RESEARCH ARTICLE OPEN ACCESS

Various Types of Soft Transient Free Switching in Embedded Systems

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
With the advancement in technology, many features of the large-scale systems are being embedded and miniaturized. This has led to investment of significant efforts on development and implementation of various ‘soft elements’ of embedded systems including their testing and verification. The ‘soft elements’ here is broadly referred to several of the dynamic elements like filters, faders or transient free switches, rate limiters etc. which are used in the soft form (as part of the computational algorithms within the software) within the embedded systems for various applications. 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 the unwanted effects, especially transients in the final outputs or commands 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. This article presents details on the fader or transient free switching computations. The article provides brief recursive formulation of the fader implementation as well as proposes various types of possible faders and their comparative analysis. The article also proposes the TFSW with the feature of termination of computations at Less than or Equal to Fader Time (LEFT), if the signal reaches to the selected output prior to the completion of the pre-fixed time. Functionality of the proposed faders is demonstrated through simulation results. Based on the experience gained over a period of time while working on safety critical embedded systems, some guidelines for formulating the TFSW related requirements have also been provided. The relevant experiences are also shared with the help of a few illustrative examples.

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

Date of Submission: 01-08-2023 Date of acceptance: 11-08-2023

I. INTRODUCTION

With the advancement in technology, many features of the large-scale application systems are being embedded and miniaturized. This has led to investment of significant efforts on development of various ‘soft elements’ of embedded systems including their testing and verification [1-5]. The ‘soft elements’ here is broadly referred to several of the dynamic elements like filters, faders or transient free switches, 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. 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 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].

This article presents details on the fader or transient free switching computations. The article provides brief recursive formulation of the fader implementation as well as propose various types of possible faders and their comparative analysis. The article also proposes the TFSW with the feature of termination of computations at Less than or Equal to Fader Time (LEFT), if the signal reaches to the selected output prior to the completion of the pre-fixed time. Based on the experience gained over a period of time while working on safety critical embedded systems, some guidelines for formulating the fader related requirements have also been provided. The relevant experiences are also shared with the help of a few illustrative examples.

The article is organized as given below. After introduction, Section 2 presents generic background of the embedded systems involving software and hardware and associated constraints which are usually encountered. Section 3 presents the details on the soft Transient Free SWitch (TFSW) which is also referred to soft fader. This section provides brief recursive formulation of the fader as well as proposes various types of possible faders and their comparative analysis. The subsection herein also presents the TFSW with the feature of termination of computations at Less than or Equal to Fader Time (LEFT), if the signal reaches to the selected output prior to the completion of the pre-fixed time. Section 4 provides the details of the proposed faders through simulation results demonstrating their functionality. Section 5 presents experiences gained over a period of time and lessons learnt. Section 6 presents some guidelines for formulating the fader related requirements. These guidelines have been arrived at based on the experiences gained over a period of time while working on the design and development of the safety critical fly-by-wire flight control system. Section 7 concludes the paper.

II. EMBEDDED SYSTEMS AND ON-BOARD CONSTRAINTS

The algorithms functioning in the form of large-scale software involve several nonlinear and dynamic computational elements, and intensive logic for reconfiguration (failure detection, isolation, selection). The logic itself is dependent on the several other discrete conditions which may be external due to user demand or the system failure or internally generated by the system due to satisfying certain criteria during the operation as intended.

The nonlinear elements such as nonlinear functions or lookup tables themselves may be fixed or reconfigurable depending upon the operating point of the plant or failure in the system. The dynamic elements involve faders (used for smooth transition of the signal from present value to the desired value over a specified time whenever reconfiguration takes place on occurrence of a specific event), filters (used for smoothening the signal within a range of frequency of interest or extracting the signal with range of frequency of interest), persistency counters (used for counting the elapsed time on occurrence of an event) etc. The analysis of the data evaluation of multi-input multi-output (MIMO) system involving such elements and their interdependencies becomes very complex. Over and above, the noise due to electronics hardware also adds some uncertainty on the exact matching of the data with the expected value. It may be noted that the ground test rigs used for evaluation of such safety critical systems have their own additional hardware to facilitate the data acquisition at various intermediate stages. The inherent characteristics of these additional hardware elements are required to be understood thoroughly. In the data analysis, it is required to identify and distinguish the effects of on-ground test rig and onboard system hardware characteristics, separately on the overall clearance of the implemented system.

After the entire embedded system (including the hardware and software) is cleared for initial onboard evaluation during prototype development, there could be more number of iterations for the software upgradation to embed all the functional requirements gradually in a step by step manner. These new requirements need to be assessed thoroughly at on-ground test platforms. The evaluation at on-ground hardware-in-loop test rig is classified as ‘black box’ testing. In the ‘black box’ testing, tester does not have an idea of how the functional requirements are written / coded but only based on the available knowledge of the functional requirements, the testing needs to be carried out and results to be assessed.

Therefore, the system designer must understand the test setup and associated features apart from the constraints of the onboard hardware. Based on relevant knowledge, one should incorporate the required test points (also called intermediate states which are to be acquired during the testing either on-ground or on-board) from the algorithms residing in the form of software in the on-board computer.

2.1 CONSTRAINTS OF THE ON-BOARD SYSTEMS

With available hardware architecture of the embedded system, the increase in software functionality leads to the constraints on the execution time, i.e., the specific computations of the algorithms should be completed within the specified time frame. In case of safety critical fly-by-wire system for the high-performance combat aircraft, these constraints play a critical role. The onboard software has to play a crucial role of sending the data on the multiple data recording devices while doing the complex and voluminous computations in real time involving several tasks [5].

III. TRANSIENT FREE SWITCH OR SOFT FADER (TFSW)

The 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. The operation of the TFSW is illustrated in Figure 1.

3.1 Various Types of Faders

Common purpose of the TFSW is to enable smooth transition between the signals on setting of relevant discrete. The signals used for such reconfiguration also vary during the time of transition. Depending upon certain metrics as listed in Table 1, like: computational complexity, error reduction characteristics, memory requirement for storing the intermediate variables, constraint of rate limiting or maximum allowable variation in the signal per frame during fader time, implementation aspects, execution of the real time software / system, etc. affects the performance of the TFSW. Associated effects or signatures are seen in the selected signals computed from the TFSW. Thus, depending upon the way error is reduced in order to reach to the required signal over a finite time of the TFSW (or transition takes place gradually from one signal to another on toggling of the event), the following three broad categories or types of TFSWs are proposed:

1) Direct Fixed Error Reducing Fader (DFERFD)
2) Direct Variable Error Reducing Fader (DVERFD)
3) Scaled Error Reducing Fader (SERFD)

The classification of the TFSWs is shown in Figure 2. A brief description of each of the above TFSW is given below. Refer to Table 1 for the details of the above mentioned three faders for comparison with respect to various metrics including the computational structure.

1) **Direct Fixed Error Reducing Fader (DFERFD):** In this TFSW or fader, the difference between the value of the required signal after event transition and the value of the signal at the instant prior to the event toggle is computed as the Total Error. This Total Error is to be nullified over the number of samples or frame count computed as per the fader time of that event. The number of samples or frame count for the given fader time is computed by dividing the Fader_Time with the sampling time (reciprocal of which indicates computation rate). Thus, the required error to be nullified or reduced per frame is computed (Total Error / frame count) at the instant of event toggle detection. This 'decrementing error per frame' remains fixed through the completion of the fader time, despite the variation in the signal to be reached (required signal) and is directly reduced from the required signal. Hence, this TFSW is named as Direct Fixed Error Reducing Fader. The input signal at the instant prior to the event toggle remains as it is, and relative to which the required signal only gradually comes in the selected output signal.
2) **Direct Variable Error Reducing Fader (DVERFD):** In this TFSW or fader, the difference between the value of the required signal after event transition and the value of the signal at the instant prior to the event toggle is computed as the Total Error. Based on this Total Error, the required error to be nullified or reduced per frame is computed at the instant of event toggle detection. Thereafter, in the subsequent frame counts or residual period of fader time, the Total Error is recomputed (between the present value of the fader output or selected signal and the signal to be reached) as well as 'decrementing error per frame' based on the remaining number of frame counts (decrementing error per frame = Present Total

Error / remaining number of frame count). Thus, depending upon the variation in the signal to be reached, the Total Error varies, as well number frame counts and hence in turn it results in making the variation in the 'decrementing error per frame' every frame. This error is directly reduced from the required signal. Hence, this TFSW is named as Direct Variable Error Reducing Fader. The input signal at the instant prior to the event toggle remains as it is and relative to which the required signal only gradually comes in the selected output signal.

3) **Scaled Error Reducing Fader (SERFD):** In this TFSW or fader, a 'normalized delta per frame' a fixed number computed at the beginning is used for reducing the scale factor which is used for gradually bringing in the required signal contribution every frame or other way gradually reduce the error between the selected output and required signal. Here the term 'reducing the scale factor' is referred to a complemented value of the 'normalized delta per frame', which can be precisely written as (1-normalized delta per frame). Further, at the same time contribution of the input signal at the instant prior to the event toggle (which is held constant) is reduced gradually by using the same scale factor 'normalized delta per frame'. Hence, this TFSW is named as Scaled Error Reducing Fader. The 'normalized delta per frame' is computed as given here: normalized delta per frame = Sample_Time / Fader Time of the Event. As can be seen, the required signal (signal to be reached) vary over a period, hence despite being the constant value of the scale factor 'normalized delta per frame', the corresponding error reduction vary every frame ('decrementing error per frame vary' over a period). Thus, the simultaneously, frozen or last good value of the input signal at the instant prior to the event toggle reduces while the required signal gradually coming in. Thus, simultaneous increase and decrease effects of required and held signal occurs. Hence rate of error conversion is faster.

In all these TFSW, one needs to ensure that the 'per frame variation' in the signal always remain well within the specified rate limit value for that path in which the TFSW is required to be implemented. Otherwise, the TFSW would act as rate limiter. Thus, while designing or arriving at the fader time for each event, one need to ensure that the 'error reduction per frame' computed based on the maximum difference between the two input signals (to the TFSW) at any instant over the number of frame count (fader time) would not exceed the specified rate limit value for that path in which the TFSW is required to be implemented. In such a situation, then it may not matter much whether error reduction is linear or nonlinear. In such a situation, the other qualitative metrics as given in Table 1 may be taken into consideration for selecting and implementing the specific TFSW.

3.2 TFSW Computation Termination Before Time: TFSW_LEFT

Depending upon the nature of variation in the required signal, it is likely that the selected signal (output of the TFSW) may reach to the required signal before completion of the fader time. In such a situation, the feature for termination of TFSW operation either at Less than or Equal to Fader Time (LEFT) can be incorporated. Such condition or situation can be found out by detecting the change in the sign of the error between the past and the present time frame. The error is referred to the difference between the required signal at that instant and the selected signal output of the corresponding past instant. The change in the sign of error indicates that the output is reached to the desired selected signal and thereafter the selected signal may or may not deviate from the required signal depending upon the change therein. Therefore, continued fader computations although shall lead to converge to the output signal at the end of fader time, however it could be redundant. Thus, it would be better to terminate the TFSW computations, once the output or selected signal is reached to the required signal before completion of the specified time. It would also help in reducing the pumping in of the unwanted, nonlinear natured error (drift or oscillatory behavior) during the fader time in the output signal. 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, if 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.

3.3 Comparative Analysis of Proposed Faders

Table 1 presents comparison of the transient free switches with reference to various qualitative metrics. Due to paucity of the space and illustration complexity, the computational structure for the 'LEFT' feature is not shown in Table 1.

IV. IMPLEMENTATION OF FADER AND SIMULATION RESULTS

Simulation results of the various TFSWs are shown in Figures 3 to 15. Table 2 presents the list / arrangements of figures showing the results of various TFSWs. Table 2 also details the description on the contents of each figure, analysis or discussion on the results therein. Thus, Table 2 and set of Figures 3 to 15 altogether provide additional details for comparative analysis and summary thereof. Figure 15 clearly brings out the efficacy of the proposed termination feature namely 'LEFT' for the TFSW.

V. EXPERIENCES GAINED OVER A PERIOD OF TIME AND LESSONS LEARNT

This section presents a few experiences gained over a period of time and lessons learnt during the design, development and evaluation of Fly-By-Wire (FBW) Flight Control System (FCS) of high-performance fighter aircraft [1-6]. These examples are based on the white box testing (as part of the software verification and validation process) and black box testing (Hardware in loop testing and analysis):

1) Initially the implemented faders were of Direct Fixed Error Reduction (DFERFD) type. It was found later that, the TFSW output signals were exceeding the limits of the source and destination signals' permitted boundaries. Therefore, an additional set of minimum and maximum limit values on the fading signals were required to be defined. It added extra database arriving at in the design, implementation and computations for applying the limits. It worked for providing bounded / limited signals.
2) Although Direct Fixed Error Reduction (DFERFD) with additional set of minimum and maximum limits worked well for providing bounded / limited signals, but after addition of few events computations resulted in exceeding execution time limit. It made to put an extra effort to compute the global maxima and minima for all such signal associated with each TFSW. The TFSWs were in two-digit numbers in the entire onboard software. Further, thereafter for any updates to the FBW FCS, it required to verify the validities of the prior limits on the signal and otherwise update it. Thus, it became a massive task.
3) Therefore, a preliminary form of Scaled Error Reduction Fader (SERFD) was formulated [6], to tackle the issue. However, such modified form TFSWs are existing in large numbers in various places and there appears to be a scope to optimize their numbers. Details on the optimization on the numbers of TFSWs is being dealt with separately.
4) Illusion on fader implementation being incorrect due the range of the output signal acquired being narrower than actually required. In real time on-board software, the faders were working correctly. However, due to clipping of the recorded signal of the then implemented TFSW due to narrower range than the actual required, it appeared that the fader time not implemented correctly in the on-board software. Down the line test points (intermediate signals in the other parts of the algorithms / software) were found to be matching correctly, which led to infer that the TFSW functionality is correctly implemented and the issue is with the range of the signal being recorded was not specified correctly [5].
5) Higher rate signal discretes missing in the recorded data: The procedures to compute the discrete were executed at higher rate, while the discrete signals were recorded at lower rate. The effect of setting of one discrete for a one frame (but not seen in recorded data) was seen in the final outputs through the fading effects and thus confirmed the toggling of the events in real.

Flight data acquired in the recorder is used during development phase as well as in final operations / services for preventive maintenance, and feedback to the design house for improvement of the product. It is an important part of any product life cycle. Therefore, for the hardware and software development, especially for the functional / algorithm level development, robustness, reliability, consistency, interpretability, analyzability of the recorded data is very essential and it needs to be taken into consideration while developing a safety critical product. A significant work has been carried out on the data attributes, data quality and metrics [5].

VI. GUIDELINES FOR FORMULATING REQUIREMENTS

There are several points worth to mention as part of the guidelines towards formulating the requirements for TFSW. However, due paucity of the space, a few points on TFSW are given below. Other points shall be dealt with in a separate article.

1) **Fader Time:** It should be set such that the rate of transition between the two signals should not exceed the permitted rate limits within the system or in the path. Let us say the rate limit for the signals in the path wherein the fader is incorporated is RL unit/sec. When the fader discrete toggles, the selected signal transits from signal to A to B or vice versa with the rate less than or equal to R unit/sec. Taking this rate of R unit/sec, the required time for transition (fader time) can be computed as given below. Note that here signals A and B could be constants or varying over the entire range of operation or envelope, but within the prior known finite bounds:

   Ftime = max(X1-X2) / RL

   Where

   Ftime = Fader Time in second

   max(A) = Maximum value of signal A over the entire range of operation or envelope

   min(B) = Minimum value of signal B over the entire range of operation or envelope

   X1 = max(max(A), max(B))

   X2 = min(min(A), min(B))

   max(X1 – X2) = indicates the maximum difference between signals X1 and X2 over the entire range of operation or envelope

   RL = permitted or specified rate limit in unit/second

2) Range of the TFSW output signal: The range of the output of TFSW which is recorded in the on-board system should be computed by taking into account the global maxima and minima for all such signal associated with each TFSW. Then doubtful interpretation on the correct implementation of TFSW time and operation / functionality thereof, from the offline data analysis can be ruled out.

3) It appears that there could be a way to optimize the number TFSWs in the entire set of algorithms / part of the software. Specifically, if say there are N number of events, and each event deals with M number of signals at multiple places in the entire application layer, then it requires implementation of MxN number of TFSWs. However, intuitively it appears that, ideally number of TFSWs should be same as that of the number of events / discretes by making the vector of inputs and outputs of all TFSWs. These vectors are updated every frame and signals therein are used at the respective place of reconfiguration for each TFSW. Then it would probably eliminate at least half the part of repetitive computations of TFSWs, and thus it would aid in reducing the memory requirement as well as accelerating the execution time. However, efficacy of such scheme needs to be assessed after actual implementation. Details of such aspects shall be dealt with in a separate article.

VII. CONCLUSION

In this article details on the fader or Transient Free SWitching (TFSW) computations and relevant aspects are presented. Brief recursive formulation of the fader implementation as well as various types of possible faders proposed and their comparative analysis is presented. Further, TFSW with the feature of termination of computations at Less than or Equal to Fader Time (LEFT), if the signal reaches to the selected output prior to the completion of the pre-fixed time is newly proposed. Functionality of the proposed faders is demonstrated through simulation results. Based on the experience gained over a period of time while working on safety critical embedded systems, some guidelines for formulating the fader related requirements have also been provided. The relevant experiences are also shared with the help of a few illustrative examples. These guidelines are expected to be applicable for most of the embedded systems and may get enriched further by the experiences and lessons learnt from other systems.

The following few aspects on the TFSW implementation can be researched in future to cater for real time computations of the embedded systems of safety critical nature:

- Multiple or combined events / discretes dependent faders, where the events are triggered sequentially or simultaneously.
- Rate of computation, memory requirements and overall execution time of the processor, etc.

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

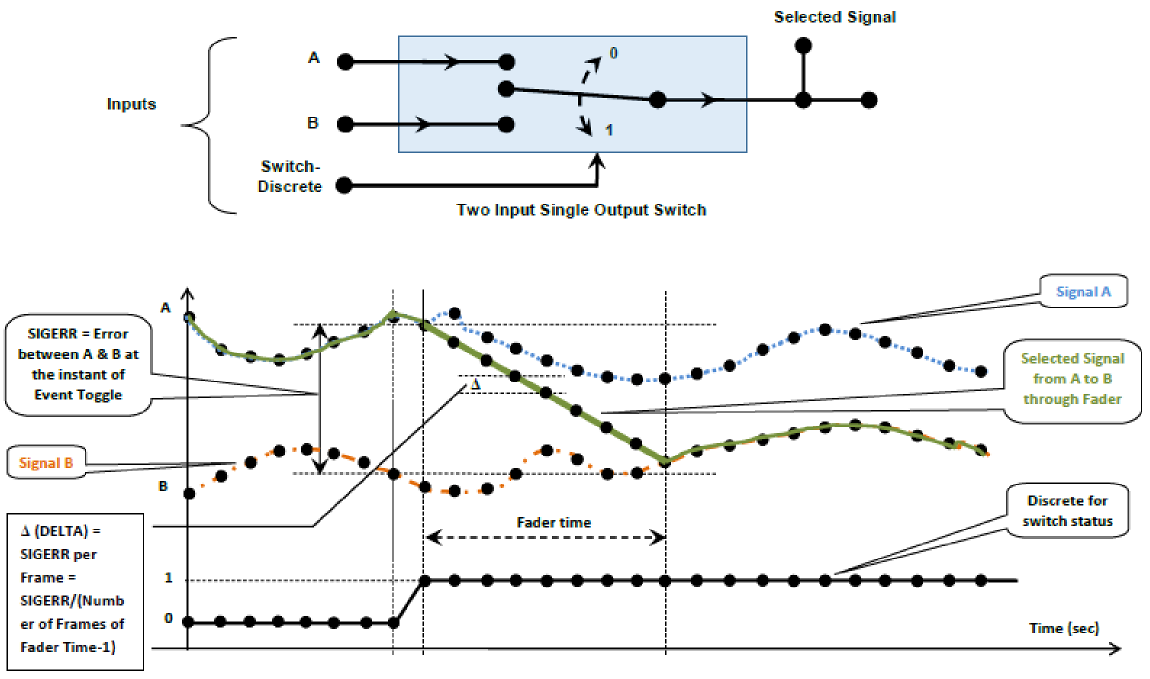
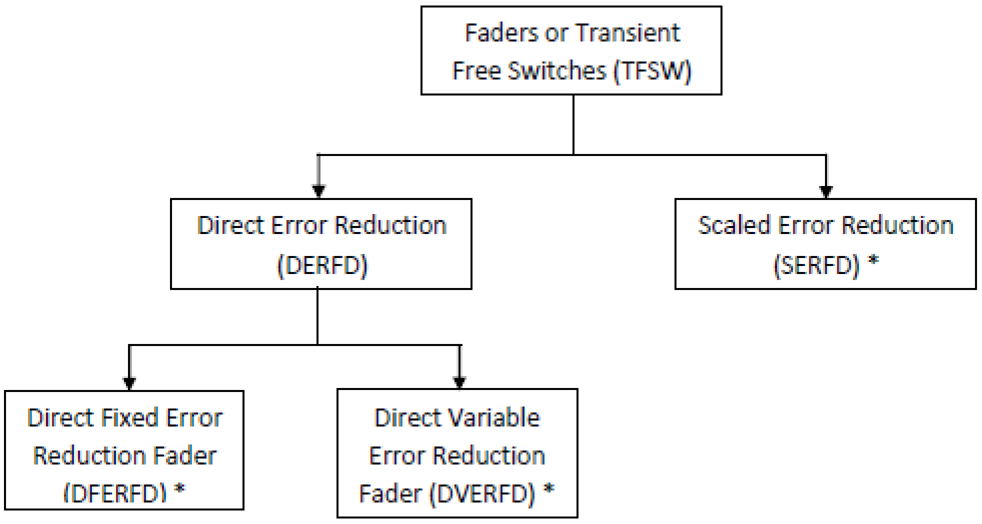

Selected Signal
Inputs
A
B
0
1
Switch-Discrete
Two Input Single Output Switch
Signal A
SIGERR = Error between A & B at the instant of Event Toggle
Selected Signal from A to B through Fader
Signal B
Δ
Δ (DELTA) = SIGERR per Frame = SIGERR/(Number of Frames of Fader Time-1)
Fader time
Discrete for switch status
1
0
Time (sec)

Figure 1: Transient Free Switch Operation: Signal Selection and Transit Process Illustration


Faders or Transient Free Switches (TFSW)
Direct Error Reduction (DERFD)
Scaled Error Reduction (SERFD) *
Direct Fixed Error Reduction Fader (DFERFD) *
Direct Variable Error Reduction Fader (DVERFD) *

* If the selected signal (output) of the Transient Free Switch / Fader (TFSW) reaches to the required signal before completion of the fader time (as the required signal may vary) then the feature for termination of TFSW operation at either Less than or Equal to Fader Time (LEFT) can be incorporated. It can be found out by detecting the change in the sign of error and then taking the suitable action for termination of fader computation (indicates that the output is reached to the desired selected signal). Thus, it helps in reducing the unwanted pumping in of error in a cyclic manner during the fader time (a nonlinear effect) in the output signal. Then such TFSW are 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'.

Figure 2: Classification of Transient Free Switches or Faders depending upon the way Error between the Required and Present Signal or Current Signal (prior to the Event Toggle) is reduced

Table 1: Comparative Analysis of Various Soft Faders or Transient Free SWitching (TFSW)

Sl. No.	Qualitative Property / Metric / Feature of TFSW	Direct Fixed Error Reducing Fader (DFERFD)	Direct Variable Error Reducing Fader (DVERFD)	Scaled Error Reducing Fader (SERFD)
1	Nature of Fader or Equation	*Y=REQSIG - CURRENT_SIGERR;* Where Y = current output REQSIG = Required Signal on Event Toggle CURRENT_SIGERR = It is computed every frame which indicates the error between the required signal and the current value. It is computed as given below as part of the Fader weight computation on Event Toggle 1) If Event is toggled, then select the applicable Maximum number of frame count (= Fader Time of the Event / Sample Time or Frame Time. It is an integer number) for that time depending upon 0 to 1 or 1 to 0 transition and compute: • CURRENT_FRAME_COUNT = MAX_FRAME_COUNT; • SIGERR_AT_EVENT_TOGGLE = REQSIG - Yp; • SIGERR_PER_FRAME = SIGERR_AT_EVENT_TOGGLE / MAX_FRAME_COUNT; • CURRENT_SIGERR = SIGERR_PER_FRAME* (CURRENT_FRAME_COUNT-1); 2) If event is not toggled, then continue computing the followings by decrementing the frame count. Frame Count Decrement in Fader Weight Limited to Minimum value of Zero • CURRENT_FRAME_COUNT = max(0, CURRENT_FRAME_COUNT_p-1); • SIGERR_AT_EVENT_TOGGLE = SIGERR_AT_EVENT_TOGGLE_p; • SIGERR_PER_FRAME = SIGERR_PER_FRAME_p; • CURRENT_SIGERR = SIGERR_PER_FRAME_p* CURRENT_FRAME_COUNT; Where • CURRENT_FRAME_COUNT = Current Frame Count • SIGERR_AT_EVENT_TOGGLE = Error between Required and Prior Frame Signal at Event Toggle • SIGERR_PER_FRAME = Signal Error Per Frame to be Reduced to Reach to Required Signal over Fader Time • MAX_FRAME_COUNT = Maximum Number of Frame Count for the specified Time on Transit of Event • T = Sample Time or Frame Time	*Y = REQSIG - CURRENT_SIGERR;* Where Y = current output REQSIG = Required Signal on Event Toggle CURRENT_SIGERR = It is computed every frame which indicates the error between the required signal and the current value. It is computed as given below as part of the Fader weight computation on Event Toggle 1) If Event is toggled, then select the applicable Maximum number of frame count (= Fader Time of the Event / Sample Time or Frame Time. It is an integer number) for that time depending upon 0 to 1 or 1 to 0 transition and compute: • CURRENT_FRAME_COUNT = MAX_FRAME_COUNT; • SIGERR_AT_EVENT_TOGGLE = REQSIG - Yp; • SIGERR_PER_FRAME = SIGERR_AT_EVENT_TOGGLE / MAX_FRAME_COUNT; • CURRENT_SIGERR = SIGERR_PER_FRAME* (CURRENT_FRAME_COUNT-1); 2) If event is not toggled, then continue computing the followings by decrementing the frame count. Frame Count Decrement in Fader Weight Limited to Minimum value of Zero • CURRENT_FRAME_COUNT = max(0, CURRENT_FRAME_COUNT_p-1); • SIGERR_AT_EVENT_TOGGLE = SIGERR_AT_EVENT_TOGGLE_p; Re-compute the 'Signal Error Per Frame' in every frame in order to avoid a large jump at the end of Fader Time, if the Required Signal is still away from the output achieved by that time. • SIGERR_NOW = REQSIG - Yp; • SIGERR_PER_FRAME = SIGERR_NOW / max(1,CURRENT_FRAME_COUNT); • if CURRENT_FRAME_COUNT==0 SIGERR_PER_FRAME = SIGERR_PER_FRAME*0; end • CURRENT_SIGERR = SIGERR_PER_FRAME * (max(0, CURRENT_FRAME_COUNT-1)); Where • CURRENT_FRAME_COUNT = Current Frame Count • SIGERR_AT_EVENT_TOGGLE = Error between Required and Prior Frame Signal at Event Toggle	*Y=REQSIG*(1-DELFD)+Yp_held*DELFD;* Where Y = current output REQSIG = Required Signal on Event Toggle Yp_held = Output prior to Event Toggle and continues to be held till fader time is complete (or the DELFD described below reaches to 0) DELFD = Normalized Delta Decrement in Fader Weight from 1 to 0 (Limited to Minimum value of Zero) It is computed as given below as part of the Fader weight computation on Event Toggle 1) If Event is toggled, then select the applicable 'Maximum Delta Per Frame' (= Sample Time or Frame Time/Fader Time of the Event. It is non-integer number) for that time depending upon 0 to 1 or 1 to 0 transition and compute: • Yp_held = Yp; • DELFD = 1; • DELFD_PER_FRAME = MAX_DELFD_PER_FRAME; 2) If event is not toggled, then continue computing the followings by decrementing the Delta (Normalized Scale Factor or Fader Weight) by 'Maximum Delta Per Frame' which is limited to Minimum value of Zero • Yp_held = Yp_held_p; • DELFD = max(0,(DELFD_p - DELFD_PER_FRAME_p)); • DELFD_PER_FRAME = DELFD_PER_FRAME_p; Where • Yp_held = Past output at the instant of Event Toggle held • DELFD = Normalized Fader Weight at current frame • DELFD_PER_FRAME = Normalized Fader Weight per Frame to be Reduced to Reach to Required Signal over Fader Time • CURRENT_SIGERR = Error between Required and Prior Frame output at present instant within the on-going Fader Time • DELFD_LOCAL = Set of Normalized Fader Weights at current frame for internal computations • MAX_DELFD_PER_FRAME = Maximum Normalized Weight Per Frame for the specified Time on Transit of Event

Sl. No.	**Qualitative Property / Metric / Feature of TFSW**	**Direct Fixed Error Reducing Fader (DFERFD)**	**Direct Variable Error Reducing Fader (DVERFD)**	**Scaled Error Reducing Fader (SERFD)**
		'_p' tag attached to the signal or variable indicates the value of that signal at the past frame.	• SIGERR_PER_FRAME = Signal Error Per Frame to be Reduced to Reach to Required Signal over Fader Time • CURRENT_SIGERR = Error between Required and Prior Frame output at present instant within the on-going Fader Time • REQSIG = Required signal on Event Transit • MAX_FRAME_COUNT = Maximum Number of Frame Count for the specified Time on Transit of Event • T = Sample Time or Frame Time '_p' tag attached to the signal or variable indicates the value of that signal at the past frame.	• T = Sample Time or Frame Time '_p' tag attached to the signal or variable indicates the value of that signal at the past frame.
2	Rate of Error Reduction (Fast / Moderate / Slow)	Input signal at the instant prior to the event toggle remains as it is and relative to which the required signal only gradually comes in the selected output signal. Therefore, rate of error conversion is moderate as compared to SERFD.	Input signal at the instant prior to the event toggle remains as it is and relative to which the required signal only gradually comes in the selected output signal. Therefore, rate of error conversion is moderate as compared to SERFD.	Simultaneously, frozen or last good value of the input signal at the instant prior to the event toggel reduces while the required signal gradually coming in. Thus, simultaneous increase and decrease effects of required and held signal occurs. Hence rate of error conversion is faster.
3	Type of Error Reduction (Linear / Nonlinear)	Linear Error Reduction	Nonlinear Error Reduction	Nonlinear Error Reduction
4	Smoothness of Signal Transition (at the end, especially)	• Error reduction per frame (or 'incremental delta magnitude per frame' for achieving the required signal) remains fixed over the enitre duration. • A Jump of the order greater than the 'incremental delta magnitude per frame' could occur in the signal transition either at the end and / or beginning can occur depending upon the variation in the required signal during fader time. • If the 'incremental delta magnitude per frame' variation is acceptable as per the design of fader time, then signal transition may be treated as smooth at the end.	• Error reduction per frame (or 'incremental delta magnitude per frame' for achieving the required signal) varies over the enitre duration depending upon the variation in the required signal during the fader time. • As such jump in the signal may not be attributable here because 'incremental delta magnitude per frame' vary over the enitre fader duration and accordingly smoothness of signal transition at the beginning and end can be seen. • If the 'incremental delta magnitude per frame' variation is acceptable as per the design of fader time, then signal transition may be treated as smooth at the end.	• 'Normalized delta per frame' used for scaling the Error is fixed over the entire duration, however, the 'scaled error per frame' with this normalized constant vary over a period. • As such jump in the signal may not be attributable here because 'scaled error per frame' vary over the enitre fader duration and accordingly smoothness of signal transition at the beginning and end can be seen. • If the 'scaled error per frame' variation is acceptable as per the design of fader time, then signal transition may be treated as smooth at the end.
5	Error limits during transition or fader time	Selected signal may exceed the limits of source and destination signals during fading time	Selected signal guarantedly remains witin the limits of source and destination signal during fading time	Selected signal guarantedly remains witin the limits of source and destination signal during fading time
6	Qualitative Number of computations	Moderate	More	Less
7	Complexity / Simplicity	Moderate	Complex	Simple
8	Execution Effectiveness	Moderate	More computationally intensive	Moderate
9	Design Acceptability	By design, if the occurrence of the jump in the signal at the end which could be more than the fixed value of the 'signal per frame', is acceptable, then is may be used.	By design, if the occurrence of the maximum variation in the 'signal per frame' anywhere during the fader time is acceptable, then it may be used.	By design, if the occurrence of the maximum variation in the 'signal per frame' (also referred to 'scaled error per frame') anywhere during the fader time is acceptable, then it may be used.
10	Usability preference (Assuming design acceptability and cosnidering computational aspects)	Preferred over DVERFD. If linear error reduction is mandatory.	If SERFD and DFERFD not suitable due to some reasons then DVERFD may be considered. If nonlinear error reduction is acceptable	Preferred over DVERFD and DFERFD If nonlinear error reduction is acceptable
11	Error Reduction Feature	Direct Error Reduction. Fixed value per fram is reduced	Direct Error Reduction with variable value per frame. It has got combined features of DFERFD and SERFD	Scaled Error Reduction with variable value
12	Memory Requirement	As compared to DVERFD less number of elements to be stored (intermediate states), but more than that of SERFD, hence moderate memory requirement	More number of elements (intermediate states) to be stored as compared to DFERFD and SERFD. Hence more memory requirement	Less number of elements (intermediate states) to be stored and hence less memory requirement

Table 2: Figure Arrangements and Analysis / Discussion of Results of Various Soft Faders or Transient Free Switches

Sl. No.	Figure No.	TFSW	Description	Additional Remarks / Analysis / Discussion
1	1	-	Transient Free Switch Operation illustration	Signal Selection and Transit Process Illustration
2	2	-	Classification of Transient Free Switches or Faders	Classification is shown depending upon the way Error between the Required and Present Signal or Current Signal (prior to the Event Toggle) is reduced
3	3	DFERFD	Event, Source (Current) signal, Destination Signal, Required, Selected Signal (Output of TFSW), and Intermediate signals in various subplots. Segment Numbers 1 to 5 are shown in Event plot are later used for showing the zoomed version of the results in Figure 9. The results in zoomed version of Figure 9 are shown in Figures 10 to 15.	Intermediate parameters or Fader Elements include: (1) Reducing Scale Factor on Error, (2) Signal Error (Required-Selected), (3) Output per Frame and Error per Frame, and (4) Frame Count. All of above are not applicable for each TFSW. However, as a common frame work for data analysis, they have been plotted in all Figures. Applicability for respective TFSW may be inferred if they found to be varying in the plots. ***Example: Reducing Scale Factor on Error not varying for DFERFD and hence it is not applicable for DFERFD TFSW.***
4	4	DFERFD_LEFT	Respective Signal similar to DFERFD	Reducing Scale Factor on Error: Not Applicable
5	5	DVERFD	Respective Signal similar to DFERFD	Reducing Scale Factor on Error: Not Applicable
6	6	DVERFD_LEFT	Respective Signal similar to DFERFD	Reducing Scale Factor on Error: Not Applicable
7	7	SERFD	Respective Signal similar to DFERFD	Frame Count Not Applicable
8	8	SERFD_LEFT	Respective Signal similar to DFERFD	Frame Count Not Applicable
9	9	All TFSW	Event, Required, Selected, Current Signals, Superimposed Outputs (Selected Signals) of TFSWs separately for with and without LEFT for ease of comparison and understanding	In this Figure Fader Elements are not shown. Instead superimposed plots of TFSW without LEFT in one subpot while results of TFSW with LEFT in another subplots are show.
10	10	All TFSW	ALL TFSW Results, With and Without LEFT: Zoomed version of Figure 9 at Event Toggle Segment No. 1	Results when Event transits from 0 to 1
11	11	All TFSW	ALL TFSW Results, With and Without LEFT: Zoomed version of Figure 9 at Event Toggle Segment No. 2	Results when Event transits from 1 to 0
12	12	All TFSW	ALL TFSW Results, With and Without LEFT: Zoomed version of Figure 9 at Event Toggle Segment No. 3	This segment shows the results when the Event got toggled back before completion of the Fading time (and operation thereof) due to the prior toggle.
13	13	All TFSW	ALL TFSW Results, With and Without LEFT: Zoomed version of Figure 9 at Event Toggle Segment No. 4	Results when Event transits from 0 to 1
14	14	All TFSW	ALL TFSW Results, With and Without LEFT: Zoomed version of Figure 9 at Event Toggle Segment No. 5	Results when Event transits from 1 to 0
15	15	All TFSW	ALL TFSW Results, With and Without LEFT: Additional Zoomed version of Figure 9 at Event Toggle Segment No. 5	Efficacy of LEFT Termination feature can be clearly seen in this Figure from the plots of with and without LEFT outputs (Selected Signals).

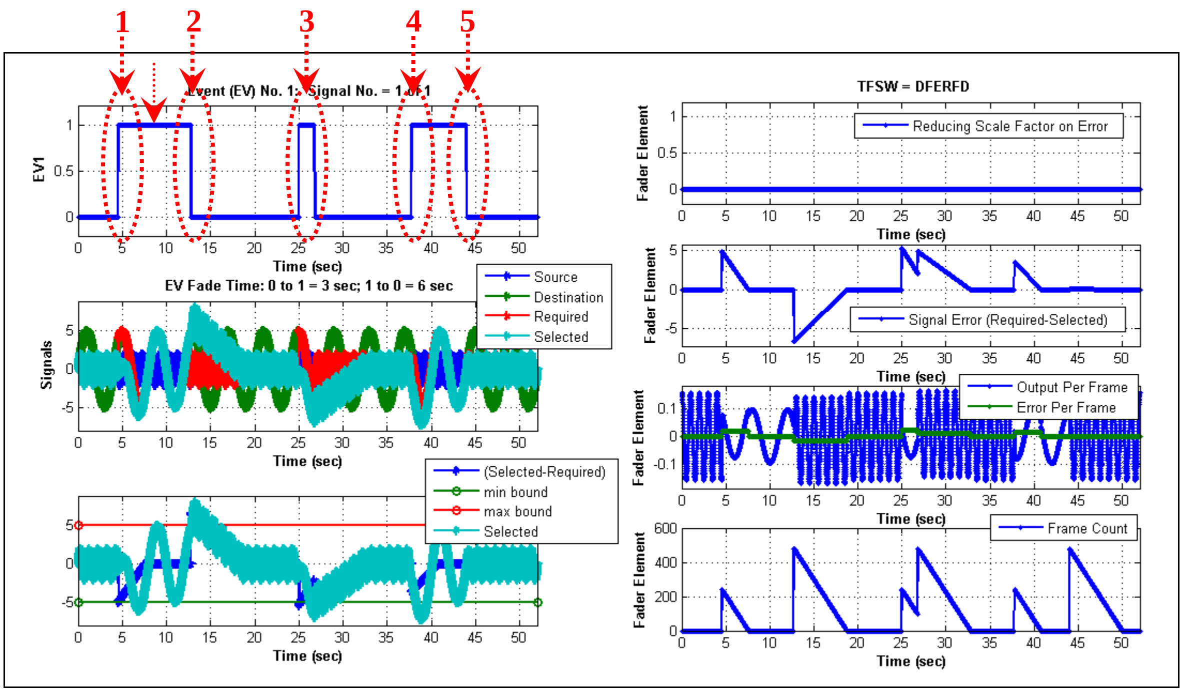


Figure 3: DFERFD TFSW Results

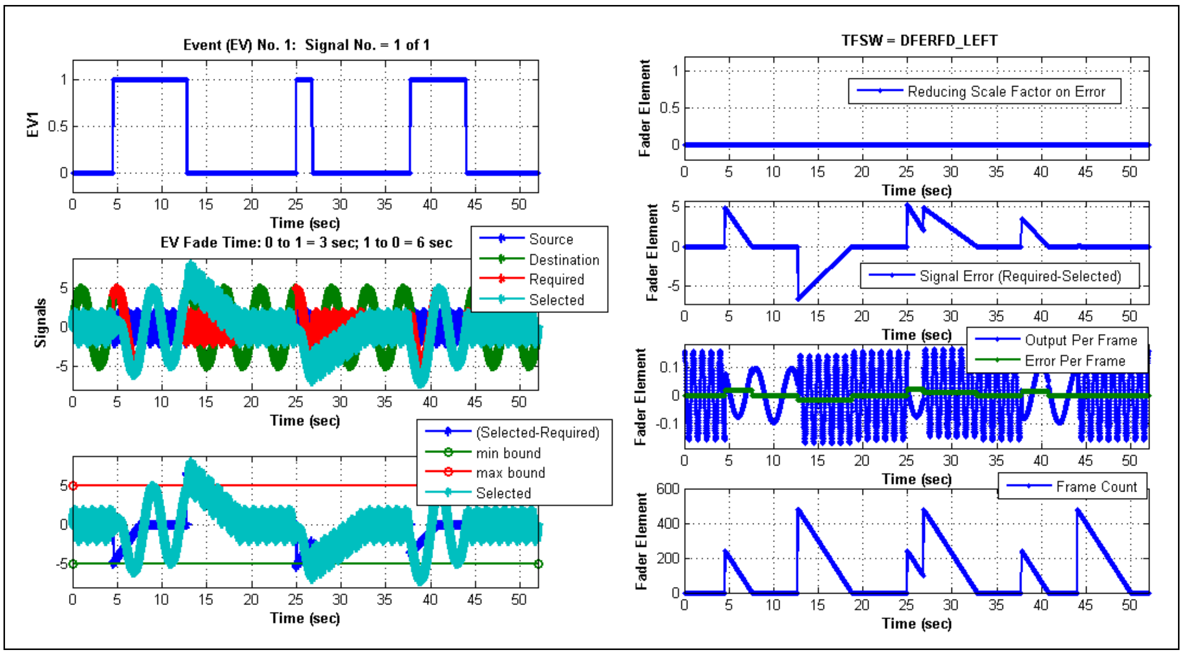


Figure 4: DFERFD_LEFT TFSW Results

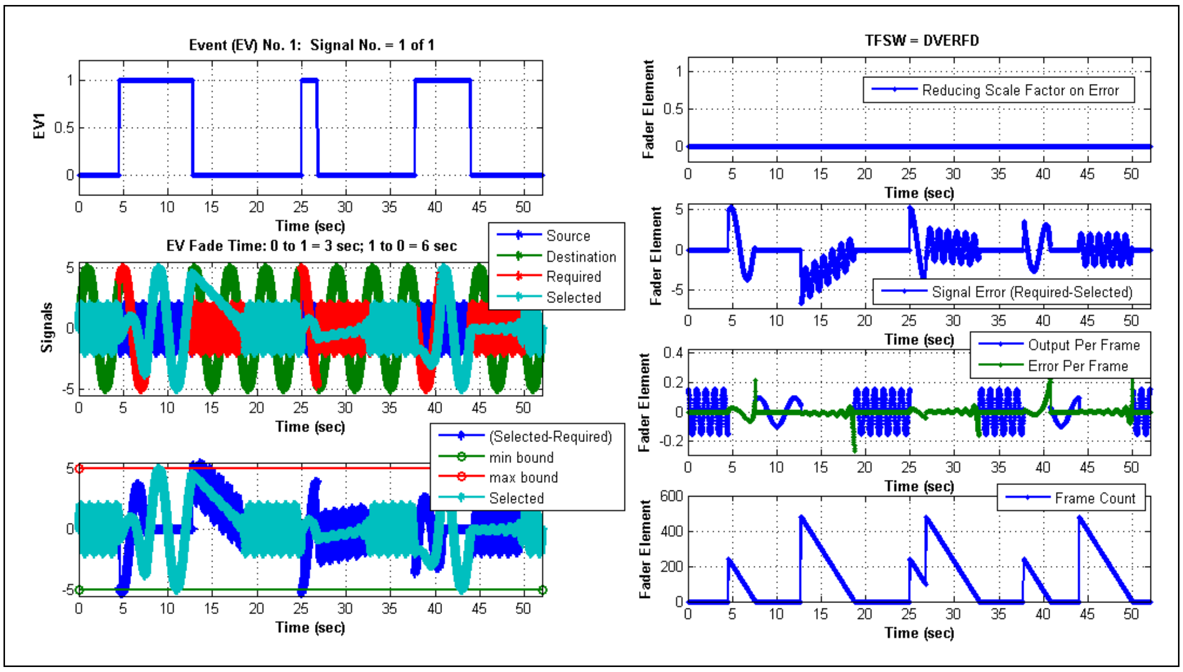


Figure 5: DVERFD TFSW Results

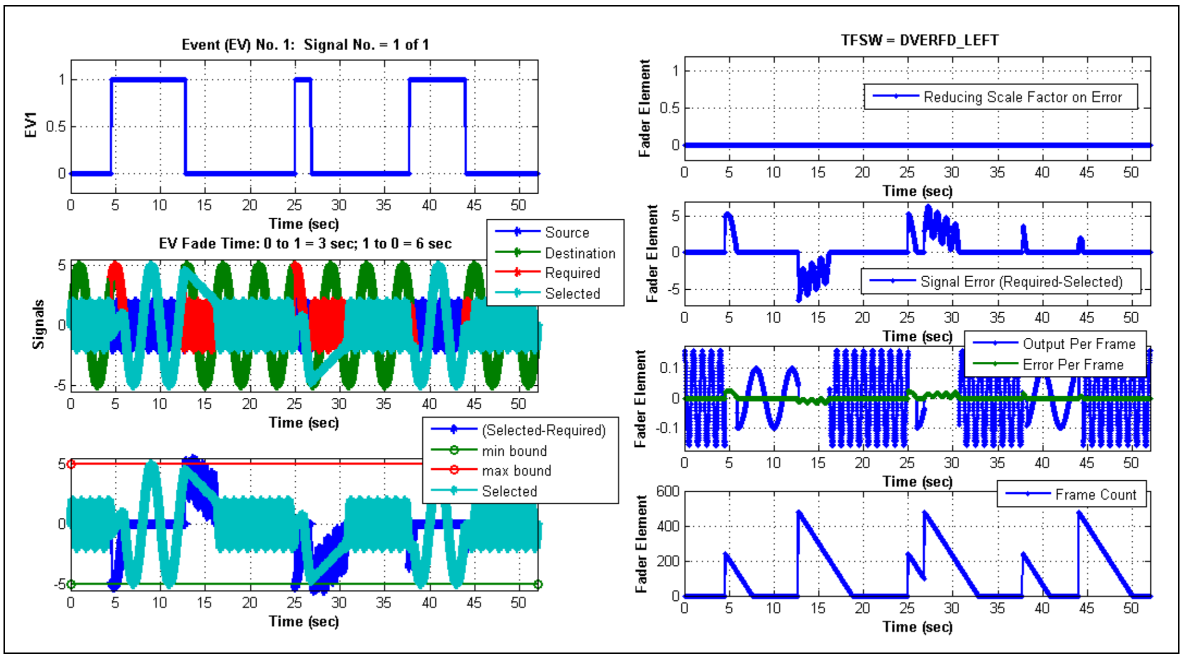


Figure 6: DVERFD_LEFT TFSW Results

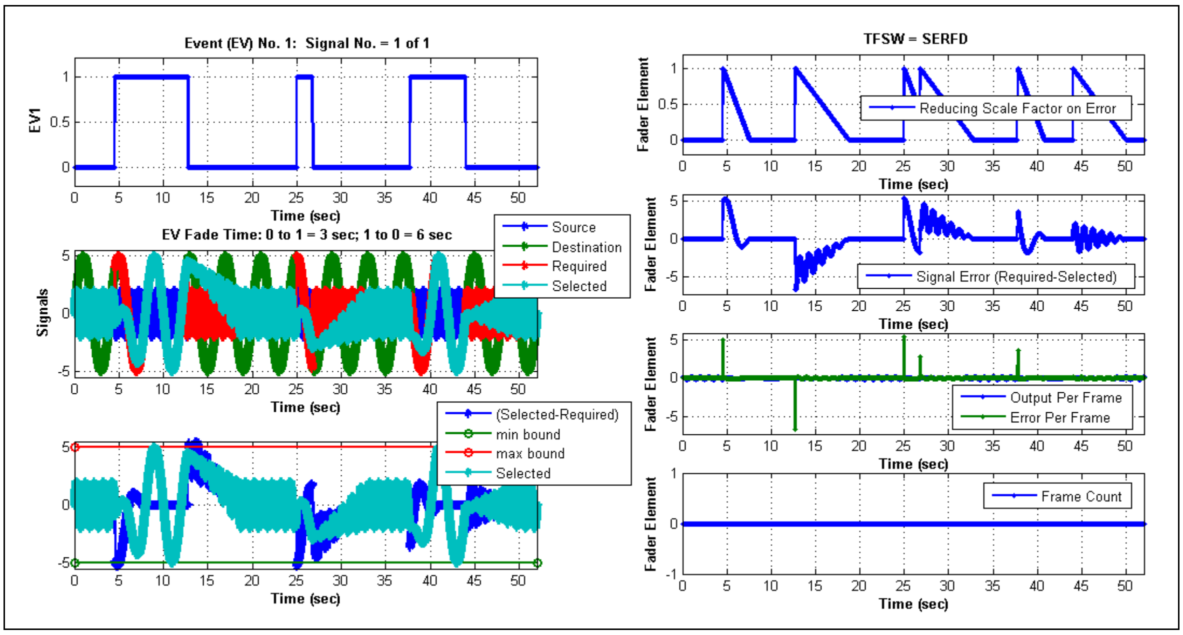


Figure 7: SERFD TFSW Results

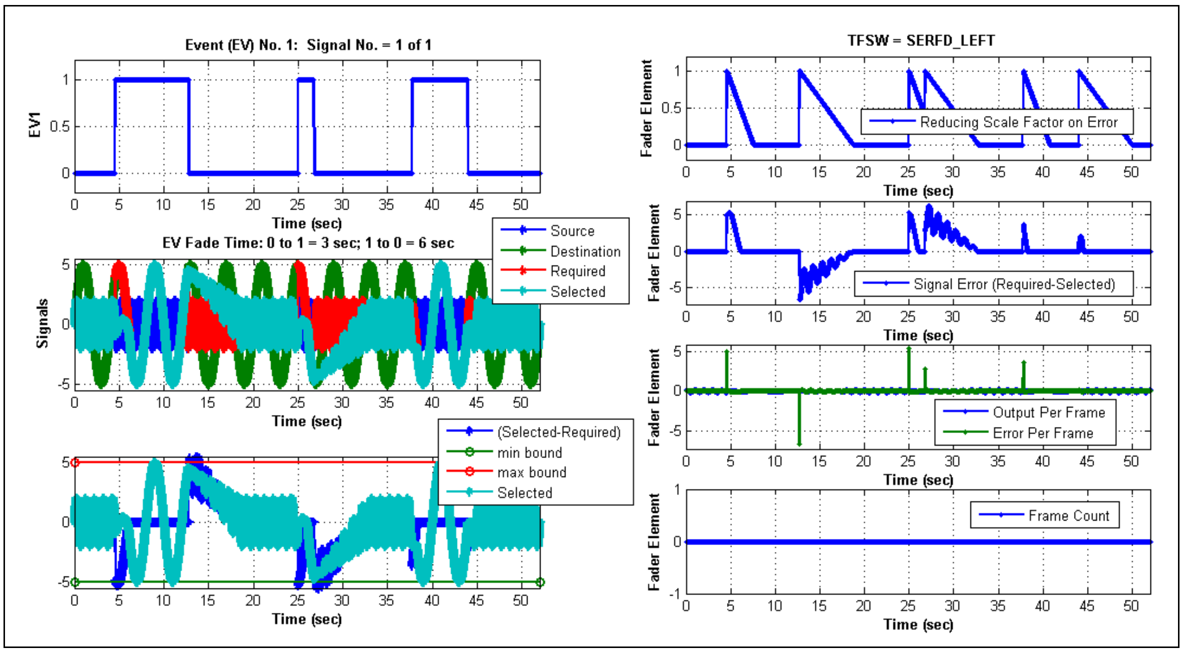


Figure 8: SERFD_LEFT TFSW Results

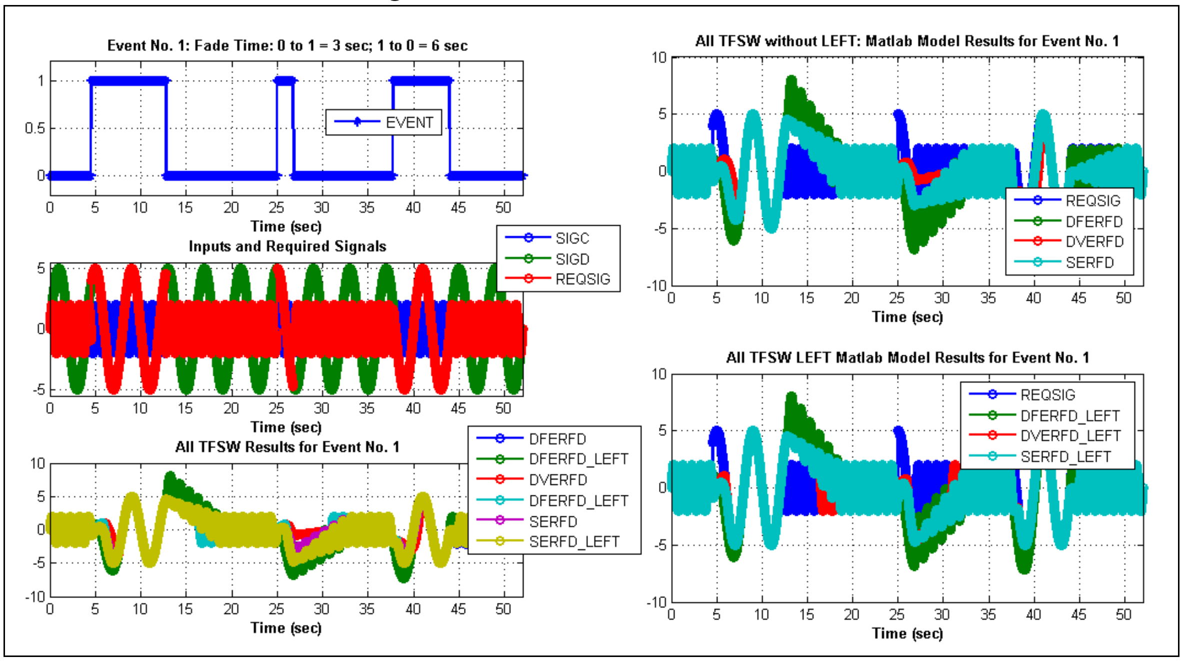


Figure 9: ALL TFSW Results, With and Without LEFT

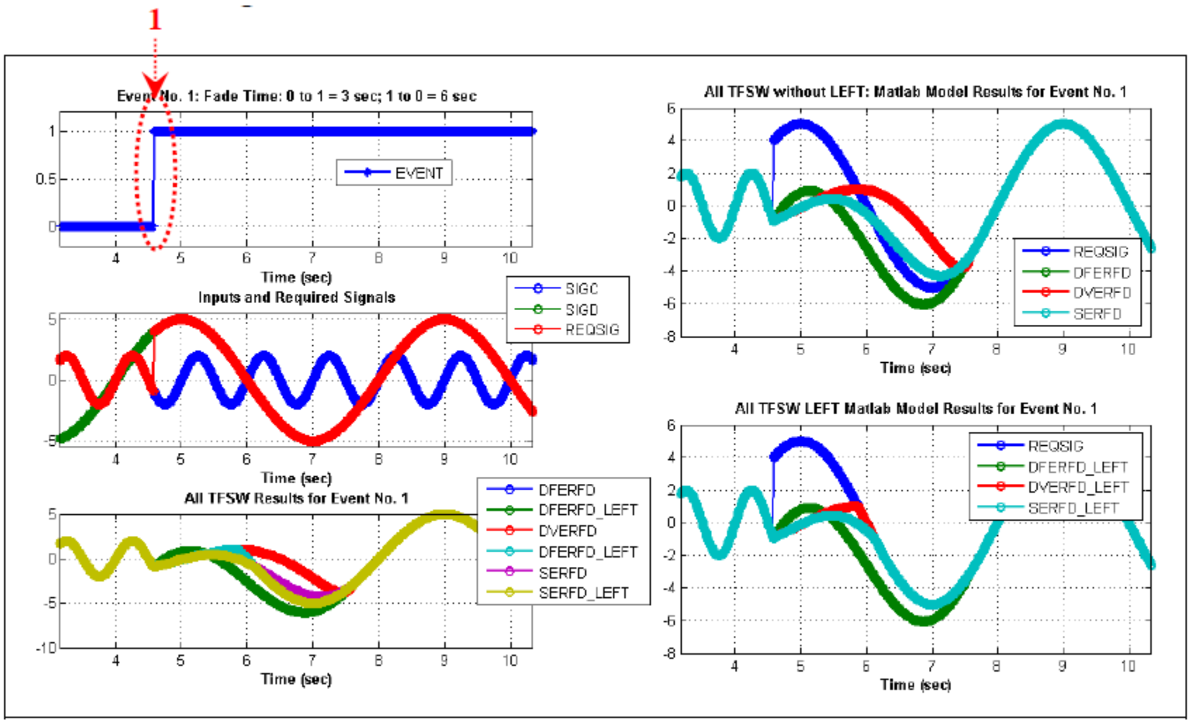


Figure 10: ALL TFSW Results, With and Without LEFT: Zoomed version of Figure 9 at Event Toggle Segment No. 1

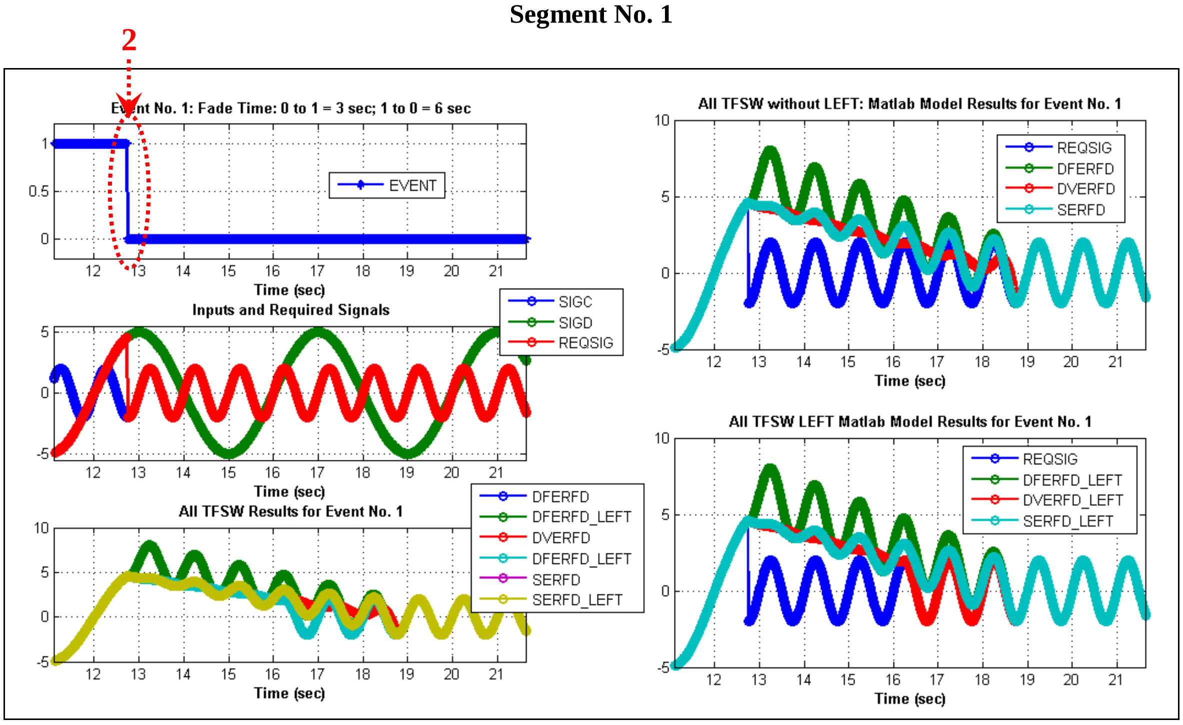


Figure 11: ALL TFSW Results, With and Without LEFT: Zoomed version of Figure 9 at Event Toggle Segment No. 2

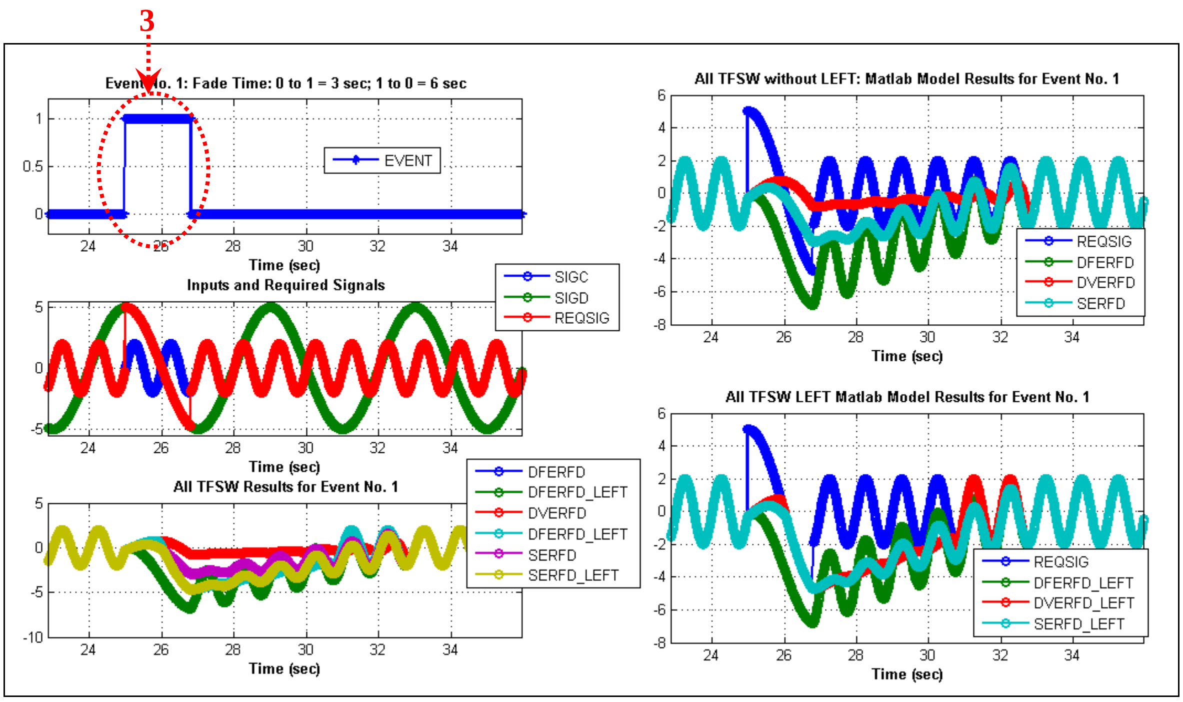


Figure 12: ALL TFSW Results, With and Without LEFT: Zoomed version of Figure 9 at Event Toggle Segment No. 3

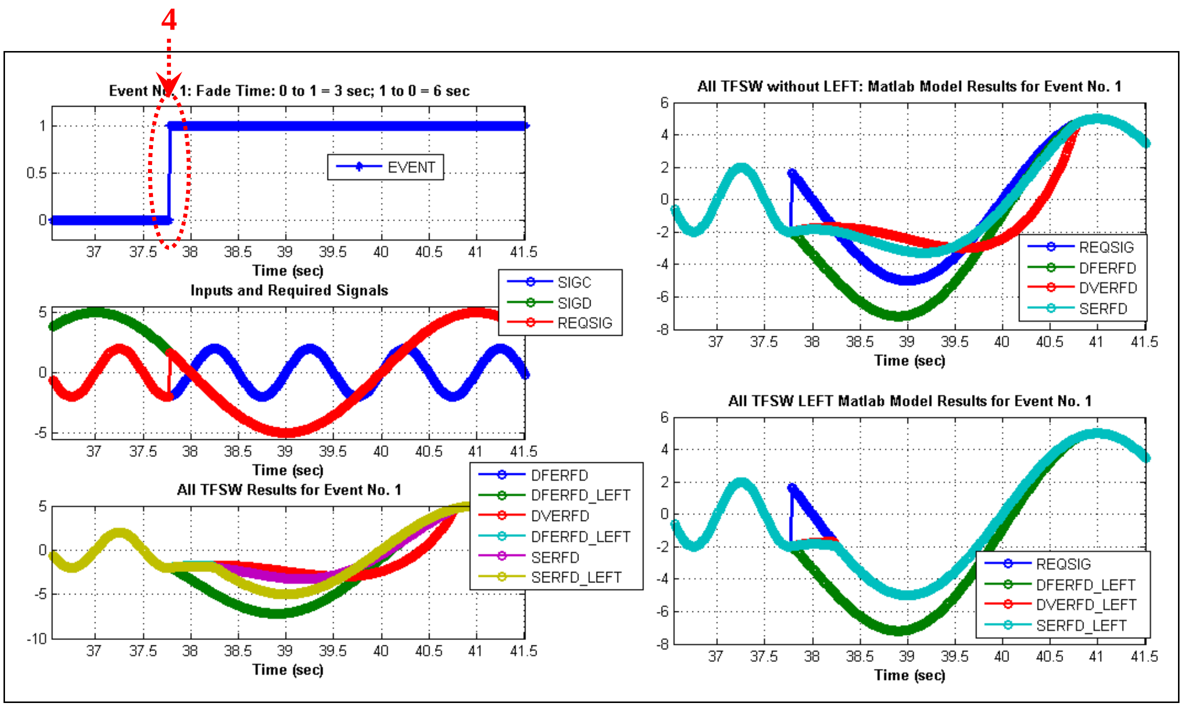


Figure 13: ALL TFSW Results, With and Without LEFT: Zoomed version of Figure 9 at Event Toggle Segment No. 4

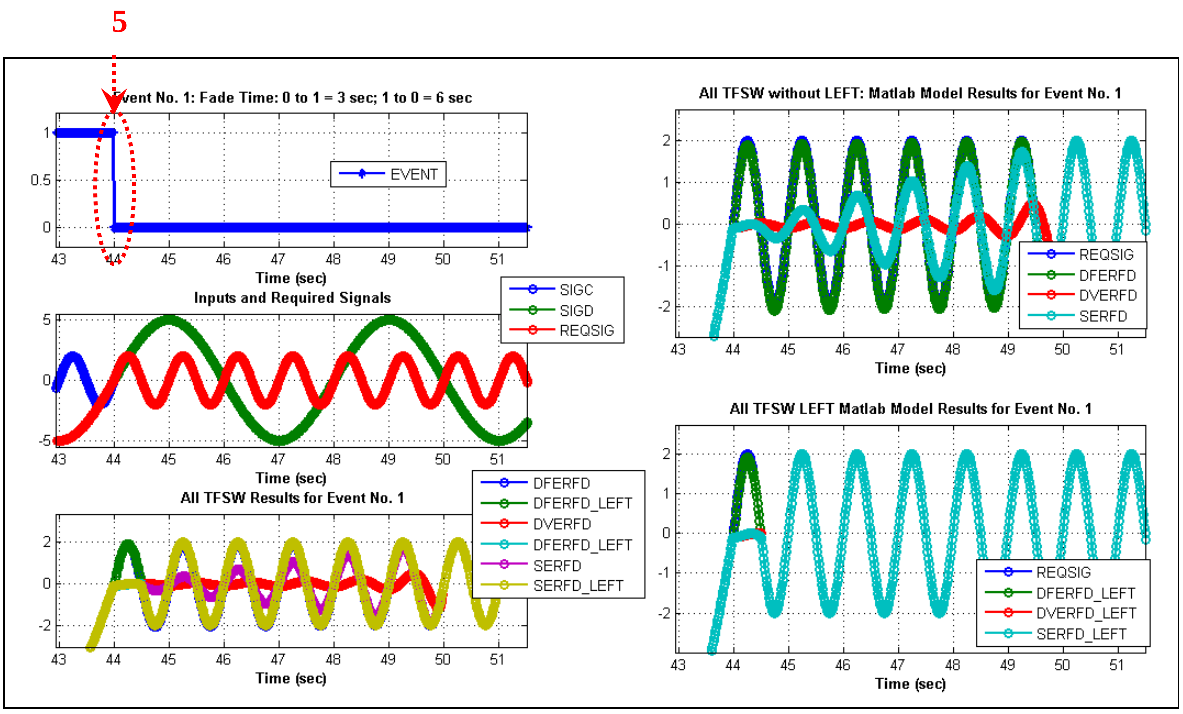


Figure 14: ALL TFSW Results, With and Without LEFT: Zoomed version of Figure 9 at Event Toggle Segment No. 5

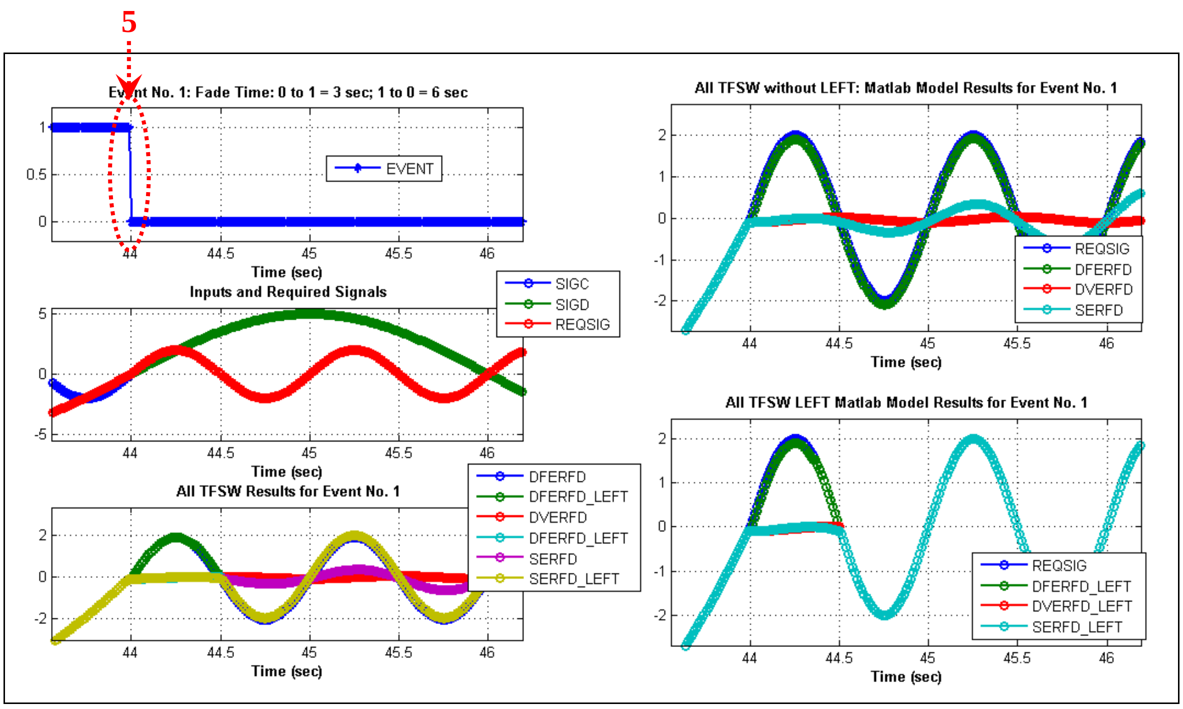


Figure 15: ALL TFSW Results, With and Without LEFT: More Zoomed version of Figure 9 at Event Toggle Segment No. 5. Efficacy of LEFT Termination feature can be clearly seen in this Figure from the plots of with and without LEFT outputs (Selected Signals).