Coupled Crash Live Deployment Simulation Using LS-DYNA® Functional Mock-up Interface

Ke Dong
General Motors Company
Xiaomeng Tong, Isheng Yeh
ANSYS

Abstract

Advanced driver-assistance systems (ADAS) consist of multiple advanced sensors. Information from these sensors has great potential to improve the performance of existing crash sensing systems. Integrated simulation of the active and passive sensing system is a key enabler to assess the potential benefit. Since ADAS sensing and crash sensing simulation are usually conducted in different environments, co-simulation capability is necessary to bring multiple different environments together to achieve the same simulation goal. In this paper, new co-simulation features have been developed in LS-DYNA using Functional Mock-up Unit (FMU). LS-DYNA is able to generate FMU and co-simulate with other software though Functional Mock-up Interface (FMI). A plugin package "FMU manager" built upon FMI 2.0 standard is provided to LS-DYNA users to implement the FMU import and export. The newly developed features were applied to a coupled live development crash simulation between Matlab/Simulink and LS-DYNA. The crash sensing algorithm was built in Matlab/Simulink and finite element vehicle and airbag models were built in LS-DYNA. Co-simulation between them demonstrated the capability of live deployment simulation under different crash scenarios without modifying the models. In future, the coupled live deployment model will be further integrated with ADAS sensing simulations to explore the benefits of the integrated sensing system.

Introduction

Increasing number of vehicles have been equipped with active safety systems. Most of active safety systems on the market provide functions as assistance to the driver to prevent frontal collisions, for example, automated emergency braking (AEB), and to maintain the vehicle maneuver, for example, lane keeping assist (LKA) and lane departure warning (LDW). Those systems are commonly known as Advanced Driver-Assistance Systems (ADAS). ADAS relies on advanced sensing technologies like radar, lidar and cameras to sense the surrounding objects and environment. These advanced sensors have the capability to detect potential crashes much earlier than traditional crash sensing system. However, those sensing technologies may have limitation on predicting the crash severity. On the other hand, crash sensing system uses accelerometers or contact based sensors to detect severity of the crash and command deployment of restraint system accordingly. Utilization of ADAS sensing system has great potential to improve the performance of crashing sensing system while maintaining the capability to predict the crash severity.

Development of such sensor fusion system will require seamlessly integrated simulation capability for both sensing systems. Currently, ADAS simulations are commonly conducted in active safety simulation software, like IPG Carmaker, TASS Pre-scan, Vires, etc. Crash sensing simulations are usually conducted in finite-element simulation software, like LS-DYNA. Research has been started recently on simulation of integrated safety system, including data transfer setup and co-simulation environment. [1] presented a methodology to study the integrated safety system, including autonomous emergency brake (AEB) and passive safety system, using co-simulation among Pre-Scan, Matlab/Simulink and Madymo. The effect of AEB on occupant injury has been studied using the co-simulation methodology. Similar study has been conducted in [2] with different environments, where CarSim was involved in the co-simulation as well. [3] proposed a data transfer method to transfer the vehicle and occupant position or motion from pre-crash simulation to in-crash simulation.

This concept can be applied to sequential co-simulations while the simulations in two stages don't have to communicate with each other. Research in integrated safety system is also progressing in the area of crash severity prediction [6], new restraint technologies [4] and co-simulated optimization methods [7]. Although LS-DYNA didn't support co-simulation with standardized interface, co-simulation with control software has been initiated through user-defined functions in some recent research on solving mechatronics problems [5]. By supporting industrial standardized co-simulation interface, it will greatly expand LS-DYNA's capability and potential applications in multi-discipline simulation area. It will also extend LS-DYNA's leading role in crash finite element simulation to integrated safety system simulation.

Function Mockup Interface (FMI) is a co-simulation standard that defines an interface to exchange dynamic models and information between different simulation solvers. It has been supported by more than 100 tools and widely used in co-simulation domain [8]. Many commonly-used active safety simulation software supports or is in development to support FMI standard.

This paper presents a newly developed feature in LS-DYNA based on FMI standard. With the new feature, LS-DYNA is capable of coupling with other FMI-compatible software and conducting multi-discipline co-simulations. A coupled crash live deployment simulation is used to demonstrate the newly developed feature.

Development of New Co-simulation Features in LS-DYNA

The FMI is a free standard that defines a container and an interface to exchange dynamic models using a combination of XML files, binaries and C code zipped into a single file [8]. The FMI can be used for both model exchange and co-simulation, which are separated in FMI 1.0 standard released in 2010 and merged in FMI2.0 in 2014 with other improvements incorporated. LS-DYNA adopts the latest FMI 2.0 standards and can generate a Functional-Mockup-Unit (FMU), which is then imported into any 3rd party software supporting the FMI co-simulation feature. During co-simulation, the two software, i.e., LS-DYNA and the 3rd party software, can either run on the same computer or two computers with the same/different operating systems, say, LS-DYNA is running on Linux HPC machine, and MATLAB is running on a Windows PC, assuming that both machines are in the same private network. The data exchange for the co-simulation is built on the TCP-IP protocol with LS-DYNA as the server and the other software as the client, and thus the IP address of the DYNA machine should be specified in the keyword for remote connection.

To enable the co-simulation feature, a necessary LS-DYNA plugin called "FMU Manager" [9] should be downloaded and configured with a compatible C compiler. The R12 and also the latest LS-DYNA developer version of SMP/MPP Single/Double precision have enabled such co-simulation capability and more features are continuously added. Multiple examples can be found in [9] with detailed instructions and manuals. During the co-simulation, LS-DYNA runs with its own explicit time step $\Delta t1$, and the data exchange occurs every $\Delta t2$, which is the co-simulation time step, usually specified in the other software and set to be greater than $\Delta t1$, i.e., $\Delta t2>\Delta t1$. Reducing $\Delta t2$ indicates that the data exchange is more frequent, which could be unnecessary, especially when $\Delta t1$ could be very small during the vehicle crash simulation.

A dual-step procedure is followed to implement the co-simulation with LS-DYNA.

(1) FMU generation. Users properly define the interface variables and other FMU settings in a keyword file and run LS-DYNA to generate an FMU. The interface variables are exported/imported variables to be exchanged during co-simulation, and mostly, the exported variables include the acceleration & velocity & displacement & position of nodes or rigid bodies in the global or local coordinates, and the imported variables include force & moment & segment pressure of nodes or rigid bodies, etc. Other general variables can be imported through defined curves, and LS-DYNA will dynamically modify the curve value at every co-simulation time step.

(2) FMU co-simulation. Users import the FMU into another software and set up the co-simulation time step. Users run LS-DYNA first, which will be blocking and waiting for the connection from another software. Users subsequently run the other software to connect to LS-DYNA, and the co-simulation will automatically begin.

The co-simulation will end when either LS-DYNA or another software terminates, whichever is earlier. The overall working flow is illustrated in Figure 1 below.

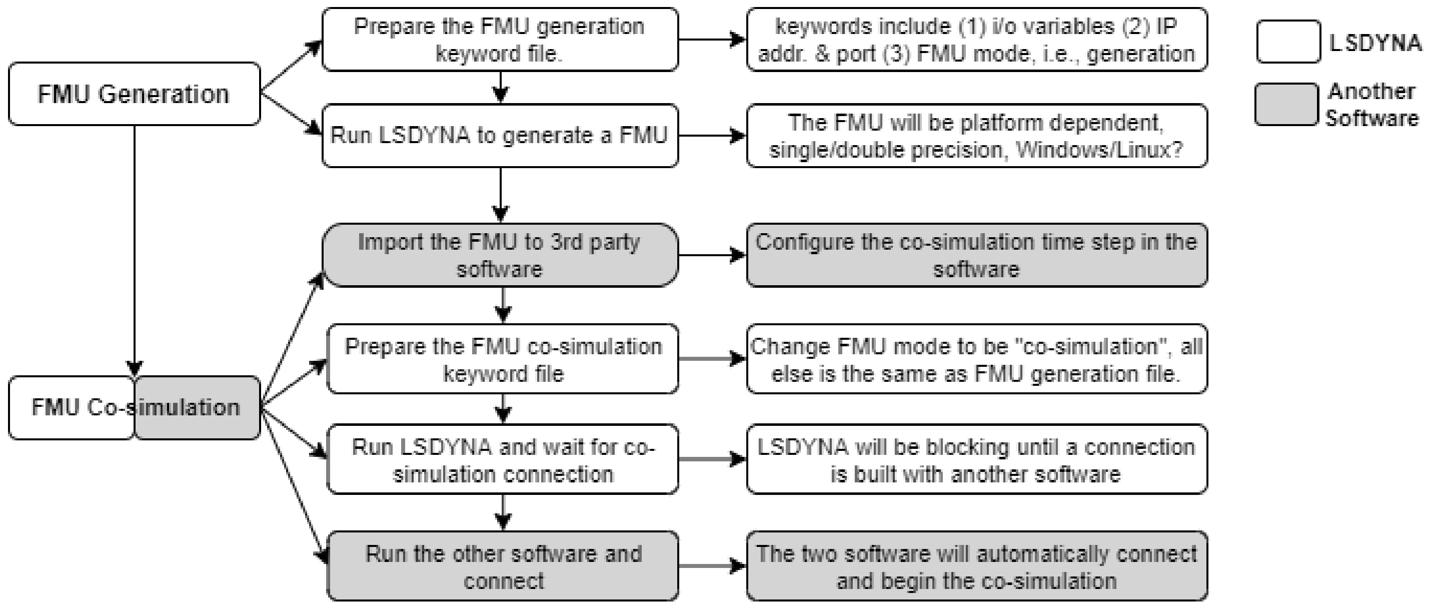

FMU Generation
Prepare the FMU generation keyword file.
keywords include (1) i/o variables (2) IP addr. & port (3) FMU mode, i.e., generation
Run LSDYNA to generate a FMU
The FMU will be platform dependent, single/double precision, Windows/Linux?
Import the FMU to 3rd party software
Configure the co-simulation time step in the software
FMU Co-simulation
Prepare the FMU co-simulation keyword file
Change FMU mode to be "co-simulation", all else is the same as FMU generation file.
Run LSDYNA and wait for co-simulation connection
LSDYNA will be blocking until a connection is built with another software
Run the other software and connect
The two software will automatically connect and begin the co-simulation
LSDYNA
Another Software

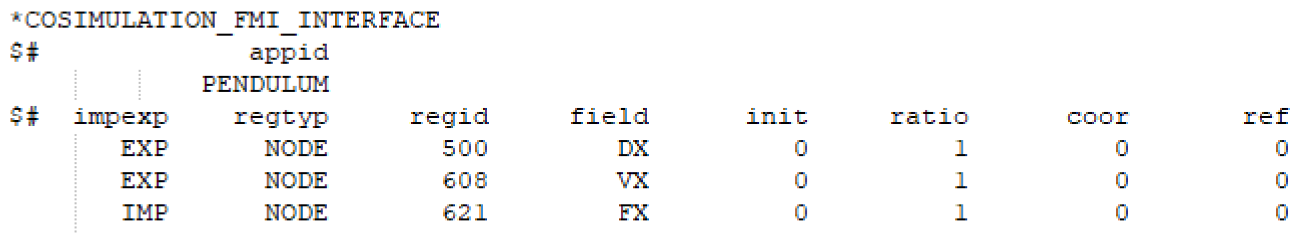
Figure 1. Flow Chart of LS-DYNA Co-simulation

Two co-simulation keywords are involved, *COSIMULATION_FMI_CONTROL and *COSIMULATION_FMI_INTERFACE. The control keyword, as shown in Figure 2 below, specifies the FMU name, i.e., "PENDULUM"; the FMU mode, i.e., "C" for co-simulation or "G" for "Generation"; the DYNA mode, i.e., "S" when DYNA is the slave and the other software is the master which controls the co-simulation time step; <home>, i.e., the directory of the FMU manager; <fmudir>, i.e., the directory of FMU to be generated or co-simulated; <tcp>, i.e., the IP address and available port of the DYNA machine for co-simulation, not the other software machine; <socket>, i.e., some general settings related to TCP socket connection.

```
*COSIMULATION_FMI_CONTROL
$#             appid   cosim/generate m/s
            PENDULUM         C         S
$#            settings
<home> C:\DYNA_bin\FMU_Manager_v4_deliver
<fmudir> C:\DYNA_bin\dyna_key
<tcp> ip=127.0.0.1,port=39400
<socket> nconnect=20,tdelay=1001,recvtimeout=30001
```

Figure 2. Keyword Demo of Co-simulation Control

The interface keyword, as shown in Figure 3 below, define the interface variables to be exchanged during the co-simulation. "EXP" indicates that the variable will be exported by DYNA to another software, and "IMP" is the opposite. Users also need to specify the region type "regtyp" to be node, node set, segment set, curve, etc. The default initial condition "init" is 0 unless specified otherwise, "ratio" is used to scale the variables to be sent or received, "coor" specifies the coordinate system, "ref" is the reference option for the fixed/changing local coordinate.



```
*COSIMULATION_FMI_INTERFACE
$#          appid
         PENDULUM
$#  impexp    regtyp     regid     field      init     ratio      coor       ref
       EXP      NODE       500        DX         0         1         0         0
       EXP      NODE       608        VX         0         1         0         0
       IMP      NODE       621        FX         0         1         0         0
```

Figure 3. Keyword Demo of Co-simulation Interface

Currently, only ONE software is allowed to co-simulate with LS-DYNA, i.e., a single FMU is supported, and new features are continuously added to enable multi-FMUs co-simulation, i.e., multiple software can exchange data with LS-DYNA simultaneously.

Application of New Features

Traditional method of crash sensing simulation is to obtain sensing signature signals from finite element analysis, then the deployment time is generated by feeding the signals through sensing calibration. Separate simulation can be conducted using generated firing time to determine occupant performance. This method involves an offline step to get the firing time through calibration. It also involves repeated simulation to incorporate the true deployment time from sensing algorithm. This dual-step method makes it difficult to investigate multiple complex scenarios in parallel. For example, if people would like to conduct a Design of Experiments (DOE) to investigate occupant injuries under multiple crash scenarios, two sets of DOE have to be run, one is used to generate sensing signals to determine the deployment time for each case, the other is to obtain the occupant performance using the deployment time generated from the first set of DOE. It doubles the computational resource to solve the single problem.

Live deployment simulation has sensing calibration embedded in the crash model. The sensing algorithm runs at the same time as the finite element simulation. Depending on the signals at sensor and Safety Diagnostic Module (SDM), the algorithm will command deployment of restraint system models during the simulation. Live deployment simulation eliminates the offline setup to get deployment time and is capable of running multiple scenarios, like DOE, at the same time.

In this section, the newly developed FMU feature in LS-DYNA is used to co-simulate with Matlab/Simulink to build a live deployment crash model. In this co-simulated model, an FMU wrapper was generated by LS-DYNA and used as co-simulation interface with Simulink. Matlab/Simulink served as the master and LS-DYNA was the slave. The co-simulation time step $\Delta t2$ was set in Matlab to be a constant 1 ms, and the explicit time step $\Delta t1$ was flexibly controlled in LS-DYNA, and could be two~three magnitudes smaller.

In the demonstration model, a simple vehicle crash model was built. The model consists of three subsystems, as indicated in Figure 4, rigid pole barrier, vehicle structure and airbag. The pole barrier is aligned at the center of the vehicle. The initial velocity of the vehicle was set at 56kph to simulate a non-regulated, but field related event. The vehicle structure consists of the vehicle body and several frontal structural parts. Several simplified parts, including bumper and hood, represent the vehicle front structure and are expected to deform during frontal impact. Acceleration of two nodes at vehicle front, which represent two front accelerometers, and acceleration of one node at vehicle center tunnel location, which represents one accelerometer at SDM, are requested in co-simulation FMI interface keyword card. Those nodal accelerations will be extracted by FMI during simulation as inputs to the sensing algorithm running in Matlab/Simulink.

An airbag model is constrained to the vehicle structure through nodal rigid bodies, which is indicated as blue dash lines in Figure 4. Deployment of the airbag is controlled by *SENSOR_CONTROL card. The initial status of the airbag is OFF. The deployment is commanded by a curve through *SENSOR_SWITCH and *SENSOR_DEFINE. When curve's value is greater than 0.5, airbag deployment is triggered. The airbag command curve will be fed through FMI from the sensing algorithm when the crash model is co-simulated with Matlab/Simulink.

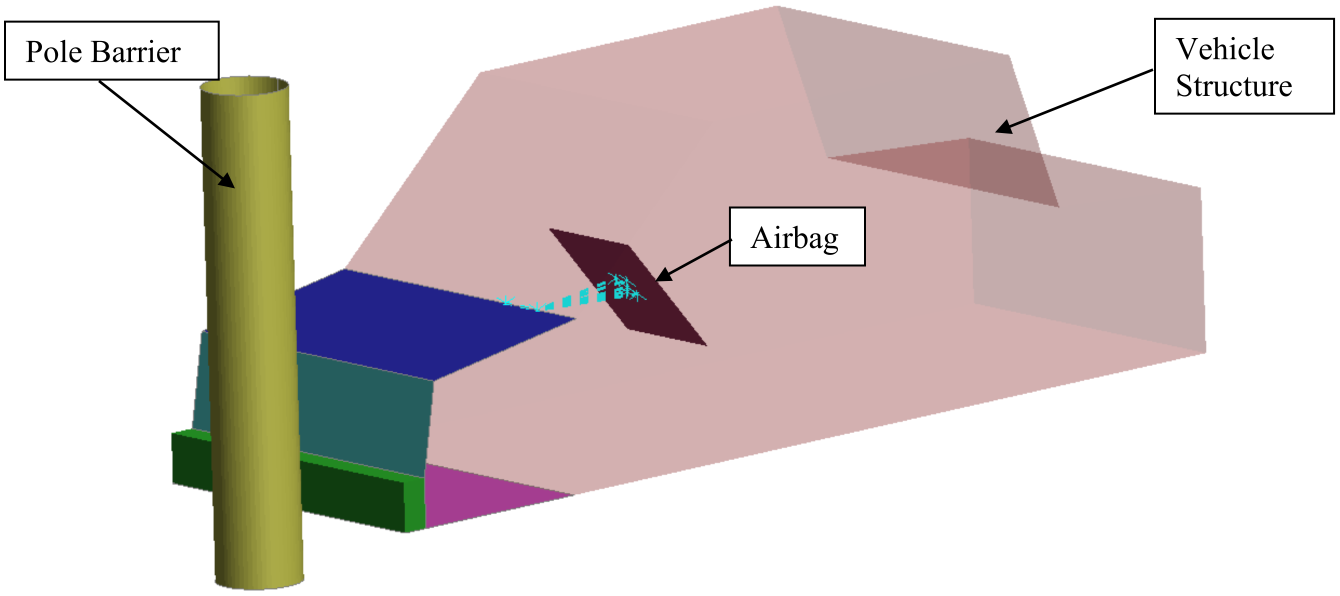


Figure 4. Simple Vehicle FE Crash Model

In order to co-simulate with Matlab/Simulink, an FMU has to be generated by LS-DYNA and the FMU will be imported as a block diagram in Matlab/Simulink. *COSIMULATION_FMI_INTERFACE and *COSIMULATION_FMI_CONTROL keywords are used for FMU generation by setting OPT in *COSIMULATION_FMI_CONTROL to "G" (generate). Three export variables and one import variable are defined in *COSIMULATION_FMI_INTERFACE. The export variables include accelerations in vehicle longitudinal direction from the three nodes, two at the front accelerometers and one at SDM. The import variable includes a curve which will command the airbag deployment. After running the model, a .fmu file is generated in the result folder. This file will be imported in FMU block in Simulink.

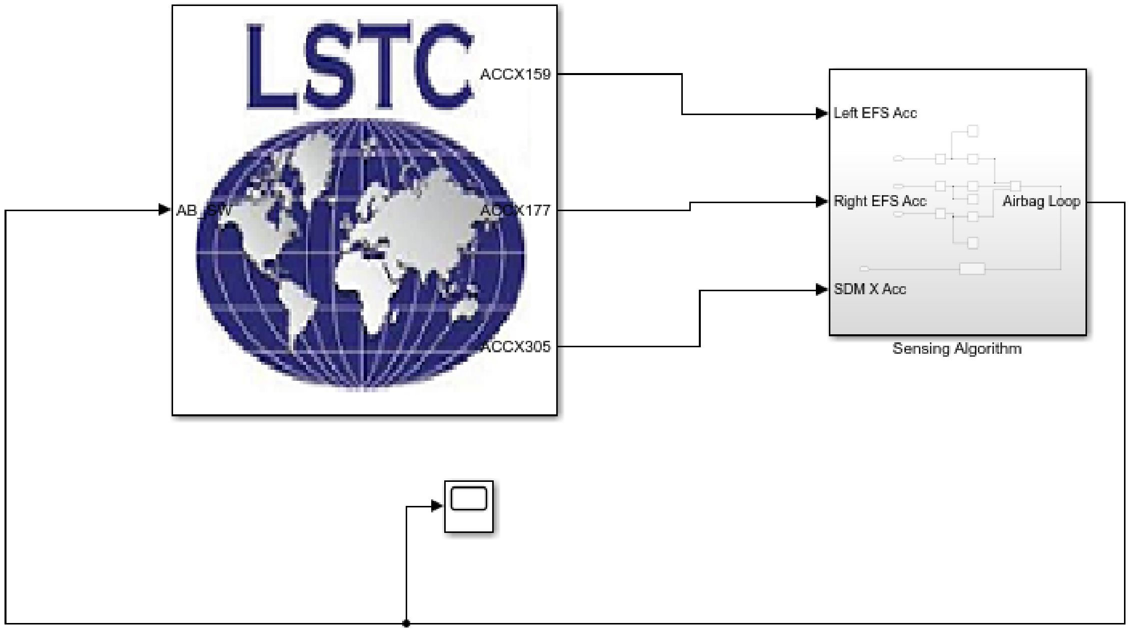
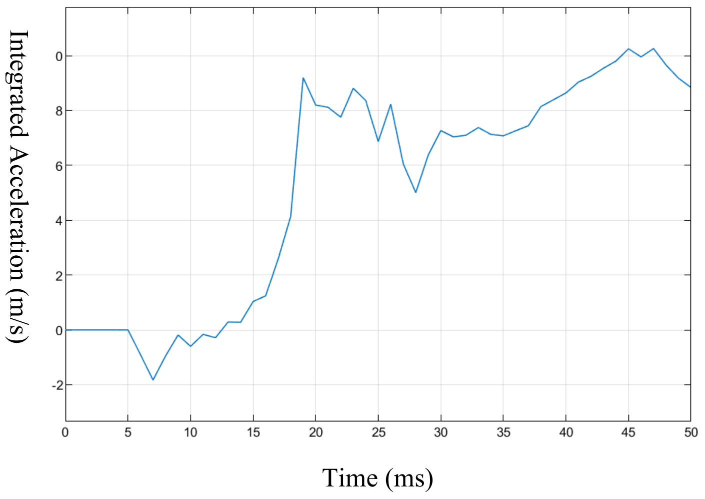


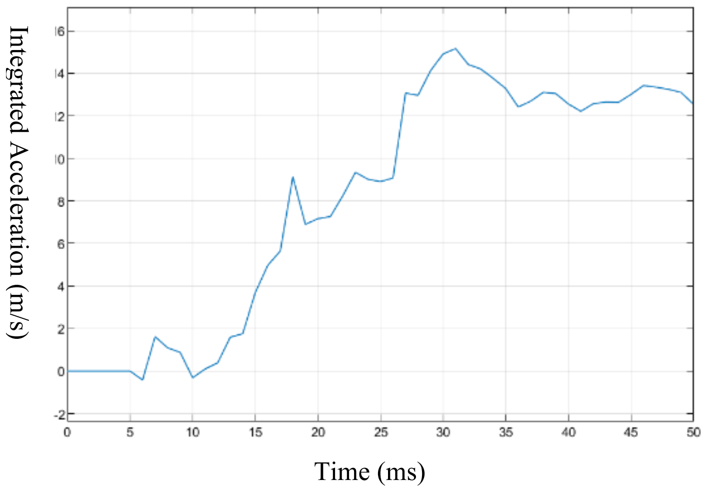
Figure 5. Simulink Diagram of Co-simulation

In parallel, co-simulation diagram with sensing algorithm was built in Simulink, as shown in Figure 5. The Simulink model includes two major blocks, FMU block and sensing algorithm block. The FMU block contains the .fmu file generated by LS-DYNA. Three export variables and one import variable appear as three output ports and one input port. The sensing algorithm block contains logic to process acceleration data from FE simulation and generate deployment command. Several scopes were added to the Simulink model to monitor various signals.

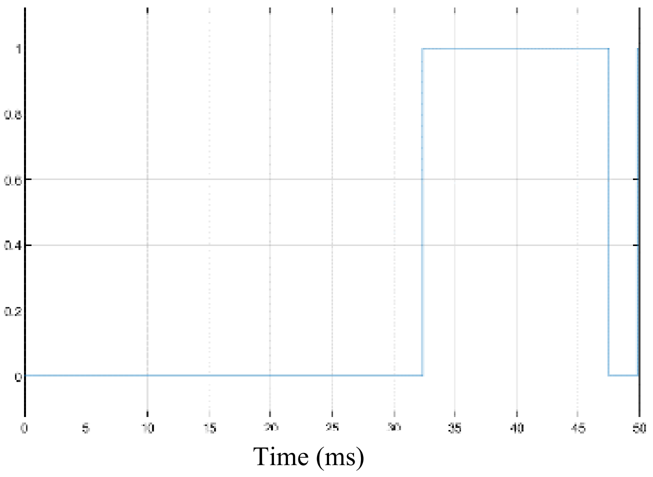
When both Simulink and FE crash models are ready for co-simulation, OPT in *COSIMULATION_FMI_CONTROL was changed to "C" (co-simulate) for switching vehicle FE model to co-simulation mode. The co-simulation was initiated by running the LS-DYNA model first, followed by the Simulink. Sensing algorithm and vehicle crash model ran in parallel by exchanging data through FMI.



(a) Left Front Accelerometer



(b) Right Front Accelerometer



(c) Airbag Deployment Loop

Figure 6. Front Accelerometer Signals and Airbag Deployment Command

Figure 6 shows several signals picked from scopes in Simulink model. Figure 6 (a) and (b) are integrated acceleration from two front sensors. These two signals were calculated by sensing algorithm based on original acceleration signals. The integrated accelerations are shown instead of because it is a good indicator of impact severity and less noisy compared to the original acceleration signals. Together with the signal from SDM, sensing algorithm made deployment decision during the simulation based on all three signals from the FE vehicle model. Figure 6 (c) is airbag deployment command generated from the sensing algorithm module. The deployment decision was made at 32ms. The airbag started inflation at 32ms and Figure 7 shows the vehicle structure deformation and airbag deployment shape at 45ms.

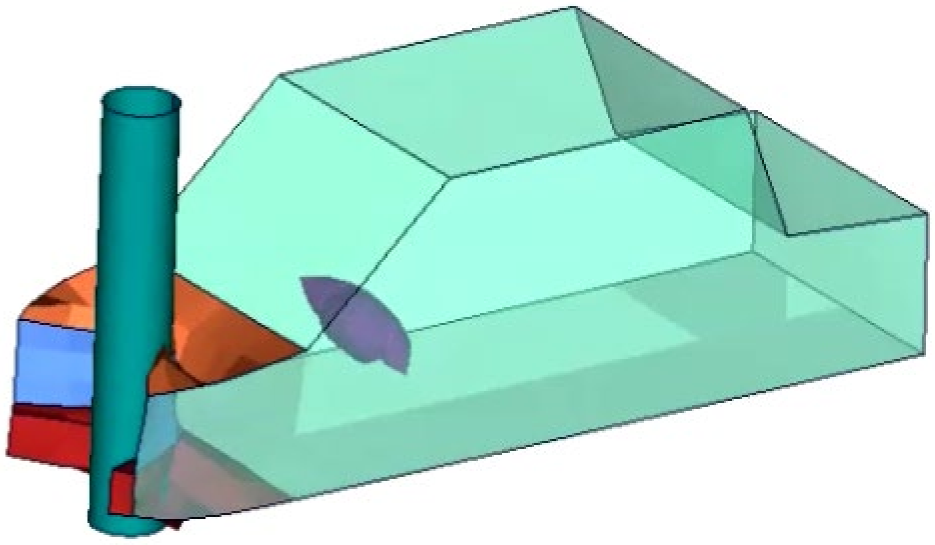


Figure 7. Vehicle Structure and Airbag at 45ms

Without changing the Simulink model, different initial speed, mode or barrier can be modified in FE model to investigate crash sensing performance or occupant performance under different circumstances. No offline sensing simulation or repeated occupant simulation is needed.

Future Applications

As the next step, co-simulation of live deployment will be further extended to couple with ADAS sensing simulation to study the benefit of integrated sensing system. In future study, Matlab/Simulink will serve as a master to communicate with both LS-DYNA and IPG Carmaker. ADAS sensing system together with vehicle dynamics will be simulated in IPG Carmaker. Vehicle motion before crash time 0 will be passed to LS-DYNA through Matlab/Simulink. Carmaker simulation will be deactivated at time 0 while Matlab/Simulink and LS-DYNA continue the FE simulation into the crash protection stage.

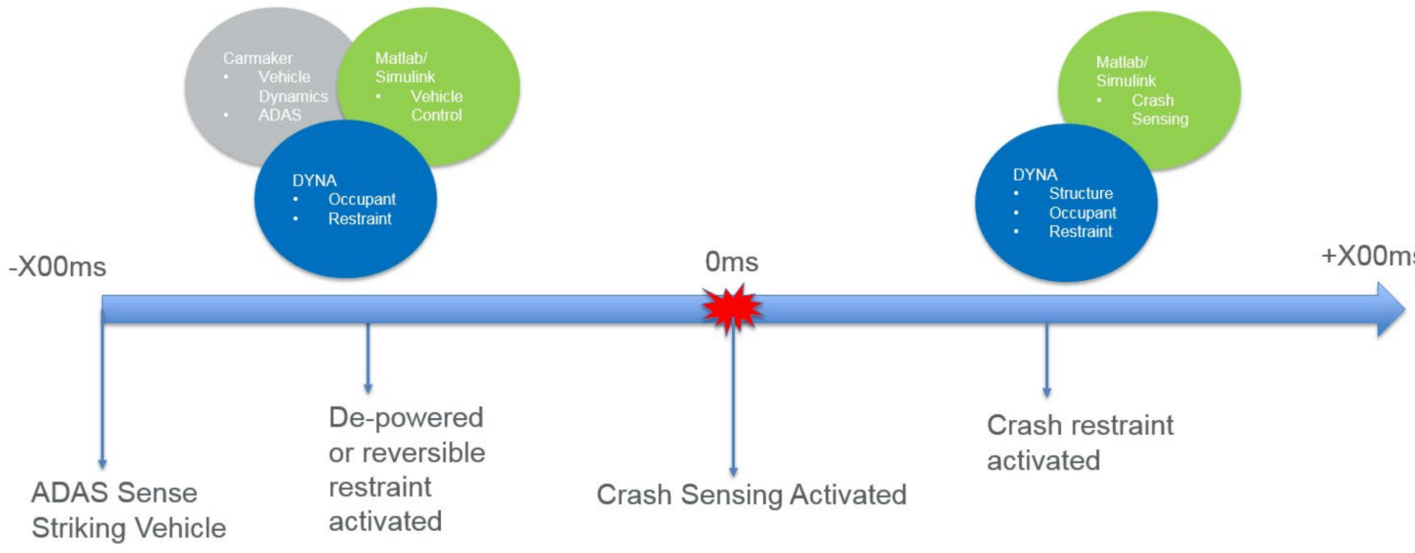


Figure 8. Co-simulation Timeline of Integrated Sensing System

Figure 8 shows a timeline of the proposed co-simulation scheme, which incorporates three software and simulates the whole event from active safety to passive safety. The simulation would start when ADAS senses striking vehicle and determines high risk of impact. The vehicle dynamics, control and ADAS are simulated in CarMaker together with Simulink. The CarMaker features are integrated with Simulink through S-functions and API functions. On the other hand, Simulink is also coupled with LS-DYNA so that it passes vehicle motion to LS-DYNA to simulate occupant motion before crash happens. The control algorithm in Simulink also controls the deployment of any reversible or de-powered restraint features in LS-DYNA. When vehicle impact happens (time 0), CarMaker will terminate the simulation while Simulink and LS-DYNA continue the co-simulation into crash stage. After time 0, LS-DYNA is in charge of both vehicle structural and occupant simulations while Simulink continues the sensing control algorithm. In this way, Simulink takes the full ownership of sensing control algorithm from ADAS to crash sensing, which makes development of integrated sensing system possible.

Conclusions

In order to utilize ADAS information to improve crash sensing performance and create the sensor fusion system, multidiscipline simulation between ADAS and crash sensing system has to be developed. Co-simulation between active and passive safety solvers not only eliminates barriers in the continuous simulation of the whole event, from ADAS sensing to crash sensing, but also significantly improves efficiency by reducing repeated simulations and automatic data transfer between solvers.
The newly developed FMI feature enables LS-DYNA to co-simulate with other active safety software under industry-standard interface. A coupled live deployment simulation demonstrated the co-simulation capability of LS-DYNA, which served as the first step to simulate and develop the sensor fusion system. The benefit of co-simulation can also be extended to other aspects of integrated safety system, including investigating injury consequences of various ADAS maneuvers and studying the benefits of de-powered or reversible restraint systems. Co-simulation will become increasingly popular when development of integrated safety system accelerates.

References

[1] Martin Tijssens, Freerk Bosma, and Kajetan Kietlinski, *A Methodology And Tool Chain To Develop Integrated Safety Systems*, 24th International Technical Conference on the Enhanced Safety of Vehicles (ESV), ESV Paper 15-0329
[2] Lee Jeong Keun, Chu Heon Jeong, and Hurh Kyung Rok, *A development of the CAE process for the AEB-occupant integrated safety system*, 25th International Technical Conference on the Enhanced Safety of Vehicles (ESV), ESV Paper 17-0316
[3] Abdulkadir Öztürk, Christian Mayer, Hemanth Kumar G, Pronoy Ghosh, Atul Mishra, Ravi Kiran Chitteti and Dirk Fressmann, *A Step Towards Integrated Safety Simulation Through Pre-Crash To In-Crash Data Transfer*, 26th International Technical Conference on the Enhanced Safety of Vehicles (ESV), ESV Paper 19-0257
[4] David Breed, Nina Yurchenko, Pavlo Vynogradskyy, Konstantin Kuzmenko, Shawe Zhang and Bobby Li, *The Analysis And Experimental Development Of Aspirated Airbags For Conventional And Autonomous Vehicles*, 26th International Technical Conference on the Enhanced Safety of Vehicles (ESV), ESV Paper 19-0025
[5] R. Cresnik, A. Rieser and H. Schluder, *Dynamic Simulation of Mechatronic Systems*, 7th European LS-DYNA Conference,
[6] Daniel Wallner, Arno Eichberger, and Wolfgang Hirschberg, *A Novel Control Algorithm For Integration Of Active And Passive Vehicle Safety Systems In Frontal Collisions*, Systemics, Cybernetics And Informatics, Vol. 8, Number 5, 2010
[7] Ke Dong, *H∞ Robust Control of a Seat Belt Load-limiting Device*, in Proc. 50th IEEE Conference on Decision and Control, pp. 6840-6845, 2011
[8] FMI Version 2.0, *FMI for Model Exchange and Co-Simulation*, http://www.fmi-standrd.org
[9] http://ftp.lstc.com/anonymous/outgoing/xiaomeng/deliver/FMU_Manager_release_note.txt