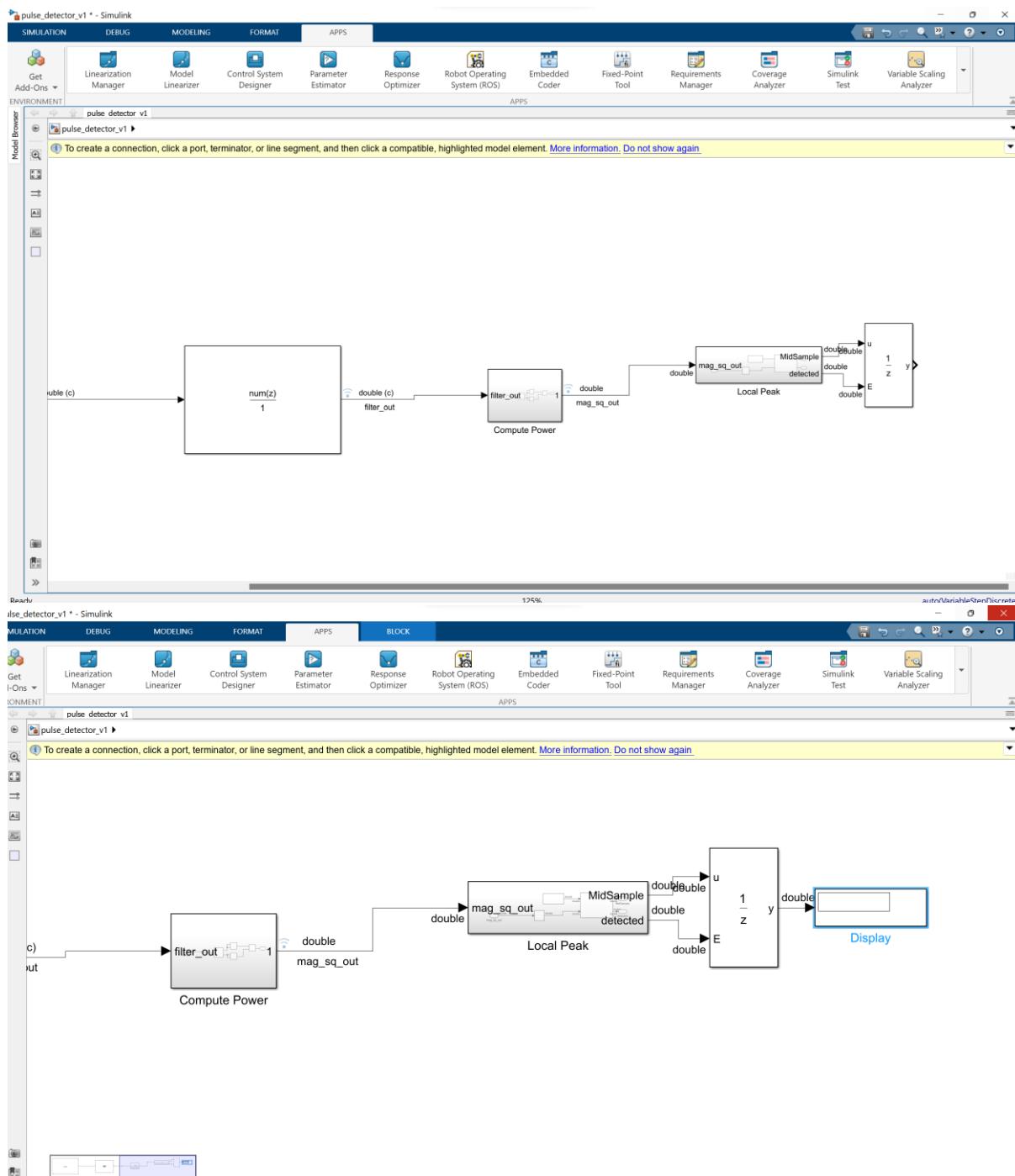
```



## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس

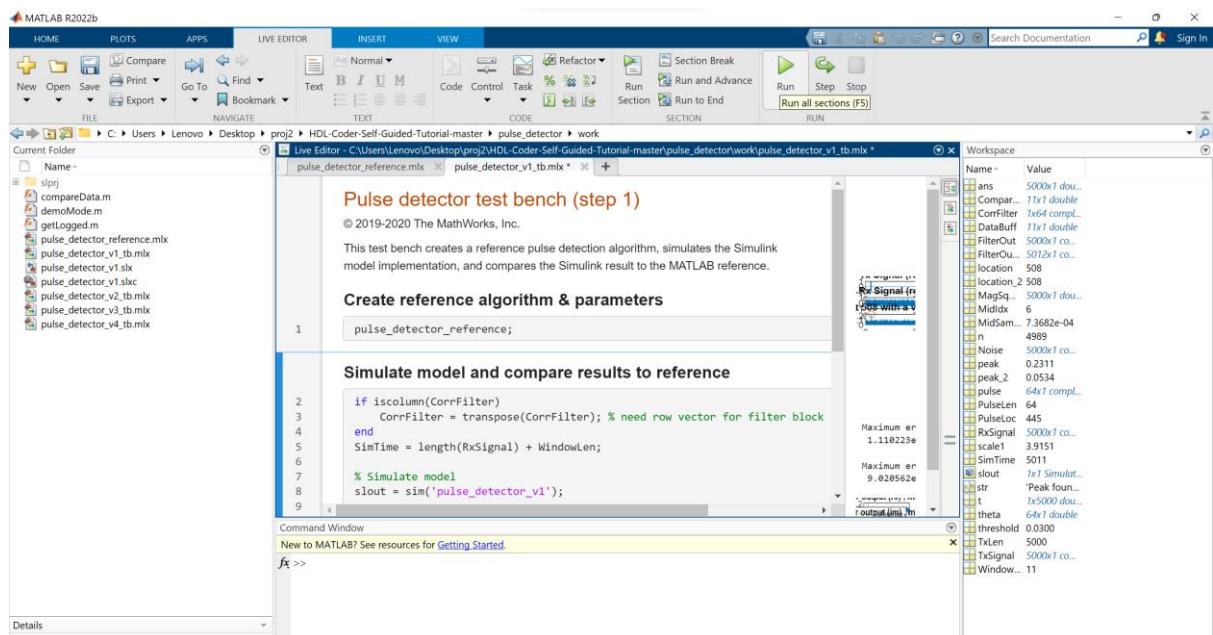


## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس

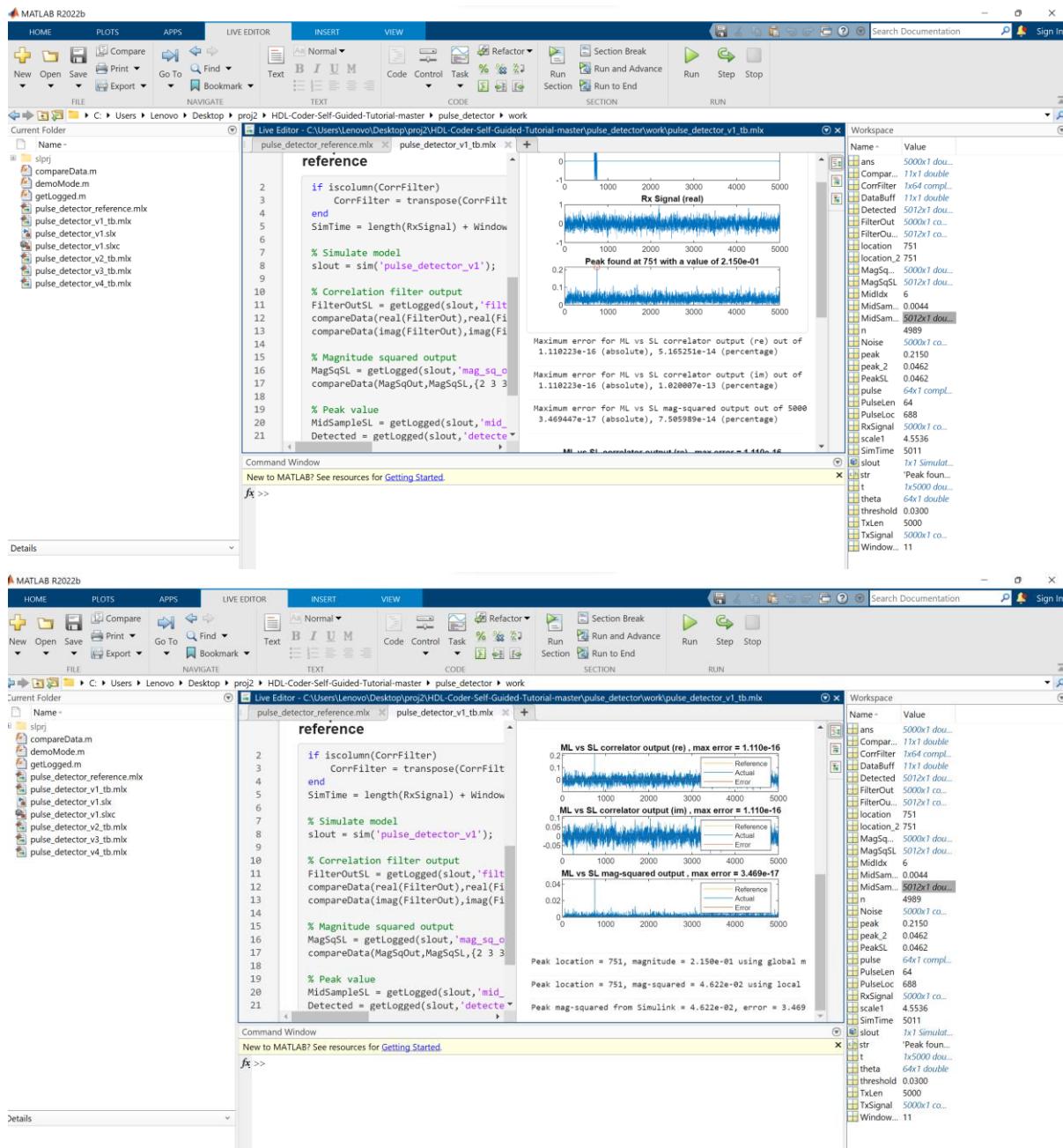


سپس دوباره تست بنج pulse\_detector\_v1\_tb.mlx را ران می کنیم اما این دفعه کل برنامه را اجرا و دوباره مقایسه می کنیم:

## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس



## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس

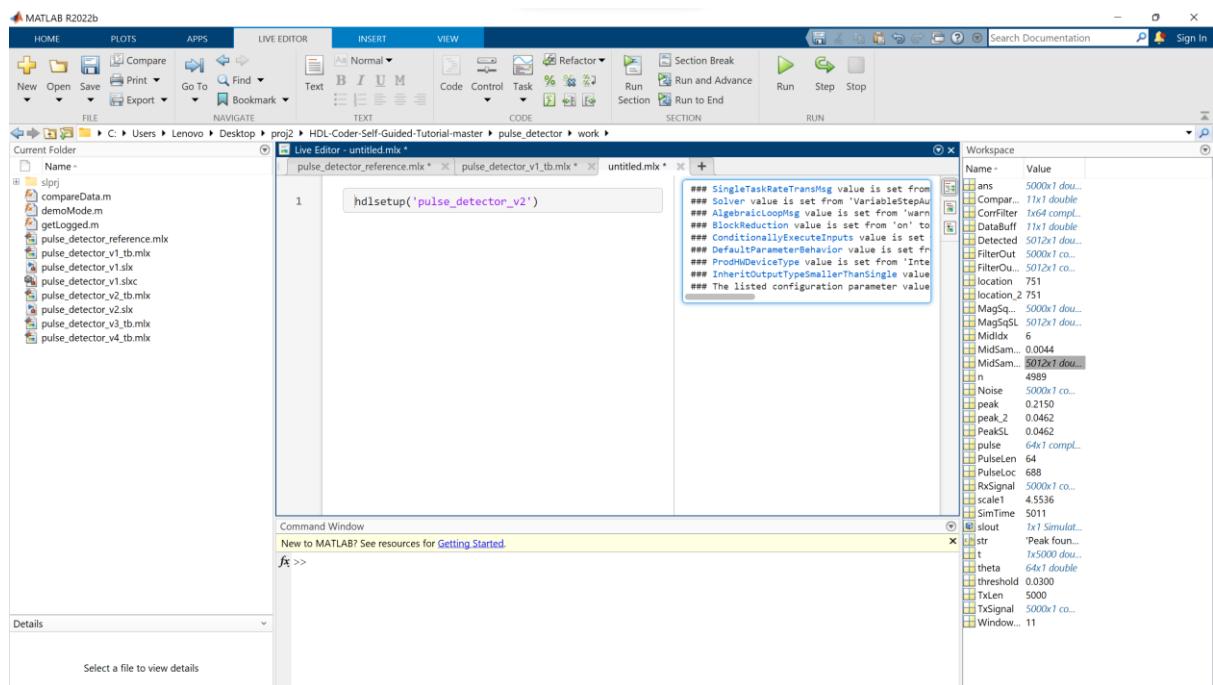


همان طور که در تصاویر بالا مشخص است اردر خطای حدود  $e^{-17}$  است.

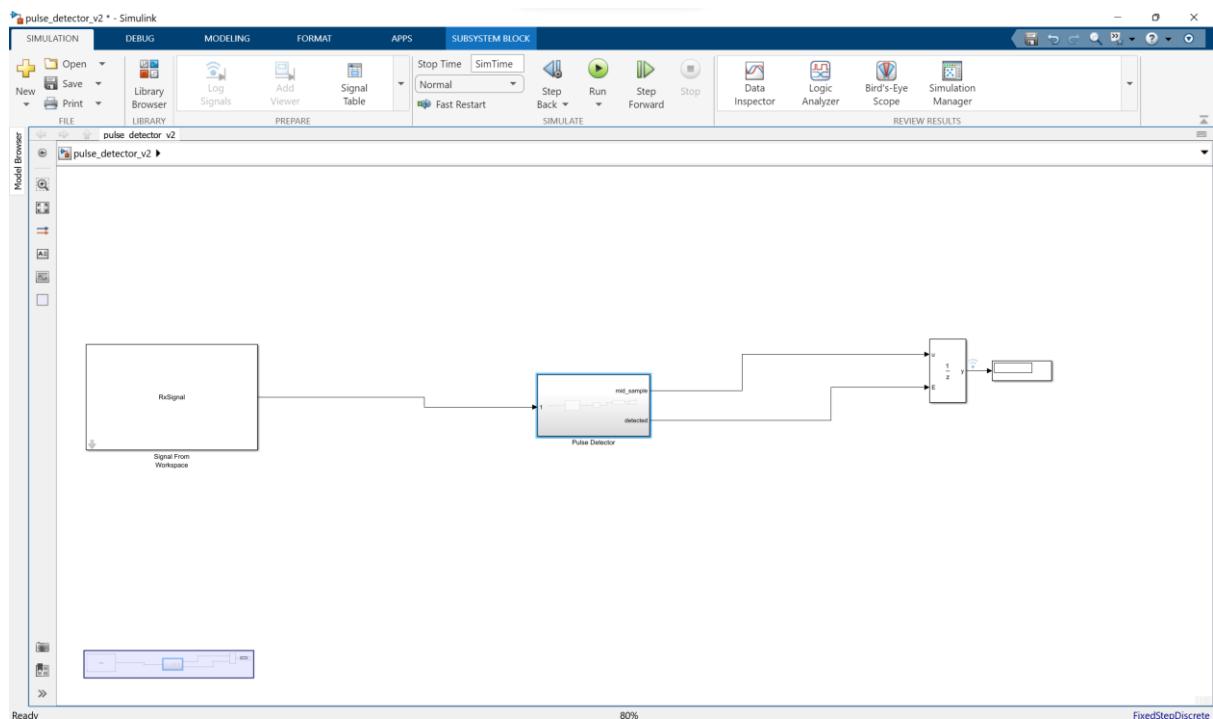
### ۲-۱ آماده سازی برای ساختن کد HDL

ابتدا دستور زیر را اجرا می کنیم که آماده استفاده از **HDL coder** برای مدل ما شود:

## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس

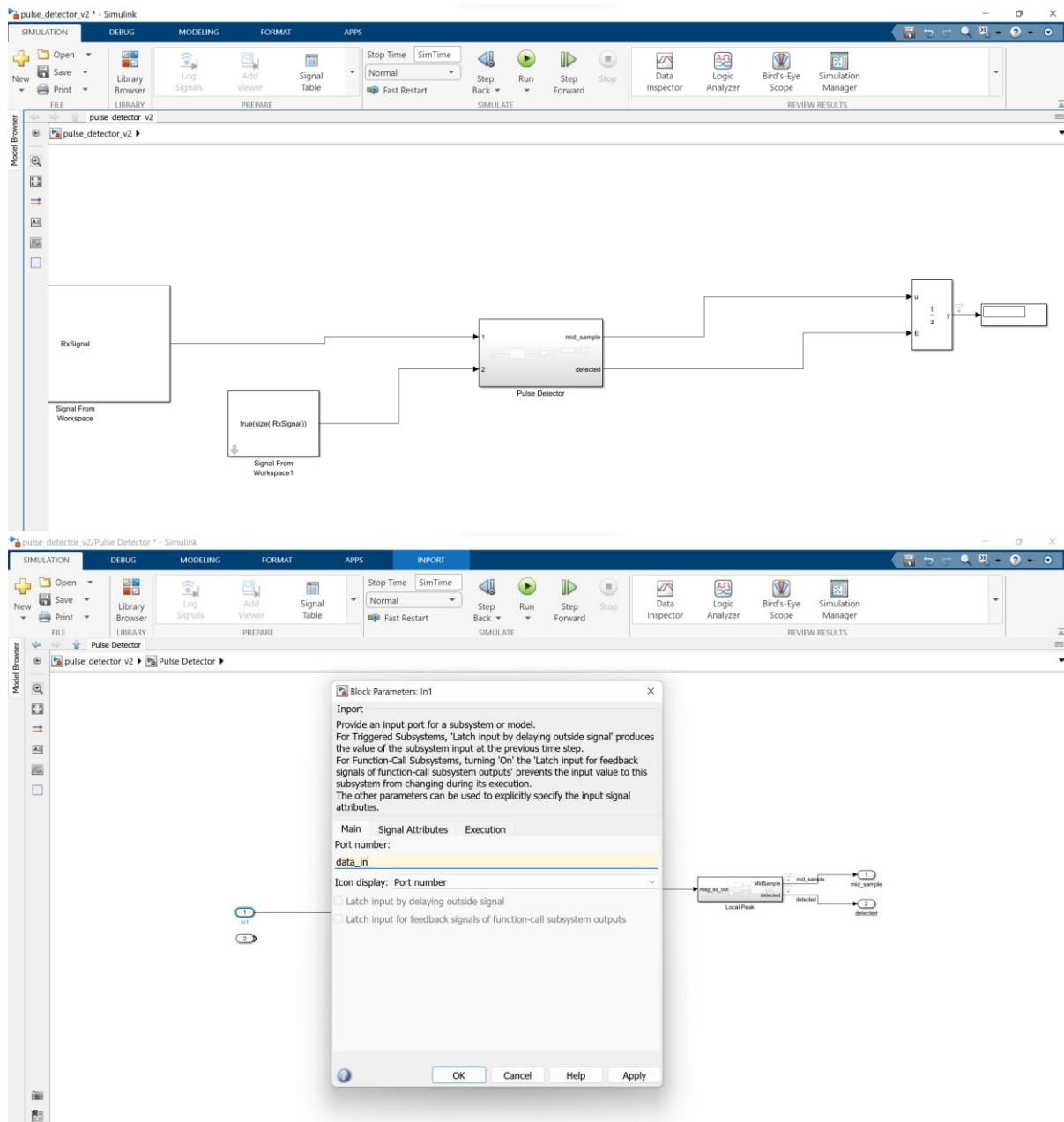


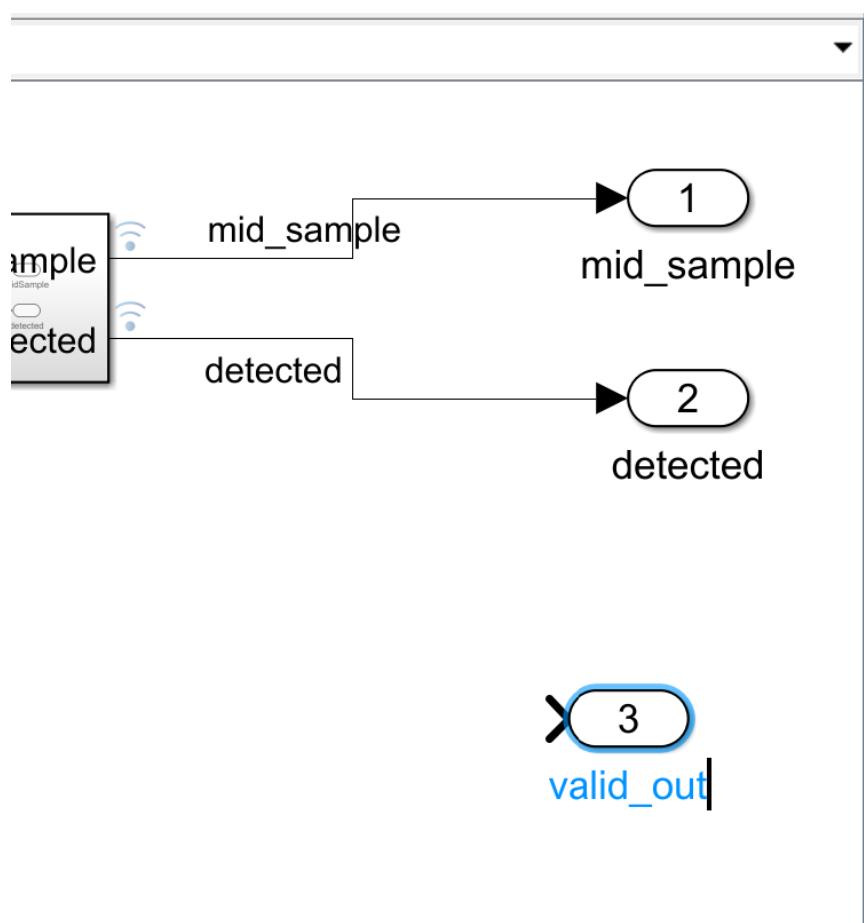
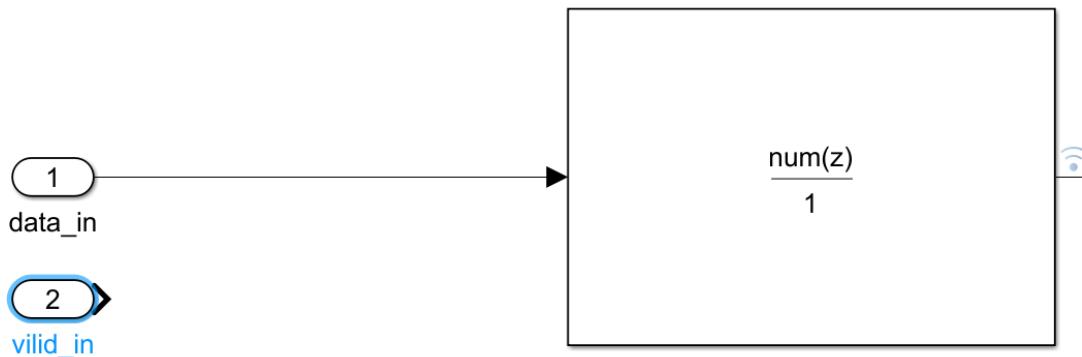
سپس مازول های فیلتر بلک، Local Peak و Compute Power Pulse Detector تبدیل می کنیم(این مازول درواقع همان مازولی است که ما در ادامه آن را سنتز و تغییر خواهیم داد)



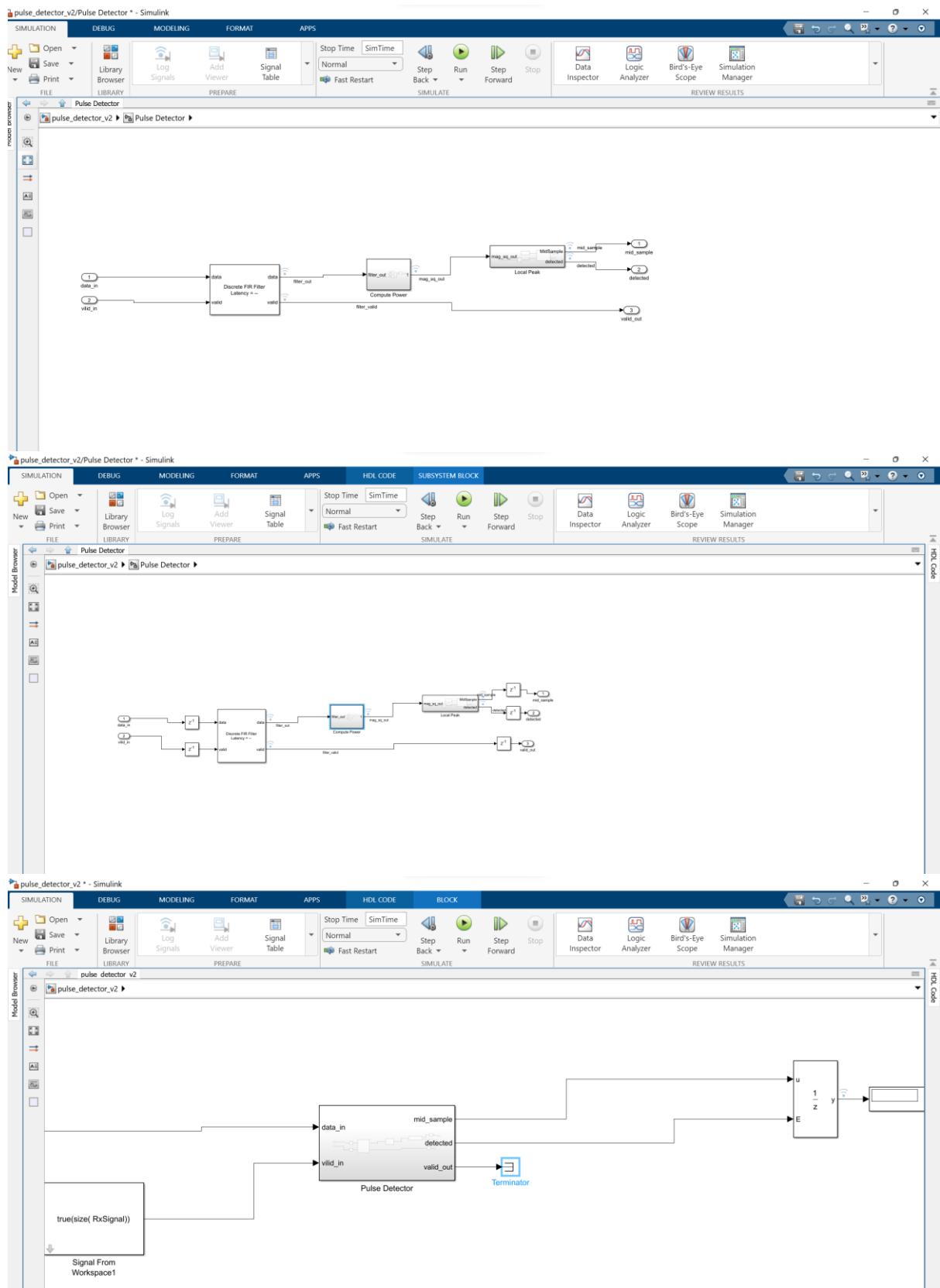
سپس به صورت مراحل زیر مازول ها را تغییر می دهیم:

## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس



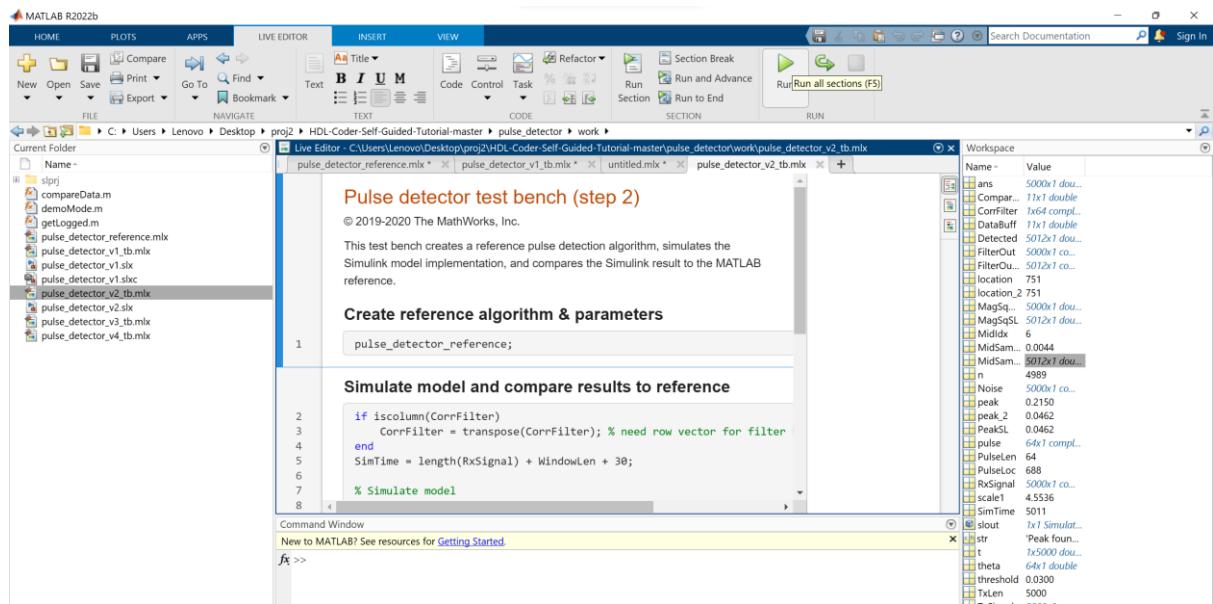


## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس

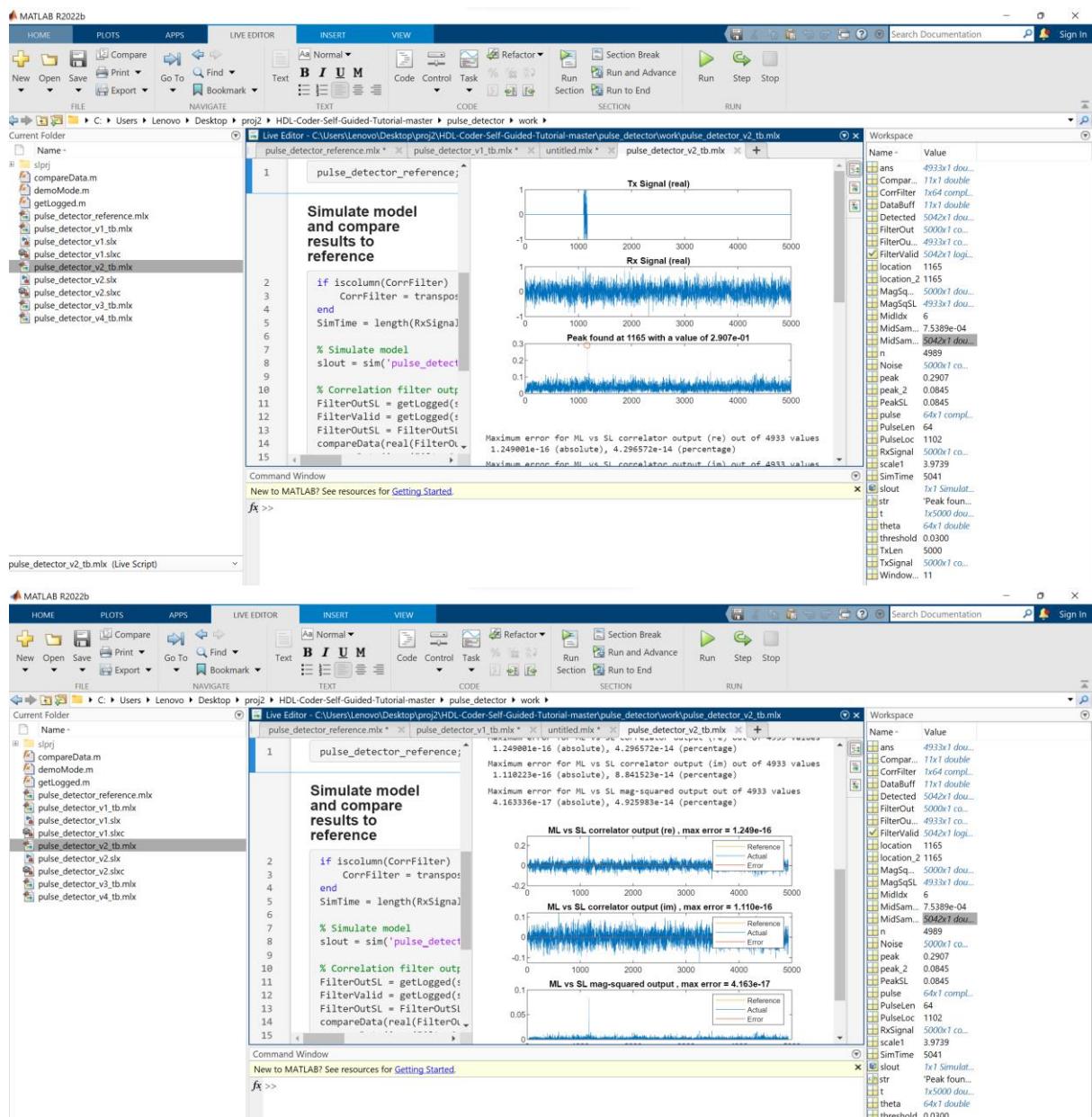


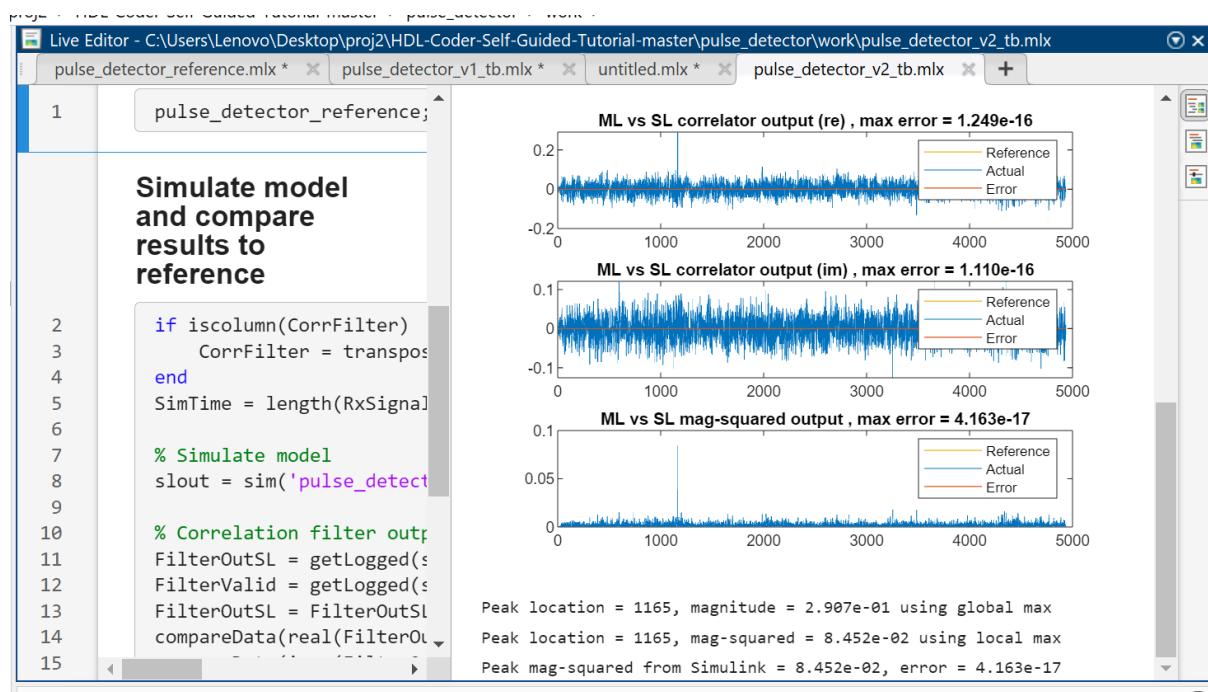
سپس مانند مراحل قبلی pulse\_detector\_v2\_tb.mlx را ران می کنیم و نتایج را مشاهده می کنیم:

## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس



## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس





### ۳-۱ تبدیل به fixed point

مانند مراحل زیر مازول هایی که لازم است را تغییر می دهیم و تعدادی مازول **convertor** اضافه می کنیم تا سیستم به تبدیل شود:

## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس

**Simulink Model:**

**Block Parameters: Data Type Conversion**

Data Type Conversion  
Convert the input to the data type and scaling of the output.

The conversion has two possible goals. One goal is to have the Real World Values of the input and the output be equal. The other goal is to have the Stored Integer Values of the input and the output be equal. Overflows and quantization errors can prevent the goal from being fully achieved.

Parameters

Output minimum: Output maximum:  
[]

Output data type: fixdt(1,16,14)

Lock output data type setting against changes by the fixed-point tools

Input and output to have equal: Real World Value (RWV)

Integer rounding mode: Floor

Saturate on integer overflow

**OK Cancel Help Apply**

**Simulink Model:**

**Block Parameters: Discrete FIR Filter**

Discrete FIR Filter  
Latency = --

Choose from fully-parallel Direct form systolic or Direct form transposed structures or a Partly serial systolic structure with configurable serialization parameters. All structures share multipliers in symmetric or antisymmetric filters. Both systolic structures make efficient use of Intel and Xilinx DSP blocks.

When using programmable coefficients, set Coefficients prototype to a coefficient vector that is representative of the symmetry and zero-value locations of the expected coefficients. The block uses this prototype to optimize multipliers in the filter implementation. If your coefficients are unknown or not expected to share symmetry or zero-value locations, set Coefficients prototype to [ ].

Main Data Types Control Ports

Coefficient source: Property  
Coefficients: DT\_coeff  
Filter structure: Direct form systolic

**OK Cancel Help Apply**

**MATLAB R2022b:**

```

% Pulse detector v3_tb.mlx
DT_coeff = fixdt(1,18); % coeff is treated as double if it's a scalar
if iscolumn(CorrFilter)
    CorrFilter = transpose(CorrFilter); % need row vector
end
SimTime = length(RxSignal) + WindowLen + 30;

% Simulate model
slout = sim('pulse_detector_v3');

% Correlation filter output
FilterOutSL = getlogged(slout,'filter_out');
FilterValid = getlogged(slout,'filter_valid');
FilterOutSL = FilterOutSL(FilterValid);
compareData(real(FilterOut),real(FilterOutSL),[2 3 1],'M'
compareData(imag(FilterOut),imag(FilterOutSL),[2 3 2],'M'

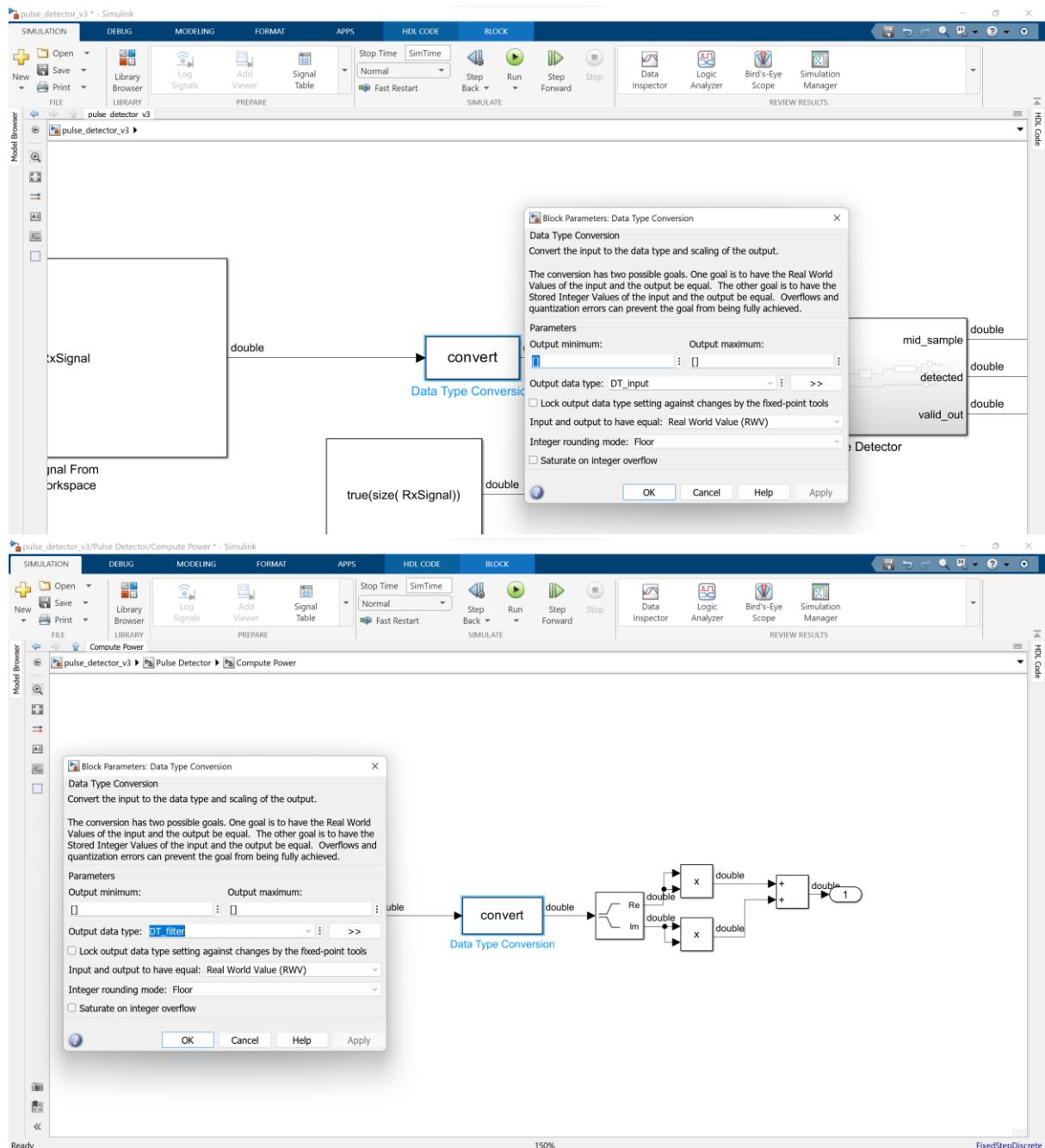
% Magnitude squared output
MagSqSL = getlogged(slout,'mag_sq_out');
MagSqSL = MagSqSL(FilterValid);
compareData(MagSqOut,MagSqSL,[2 3 3],'ML vs SL mag-squared');

```

Workspace

Name	Value
Compar...	1x1 double
CorrFilter	1x61 compl...
DataBuff...	1x1 double
DT_coeff	1x7 Numer...
DT_filt...	1x7 Numer...
DT_inpu...	1x7 Numer...
DT_powe...	1x7 Numer...
FilterOut...	600x1 co...
fopt_m...	1
location_...	514
location_2...	514
MagSq...	5000x1 dou...
MidIdx...	6
MidSam...	0.0018
n	4989
Noise	5000x1 co...
peak	0.2893
peak_2	0.0837
pulse	6dx1 compl...
PulseLen	64
PulseLoc	451
RxSignal	5000x1 co...
scale1	4.3789
SimTime	5041
str	'Peak foun...
t	7x5000 dou...
theta	6dx1 dou...
threshold	0.0300
TxLen	5000
TxSignal	5000x1 co...
Window...	11

## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس



## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس

The figure displays three screenshots of the MATLAB/Simulink environment, illustrating the design and implementation of a digital pulse detection system.

**Screenshot 1:** Simulink Model Browser showing the 'pulse\_detector\_v3/Pulse Detector/Local Peak' model. The diagram consists of several blocks: a 'threshold' block, a '11 Delays' block, a 'DataBuff fcn' block, and a 'same DT' block. A signal labeled 'mag\_sq\_out' enters the '11 Delays' block. The output of the '11 Delays' block goes to both the 'DataBuff fcn' block and the 'threshold' block. The 'DataBuff fcn' block has two outputs: 'MidSample' and 'detected'. The 'MidSample' output goes to a scope, and the 'detected' output goes to the 'same DT' block.

**Screenshot 2:** Simulink Model Browser showing the same model. A 'Constant' block is selected, and its parameters are shown in a dialog box. The 'Value' field is set to '1'. Other settings include 'Output data type: Inherit: Inherit via back propagation'.

**Screenshot 3:** MATLAB Function Editor showing the code for the 'DataBuff fcn' block. The code is as follows:

```

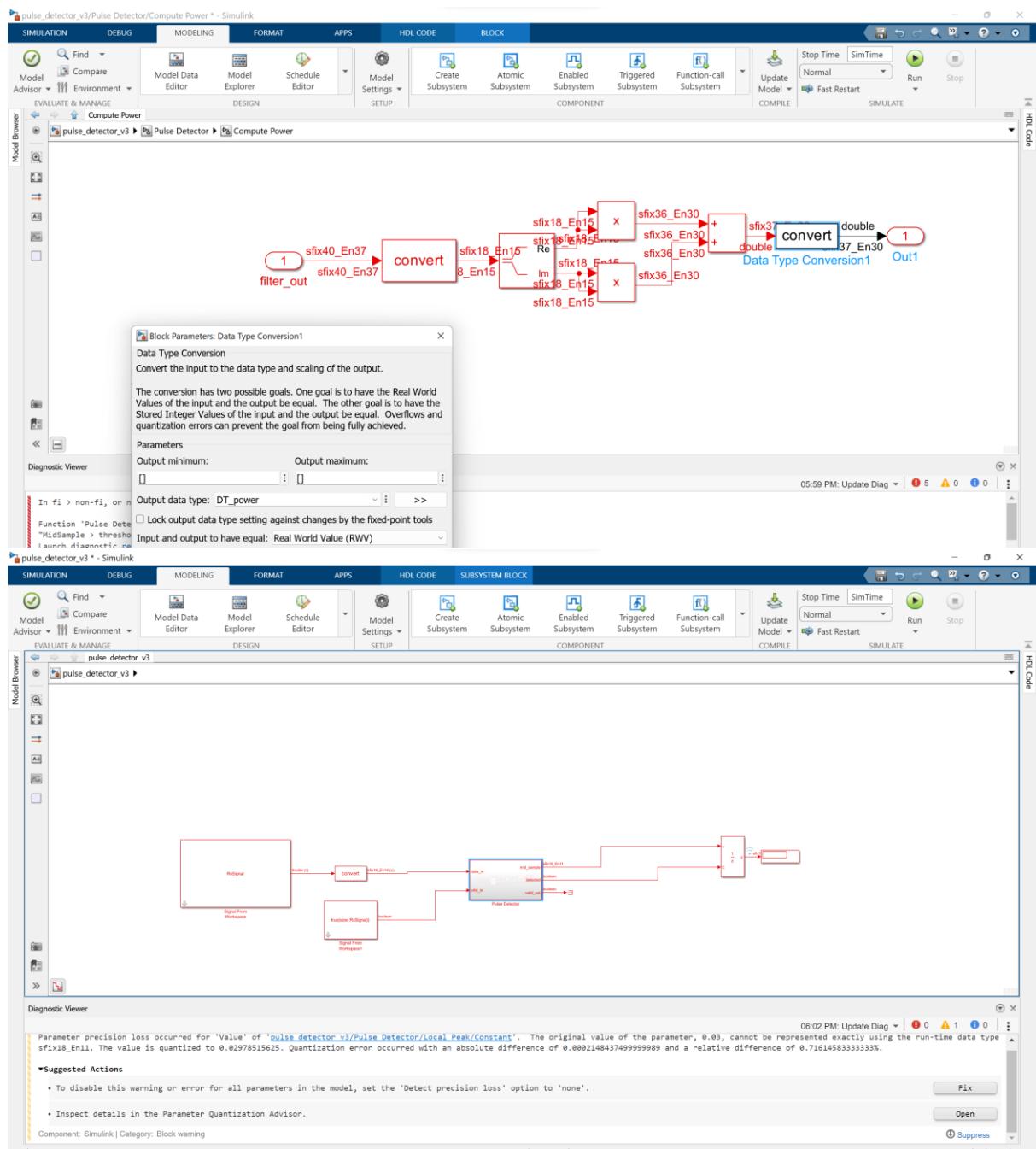
function [MidSample,detected]=fcn(threshold,DataBuff,WindowLen)
    MidIdx = ceil(WindowLen/2);
    MidSample=DataBuff(MidIdx);
    CompareOut=DataBuff-MidSample;
    % if all values in the result are negative and the middle sample is
    % greater than a threshold, it is a local max
    if all(CompareOut < 0) && (MidSample > threshold)
        detected=true;
    else
        detected=false;
    end

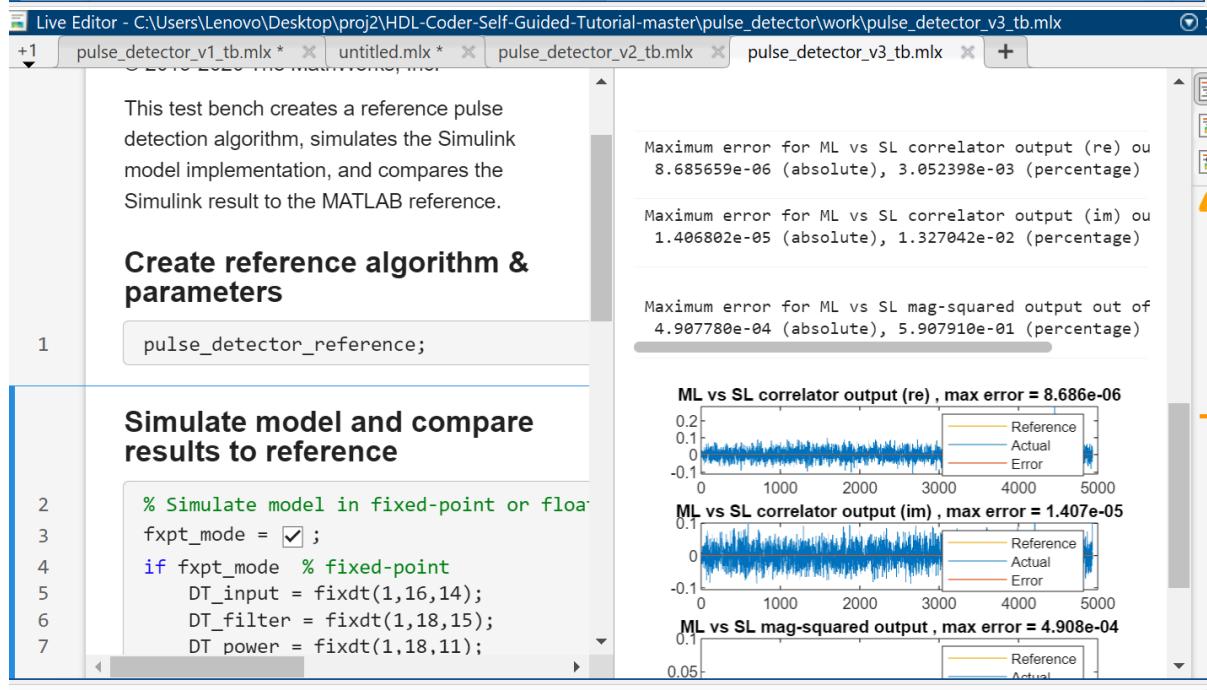
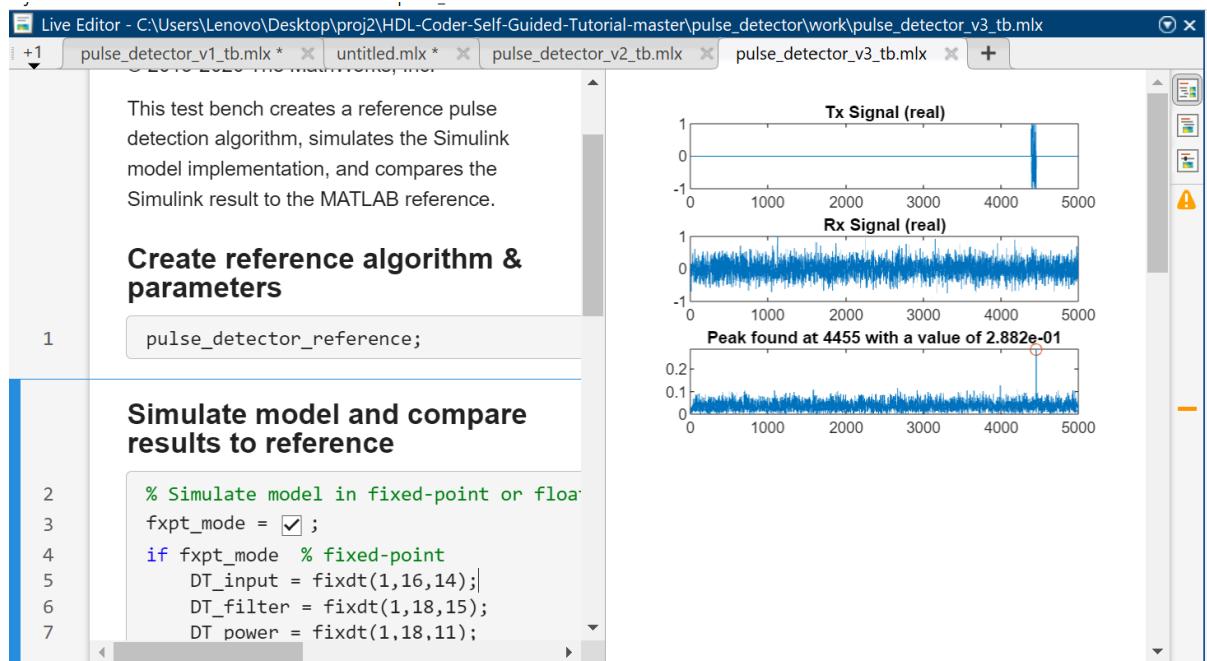
```

The right side of the editor shows a 'Symbols' table with the following entries:

TYPE	NAME	VALUE	PORT
Block	MidSample	1	
Block	threshold	1	
Block	DataBuff	2	
Block	detected	2	
Block	WindowLen		

## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس





proj2 HDL-Coder-Self-Guided-Tutorial-master pulse\_detector WORK

Live Editor - C:\Users\Lenovo\Desktop\proj2\HDL-Coder-Self-Guided-Tutorial-master\pulse\_detector\work\pulse\_detector\_v3\_tb.mlx

+1 pulse\_detector\_v1\_tb.mlx \* untitled.mlx \* pulse\_detector\_v2\_tb.mlx pulse\_detector\_v3\_tb.mlx +

This test bench creates a reference pulse detection algorithm, simulates the Simulink model implementation, and compares the Simulink result to the MATLAB reference.

**Create reference algorithm & parameters**

1 pulse\_detector\_reference;

**Simulate model and compare results to reference**

2 % Simulate model in fixed-point or float  
3 fxpt\_mode = ;  
4 if fxpt\_mode % fixed-point  
5 DT\_input = fixdt(1,16,14);  
6 DT\_filter = fixdt(1,18,15);  
7 DT\_power = fixdt(1,18,11);

Maximum error for ML vs SL mag-squared output out of 4.907780e-04 (absolute), 5.907910e-01 (percentage)

ML vs SL correlator output (re) , max error = 8.686e-06

ML vs SL correlator output (im) , max error = 1.407e-05

ML vs SL mag-squared output , max error = 4.908e-04

Peak location = 4455, magnitude = 2.882e-01 using gl

Peak location = 4455, mag-squared = 8.307e-02 using

Peak mag-squared from Simulink = 8.301e-02, error =

Peak location = 4455, magnitude = 2.882e-01 using global max

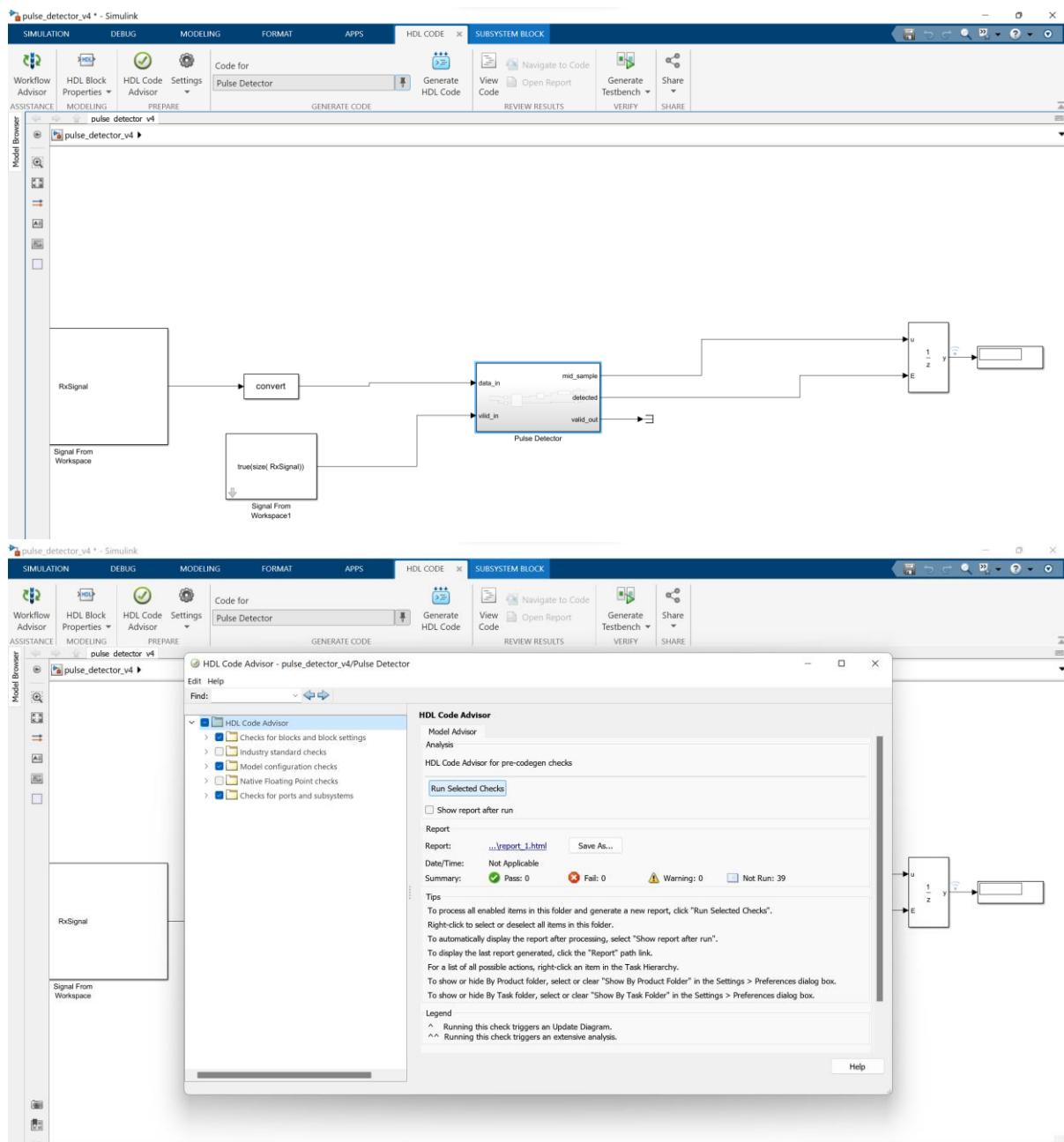
Peak location = 4455, mag-squared = 8.307e-02 using local max

mag-squared from Simulink = 8.301e-02, error = 6.353e-05

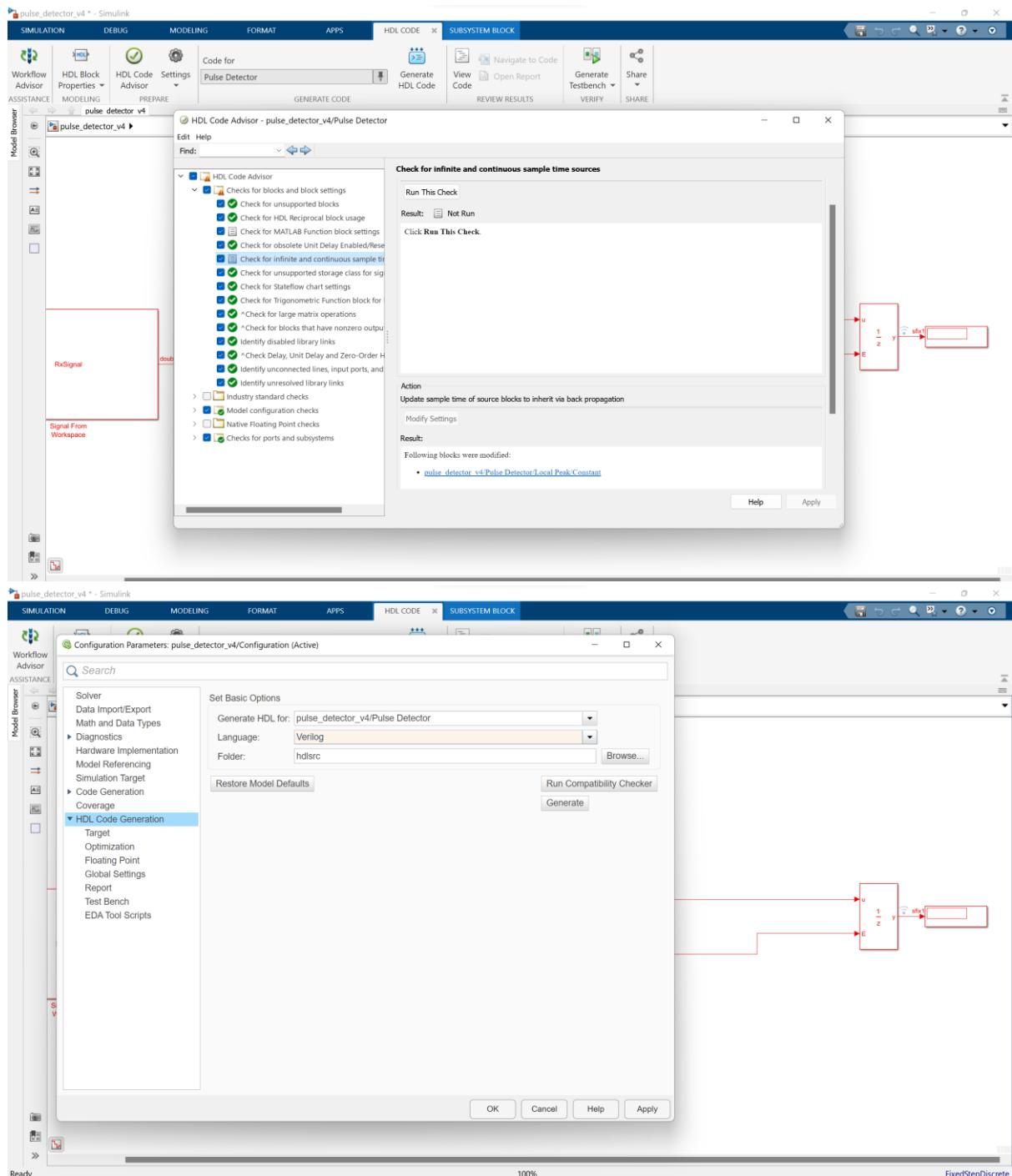
#### ۴-۱ تولید کد وریلگ و سنتز

مطابق مراحل زیر مازول Pulse Detector را به کد وریلگ تبدیل و سسپ با استفاده از vivado سنتز می کنیم و پروژه می سازیم:

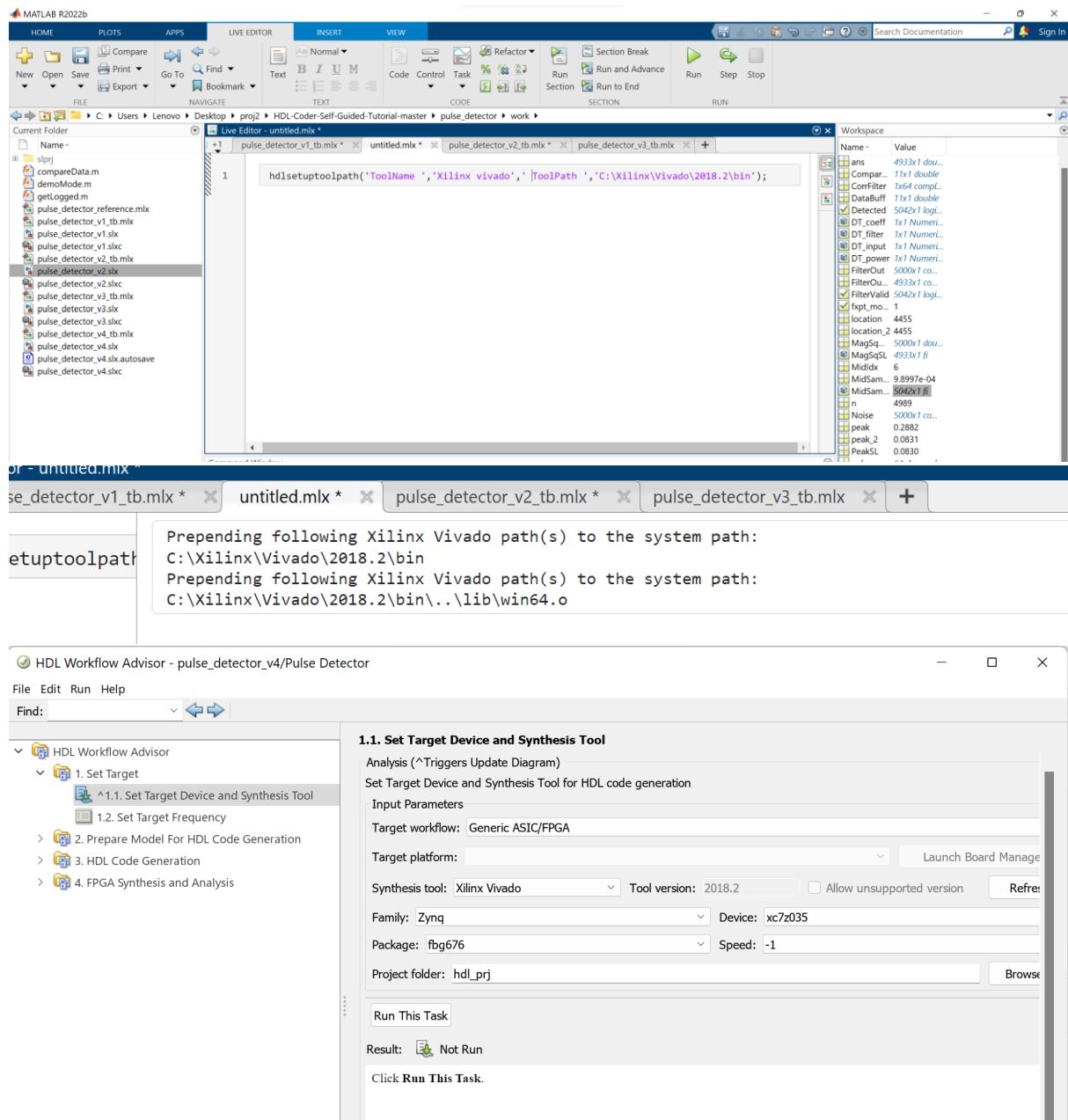
## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس



## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس



## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس



## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس

Configuration Parameters: pulse\_detector\_v4/Configuration (Active)

Solver  
Data Import/Export  
Math and Data Types  
▶ Diagnostics  
Hardware Implementation  
Model Referencing  
Simulation Target  
▶ Code Generation  
Coverage  
▼ HDL Code Generation  
Target  
Optimization  
Floating Point  
**Global Settings**  
Report  
Test Bench  
EDA Tool Scripts

Clock settings

Reset type:	Synchronous	Reset asserted level:	Active-high
Clock input port:	clk	Clock enable input port:	clk_enable
Reset input port:	reset	Clock inputs:	Single
Oversampling factor:	1	Clock edge:	Rising

Additional settings

General	Ports	Coding style	Coding standards	Comments	Model Generation	Advanced
Verilog file extension:	.v	VHDL file extension:	.vhd			
Entity conflict postfix:	_block	Package postfix:	_pkg			
Reserved word postfix:	_rsvd	Split entity file postfix:	_entity			
Clocked process postfix:	_process	Split arch file postfix:	_arch			
Complex real part postfix:	_re	Split entity and architecture				
Complex imaginary part postfix:	_im	VHDL architecture name:	rtl			
Enable prefix:	enb	Module name prefix:	<empty>			
		Timing controller postfix:	tc			

Configuration Parameters: pulse\_detector\_v4/Configuration (Active)

Solver  
Data Import/Export  
Math and Data Types  
▶ Diagnostics  
Hardware Implementation  
Model Referencing  
Simulation Target  
▶ Code Generation  
Coverage  
▼ HDL Code Generation  
Target  
Optimization  
Floating Point  
**Global Settings**  
**Report**  
Test Bench  
EDA Tool Scripts

Traceability Reports

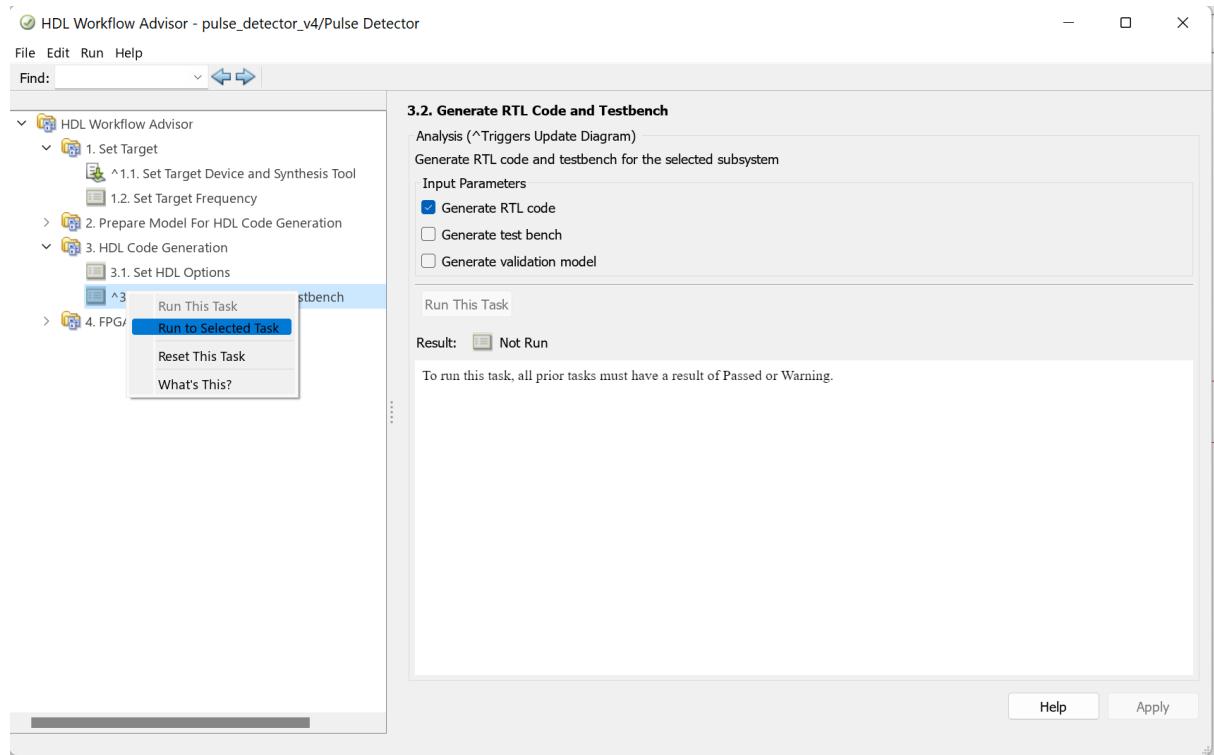
<input type="checkbox"/> Generate traceability report
Traceability style: Line Level
<input type="checkbox"/> Generate model Web view

Optimization Reports

<input checked="" type="checkbox"/> Generate resource utilization report
<input checked="" type="checkbox"/> Generate optimization report

Timing Reports

<input type="checkbox"/> Generate high-level timing critical path report
Custom Timing Database Directory: <empty> <input type="button" value="Browse"/>



نتایج این سنتز که در خود Simulink و با استفاده از vivado انجام شده به صورت زیر است(در ادامه در خود vivado هم سنتز خواهیم کرد):

Code Generation Report

Find: Match Case

**Contents**

- [Summary](#) **Summary**
- [Clock Summary](#)
- [Code Interface Report](#)
- Timing And Area Report
  - [High-level Resource Report](#) **High-level Resource Report**
- Optimization Report
  - [Distributed Pipelining](#)
  - [Streaming and Sharing](#)
  - [Delay Balancing](#)
  - [Adaptive Pipelining](#)
  - [Hierarchy Flattening](#)
  - [Target Code Generation](#)
  - [Code Reuse](#)

**Referenced Models**

HDL Code Generation Report Summary for pulse\_detector\_v4

### Summary

Model	<a href="#">pulse_detector_v4</a>
Model version	1.12
HDL Coder version	4.0
HDL code generated on	2023-07-02 17:41:52
HDL code generated for	<a href="#">Pulse Detector</a>
Target Language	Verilog
Target Directory	hdl_prj\hdlsrc

### Non-default model properties

Backannotation	on
HDLSubsystem	pulse_detector_v4/Pulse Detector
OptimizationReport	on
ResetType	Synchronous
ResourceReport	on
SynthesisTool	Xilinx Vivado
SynthesisToolChipFamily	Zynq
SynthesisToolDeviceName	xc7z035
SynthesisToolPackageName	fbg676
SynthesisToolSpeedValue	-1

OK Help

Code Generation Report

Find: Match Case

**Contents**

- [Summary](#)
- [Clock Summary](#)
- [Code Interface Report](#)
- Timing And Area Report
  - [High-level Resource Report](#) **High-level Resource Report**
- Optimization Report
  - [Distributed Pipelining](#)
  - [Streaming and Sharing](#)
  - [Delay Balancing](#)
  - [Adaptive Pipelining](#)
  - [Hierarchy Flattening](#)
  - [Target Code Generation](#)
  - [Code Reuse](#)

**Referenced Models**

Generic Resource Report for pulse\_detector\_v4

### Summary

Multipliers	194
Adders/Subtractors	207
Registers	1392
Total 1-Bit Registers	30827
RAMs	0
Multiplexers	1347
I/O Bits	57
Static Shift operators	0
Dynamic Shift operators	0

### Detailed Report

---

Report for Subsystem: [Pulse Detector](#)

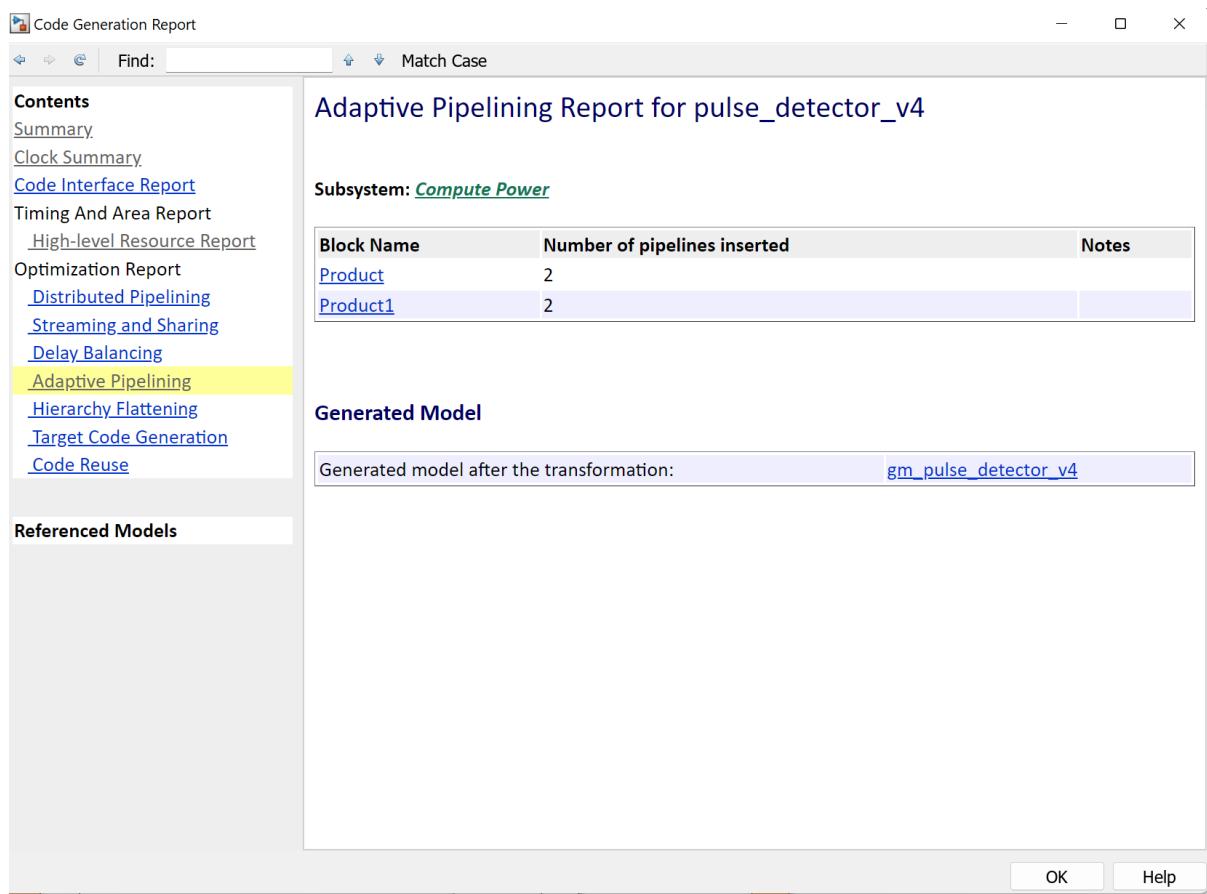
Multipliers (192)

17x19-bit Multiply : 192

Adders/Subtractors (195)

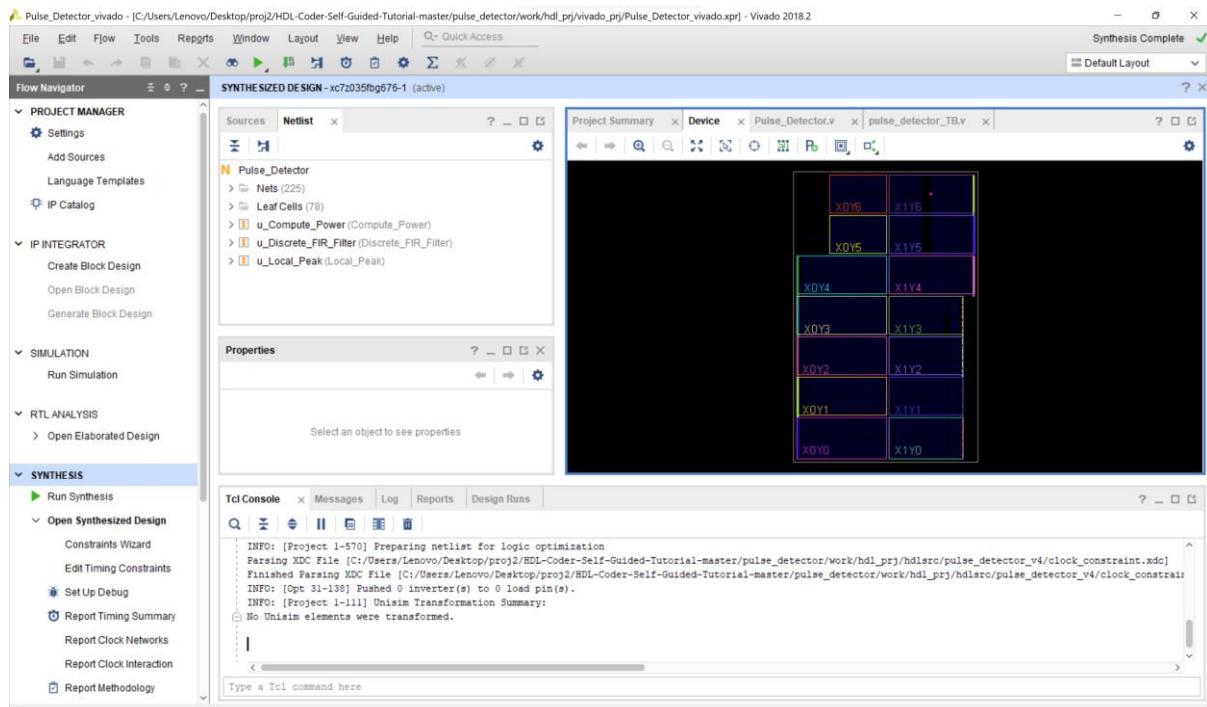
OK Help

## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس



## ۲ سنتز و شبیه سازی با استفاده از vivado

ابتدا پروژه ساخته شده را در vivado باز می کنیم و سنتز می کنیم:



## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس

نتایج سنتر به صورت زیر است:

**Tile Properties**

PSS2_X32Y313	
Name:	PSS2_X32Y313
Type:	PSS2
Row:	51
Column:	32
Number of cell pins:	0
Number of cells:	0
Number of ports:	0
Number of BELs:	391
Number of sites:	131
Number of nodes:	10,864
Number of switchboxes:	0
Number of clock regions:	2

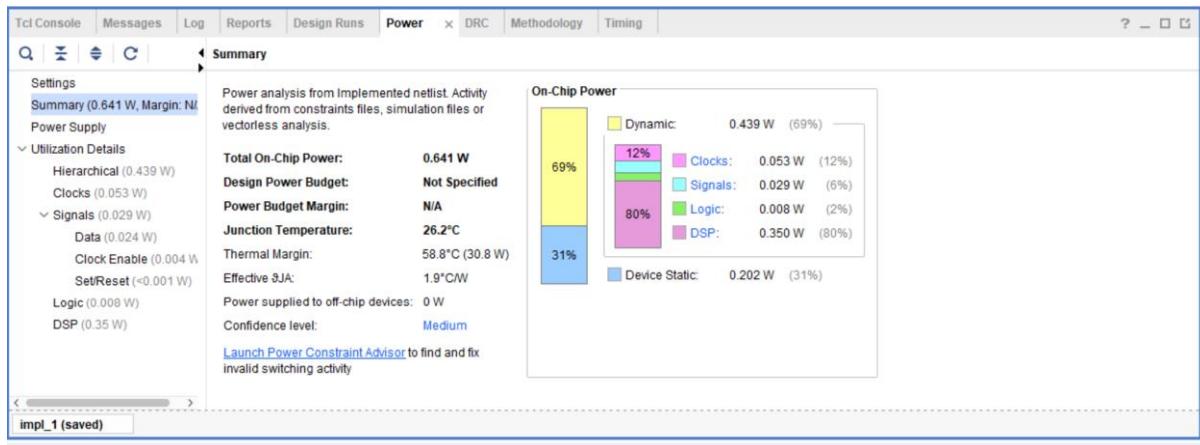
  

**Tile Properties**

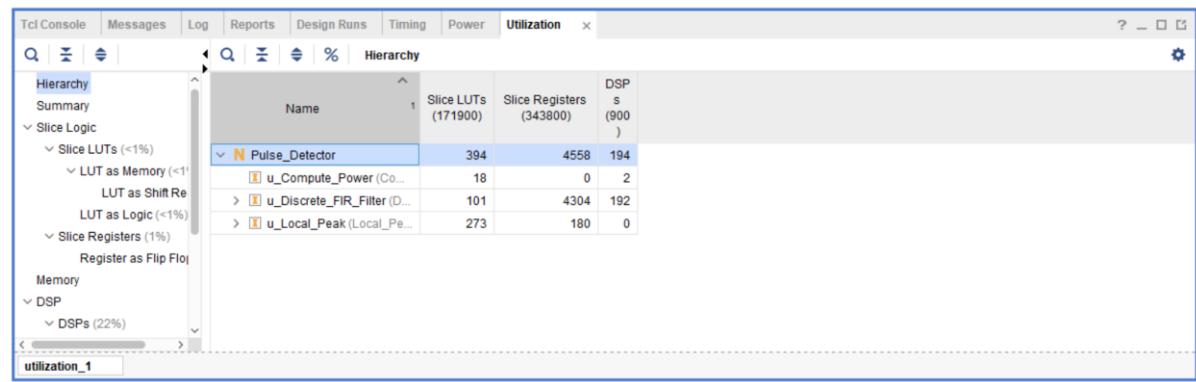
PSS2_X32Y313	
CLASS	tile
COLUMN	32
DEVICE_ID	0
FIRST_SITE_ID	11415
GRID_POINT_X	32
GRID_POINT_Y	51
INDEX	13649
INT_TILE_X	-1
INT_TILE_Y	-1
IS_CENTER_TILE	
IS_DCM_TILE	
IS_GT_CLOCK_SITE_TILE	
IS_GT_SITE_TILE	
NAME	PSS2_X32Y313
NUM_ARCS	3712
NUM_SITES	131
ROW	51
SLR_REGION_ID	0
TILE_PATTERN_IDX	1908
TILE_TYPE	PSS2
TILE_TYPE_INDEX	116
TILE_X	-187522
TILE_Y	403692
TYPE	PSS2

## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس

ریپورت توان مصرفی و power



ریپورت منابع مصرف شده:



Name	Slice LUTs (171900)	Slice Registers (343800)	DSP s (900 )
脉冲检测器 (Pulse_Detector)	394	4558	194
本地峰值检测 (u_Local_Peak)	273	180	0
MATLAB 函数 (u_MATLAB_Function)	273	0	0
离散 FIR 滤波器 (u_Discrete_FIR_Filter)	101	4304	192
滤波器银行 (u_FilterBank)	101	4304	192
子滤波器 1 (u_subFilter_1)	41	2056	64
滤波器带 9 (u_FilterTap_9)	0	32	1
滤波器带 8 (u_FilterTap_8)	0	32	1
滤波器带 7 (u_FilterTap_7)	0	32	1
滤波器带 64 (u_FilterTap_64)	0	0	1
滤波器带 63 (u_FilterTap_63)	1	32	1
滤波器带 62 (u_FilterTap_62)	0	32	1
滤波器带 61 (u_FilterTap_61)	0	32	1
滤波器带 60 (u_FilterTap_60)	0	32	1
滤波器带 6 (u_FilterTap_6)	0	32	1
滤波器带 59 (u_FilterTap_59)	0	32	1
滤波器带 58 (u_FilterTap_58)	0	32	1
滤波器带 57 (u_FilterTap_57)	0	32	1
滤波器带 56 (u_FilterTap_56)	0	32	1
滤波器带 55 (u_FilterTap_55)	0	32	1
滤波器带 54 (u_FilterTap_54)	0	32	1
滤波器带 53 (u_FilterTap_53)	0	32	1
滤波器带 52 (u_FilterTap_52)	0	32	1
滤波器带 51 (u_FilterTap_51)	0	32	1

Name	Slice LUTs (171900)	Slice Registers (343800)	DSP s (900 )
u_FilterTap_12 (...	0	32	1
u_FilterTap_11 (...	0	32	1
u_FilterTap_10 (...	0	32	1
u_FilterTap_1 (F...	0	32	1
u_subFilter_1_reP...	40	2057	64
u_FilterTap_9 (F...	0	32	1
u_FilterTap_8 (F...	0	32	1
u_FilterTap_7 (F...	0	32	1
u_FilterTap_64 (...	0	0	1
u_FilterTap_63 (...	0	32	1
u_FilterTap_62 (...	0	32	1
u_FilterTap_61 (...	0	32	1
u_FilterTap_60 (...	0	32	1
u_FilterTap_6 (F...	0	32	1
u_FilterTap_59 (...	0	32	1
u_FilterTap_58 (...	0	32	1
u_FilterTap_57 (...	0	32	1
u_FilterTap_56 (...	0	32	1
u_FilterTap_55 (...	0	32	1
u_FilterTap_54 (...	0	32	1
u_FilterTap_53 (...	0	32	1
u_FilterTap_52 (...	0	32	1
u_FilterTap_51 (...	0	32	1
u_FilterTap_50 (...	0	32	1

تست بنج نوشته شده به صورت زیر است:

```

module pulse_detector_TB();
reg clk;
reg reset;
reg clk_enable;
reg signed [15:0] data_in_re; // sfix16_En14
reg signed [15:0] data_in_im; // sfix16_En14
reg vild_in;
wire ce_out;
wire signed [17:0] mid_sample; // sfix18_En11
wire detected;
wire valid_out;
Pulse_Detector PD(clk,reset,clk_enable,data_in_re,data_in_im,vild_in,ce_out,mid_sample,detected,valid_out);
initial clk<=1'b0;

initial
begin
clk_enable<=1;
#10 reset=1'b0;
data_in_re<=12;
data_in_im<=10;
vild_in<=1;
#5 clk<=~clk;
#10 $stop;
end
endmodule

```

نتایج این تست بنج و خروجی مازول تولید شده در Simulink یکسان است و این با انتظار ما تطابق دارد زیرا ما چیزی از این مازول کم نکرده و دقیقاً مانند مازول ساخته شده در Simulink است پس طبیعی است که یک خروجی به ازای یک ورودی داشته باشند.

### ۳ کاهش دقیق و سنتز و شبیه سازی دوباره

برای کاهش دقت می توانیم بیت ورودی و خروجی را کم کنیم تا باعث کم شدن حجم کلی عملیات های ماژول شویم و در نهایت ماژول کمی بینه تر شده باشد. برای این کار به صورت زیر عمل کردیم:

Pulse\_Detector.v \* x pulse\_detector\_TB.v \*

C:/Users/Lenovo/Desktop/proj2/HDL-Coder-Self-Guided-Tutorial-master/pulse\_detector/work/hdl\_prj/hdlsrc/pulse\_detector\_v4/Pulse\_

```
49         ce_out,
50         mid_sample,
51         detected,
52         valid_out);
53
54
55     input  clk;
56     input  reset;
57     input  clk_enable;
58     input  signed [7:0] data_in_re; // sfix16_En14
59     input  signed [7:0] data_in_im; // sfix16_En14
60     input  valid_in;
61     output ce_out;
62     output signed [10:0] mid_sample; // sfix18_En11
63     output detected;
64     output valid_out;
65
66
67     wire enb;
68     reg signed [7:0] Delay_outl_re; // sfix16_En14
69     reg signed [7:0] Delay_outl_im; // sfix16_En14
70     reg Delayl_outl;
71     wire signed [19:0] filter_out_re; // sfix40_En37
72     wire signed [19:0] filter_out_im; // sfix40_En37
73     wire filter_valid;
74     wire signed [10:0] mag_sq_out; // sfix18_En11
75     wire signed [10:0] mid_sample_l; // sfix18_En11
76     wire detected_l;
77     reg signed [10:0] Delay2_outl; // sfix18_En11
78     reg Delay3_outl;
79     reg Delay4_outl;
80     reg [0:1] delayMatch_reg; // ufix1 [2]
81     wire [0:1] delayMatch_reg_next; // ufix1 [2]
82     wire Delay4_outl_l;
```

پار دیگر سنتز می کنیم و به نتایج زیر دست پیدا می کنیم:

## پروژه درس طراحی سیستم دیجیتال - آشکار ساز پالس

ریپورت پاور:



همان طور که قابل مشاهده است total on-chip power تقریبا به میزان ۰،۰۴ وات (درصد) کم شده است.

:utilization ریپورت

Name	1	Slice LUTs (171900)	Slice Registers (343800)	DSPs (900)
脉冲检测器 (Pulse_Detector)	346	2322	194	2
u_CompPower (C...)	18	0	2188	192
u_Discrete_FIR_Filter ...	53	273	100	0
u_Local_Peak (Local_...)				

همان طور که مشخص است استفاده از منابع به میزان قابل توجهی از مازول اصلی کمتر است که این نشان دهنده بهینه بودن مازول جدید است.

برای این مورد موفق به نوشتن تست بنج نشديم اما می توان حدس زد که دقت ما باید پایین آمده باشد زیرا بخش خوبی از لاجیک کم شده و در نتیجه دقت محاسبات پایین خواهد آمد که منجر به دور شدن جواب نهایی تولید شده با جواب نهایی اصلی خواهد شد.