Matlab component

Creating a component to handle binary signals

Opti*System* Application Note

Optiwave

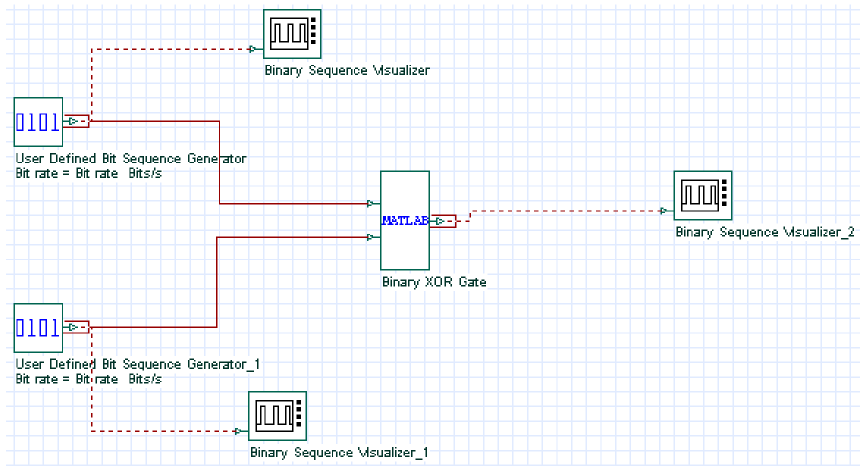
Matlab component – Creating a component to handle binary signals

1. Logical XOR Component

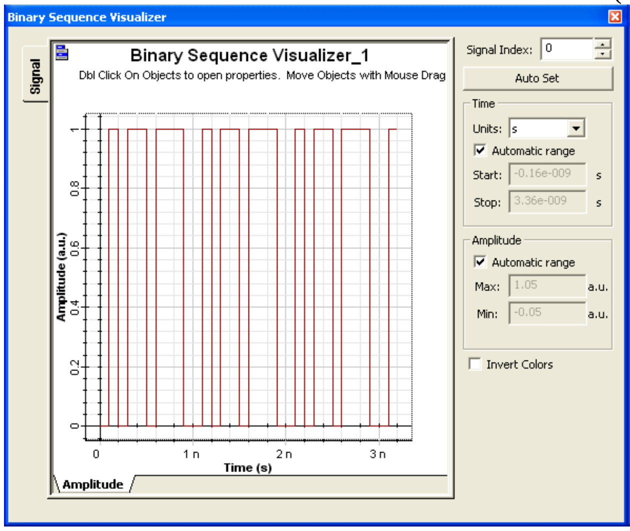
In order to create a binary component in Matlab for co-simulation with OptiSystem, first we need to understand the binary signal format that OptiSystem can generate and the structure of that signal launched into the Matlab workspace.

Following is an example to create a *Logical XOR Gate* using the Matlab component. In this example, first we introduce the signal format in OptiSystem, and then we show how to use Matlab to process that signal.

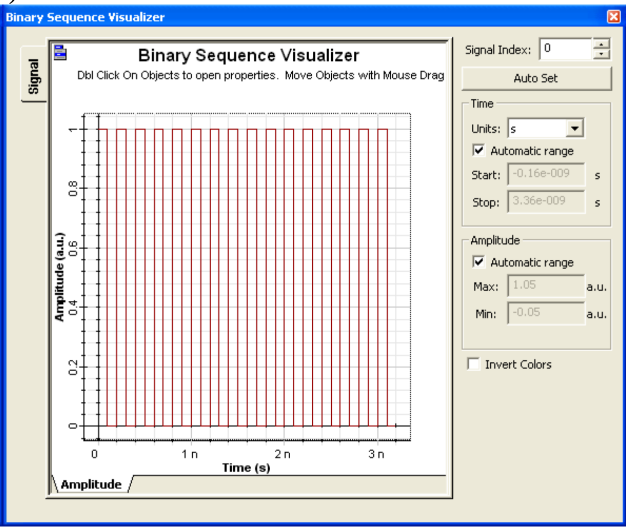
Binary Sequence Signal – Figure 1 demonstrates the system layout to build a binary XOR gate. Two Binary Bit Sequence Generators are used as the input to the XOR gate. Using two Binary Sequence Visualizers, these two signals are shown in Figure 1 (b) and (c).


Binary Sequence Visualizer
0101
User Defined Bit Sequence Generator
Bit rate = Bit rate Bits/s
MATLAB
Binary XOR Gate
Binary Sequence Visualizer_2
0101
User Defined Bit Sequence Generator_1
Bit rate = Bit rate Bits/s
Binary Sequence Visualizer_1

(a)


Binary Sequence Visualizer
Binary Sequence Visualizer_1
Dbl Click On Objects to open properties. Move Objects with Mouse Drag
Signal
Amplitude (a.u.)
Time (s)
Amplitude
Auto Set

Signal Index: 0
Time
Units: s
☑ Automatic range
Start: -0.16e-009 s
Stop: 3.36e-009 s
Amplitude
☑ Automatic range
Max: 1.05 a.u.
Min: -0.05 a.u.
☐ Invert Colors

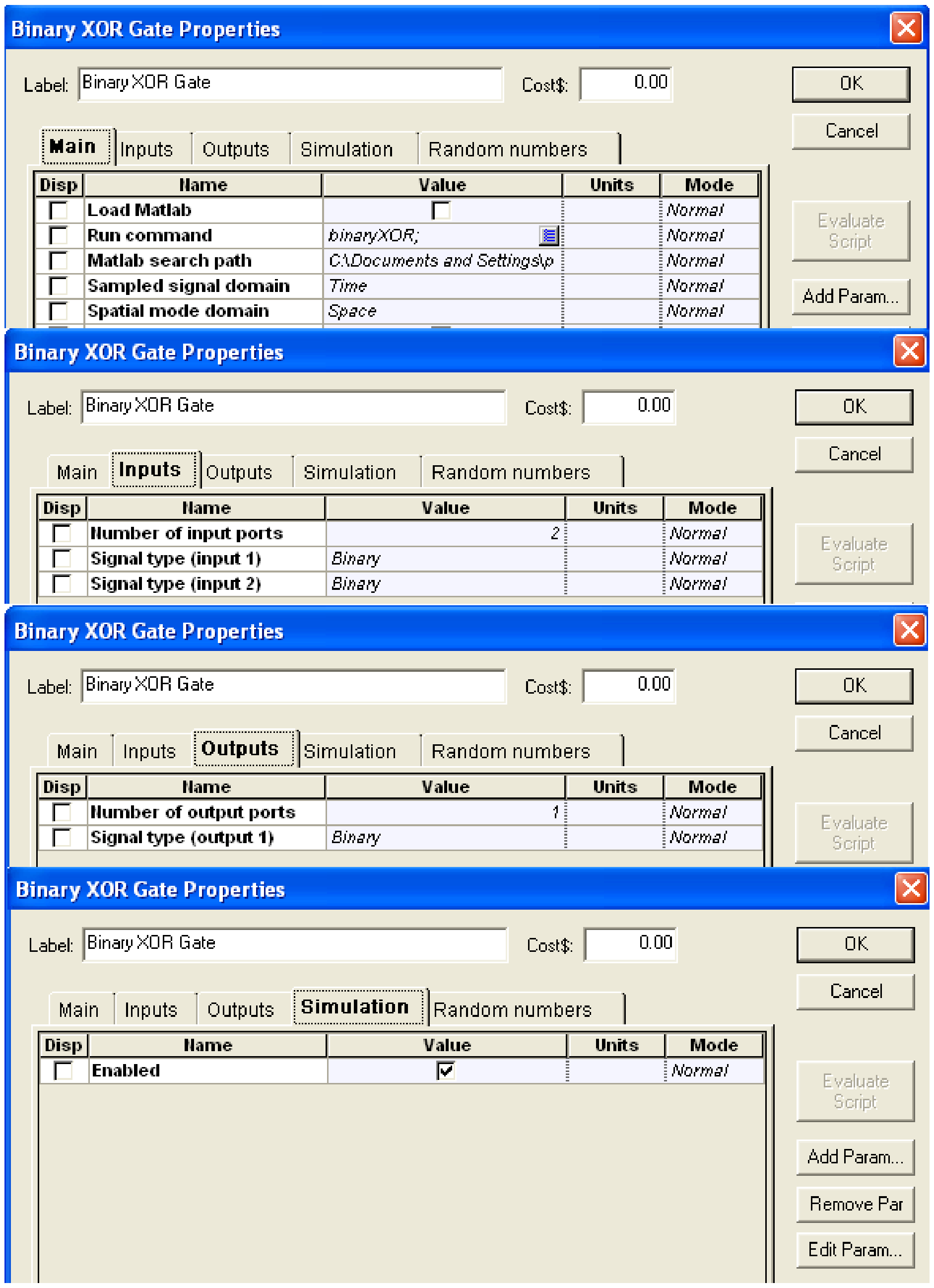

Binary Sequence Visualizer
Binary Sequence Visualizer
Dbl Click On Objects to open properties. Move Objects with Mouse Drag
Signal
Amplitude (a.u.)
Time (s)
Amplitude
Auto Set

Signal Index: 0
Time
Units: s
☑ Automatic range
Start: -0.16e-009 s
Stop: 3.36e-009 s
Amplitude
☑ Automatic range
Max: 1.05 a.u.
Min: -0.05 a.u.
☐ Invert Colors

(b) (c)

Figure 1: System layout and two binary sequences in the time domain.

Using the MATLAB Library, we have added a Matlab component to the layout. By clicking on the component properties, on the Main tab, you see the properties of the Matlab component. Using the Inputs and Outputs tap, we define the number of inputs and outputs and also their format which is binary in this example. The User Parameter tab is used to define the input parameters of the component which is not applicable here. Different tabs of the Matlab component properties are shown in Figure 2.



Binary XOR Gate Properties

Label: Binary XOR Gate   Cost$: 0.00

Main | Inputs | Outputs | Simulation | Random numbers

Disp	Name	Value	Units	Mode
☐	Load Matlab	☐		Normal
☐	Run command	binaryXOR;		Normal
☐	Matlab search path	C:\Documents and Settings\p		Normal
☐	Sampled signal domain	Time		Normal
☐	Spatial mode domain	Space		Normal

Inputs:

Disp	Name	Value	Units	Mode
☐	Number of input ports	2		Normal
☐	Signal type (input 1)	Binary		Normal
☐	Signal type (input 2)	Binary		Normal

Outputs:

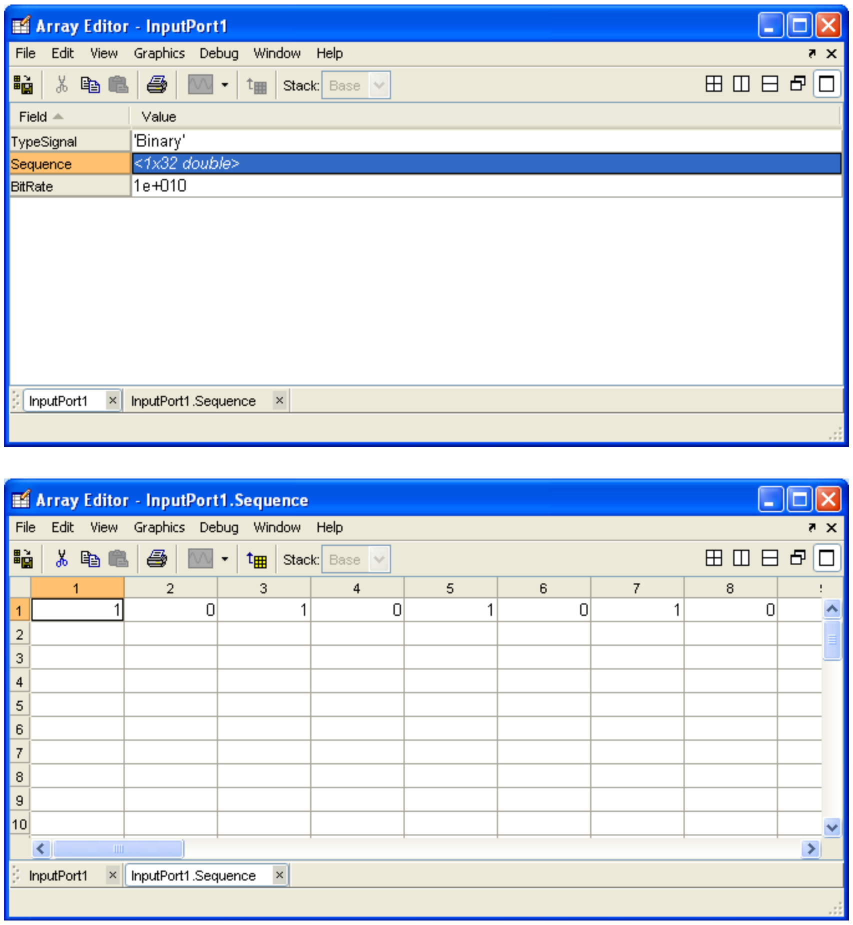
Disp	Name	Value	Units	Mode
☐	Number of output ports	1		Normal
☐	Signal type (output 1)	Binary		Normal

Simulation:

Disp	Name	Value	Units	Mode
☐	Enabled	✓		Normal

Figure 2: MATLAB Component Properties

Figure 3 shows the structure of the launched signal in the Matlab workspace. Each binary signal is defined by the sequence structure and the Bit Rate. The format of the binary sequence is double. As expected, unlike optical signals, binary signals only have one dimension. Figure 4 demonstrates the output of the Matlab component which is the logical XOR of the two inputs.

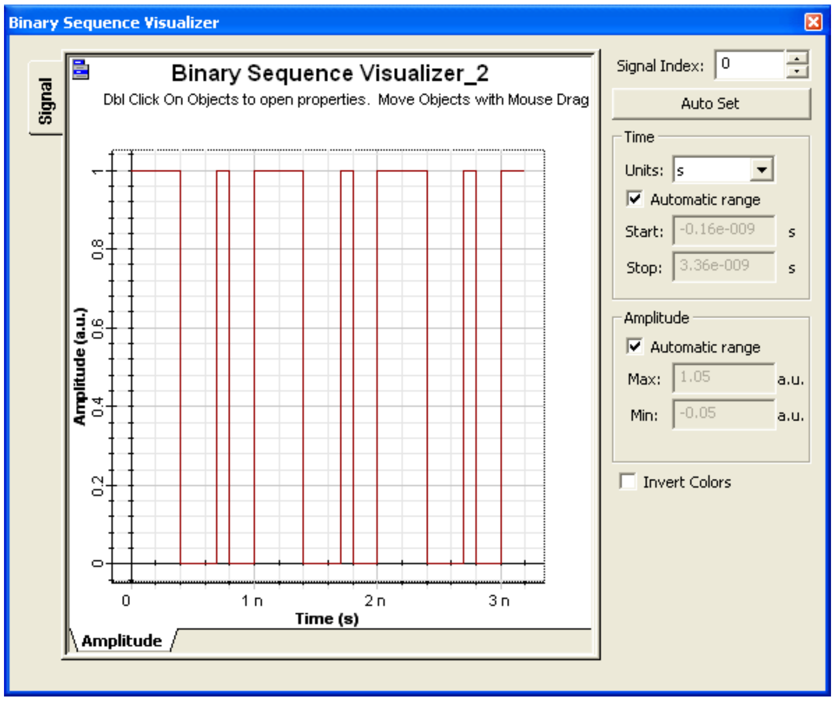

Array Editor - InputPort1
File Edit View Graphics Debug Window Help
Stack: Base
InputPort1
InputPort1.Sequence

Field	Value
TypeSignal	'Binary'
Sequence	<1x32 double>
BitRate	1e+010

Array Editor - InputPort1.Sequence
File Edit View Graphics Debug Window Help
Stack: Base
InputPort1
InputPort1.Sequence

	1	2	3	4	5	6	7	8
1	1	0	1	0	1	0	1	0
2								
3								
4								
5								
6								
7								
8								
9								
10								

Figure 3: Binary signal structure launched into the Matlab workspace.


Binary Sequence Visualizer
Signal
Binary Sequence Visualizer_2
Dbl Click On Objects to open properties. Move Objects with Mouse Drag
Amplitude (a.u.)
0
0.2
0.4
0.6
0.8
1
0
1 n
2 n
3 n
Time (s)
Amplitude
Auto Set

Signal Index: 0

Time
Units: s
[✓] Automatic range
Start: -0.16e-009 s
Stop: 3.36e-009 s

Amplitude
[✓] Automatic range
Max: 1.05 a.u.
Min: -0.05 a.u.

[ ] Invert Colors

Figure 4: The output of the Matlab component (logical XOR gate).

1.1 Matlab program:

Based on the binary signal structure, the logical XOR gate is implemented using the Matlab component. In this case the matlab component is set to have two input ports and one output port in binary format. The Matlab program (m-file) that simulates the logical gate is named *binaryXOR* and that is the command at ‘Run command’ parameter field. To be able to run this program, the path for this file must be in the Matlab search path, in this example the *binaryXOR.m* is located at ‘c:\temp’.

In order to see the m-file, go to the Matlab component properties, on the main tab, check ‘Load Matlab’ box and click OK. This will open the Matlab Command Window. In this window, first choose the code directory by ‘cd (‘c:\temp’)’ command. Using the command ‘open *binaryXOR.m*’, you can open the m-file in the Matlab Editor.

In this example, there is no parameter needed for a Logical XOR Gate; however for other applications, using the ‘User Parameters tab’, input parameters can be defined for the Matlab component by using ‘Add Parameter’ button.

After defining the Matlab component properties, you can start writing the program (Matlab code) that simulates the logical XOR gate. Following is the Matlab code:

```matlab
%
% This program simulates a Binary XOR Gate
% There is no input parameter for this example

% Creating an output structure similar to the input
OutputPort1 = InputPort1;

% Defining the parameter if applicable
%Param = Parameter0;

if strcmp(InputPort1.TypeSignal, 'Binary')

   cs = length(InputPort1.Sequence);

   if( cs > 0 )

       for m = 1 : cs

           if InputPort1.Sequence(m) == InputPort2.Sequence (m)
               OutputPort1.Sequence(m) = 0;
           else
               OutputPort1.Sequence(m) = 1;
           end
       end
   end

end
```

In order to see the file structure in the Matlab workspace (Figure 3), first you need to run the OptiSystem program, then use the command ‘open InputPort1’in the Matlab Command Window, to see the file structure for each port. You can also use the command ‘workspace’ to open the Matlab workspace and see all the variables in your Matlab simulation file.

Please refer to “Creating a component to handle optical signals” Application Note for the instructions of the Matlab debug mode.

Optiwave
7 Capella Court
Ottawa, Ontario, K2E 7X1, Canada

Tel.: 1.613.224.4700
Fax: 1.613.224.4706

E-mail: support@optiwave.com
URL: www.optiwave.com
Forum: www.optiwave.us