Aishwarya M Y, et. al. International Journal of Engineering Research and Applications
www.ijera.com
ISSN: 2248-9622, Vol. 13, Issue 10, October 2023, pp 132-145

RESEARCH ARTICLE OPEN ACCESS

Soft Transient Free Switching in Embedded Systems: MATLAB-Simulink Models

Ambalal V. Patel
Flight Control Laws (CLAW) Directorate,
Aeronautical Development Agency
(Ministry of Defence, Govt. of India),
P.B. 1718, Vimanapura Post, Bangalore – 560017, India

ABSTRACT
The soft faders or Transient Free SWitches (TFSW) are used for gradual transition between signals (instead of instantaneous transition) over a finite time on occurrence of the specific event or setting of a defined discrete. Thus, it helps in eliminating transients in the final outputs and in turn maintaining the safety and performance of the overall system. Implementation of the fader itself involves other levels of considerations like rate of computation, maintaining the memory requirements and overall execution time of the processor to cater for real time computations of the embedded systems of safety critical nature like fly-by-wire flight control system. Various types of proposed TFSWs, their implementation aspects including their implementation in MATLAB *.m file in function forms are available in literature. This article provides the implementation of various TFSWs in MATLAB-SIMULINK (*.mdl) models and their validation with MATLAB (*.m) models. The functionality of both models is exactly replicated and thus they can be used in lieu of each other in the respective platforms

Keywords **-** Fader, Transient Free Switch, data acquisition, algorithms, embedded systems, requirements, specifications

---

Date of Submission: 09-10-2023 Date of acceptance: 22-10-2023

I. INTRODUCTION

Various types of soft elements are used in the embedded systems, especially softwares therein. The 'soft elements' here broadly referred to several of the dynamic elements like filters, faders or Transient Free Switches (TFSW), rate limiters etc. are used in the soft form (as part of the computational algorithms within the software) within the embedded systems for various applications [6-7]. The soft faders or transient free switches are used for gradual transition between signals (instead of instantaneous transition) over a finite time on occurrence of the specific event or setting of a defined discrete. Thus, it helps in eliminating the unwanted effects, especially transients in the final outputs or commands and in turn maintaining the safety and performance of the system. Implementation of the fader itself involves other levels of considerations like rate of computation, maintaining the memory requirements and overall execution time of the processor to cater for real time computations of the embedded systems of safety critical nature like fly-by-wire flight control system. There are examples where the implementation of the faders has affected the software functioning and the updates required [6]. Significant efforts have been spent on development of various 'soft elements' of embedded systems including their testing and verification [1-5]. Various types of proposed TFSWs and their comparative analysis are detailed in [7]. Further, implementation aspects including their implementation in MATLAB *.m file in function forms are available in literature [8].

This article provides the implementation of various TFSWs in MATLAB-SIMULINK (*.mdl) models and their validation with MATLAB (*.m) models. The functionality of both models is exactly replicated and thus they can be used in lieu of each other in the respective platforms.

The article is organized as given below. After introduction, Section 2 presents brief review of TFSWs including their types. Section 3 deals implementation of the TFSWs in MATLAB-SIMULINK (*.mdl) model and relevant results demonstrating their functionality. This section also presents the comparison between the MATLAB (*.m) models and SIMULINK (*.mdl) models of the TFSWs for with and without LEFT features. Section 4 concludes the paper

II. TRANSIENT FREE SWITCH (TFSW) OR SOFT FADER AND THEIR TYPES

The soft faders or transient free switches are used for gradual transition between signals (instead of instantaneous transition) over a finite time on occurrence of the specific event or setting of defined discrete. Thus, it helps in eliminating the unwanted effects, especially transients in the final outputs or commands and in turn maintaining the safety and performance. Implementation of the fader itself involves other levels of considerations like rate of computation, maintaining the memory requirements and overall execution time of the processor to cater for real time computations of the embedded systems of safety critical nature like fly-by-wire flight control system. Various types of TFSWs as listed below have been proposed in Ref. [7], including the feature of termination of operation at either Less than or Equal to Fader Time (LEFT):

1) Direct Fixed Error Reducing Fader (DFERFD): A direct fixed error (between required and selected output at the instant of Event toggle) per frame (or sampling instant) is reduced from the required signal to reach the desired output smoothly over a specified fader-time.
2) Direct Variable Error Reducing Fader (DVERFD): A direct variable error (between required and selected output at the instant of Event toggle) per frame (or sampling instant) recomputed based n the number of remaining frame counts (or remaining fader-time) is reduced from the required signal to reach the desired output smoothly over a specified fader-time.
3) Scaled Error Reducing Fader (SERFD): A normalized scale factor used for reducing the past signal while increasing the required signal to gradually bring into the selected output over a specified fader-time.

The TFSW of such type of feature can be identified by the nomenclature: TFSW_LEFT, where TFSW could be ‘DFERFD’, ‘DVERFD’, ‘SERFD’. Thus, they could be identified as ‘DFERFD_LEFT’, ‘DVERFD_LEFT’, ‘SERFD_LEFT’. The LEFT feature is invoked under the following conditions satisfied together:

1) Fader computation is progressing (event toggle detected and thereafter computations continued) but not completed (before fader time completion which can be found out from the frame counter), and
2) Change in the sign of the error between the past and the present samples is detected. Here error is referred to the difference between the required signal at that instant and selected output of the corresponding past instant.

III. IMPLEMENTATION OF TFSW IN SIMULINK (*.MDL MODEL) FILES AND RESULTS

3.1 Implementation with MATLAB-Simulink files

A set of figures referred to Figure 1 shows the MATLAB Simulink (*.mdl) models of the DFERFD_LEFT TFSW. Similarly, a set of figures referred to Figures 2 and 3 show MATLAB Simulink (*.mdl) models of the DVERFD_LEFT and SERFD_LEFT TFSW, respectively. Although name of the each TFSW is ‘TFSW_LEFT’ however they are implemented in unified form, i.e., they can used for with and without LEFT feature by setting the input discrete LEFT_DI to True and False, respectively. Detailed description and dimensions of the inputs, outputs, and intermediate signals for each TFSW function are given alongside of the first figure of each TFSW. These inputs to and outputs from each block are self-explanatory. User may prepare and provision for the set or vector of inputs and outputs signals in the external interface file (usually referred to driver file), and then use these function-form files like a library module. It may be noted that the TFSWs implemented in SIMULINK (*.mdl) models provided along with article has got the feature to deal with the proposed concept of vector of inputs, outputs, and intermediate elements for each specific event, step towards the ‘Optimization of Number of Independent TFSWs’. Refer to Table 1 for the arrangement of figures showing the models of the various TFSWs implemented in SIMULINK (‘*.mdl’ models) and corresponding simulation results

3.2 Results

Refer to Table 1 for the arrangement of figures showing the simulation results. Results of the all TFSWs implemented using the *.m files given in Ref. [8] are shown in Figure 4. It includes subplots of Event, Required, Selected, Current Signals, and Superimposed Outputs (Selected Signals) of TFSWs separately for with and without LEFT for ease of comparison and understanding. In a similar manner as that of Figure 4, the results of the SIMULINK (*.mdl) models are shown in Figure 5. Thus, the results in Figures 4 and 5 show the functionality of the TFSWs implemented in MATLAB ‘*.m’ file [8] and SIMULINK (*.mdl) model, and both found to be consistent. Figures 6, 7, and 8 show the results of DFERFD, DVERFD, and SERFD TFSWs from MATLAB and SIMULINK models, that also with and without LEFT features. Figure 9 shows the superimposed outputs of the two models. The

selected time shows the results for transition of event from 0 to 1 and 1 to 0, both. The superimposed outputs of two models (MATLAB and SIMULINK) and corresponding differences are also shown in Figures 6, 7, and 8. The difference between the two outputs at the selected tine segment is zero, which indicates that functionality of both models is exactly replicated and thus they can be used in lieu of each other in the respective platforms.

IV. CONCLUSION

A conclusion section must be included and the implementation of the various unified TFSWs with the help of MATLAB-SIMULINK (*.mdl) models are provided along with this article which may be used as library functions. The unified refers to the functionality of with and without LEFT feature, which is used for termination of TFSW operation before specified time, if the selected signal reaches to the required signal. The outputs of the SIMULINK models of the TFSWs are compared with that of the corresponding MATLAB (*.m file) models. It is seen that the functionality of both models is exactly replicated and thus they can be used in lieu of each other in the respective platforms

Acknowledgements

The author is grateful to the Aeronautical Development Agency (ADA), Bangalore, India for permitting to publish this work.

REFERENCES

[1] Ambalal V. Patel, Vijay V. Patel, Girish S. Deodhare, and Shyam Chetty, "Clearance of Flight-Control-System Software with Hardware-in-Loop Test Platform", AIAA Journal of Aircraft, Vol. 51, No. 3, May-June 2014, DOI 10.2514/1.C032404.

[2] Ambalal V. Patel, Vijay V. Patel, Girish Deodhare and Shyam Chetty, "Flight Control System clearance using dynamic tests at Hardware-In-Loop Test Platform", Proceedings of International Conference on Avionics Systems (ICAS) 2008, held at RCI, Hyderabad, during February 22-23, 2008.

[3] Guruganesh R., Shyam Chetty, Ambalal V. Patel and Girish Deodhare, "Clearance of LCA Flight Control Laws on Various Ground Test Simulation Platforms", Proceedings of International Conference on Avionics Systems (ICAS) 2008, held at RCI, Hyderabad, during February 22-23, 2008.

[4] Ambalal V. Patel, Vijay V. Patel, Girish Deodhare and Shyam Chetty, "Flight Control System clearance using static tests at Iron Bird", Proceedings AIAA Guidance, Navigation and Control (GNC) conference and exhibit, paper No. 6203 in session No. 31-GNC-14, held at Keystone, Colorado, USA during August 21-24, 2006.

[5] Ambalal V. Patel**,** "Functional Level Data Acquisition Requirement Specification Formulation for Embedded Systems: Challenges, Experiences and Guidelines", International Journal of Engineering Research and Application (IJERA), ISSN: 2248-9622, Vol. 7, Issue 10, (Part -5) October 2017, pp.75-84, DOI: 10.9790/9622-0710057584

[6] Yogananda Jeppu, "Flight Control Software: Mistakes made and Lessons learnt", IEEE Software, PP 67-73, May/June 2013

[7] Ambalal V. Patel, "Various Types of Soft Transient Free Switching in Embedded Systems" International Journal of Engineering Applications and Research, ISSN: 2248-9622, Volume 13, Issue 08, August 2023, PP 01-17

[8] Ambalal V. Patel, "Some Aspects of Implementation of Soft Transient Free Switching in Embedded Systems", International Journal of Engineering Applications and Research, ISSN: 2248-9622, Volume 13, Issue 09, September 2023, PP 32-45

Table 1: Arrangement of Figures Showing the Models of the Various Transient Free Switches Implemented in SIMULINK ('*.mdl' models) and Corresponding Simulation Results

Sl. No.	Figure No.	Results of TFSW	Description	Additional Remarks
1	1	-	Set of Figures: Simulink model (*.mdl file) of DFERFD_LEFT function (Direct Fixed Error Reducing Fader) and Subsystems	These are unified TFSWs, i.e., by setting LEFT_DI to 0 or 1, it can be used for without or with LEFT termination feature, respectively
2	2	-	Set of Figures: Simulink model (*.mdl file) of DVERFD_LEFT function (Direct Variable Error Reducing Fader) and Subsystems	
3	3	-	Set of Figures: Simulink model (*.mdl file) of SERFD_LEFT function (Scaled Error Reducing Fader) and Subsystems	
4	4	All TFSW	Results obtained from **MATLAB *.m file Model of TFSWs**: Event, Required, Selected, Current Signals, Superimposed Outputs (Selected Signals) of TFSWs separately for with and without LEFT for ease of comparison and understanding	The results here show the functionality of the TFSWs implemented in MATLAB '*.m' file
5	5	All TFSW	Results obtained from **SIMULINK *.mdl file Model of TFSWs**: Arrangement Similar to that of Figure 9	The results here show the functionality of the TFSWs implemented in SIMULINK '*.mdl' file
6	6	DFERFD and DFERFD_LEFT	DFERFD TFSW Results from Matlab and Simulink model.	The superimposed and difference of output plots shows the correct replication of *.m file and *.mdl models of TFSWs
7	7	DVERFD and DVERFD_LEFT	DVERFD TFSW Results from Matlab and Simulink model	
8	8	SERFD and SERFD_LEFT	SERFD TFSW Results from Matlab and Simulink model	
9	9	All TFSW	All TFSW Results of Matlab (*.m file) and Simulink (*.mdl) model outputs	

DFERFD: Direct Fixed Error Reducing Fader
DVERFD: Direct Variable Error Reducing Fader
SERFD: Scaled Error Reducing Fader
TFSW_LEFT: Indicates TFSW with Termination Feature at either Less than OR Eaqual Fadet Time. TFSW could be DFERFD / DVERFD / SERFD

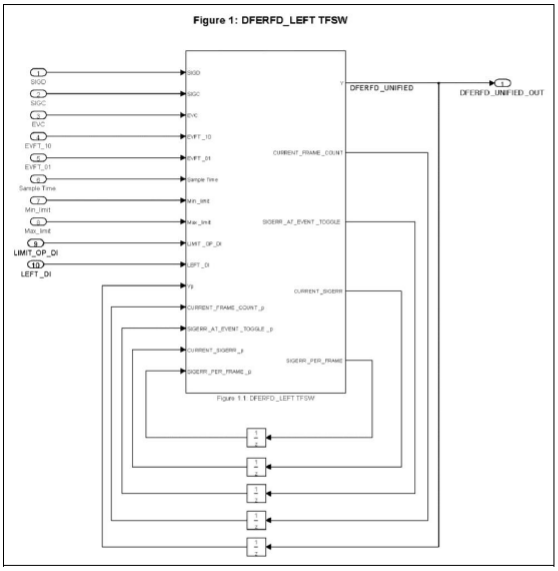


Figure 1: Set of Figures: Simulink model (*.mdl file) of DFERFD function (Direct Fixed Error Reducing Fader) and Subsystems

DFERFD_LEFT TFSW Function: Inputs, Outputs and signal sizes

Outputs:
1) Y = 1 x Ns_EVF = TFSW (Fader) Output where Ns_EVF is a vector of numbers indicating the number of signals associated with each fader for the specific event
2) CURRENT_FRAME_COUNT = 1x1 = Current Frame Count
3) SIGERR_AT_EVENT_TOGGLE = 1 x Ns_EVF = Error between Required and Prior Frame Signal at Event Toggle
4) CURRENT_SIGERR = 1 x Ns_EVF = Error between Required and Prior Frame output at present instant within the on-going Fader Time
5) SIGERR_PER_FRAME = 1 x Ns_EVF = Signal Error Per Frame to be Reduced to Reach to Required Signal over Fader Time
6) REQSIG = 1 x Ns_EVF = Required signal on Event Transit (internally it can be tapped out, not shown in this Figure)

Inputs:
1) SIGD = 1 x Ns_EVF = Set of Signals When Event Status is TRUE
2) SIGC = 1 x Ns_EVF = Set of Signals When Event Status is FALSE
3) EVC = 1x1= Event Status (True (1) or False (0))
4) MAX_FRAME_COUNT_10 (EVFT_10) = 1x1 = Maximum Number of Frame Count for the specified Time on Transit of Event from 1 to 0
5) MAX_FRAME_COUNT_01 (EVFT_01) = 1x1 = Maximum Number of Frame Count for the specified Time on Transit of Event from 0 to 1
6) T = 1x1 = Sample Time or Frame Time
7) MIN_LIMIT = 1 x Ns_EVF = Minimum Limit on Output for Each Signal of the Fader
8) MAX_LIMIT = 1 x Ns_EVF = Maximum Limit on Output for Each Signal of the Fader
9) LIMIT_OP_DI = 1x1 = Discrete for Limit (1) / Do not Limit (0) output
10) LEFT_DI = 1x1 = Discrete for Termination of TFSW Operation if 'Less Than OR Equal to Fader Time' (LEFT) Criteria satisfied (True(1): Opted; False(0): Not Opted)
11) Yp = 1 x Ns_EVF = Set of outputs at the Past Frame
12) CURRENT_FRAME_COUNT_p = 1x1 = Past Frame Count
13) SIGERR_AT_EVENT_TOGGLE_p = 1 x Ns_EVF = Past Signal Error at Event Toggle held
14) CURRENT_SIGERR_p = 1 x Ns_EVF = Signal Error (Required and Output) at the Past Frame
15) SIGERR_PER_FRAME_p = 1 x Ns_EVF = Past Signal Error Per Frame held

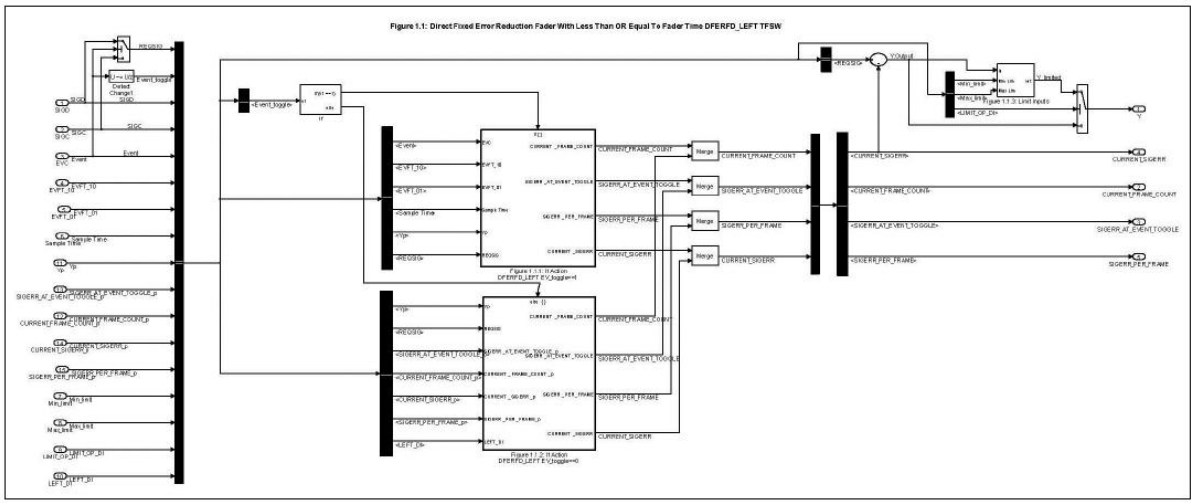
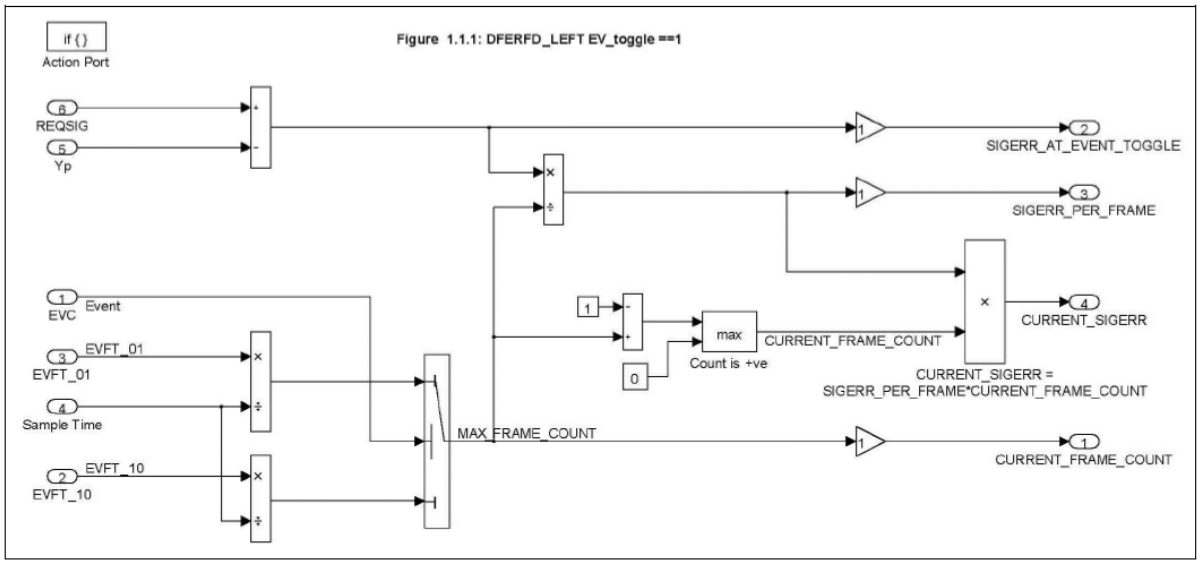
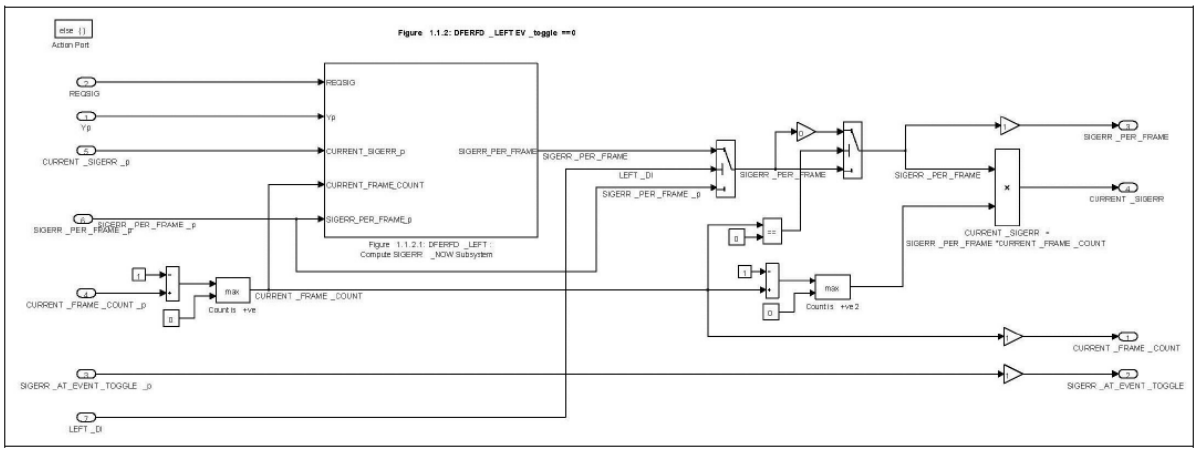
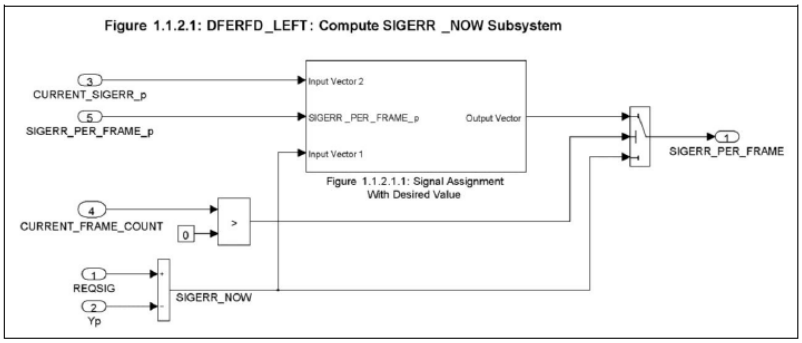
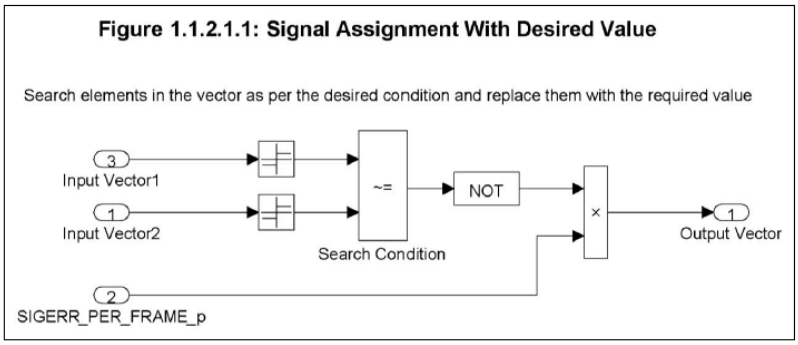


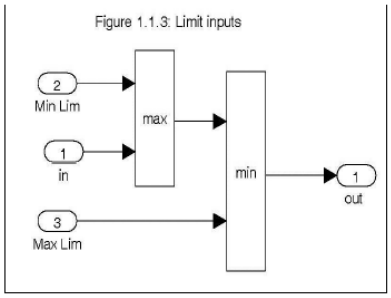
Figure 1.1: Direct Fixed Error Reduction Fader With Less Than OR Equal To Fader Time DFERFD_LEFT TFSW


if ( )
Action Port
Figure 1.1.1: DFERFD_LEFT EV_toggle ==1
REQSIG
Yp
Event
EVC
EVFT_01
Sample Time
EVFT_10
SIGERR_AT_EVENT_TOGGLE
SIGERR_PER_FRAME
CURRENT_SIGERR
CURRENT_SIGERR =
SIGERR_PER_FRAME*CURRENT_FRAME_COUNT
max
CURRENT_FRAME_COUNT
Count is +ve
MAX_FRAME_COUNT
CURRENT_FRAME_COUNT


else ( )
Action Port
Figure 1.1.2: DFERFD _LEFT EV _toggle ==0
REQSIG
Yp
CURRENT _SIGERR _p
SIGERR _PER _FRAME _p
CURRENT _FRAME _COUNT _p
SIGERR _AT _EVENT _TOGGLE _p
LEFT _DI
SIGERR_PER_FRAME
CURRENT_FRAME_COUNT
Figure 1.1.2.1: DFERFD _LEFT :
Compute SIGERR _NOW Subsystem
max
Count is +ve
SIGERR _PER _FRAME
CURRENT _SIGERR
CURRENT _FRAME _COUNT
SIGERR _AT _EVENT _TOGGLE







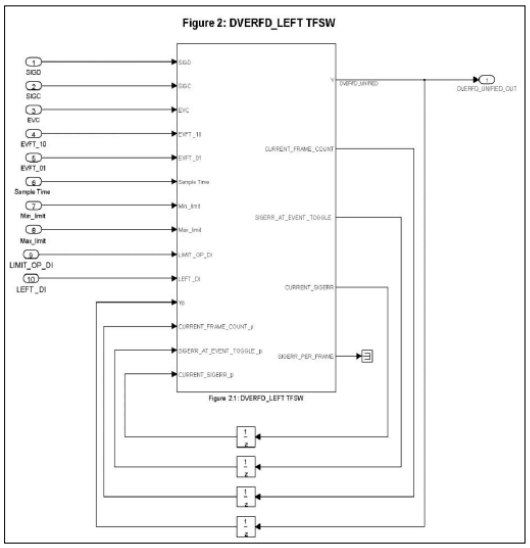


Figure 2: Set of Figures: Simulink model (*.mdl file) of DVERFD function (Direct Variable Error Reducing Fader) and Subsystems

DVERFD_LEFT TFSW Function: Inputs, Outputs and Signal sizes

Outputs:
1) Y = 1 x Ns_EVF = TFSW (Fader) Output where Ns_EVF is a vector of numbers indicating the number of signals associated with each fader for the specific event
2) CURRENT_FRAME_COUNT = 1x1 = Current Frame Count
3) SIGERR_AT_EVENT_TOGGLE = 1 x Ns_EVF = Error between Required and Prior Frame Signal at Event Toggle
4) CURRENT_SIGERR = 1 x Ns_EVF = Error between Required and Prior Frame output at present instant within the on-going Fader Time
5) SIGERR_PER_FRAME = 1 x Ns_EVF = Signal Error Per Frame to be Reduced to Reach to Required Signal over Fader Time
6) REQSIG = 1 x Ns_EVF = Required signal on Event Transit (internally it can be tapped out, not shown in this Figure)

Inputs:
1) SIGD = 1 x Ns_EVF = Set of Signals When Event Status is TRUE
2) SIGC = 1 x Ns_EVF = Set of Signals When Event Status is FALSE
3) EVC = 1x1= Event Status (True (1) or False (0))
4) MAX_FRAME_COUNT_10 (EVFT_10) = 1x1 = Maximum Number of Frame Count for the specified Time on Transit of Event from 1 to 0
5) MAX_FRAME_COUNT_01 (EVFT_01) = 1x1 = Maximum Number of Frame Count for the specified Time on Transit of Event from 0 to 1
6) T = 1x1 = Sample Time or Frame Time
7) MIN_LIMIT = 1 x Ns_EVF = Minimum Limit on Output for Each Signal of the Fader
8) MAX_LIMIT = 1 x Ns_EVF = Maximum Limit on Output for Each Signal of the Fader
9) LIMIT_OP_DI = 1x1 = Discrete for Limit (1) / Do not Limit (0) output
10) LEFT_DI = 1x1 = Discrete for Termination of TFSW Operation if 'Less Than OR Equal to Fader Time' (LEFT) Criteria satisfied (True(1): Opted; False(0): Not Opted)
11) Yp = 1 x Ns_EVF = Set of outputs at the Past Frame
12) CURRENT_FRAME_COUNT_p = 1x1 = Past Frame Count
13) SIGERR_AT_EVENT_TOGGLE_p = 1 x Ns_EVF = Past Signal Error at Event Toggle held
14) CURRENT_SIGERR_p = 1 x Ns_EVF = Signal Error (Required and Output) at the Past Frame

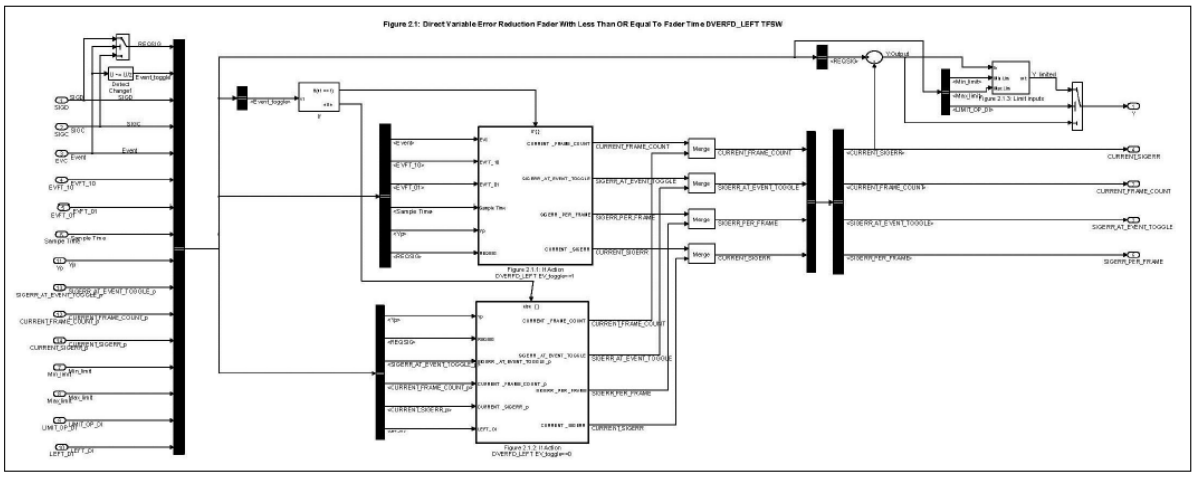
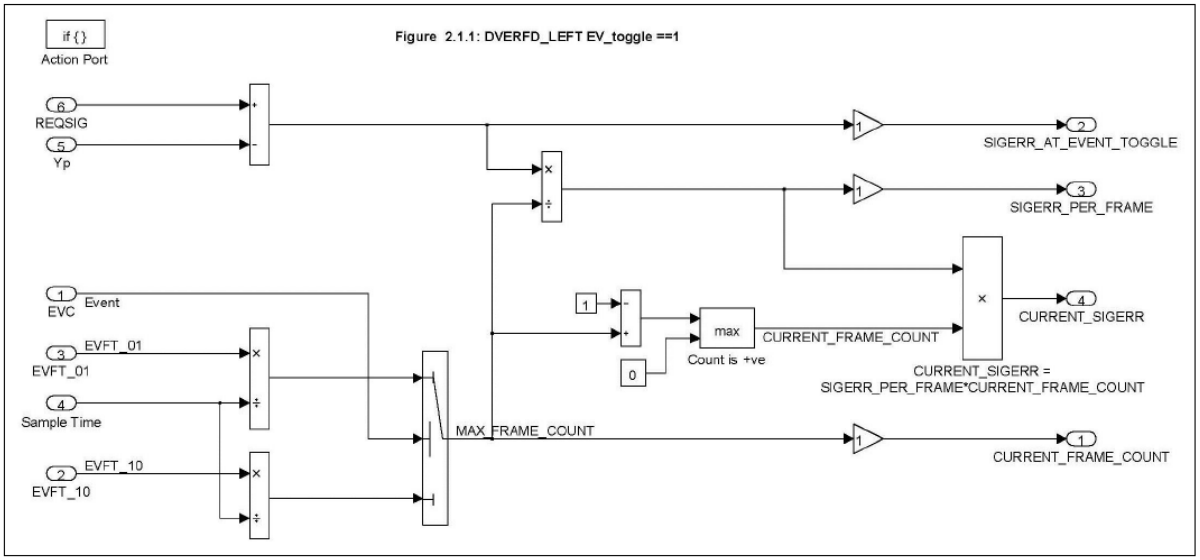
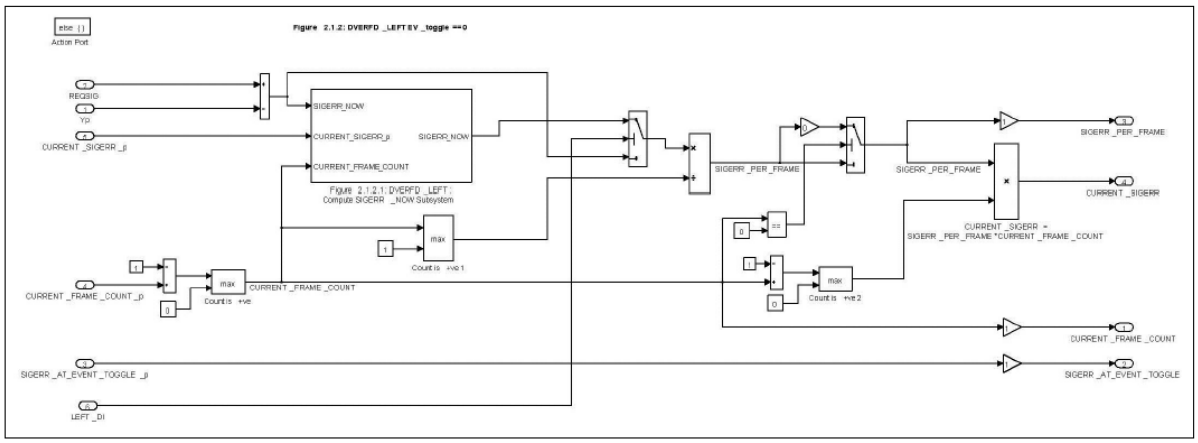
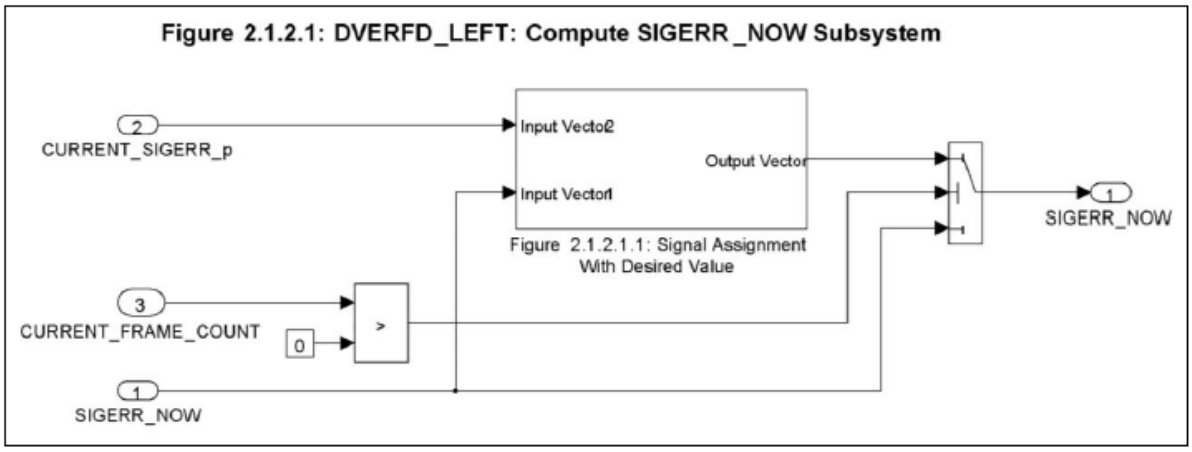
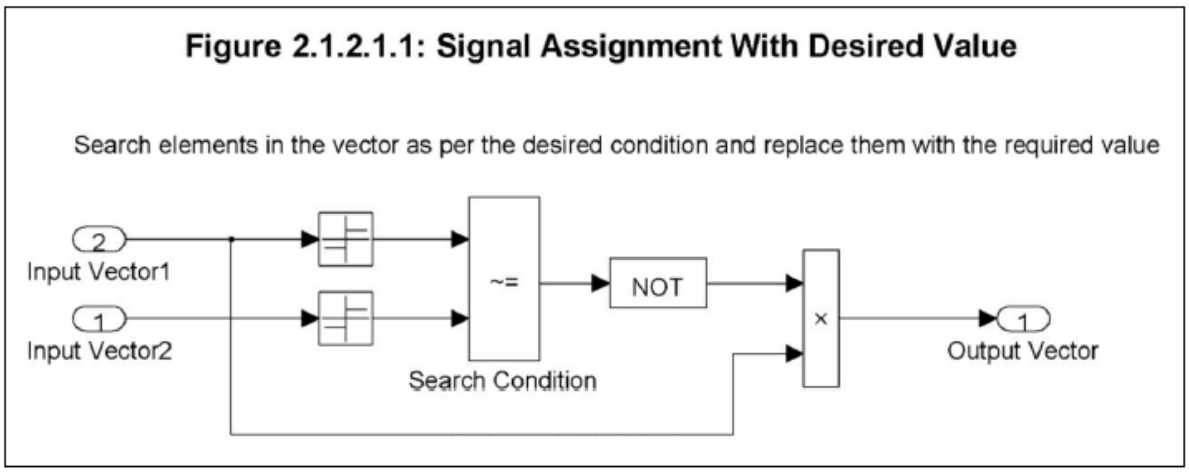


Figure 2.1.3 is same as Figure 1.1.3, hence it is not shown separately









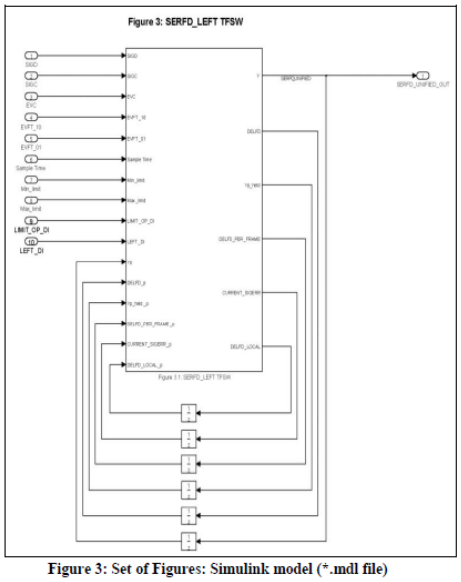


Figure 3: Set of Figures: Simulink model (*.mdl file) of SERFD function (Scaled Error Reducing Fader) and Subsystems

SERFD_LEFT Function: Inputs, Outputs and signal sizes

Outputs:
1) Y = 1 x Ns_EVF = TFSW (Fader) Output where Ns_EVF is a vector of numbers indicating the number of signals associated with each fader for the specific event
2) DELFD = 1x1 = Normalized Fader Weight at current frame
3) Yp_held = 1 x Ns_EVF = Past output at the instant of Event Toggle held
4) DELFD_PER_FRAME = 1x1 = Normalized Fader Weight per Frame to be Reduced to Reach to Required Signal over Fader Time
5) CURRENT_SIGERR = 1 x Ns_EVF = Error between Required and Prior Frame output at present instant within the on-going Fader Time
6) DELFD_LOCAL = 1 x Ns_EVF = Set of Normalized Fader Weights at current frame for internal computations
7) REQSIG = 1 x Ns_EVF = Required signal on Event Transit (Not shown in the Figure)

Inputs:
1) SIGD = 1 x Ns_EVF = Set of Signals When Event Status is TRUE
2) SIGC = 1 x Ns_EVF = Set of Signals When Event Status is FALSE
3) EVC = 1x1= Event Status (True (1) or False (0))
4) MAX_DELFD_PER_FRAME_EVT_10 (EVFT_10) = 1x1 = Maximum Normalized Weight Per Frame for the specified Time on Transit of Event from 1 to 0
5) MAX_DELFD_PER_FRAME_EVT_01 (EVFT_01) = 1x1 = Maximum Normalized Weight Per Frame for the specified Time on Transit of Event from 0 to 1
6) T = 1x1 = Sample Time or Frame Time
7) MIN_LIMIT = 1 x Ns_EVF = Minimum Limit on Output for Each Signal of the Fader
8) MAX_LIMIT = 1 x Ns_EVF = Maximum Limit on Output for Each Signal of the Fader
9) LIMIT_OP_DI = 1x1 = Discrete for Limit (1) / Do not Limit (0) output
10) LEFT_DI = 1x1 = Discrete for Termination of TFSW Operation if 'Less Than OR Equal to Fader Time' (LEFT) Criteria satisfied (True(1): Opted; False(0): Not Opted
11) Yp = 1 x Ns_EVF = Set of outputs at the Past Frame
12) DELFD_p = 1x1 = Past Frame Normalized Fader Weight
13) Yp_held_p = 1 x Ns_EVF = Past output at the instant of Event Toggle held
14) DELFD_PER_FRAME_p = 1x1 = Past Frame Normalized Fader Weight per Frame to be Reduced to Reach to Required Signal over Fader Time
15) CURRENT_SIGERR_p = 1 x Ns_EVF = Signal Error (Required and Output) at the Past Frame
16) DELFD_LOCAL_p = 1 x Ns_EVF = Set of Normalized Fader Weights at Past frame for internal computations

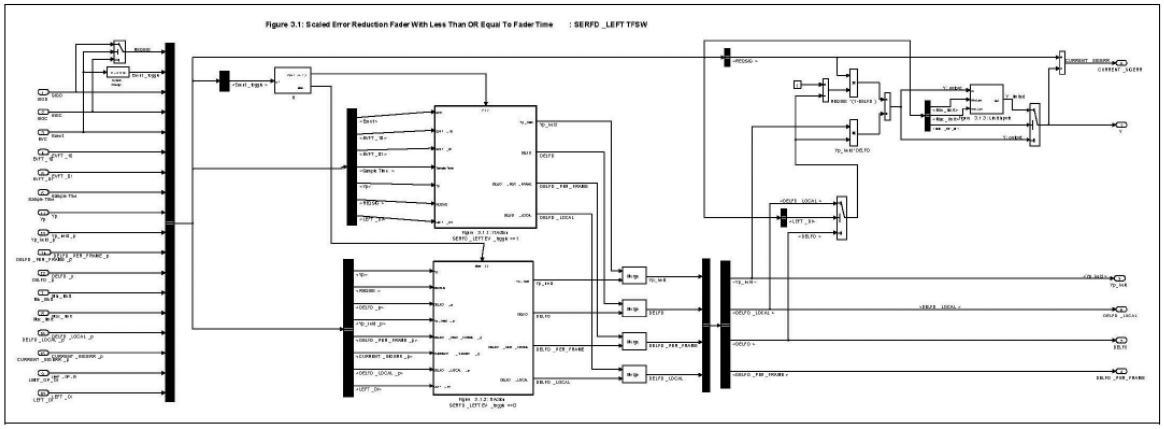
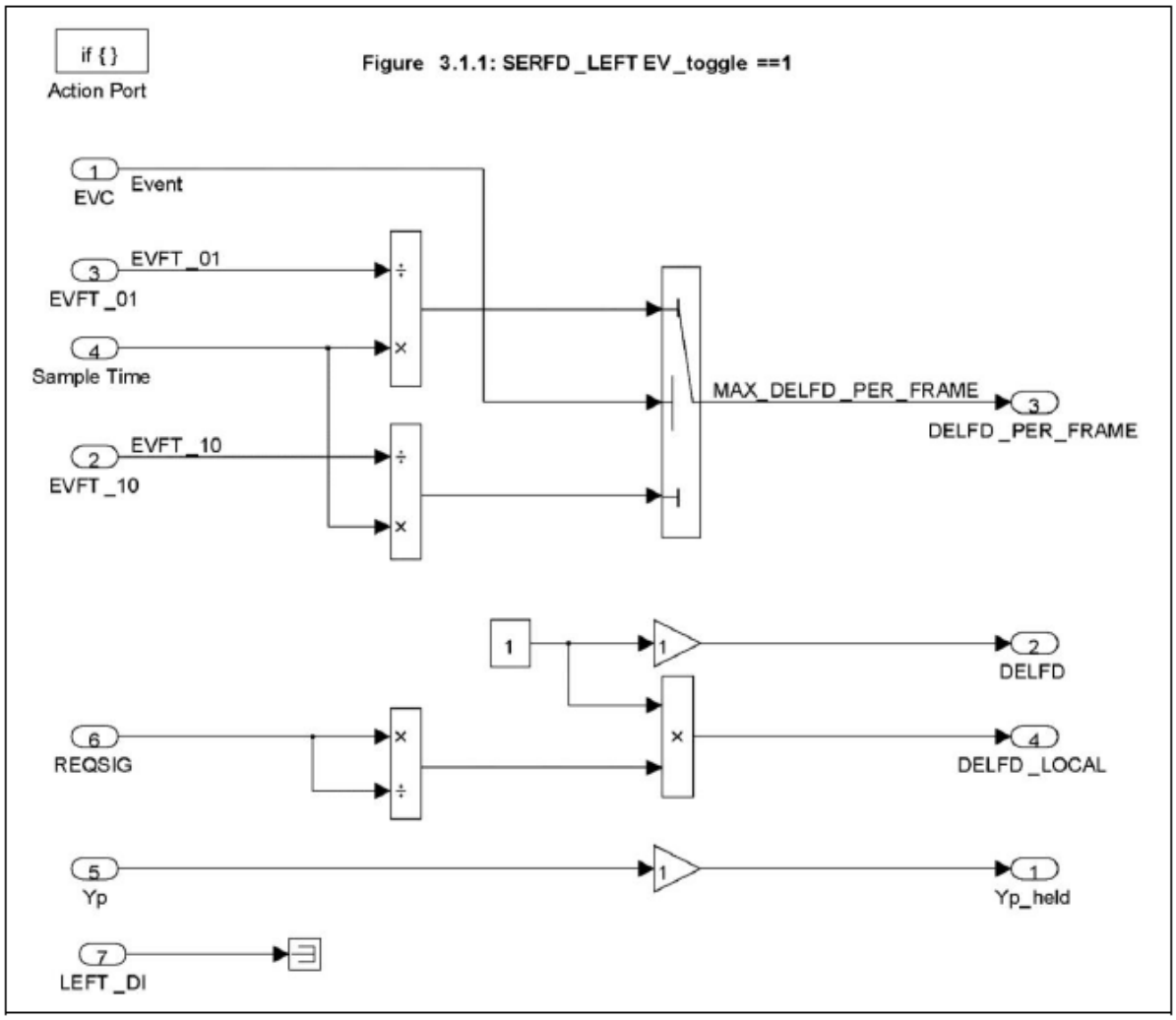
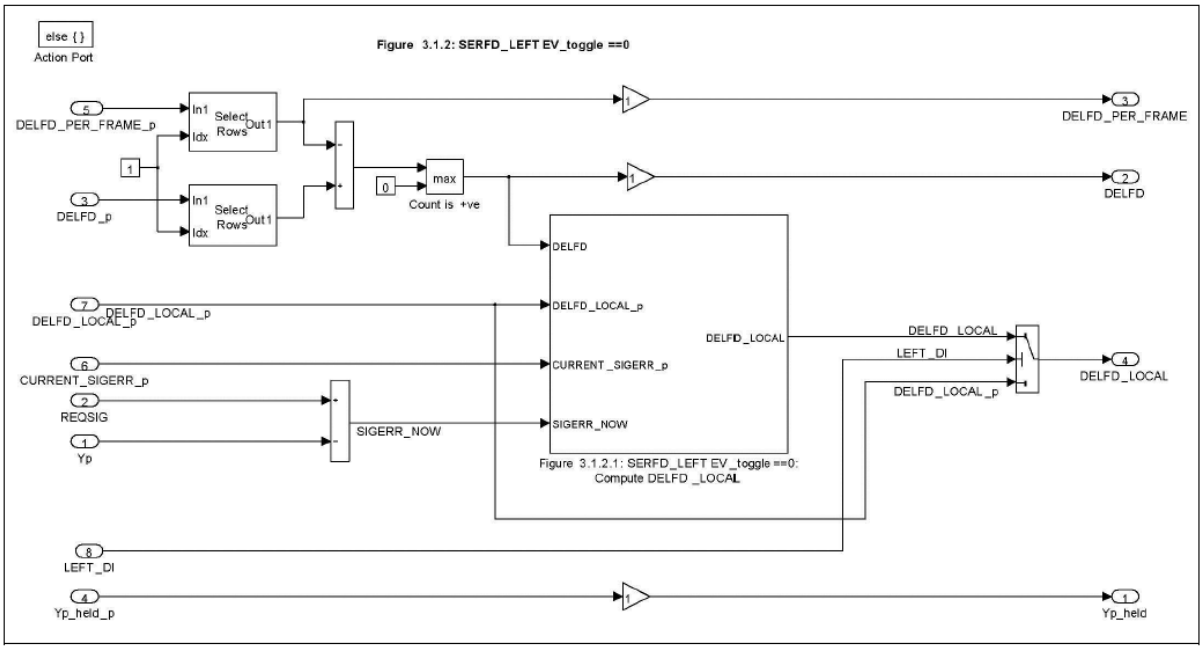


Figure 3.1.3 is same as Figure 1.1.3, hence it is not shown separately




else { }
Action Port
Figure 3.1.2: SERFD_LEFT EV_toggle ==0
DELFD_PER_FRAME_p
DELFD_p
Select Rows
max
Count is +ve
DELFD_PER_FRAME
DELFD
DELFD_LOCAL_p
CURRENT_SIGERR_p
REQSIG
Yp
SIGERR_NOW
DELFD_LOCAL
LEFT_DI
Yp_held_p
Yp_held
Figure 3.1.2.1: SERFD_LEFT EV_toggle ==0: Compute DELFD _LOCAL

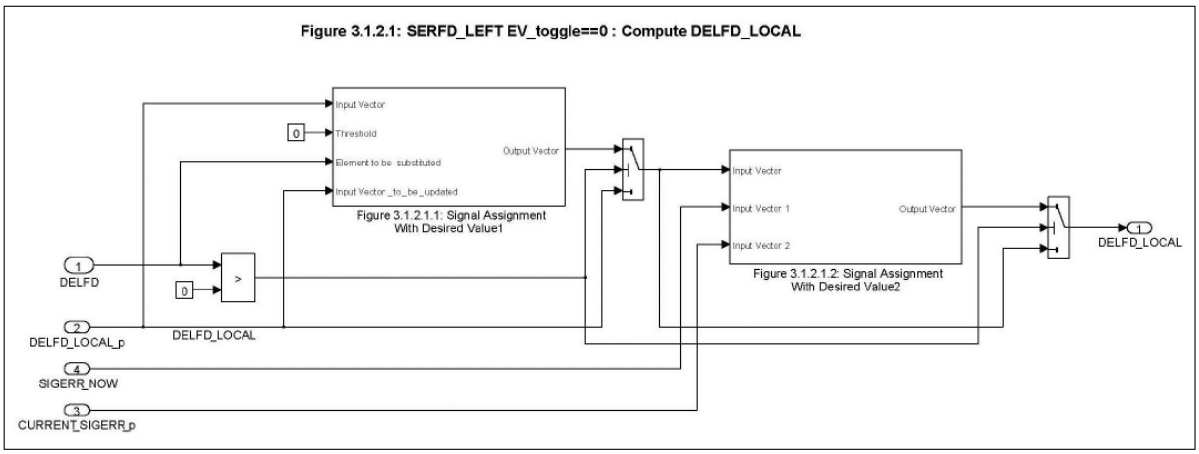


Figure 3.1.2.1: SERFD_LEFT EV_toggle==0 : Compute DELFD_LOCAL
Input Vector
Threshold
Element to be substituted
Input Vector _to_be_updated
Output Vector
Figure 3.1.2.1.1: Signal Assignment With Desired Value1
Input Vector 1
Input Vector 2
Figure 3.1.2.1.2: Signal Assignment With Desired Value2
DELFD
DELFD_LOCAL_p
SIGERR_NOW
CURRENT_SIGERR_p
DELFD_LOCAL

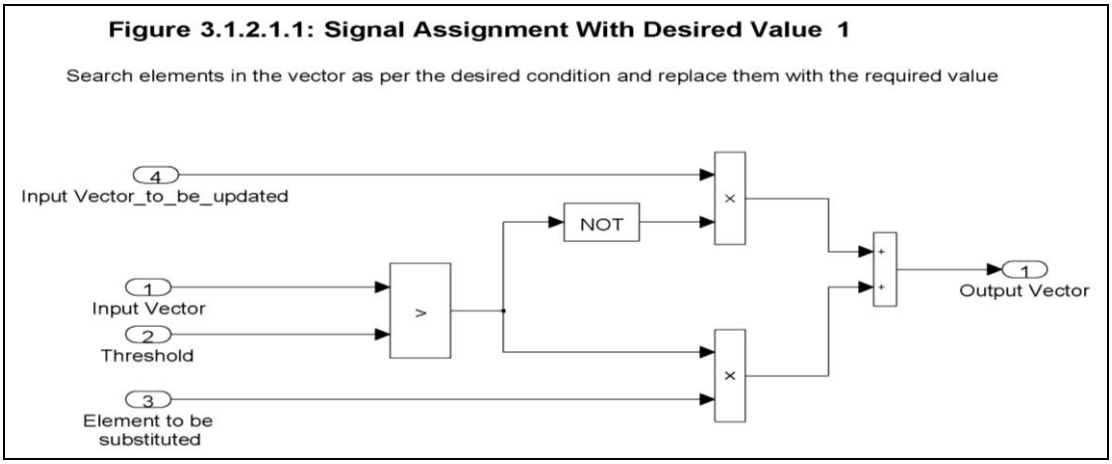


Figure 3.1.2.1.1: Signal Assignment With Desired Value 1
Search elements in the vector as per the desired condition and replace them with the required value
4
Input Vector_to_be_updated
NOT
×
1
Input Vector
>
2
Threshold
3
Element to be substituted
×
+
+
1
Output Vector

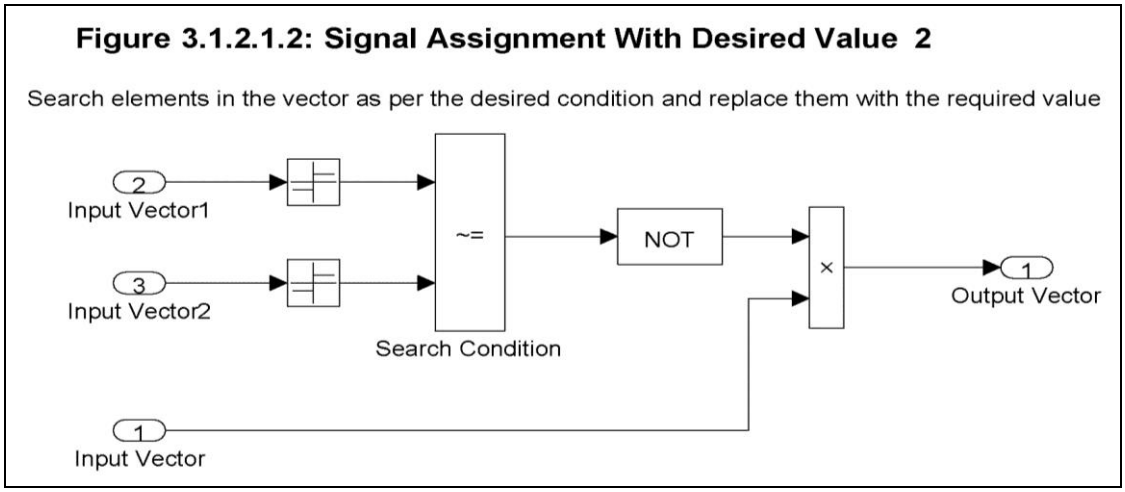


Figure 3.1.2.1.2: Signal Assignment With Desired Value 2
Search elements in the vector as per the desired condition and replace them with the required value
2
Input Vector1
3
Input Vector2
~=
Search Condition
NOT
×
1
Output Vector
1
Input Vector

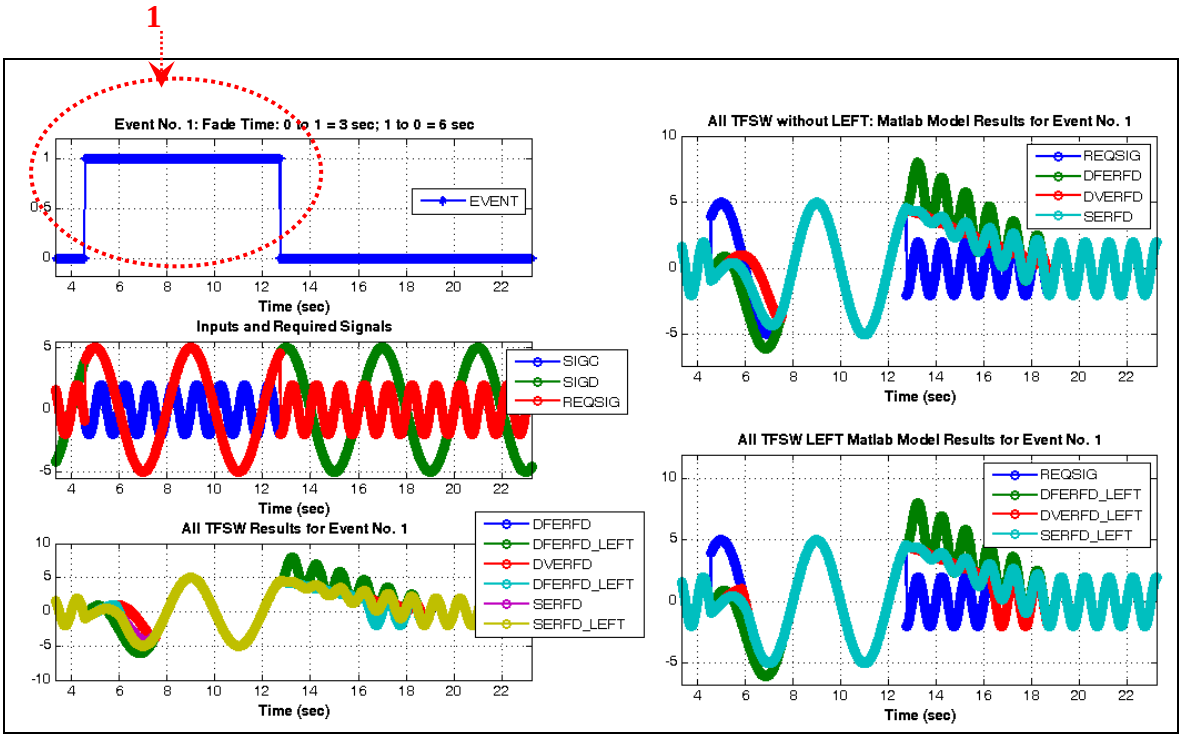


Figure 4: ALL TFSW Results of Matlab *.m file model, With and Without LEFT

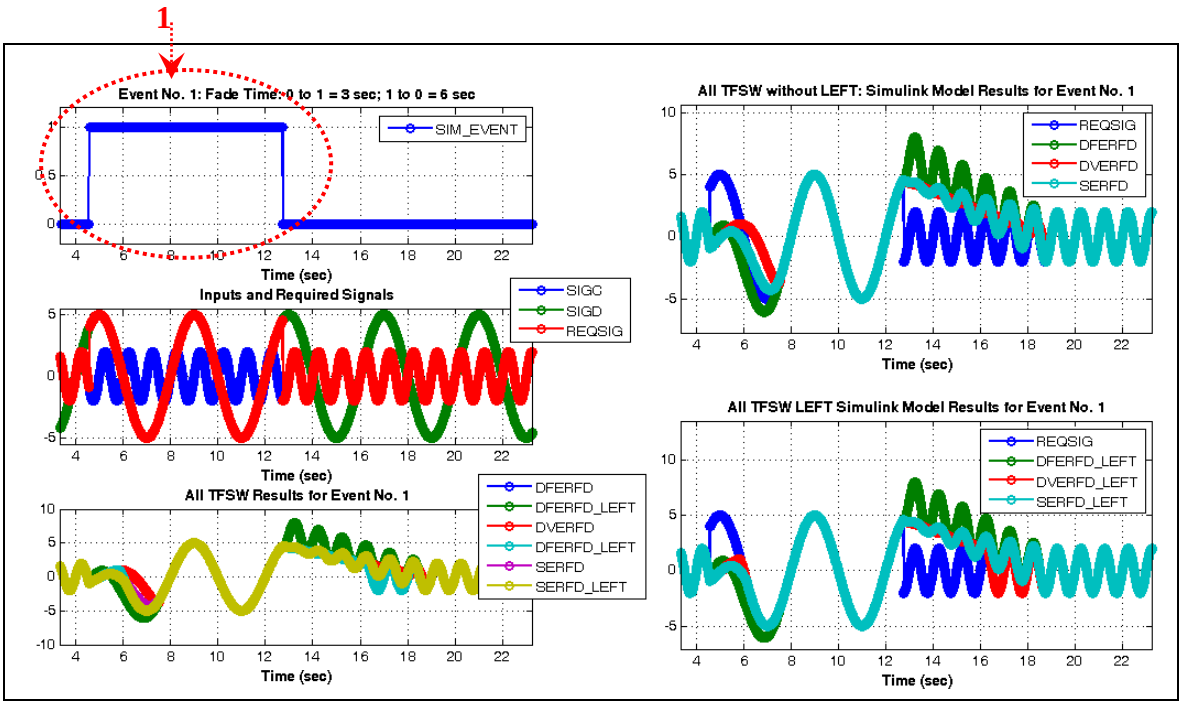


Figure 5: ALL TFSW Results of Simulink *.mdl file model, With and Without LEFT

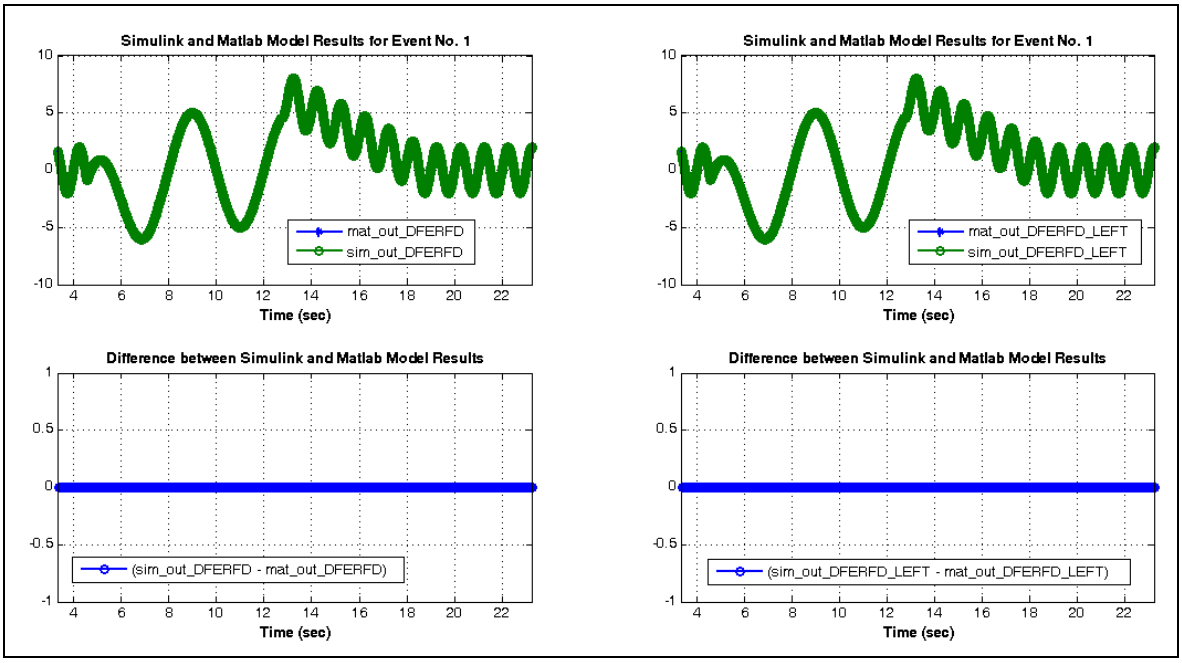


Figure 6: DFERFD TFSW Results from Matlab and Simulink model

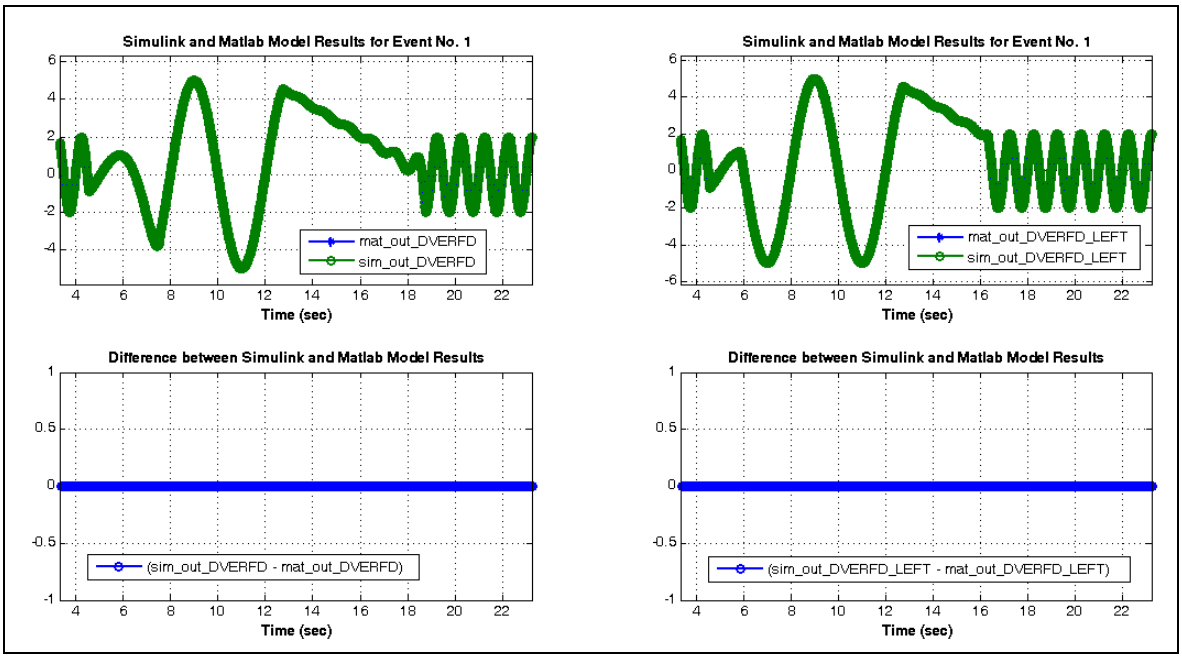


Figure 7: DVERFD TFSW Results from Matlab and Simulink model

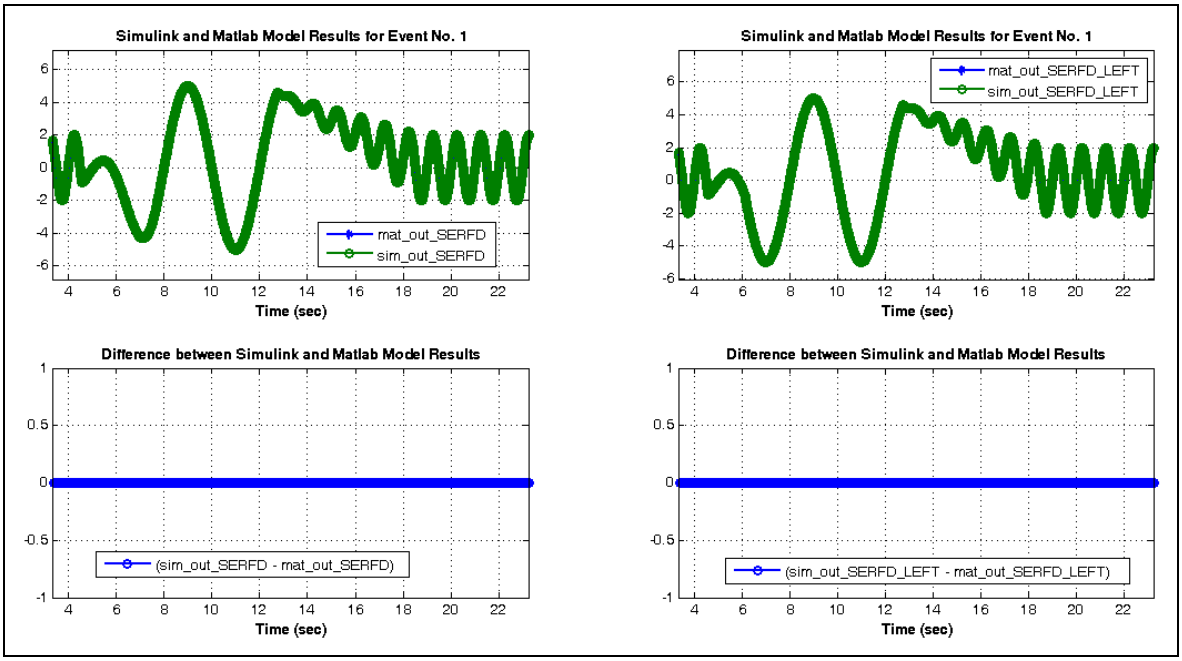


Figure 8: SERFD TFSW Results from Matlab and Simulink model

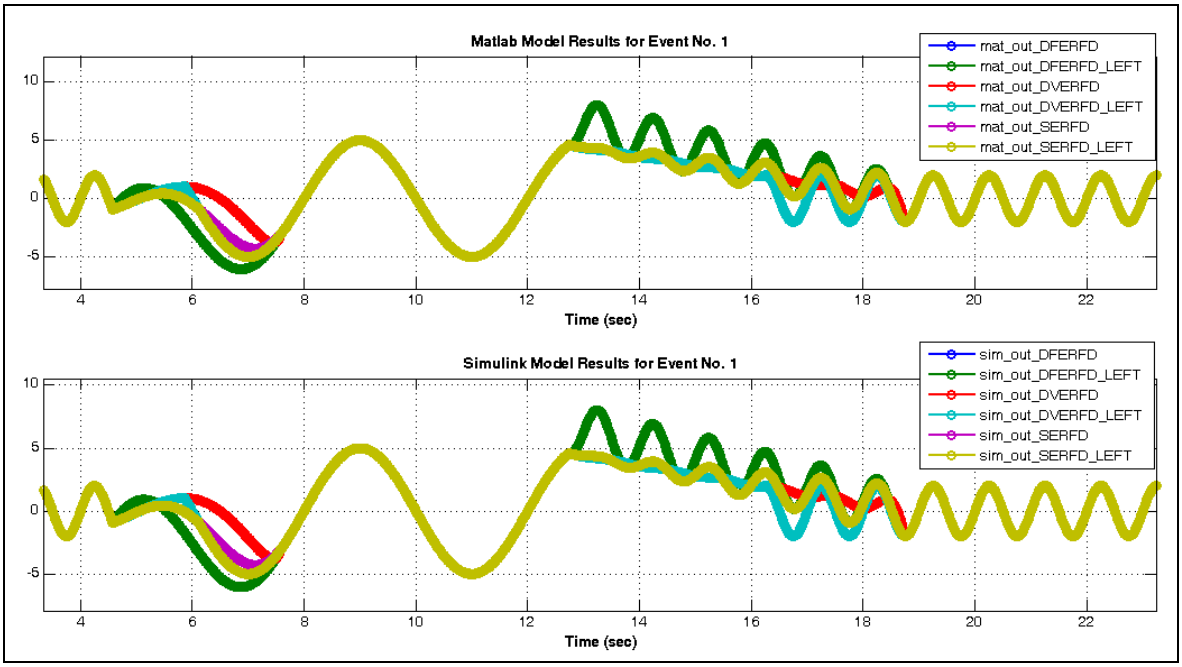


Figure 9: All TFSW Results of Matlab (*.m file) and Simulink (*.mdl) model outputs