Description

In this example, we will build a host-target (master-slave) communication system to exchange a row vector—multiple scalar signals grouped as arrays (e.g., [1×3])—as illustrated in the diagram below.

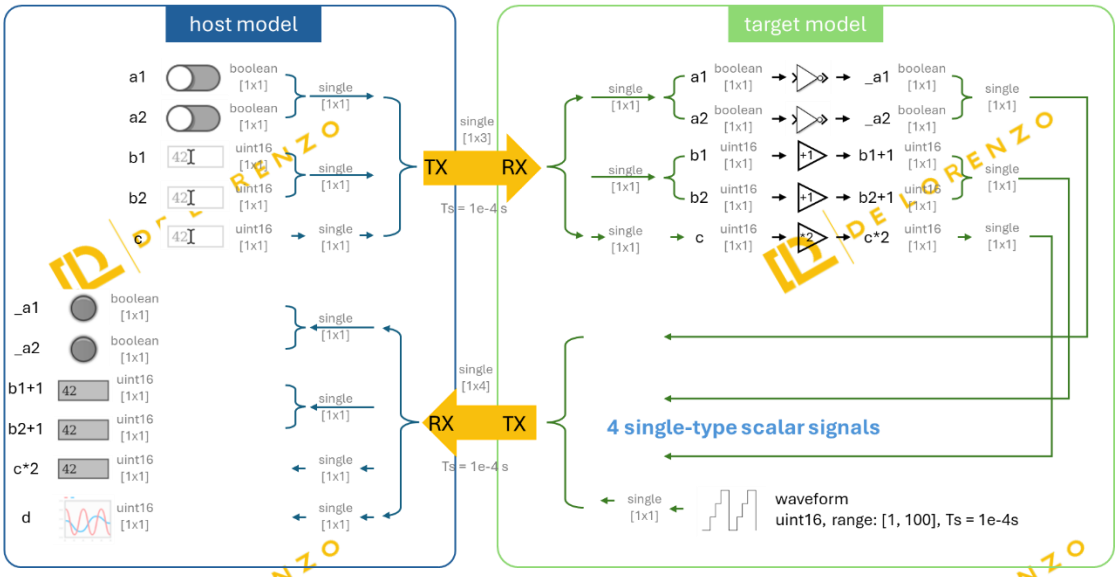
The host model running in the master (PC) will package and transmit data to the target model running in the slave device (DL RCPCORE), including 2 Boolean scalar values, 3 uint16 scalar values and 1 one-way (target-host) uint16 scalar value.

The DL RCPCORE (slave) will unpack the received data, perform simple processing, then package and return the processed data along with a discrete sawtooth waveform back to the master.

The master will then receive and unpack the data, display the results, and verify the correctness of the communication.

TX/RX Timing & Signal Dimensions

The host model performs a TX [1x3] at regular intervals of 1e-4 s;
The target model performs RX [1x3] at regular intervals of 1e-4 s;
The target model performs TX [1x4] at regular intervals of 1e-4 s;
The host model performs RX [1x4] at the same interval as the target TX, i.e. 1e-4 s.



Learning Objectives

- how to set up a host-target communication system that supports multiple data types and multiple signals
- understand and differentiate the host model and the target model through hands-on practice
- data serialization and parsing
- understand the significance of time in communication
- dashboard development

Prior Knowledge

- Create New Project
- Control Model Settings
- Host-Target Communication
  - Fundamentals
  - Scalar Demo

Demo Models

If you want to run the demo directly, please execute the following command:

openDLDemo('comm_vector')

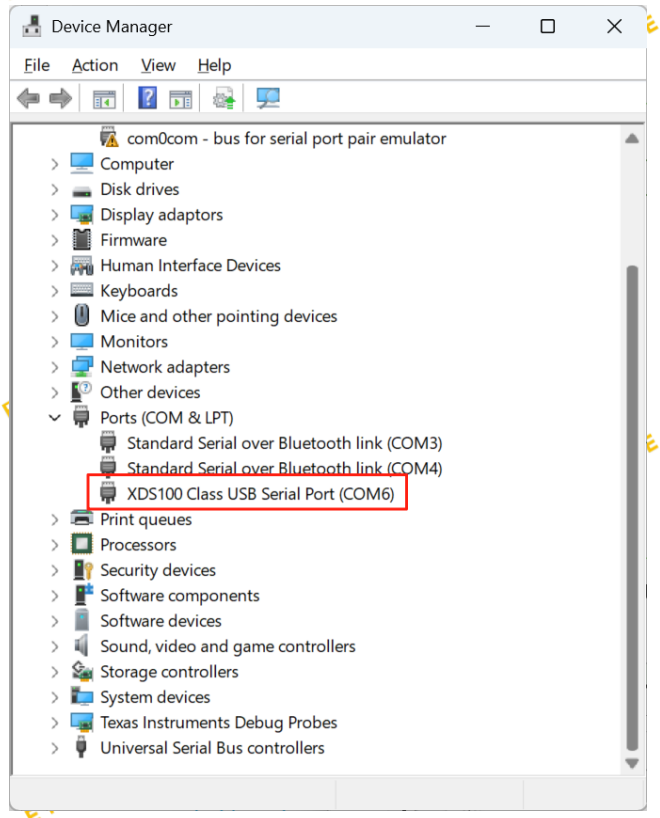
Step 0. Model creation & parameter settings

Create new host and target models using the *CreateNew* command.

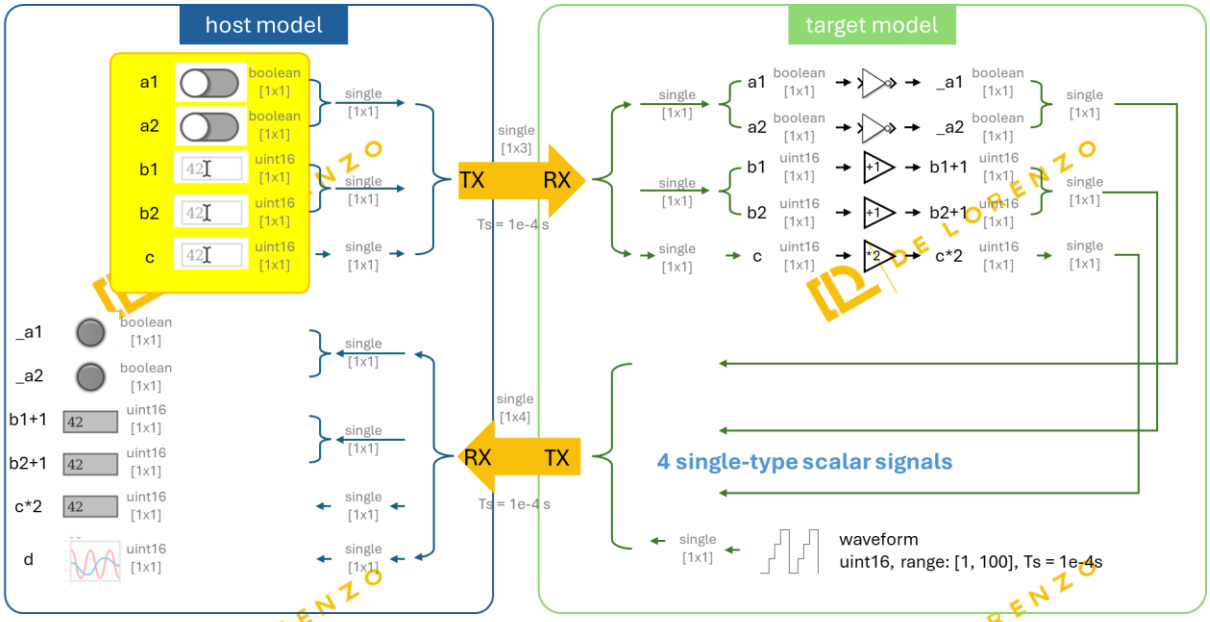
Click on ' 📌 Click here to edit model parameters' to edit the parameters.

Set the model base sample time **Ts** (which here corresponds to the transmission interval) and the **COM port** recognized by the hardware in Device Manager.

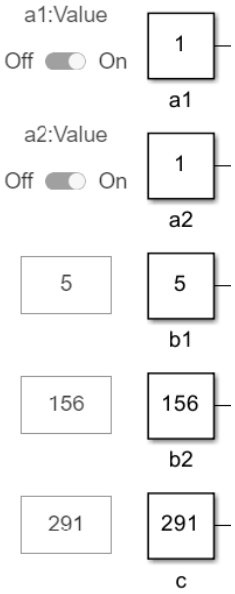
```
%% Set PWM Switching frequency
PWM_frequency    = 4e3;          %Hz          // Converter
T_pwm            = 1/PWM_frequency;  %s  // PWM switching
deadtime         = 5e-6;         %s //rising edge delay fo

%% Model Parameters
Ts               = 1e-4;         %s      // Sample time for
dataType         = 'single';             % Floating point

%% Hareware parameters
% Set controller parameters
target           = SetControllerDetails('DL RCPCORE', PWM_
target.comport   = 'COM6';       % Update the appropriate
converter        = SetConverterParameters('DL 2106T06');
```



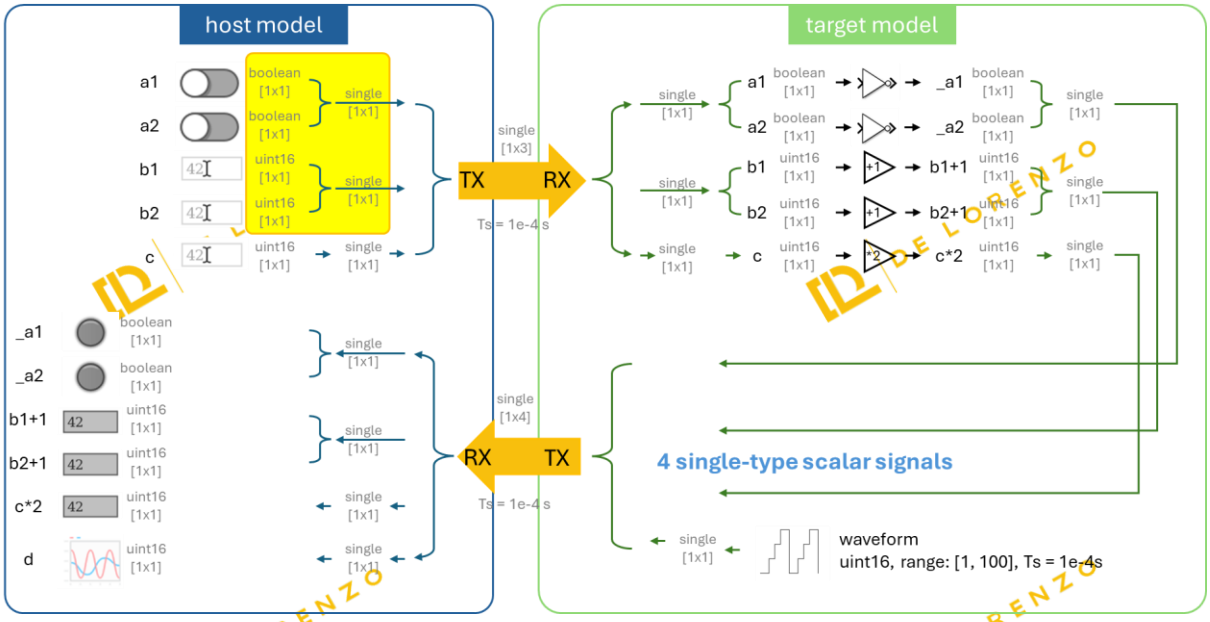
Step 1. Create controlled variables



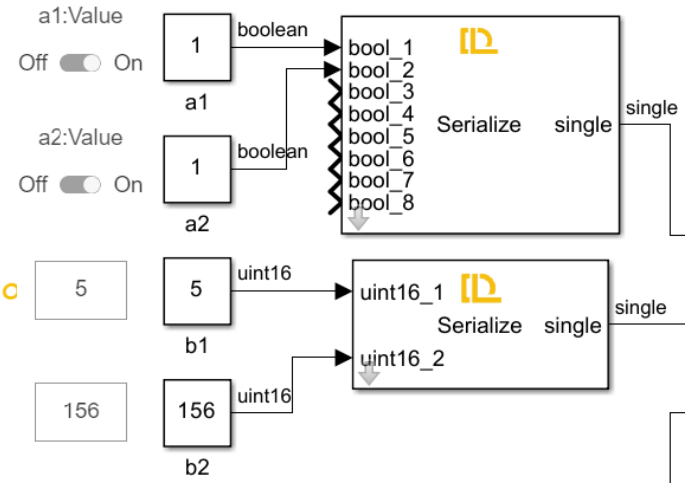
First, we add the corresponding control variables to the host model according to the requirements.



Step 2. Serialize multiple same-type signals into a single signal

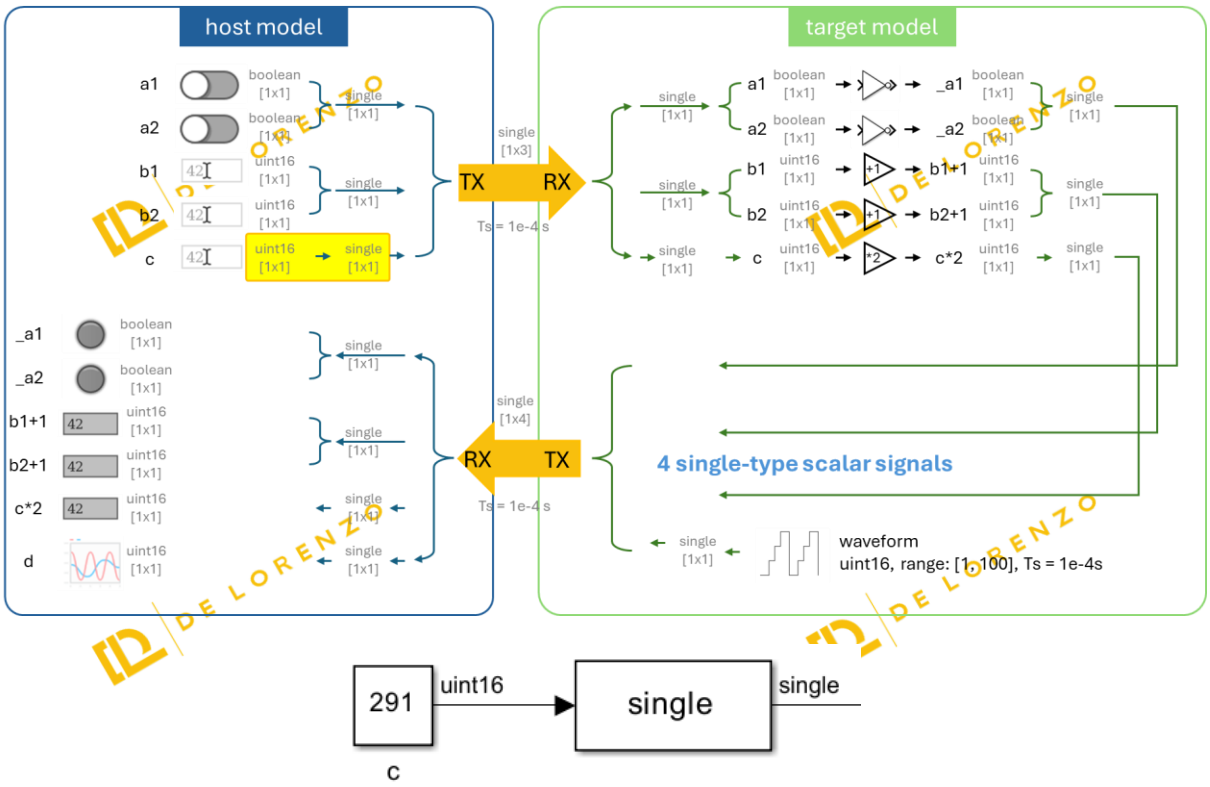


Use the **'Serialize Boolean into Single'** block and the **'Serialize Uint16 into Single'** block to combine multiple **Boolean** and **uint16** signals into **single**-precision values.

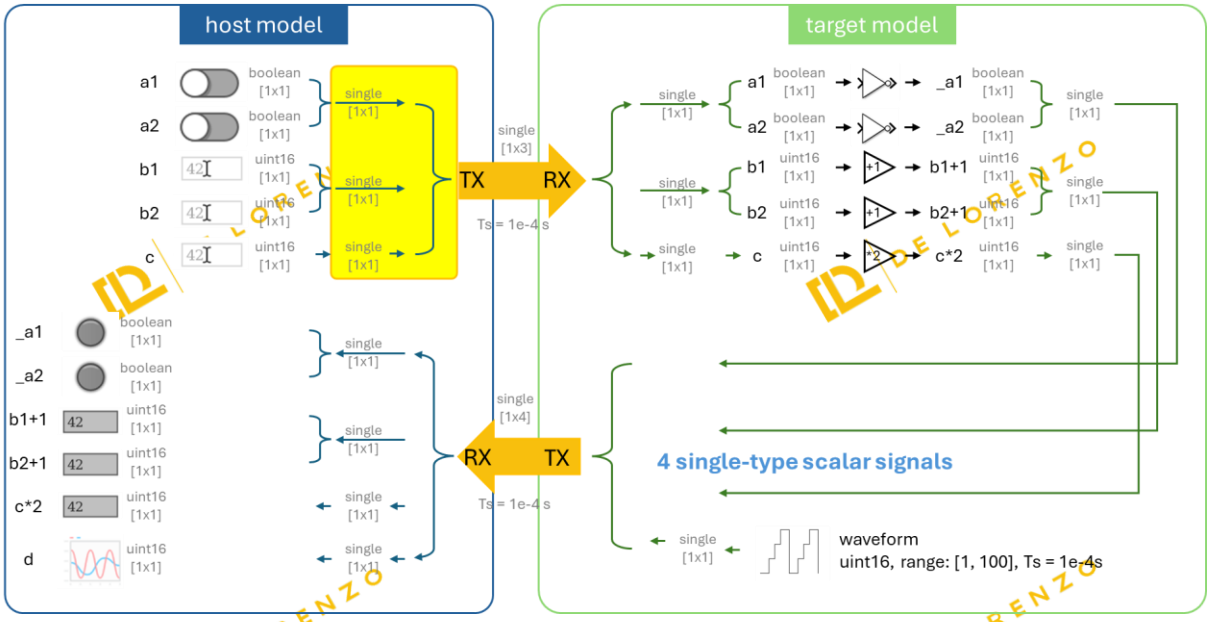


Step 3. Unify all types into single type

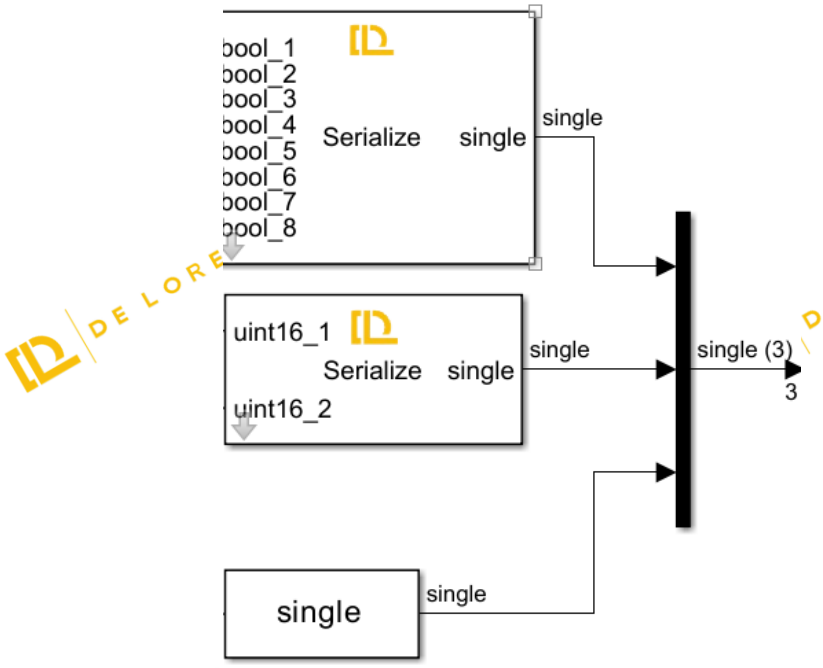
Unify all data types to single precision with the **'data type conversion'** block.



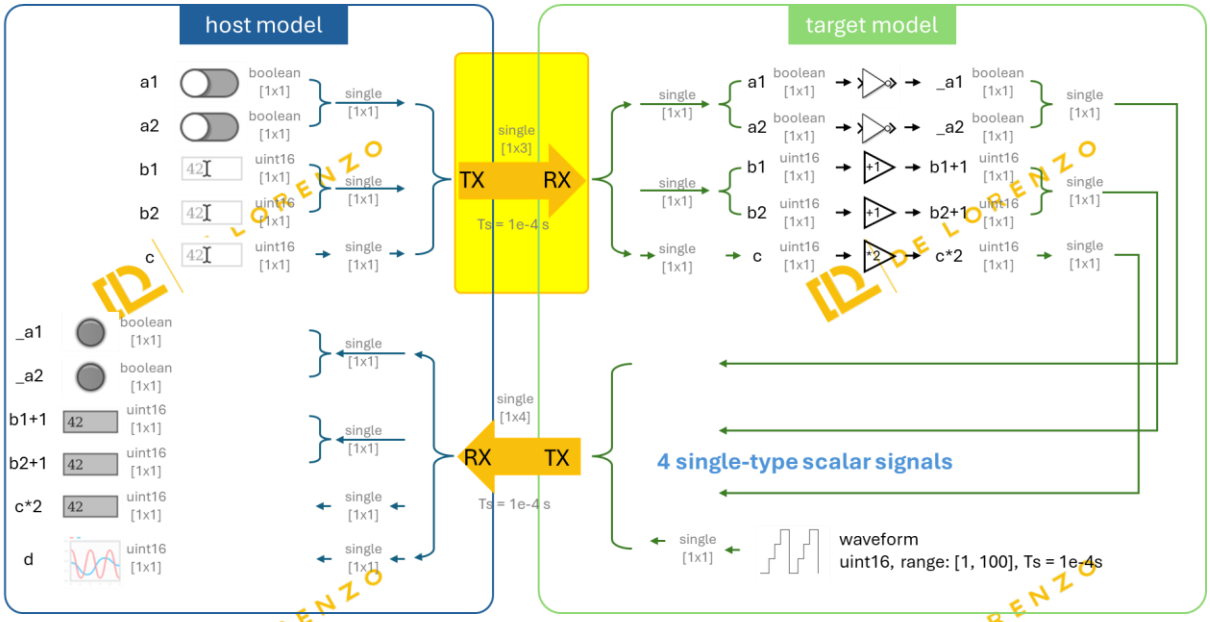
Step 4. Mux multiple signals of the same type



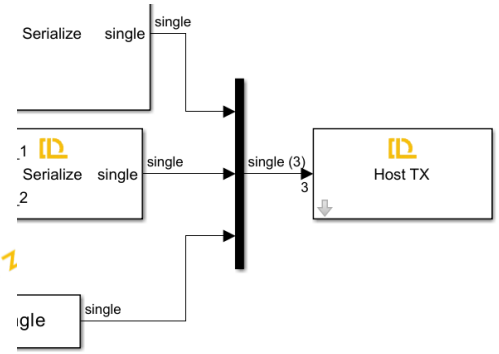
Use a **'Mux'** block to multiplex the three single-type signals into a [1×3] vector in preparation for transmission (TX).



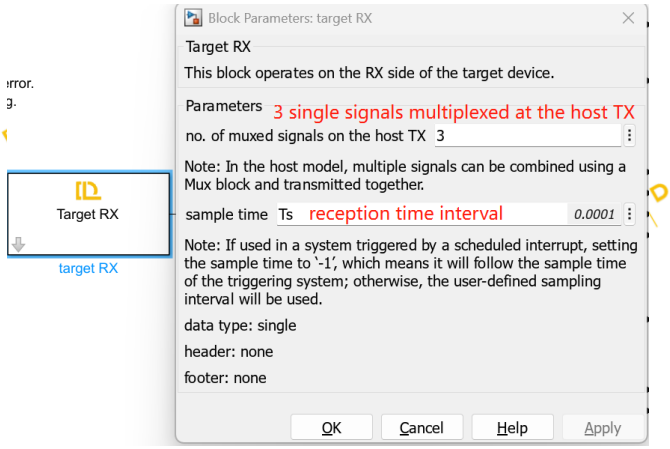
Step 5. Host-to-target transmission



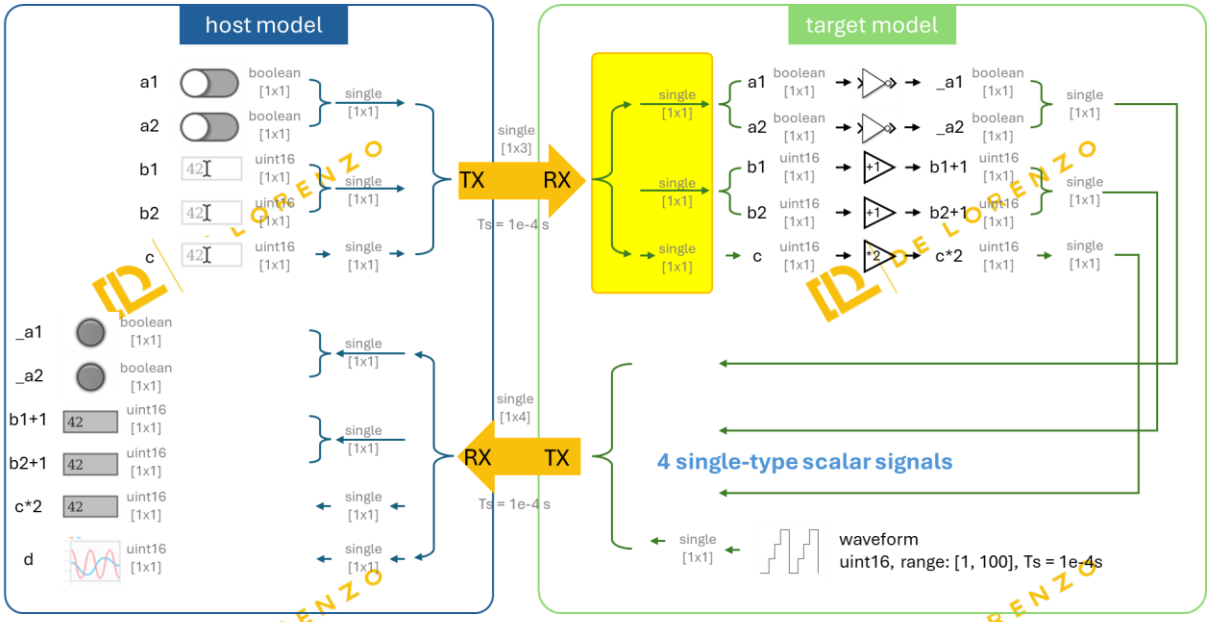
In the **host model**, place the **'Host TX'** block. The transmission rate will follow the sampling interval of the controlled variables.



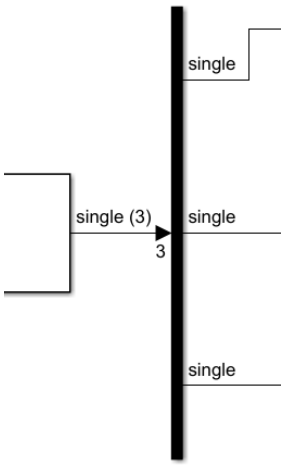
In the **target model**, place the **'Target RX'** block. Since a vector of [1x3] will be transmitted from the host model, the number of columns (i.e., the number of single-type signals) at the RX end on the target is set to 3. The reception interval is identical to the transmission interval.



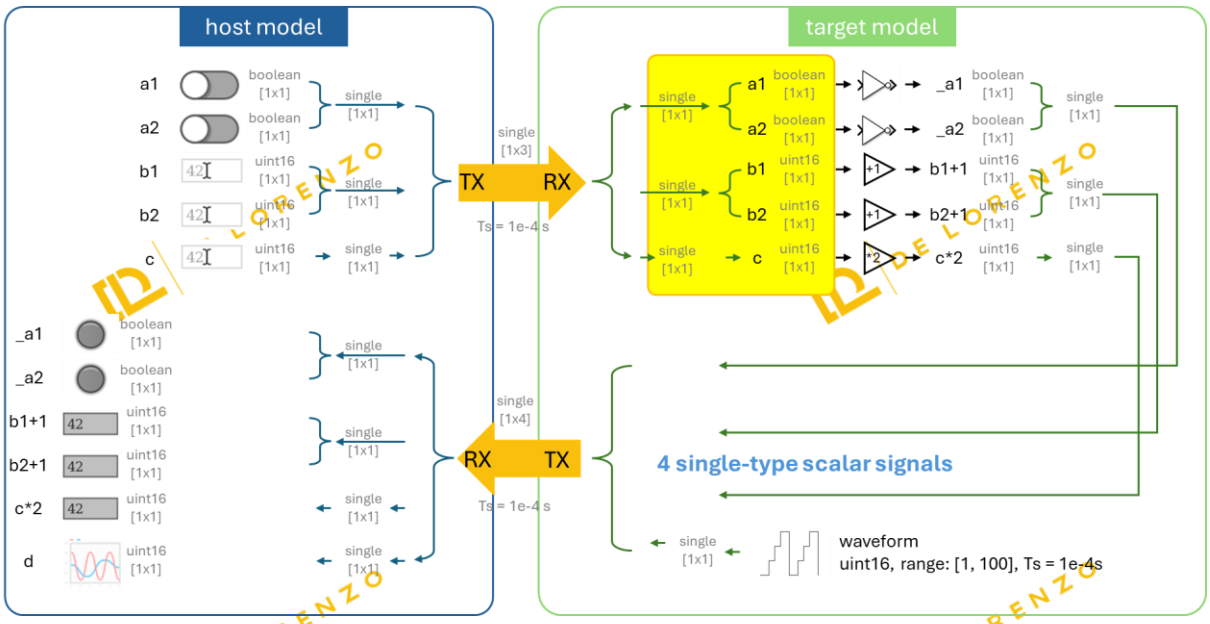
Step 6. Demux multiple signals of the same type



Use the **'Demux'** block to split a [1×3] vector into three [1×1] scalar signals.



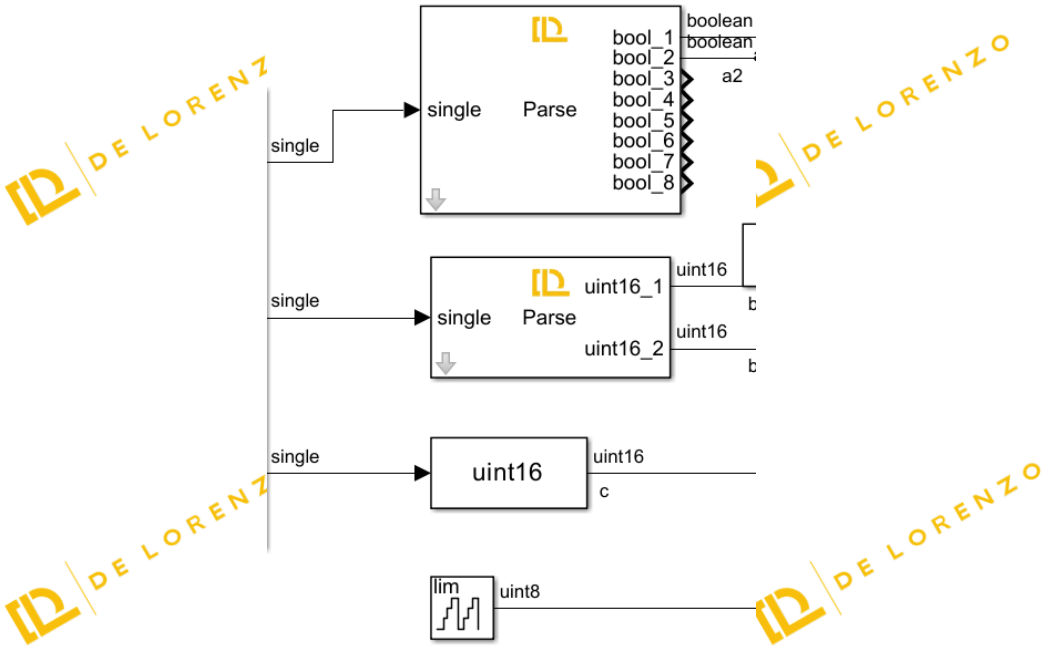
Step 7. Parse and data type convert to recover the controlled variables



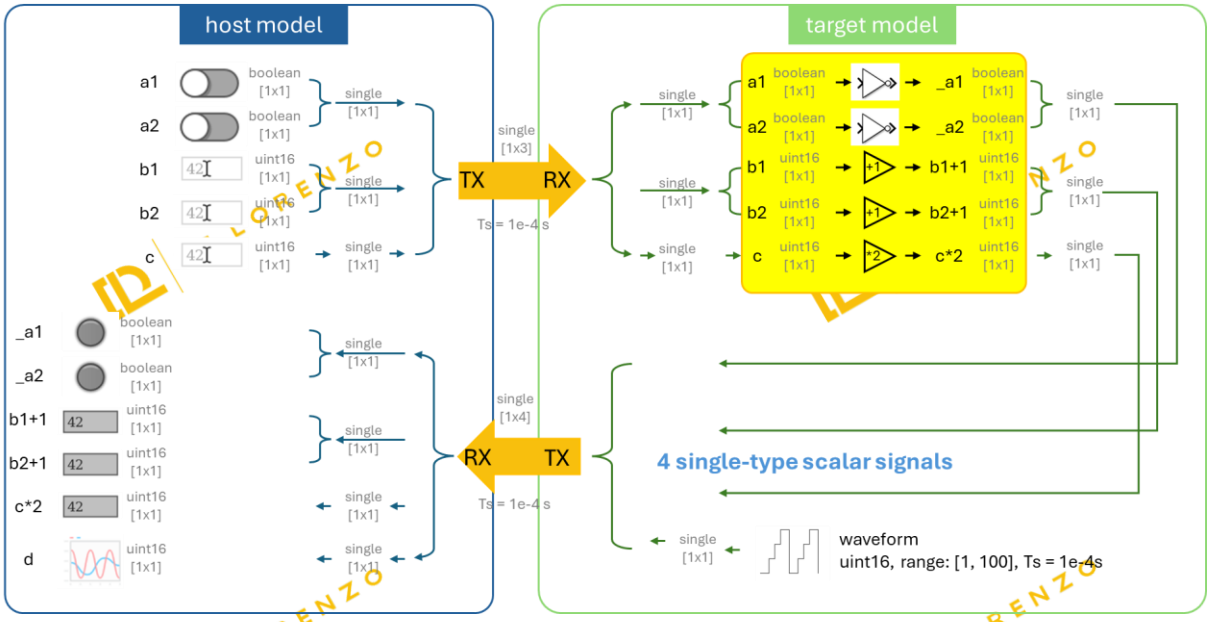
Use the **'Parse Single into Boolean'** block to convert a **single**-type signal into multiple **Boolean** signals and extract the controlled variables from the corresponding channels.

Use the **' Parse Single into Uint16'** block to convert a **single**-type signal into 2 **uint16** signals and extract the controlled variables from the corresponding channels.

Use the **'Data Type Conversion'** block to restore the data to its original type.



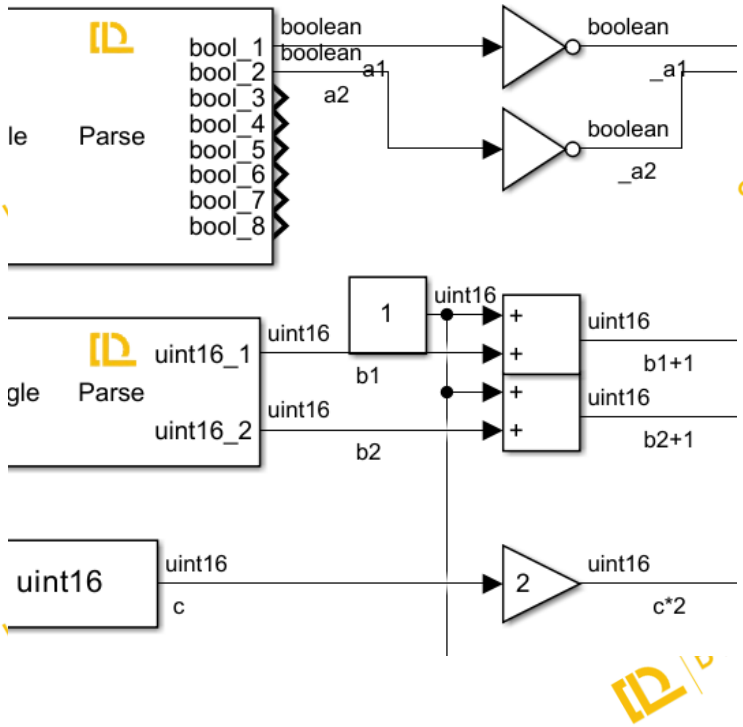
Step 8. Signal processing



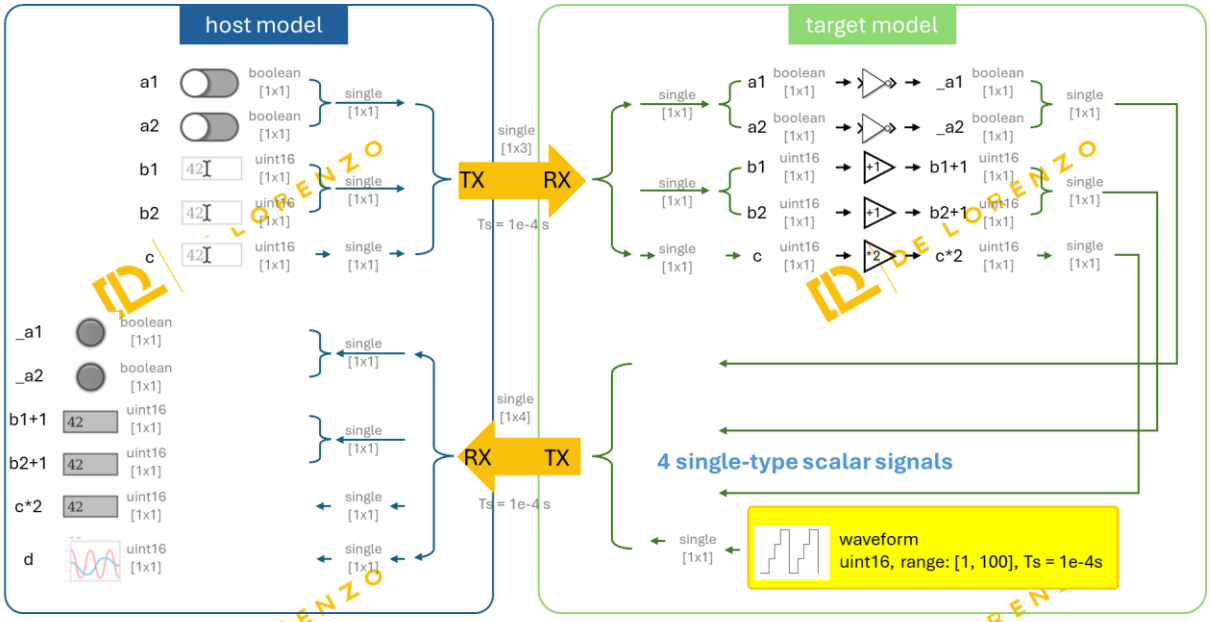
Invert Boolean signals a1 and a2 to obtain _a1 and _a2.

Add 1 to b1 and b2 to get b1+1 and b2+1.

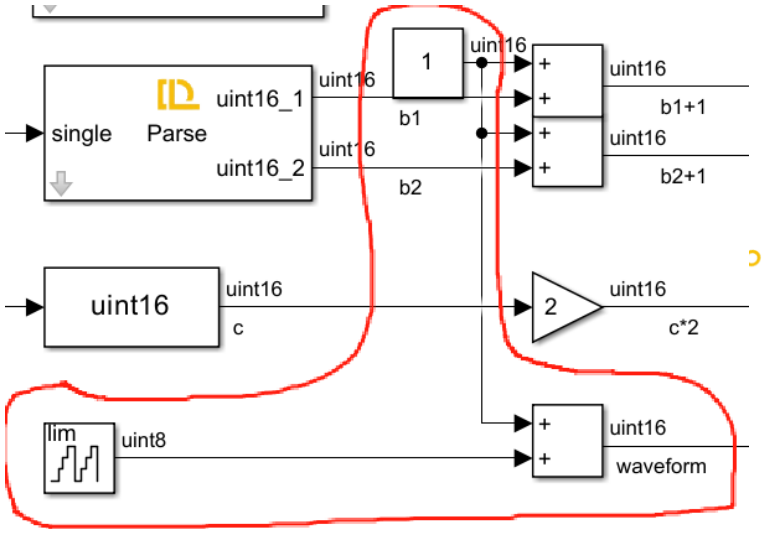
Multiply c by 2 to get c*2.



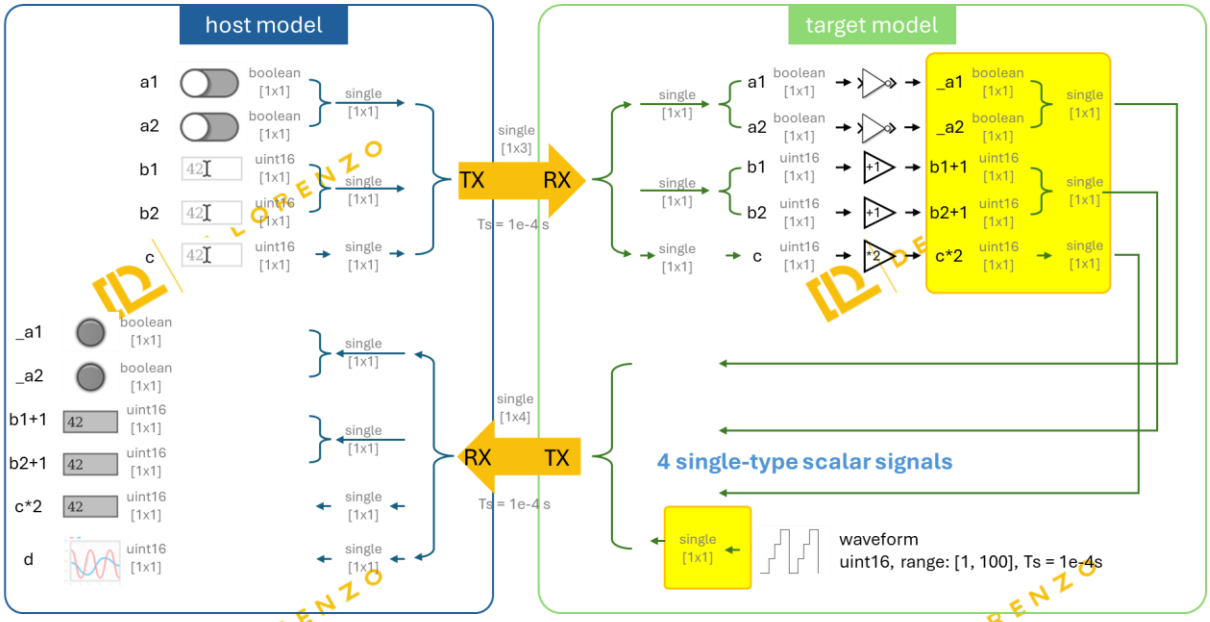
Step 9. Create the counter



Use the **'Counter Limited'** block to create a discrete sawtooth waveform ranging from 1 to 100 with sample time of 1e-4s. Convert the final output into single type to be consistent with the other serialized signals.

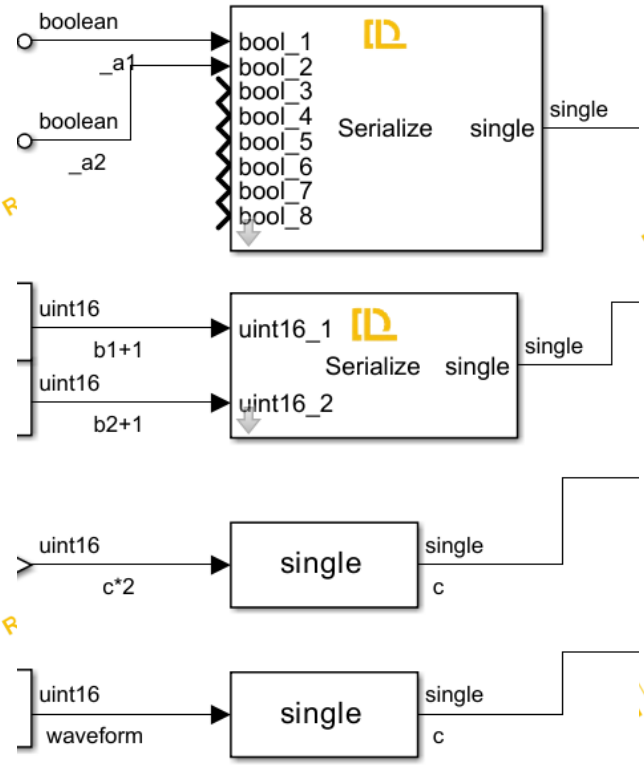


Step 10. Serialize and unify all types into single

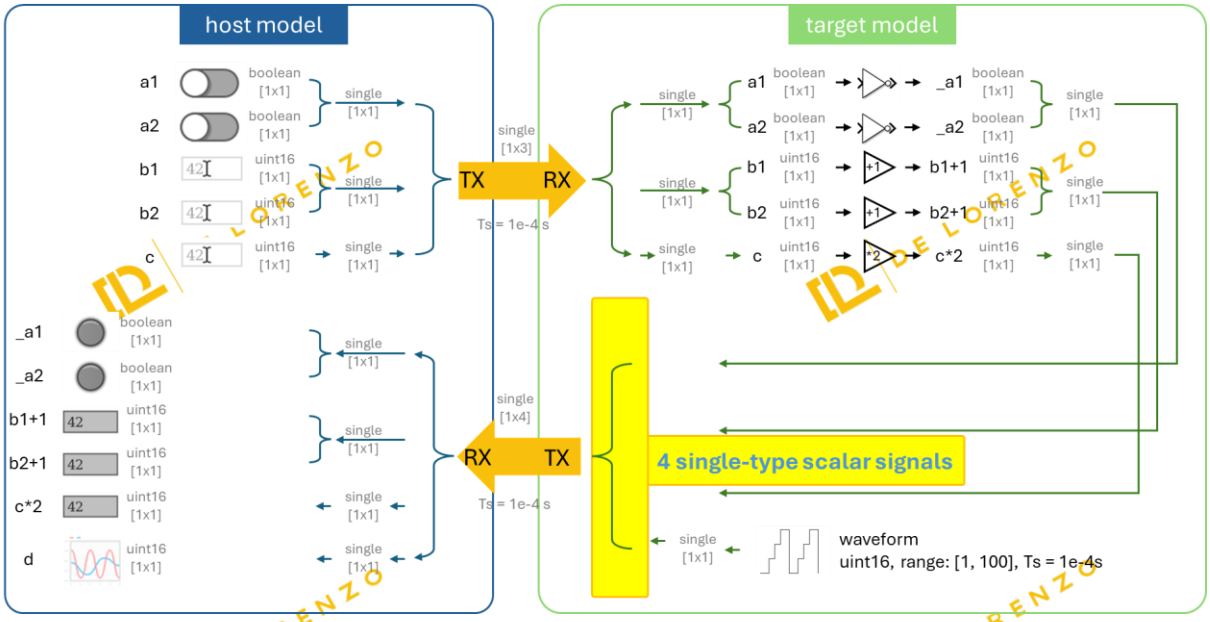


Use the **'Serialize Boolean into Single'** block and the **'Serialize Uint16 into Single'** block to combine multiple **Boolean** and **uint16** signals into **single**-precision values.

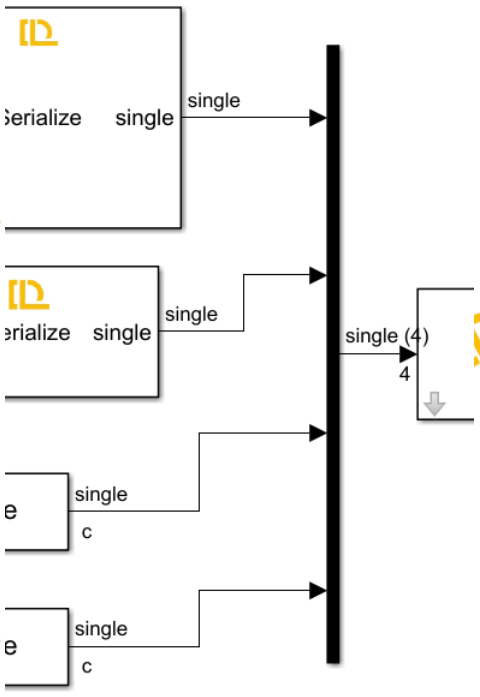
Unify all data types to single precision.



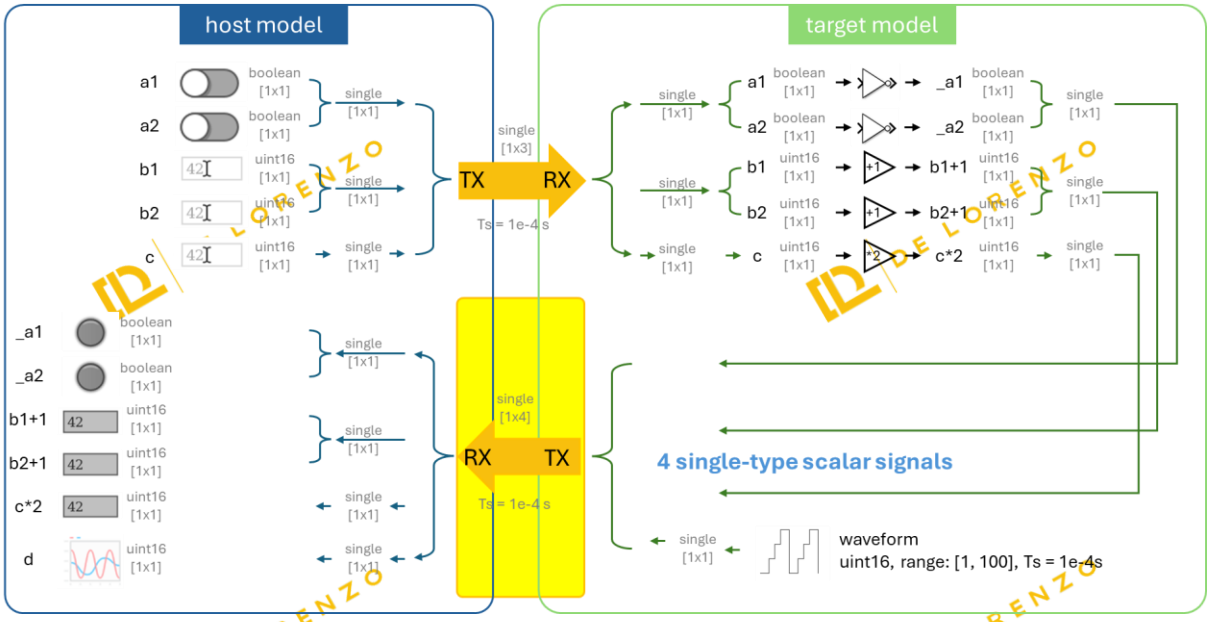
Step 11. Mux multiple signals of the same type



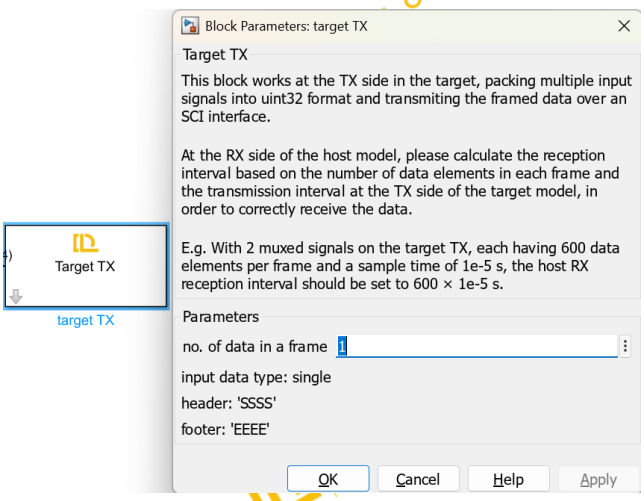
Use a **'Mux'** block to multiplex the four single-type signals into a [1×4] vector in preparation for transmission (TX).

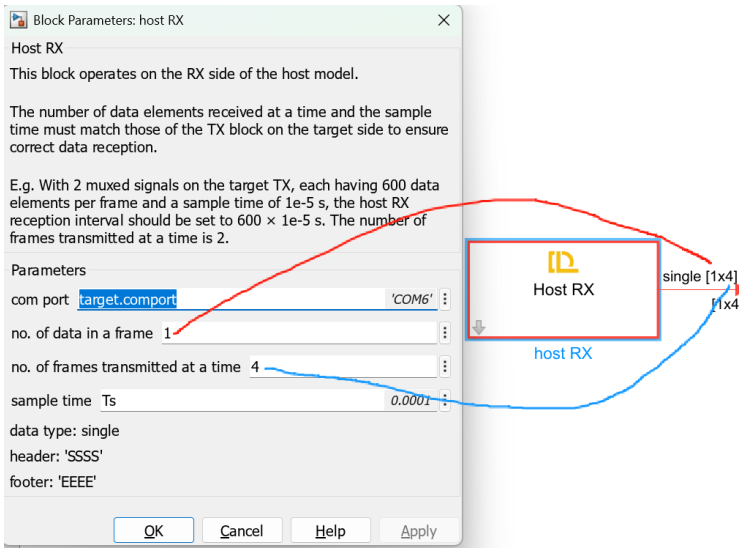


Step 12. Target-to-host transmission



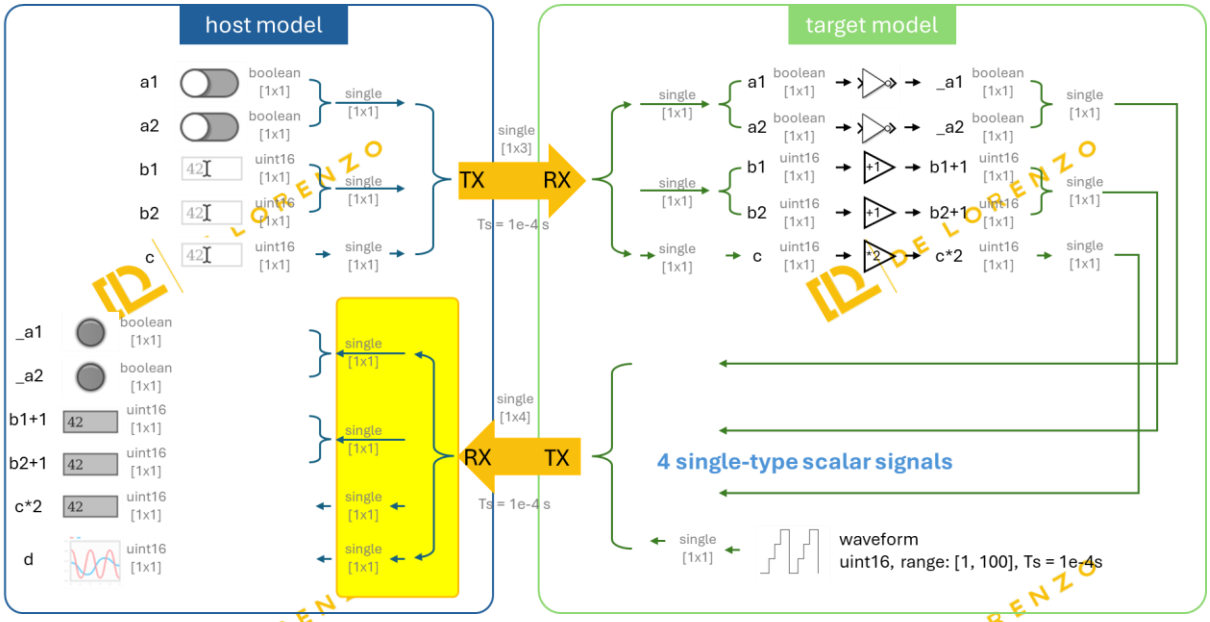
In the **target model**, place the **'Target TX'** block. The transmission rate will follow the target RX interval. Since the transmission and reception intervals on both the host and target sides are identical, and four single-scalar signals ([1×4]) are being transferred, the number of rows in the data—i.e., the amount of data per frame—is set to 1.



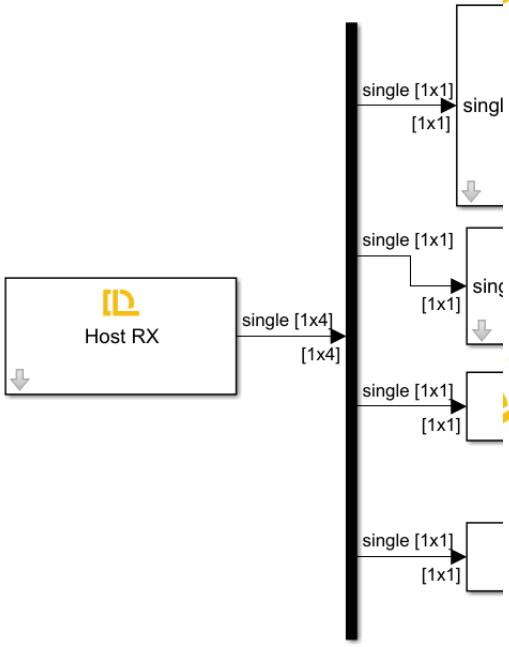


In the host model, place the **'Host RX'** block. For receiving four single-scalar signals ([1×4]), set the number of rows to 1 and the number of columns to 4. Ensure that the reception interval matches the transmission interval.

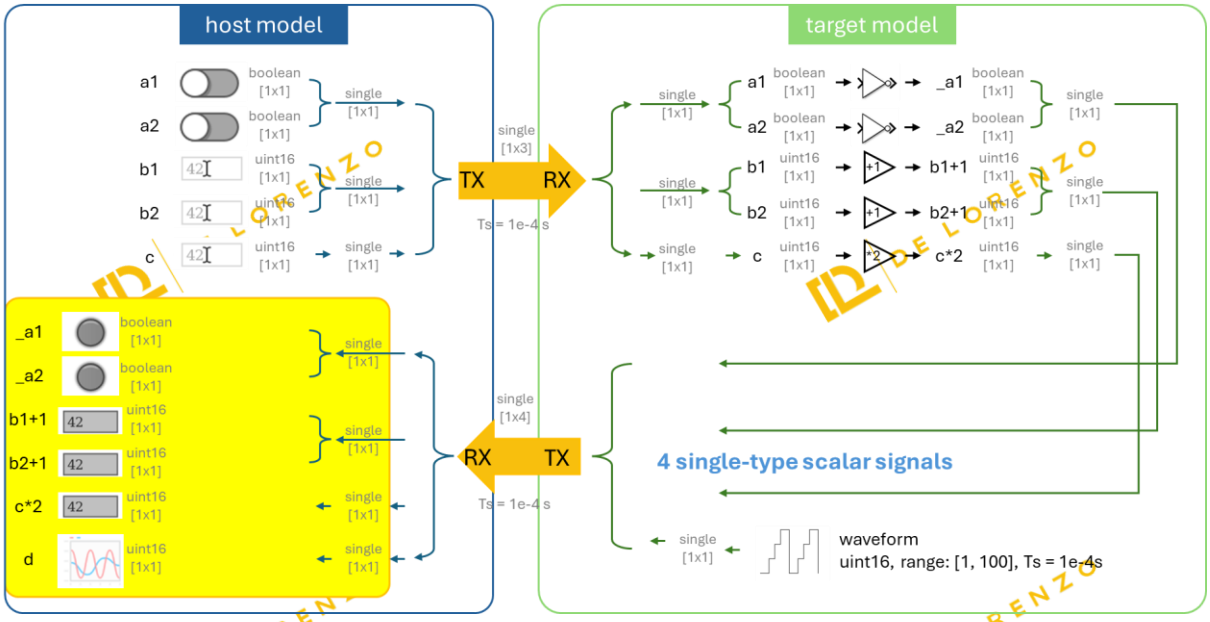
Step 13. Demux multiple signals of the same type



Use the **'Demux'** block to split a [1×4] vector into four [1×1] scalar signals.

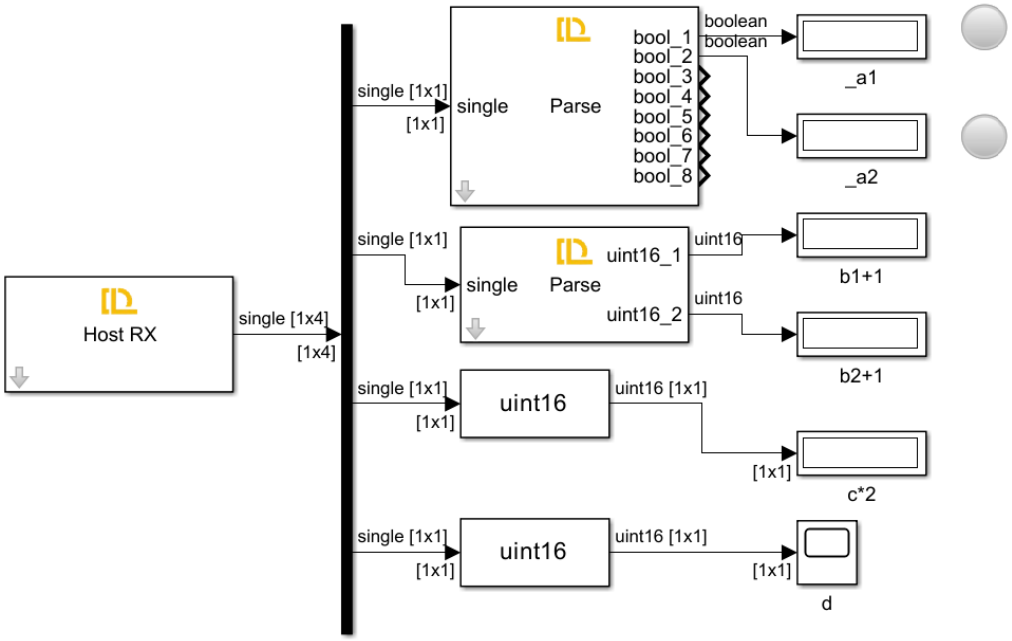


Step 14. Parse and data type convert to recover the controlled variables



Use the **Parse** blocks to convert the **single**-type signals into multiple **Boolean** and **uint16** signals and extract the controlled variables from the corresponding channels.

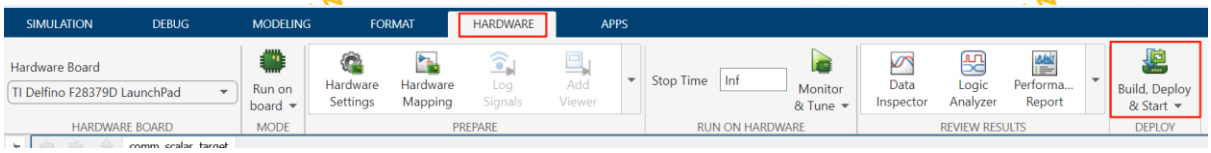
Use the **'Data Type Conversion'** block to restore the data to its original type.



Step 15. Debug

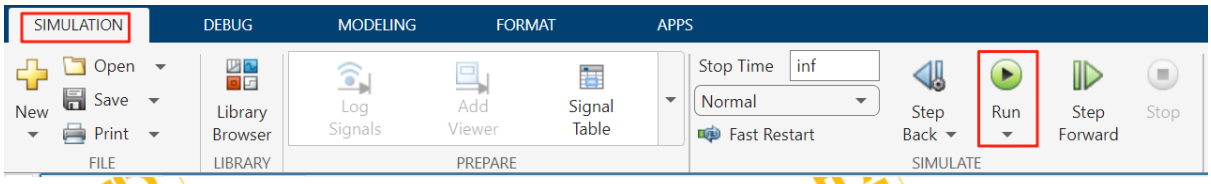
In the **target model**, compile and deploy the model to the DL RCPCORE hardware.

Wait for the 'Ready' message in the lower-left corner. If an error occurs, check and resolve it.

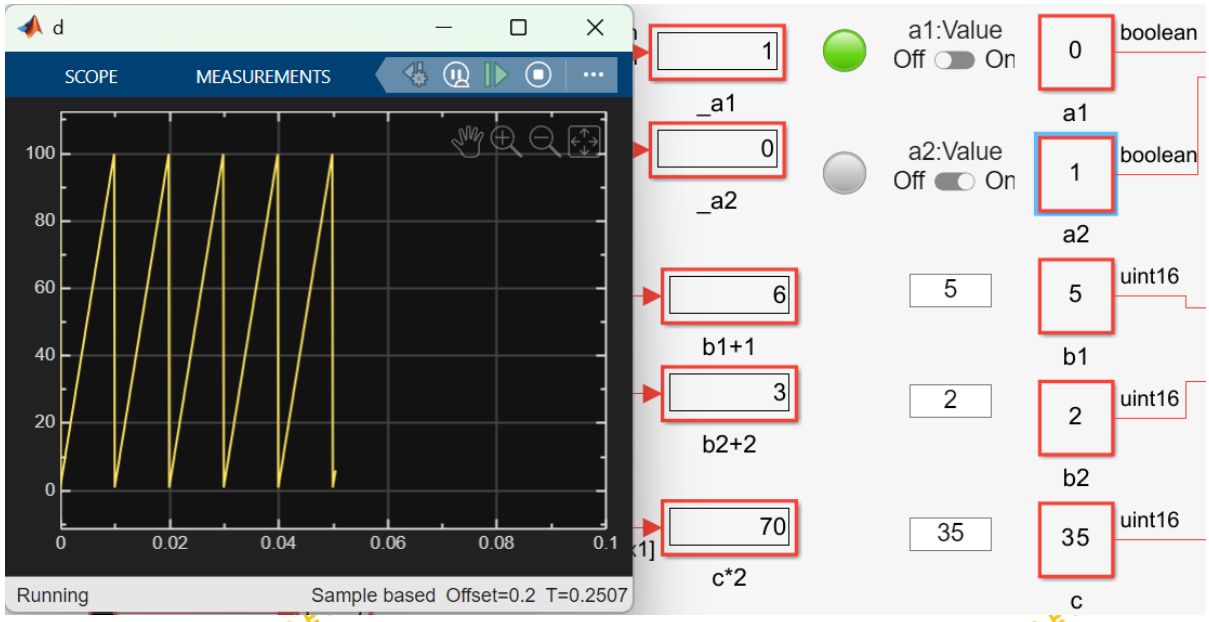


In the **host model**, after confirming that the target model has been deployed and is running on the DL RCPCORE, run the host model to begin debugging.

If an error occurs, check and resolve it.



Modify the control variables and verify whether the values processed by the target model are correctly sent back.



***Note**: Pay attention to the lamp color settings.*