APPLICATION NOTE

Radar Development Using Model-Based Engineering

Introduction to MBE

The Radar lifecycle comprises development, deployment, maintenance, and enhancement for future requirements; each must be considered upfront for developing an effective and efficient system and to achieve low cost of ownership. Highly complex Radars have been developed and deployed in complex environments with great performance and superb results since World War II. Since then; however, signal complexity has gone up by several orders of magnitude and new technological innovations have been introduced to realize ever increasing capabilities. These two aspects alone make the Radar development lifecycle very complex. In fact, without modern tools and processes it would be almost impossible to meet today's delivery times and budget requirements. Because of this, several big organizations in the Unites States—Raytheon, Northrop Grumman, Lockheed Martin, Boeing, and others—came together to evolve new methods for Radar development. Model Based Engineering (MBE) is one such technique.

This application note briefly reviews the MBE technique. It also examines in detail how a set of simulation and modeling tools can work together to support MBE.

This application note describes how SystemVue is well suited for use in model-based engineering. It is targeted at system engineers, directors of engineering and equivalent job functions who are involved in the development of Radar.

KEYSIGHT TECHNOLOGIES

MBE Basics

As per the final report[1] of the MBE subcommittee of National Defense Industrial Association (NDIA), MBE is defined as:

"An approach to engineering that uses models as an integral part of the technical baseline that includes the requirements, analysis, design, implementation, and verification of a capability, and/or a product throughout the acquisition cycle."

During development, therefore, all aspects of Radar acquisition have to be considered. MBE was originally derived because of several gaps in the practice and the process of developing complex systems. While the negatives may have necessitated MBE, the advantages have accelerated the focus and attention on it. MBE's main advantages include the ability to:

- Quickly evaluate what's not possible by exploring the entire project scope
- Derive all downstream activities
- Come up with quick proposals and Rough Order of Magnitude (ROM) estimates
- Rapidly evaluate the system performance for changing requirements
- Use the top-level model as the beacon for every subsystem development

With a main system model, it's possible to derive all subsystems' requirements and partition the design margins more meaningfully; a process generally known as top-down design. In the case of MBE, the main system model not only guides the subsystem design, but also helps define the system test, manufacturability and cost of production, among other things. This also leads to the ability to generate a quick proposal that is totally devoid of impossibilities and has a decent probability of success, as well as a ROM estimate of cost.

What are the impossibilities in the context of Radar? Certainly Radar cannot have infinite range and a pulse Radar requires a minimum range dictated by the finite speed-of-light. A pulse Doppler Radar does not receive while transmitting. Hence, any signal that returns during transmission is not detected. This limits the detection range and this limitation comes from the physics of the system. Similarly, we can identify other limitations associated with such things as resolution and Doppler ambiguity. With a good main system model upfront, one can quickly examine these limitations early in the design phase.

Clearly the first, and perhaps the only, critical step in MBE is the development of the model! When developing the model we need to consider two aspects:

Width and versatility. How wide should the model be? The Radar model has to predict the performance under a variety of conditions, such as frequency range, complex environments, waveforms, wide variety of threats, and various interferences. It is generally desirable to build a model as wide and versatile as possible so that one can explore all of the above conditions.

Depth. How deep should the model be? This is an engineering and business decision. There are tools that allow both the fidelity and accuracy of the building block models to be increased infinitely. Doing so; however, would increase the simulation time enormously and could be prohibitive and counterproductive from a business point of view. Hence, a good main system model is one that has enough accuracy, but that also simulates quickly. A good main system model also has fidelity that can be progressively increased on demand.

Let's now go to the question of: What is being engineered? In the context of Radar, this can be several systems. We can view Radar as a system of systems as shown in Figure 1.

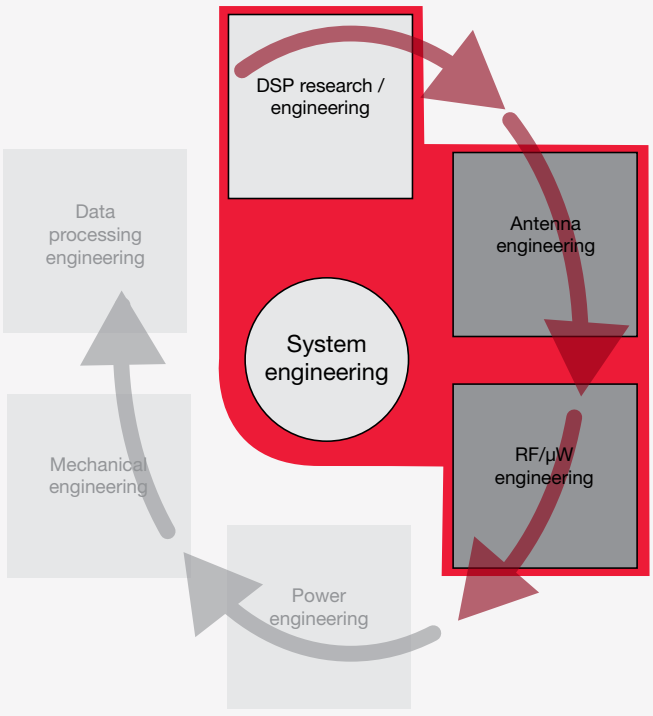

Figure 1. Radar can be viewed as a system of systems.

For the rest of the discussion we will consider only the electronic system, which includes the DSP and RF/MW subsystems in the transmitter and receiver, and the antenna. However, the methodology applies to all other systems each requiring a set of special tools.

The MBE approach can be efficiently described by a V-diagram (Figure 2). The V-diagram is adopted by systems engineering and is available from various papers and text books. The specific one shown in Figure 2 is from the presentation at the INCOSE workshop[2].

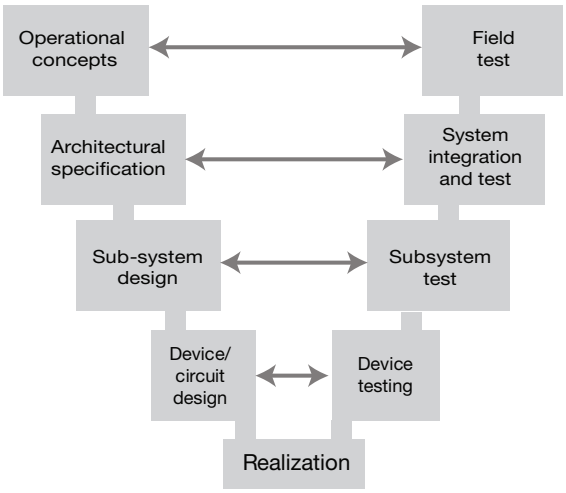

Figure 2. A V-diagram of an MBE approach.

Figure 2 shows both a top-down and bottom-up flow. The definition and decomposition flow dictates the specification of the subsystems and their margins. It is common that the subsystems themselves are pretty complex and hence, they can also be described with individual V-diagrams as shown in Figure 3.

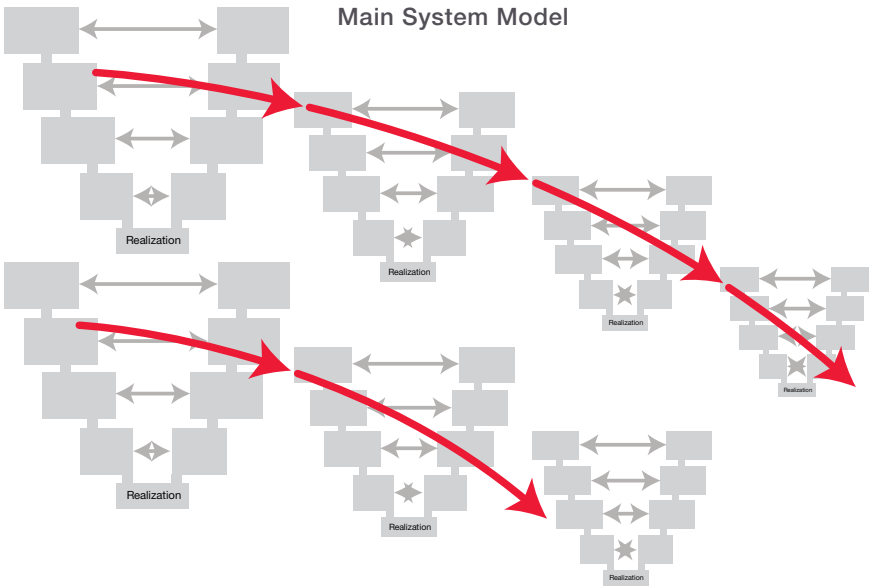

Figure 3. Individual V-diagrams of the individual subsystems in an electronic system.

Figure 3 implies that the subsystems themselves can be complex and as such need to be treated as individual systems that should also follow the MBE approach. For example, the phased-array subsystem in an antenna is very complex and needs to be treated with MBE. Similarly, the Transmit-Receive (TR) module, designed as an integrated circuit, may have to be treated like a system and all the system concepts should be applied.

Each of these subsystems must be designed using an appropriate tool. Let's call the V-diagram one level below a child-V, and the one above child-V, a mother-V. An important requirement of MBE is that a mother-V be able to call, command and control child-V. It should be able to simulate child-V, keep it in standby mode, examine the output from child-V, and change the inputs to child-V to continue with the simulation on demand.

The tools used to design each of these systems—which by itself is a subsystem for the higher level system—are very likely different tools. For example, a system-level simulation is done using popular tools like SystemVue, MATLAB, and System Tool Kit (STK), while the subsystem (e.g., a TR module) is designed using a circuit simulator like Advanced Design System (ADS). Depending on the situation, some of the subsystems could be actual hardware connected to measuring instruments. It quickly becomes clear that for maximum effectiveness, however, all of these tools should be interoperable. In other words, a higher level system simulating tool should be able to call a subsystem simulating tool and put it in a command and control mode.

Once the above requirements are satisfied, one can build the main system model to simulate the end-to-end performance of the system. Let's see how such a main system model can be created for a Radar using a set of tools.

On close observation of the V-diagram in Figure 2, notice that the left side of the V pertains to top-down design, while the right pertains to integration and verification. Verification is usually done with actual hardware, meaning that it is not performed until the actual hardware is available. With modern tools like SystemVue and ADS, a top-level model can be built that allows verification to be done in simulation. To highlight the difficulty, imagine having to co-simulate RF and DSP circuits. Very few tools in the industry support this kind of co-simulation. With an improved co-simulation capability though, both sides of the V-diagram (top-down design and bottom-up verification) can be done entirely in simulation. Final verification is then done with actual hardware. Performing the entire V in simulation eliminates major risk factors early in the development cycle.

Building a main system model using SystemVue, MATLAB, Google Maps, and STK

To better understand how to build a main system model, consider the example of a mono-static pulse Doppler Radar, including the Radar platform and target. We will show that this model can be easily expanded to bi-static and multi-static Radar systems, as well as multiple targets. We will also show how clutter, jammer, interferer, and noise can be added to the target return. The Radar and the environment being built will cater to a typical operational scenario depicted in Figure 4.

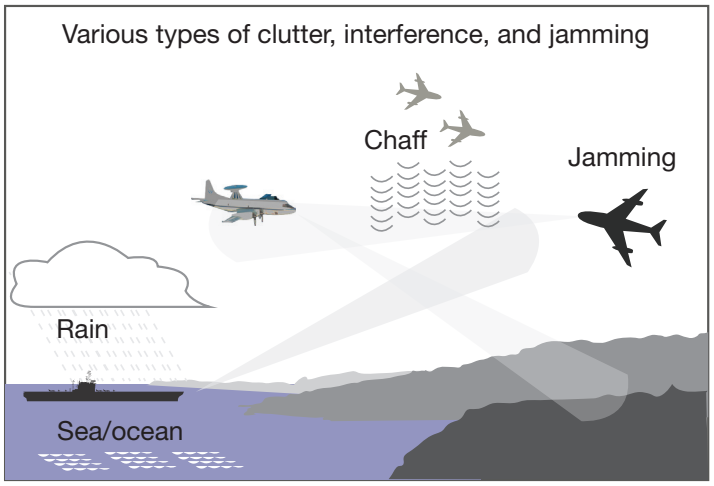

Figure 4. This image depicts the typical operation of a mono-static pulse Doppler Radar.

From Figure 4, it's apparent that the Radar is on a moving platform, a ship, and that there are three targets, one jammer and the clutter caused by sea, rain and chaff. The communication signals from the city environment may act as interferers. Also, the Radar could be treated as bi-static with the Radar receiver on the reconnaissance aircraft. **It is important to note that pictures like that shown in Figure 4 have to be made visible throughout the organization** engaged in MBE. Everyone then will be able to check the performance of their individual subsystems through the main system model.

We begin by building a model in SystemVue that can potentially capture scenarios like that in Figure 4. SystemVue's framework infrastructure has three layers to express a system's operation:

Trajectory Layer: In the trajectory layer we model the positions and motion of the Radar transmitter and receiver, as well as a single target or multiple targets. The transmitter and receiver antennae are specified independently so that they can be co-located as in mono-static Radar or separated as in bi-static Radar. Multiple receivers can also be specified to model multi-static Radar. The Radar transmitter or receiver antenna model has the same parameters; however, multiple instances have to be placed to specify them independently. The resulting model is shown in Figure 5.

There are two modes in which the position and motion parameters, velocity, acceleration and jerk can be specified: RunTimeGeneration and UserDefined. In the RunTimeGeneration mode, the initial position is specified using the Longitude, Latitude and Altitude (LLA) coordinate format. In the UserDefined mode, the positions are specified in the Earth Centered Inertial Frame (ECI) format, which is essentially a rectangular coordinate system with Earth's center at the origin. The motion of the Radar antenna can be specified in terms of velocity, acceleration and jerk. In addition to these specifications, the orientation of the antenna can be specified using three input pins—yaw, pitch and roll—as shown in Figure 6.

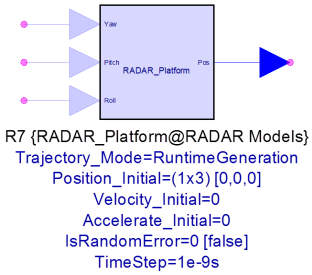

Figure 5. Shown here is the model derived in SystemVue's trajectory layer.

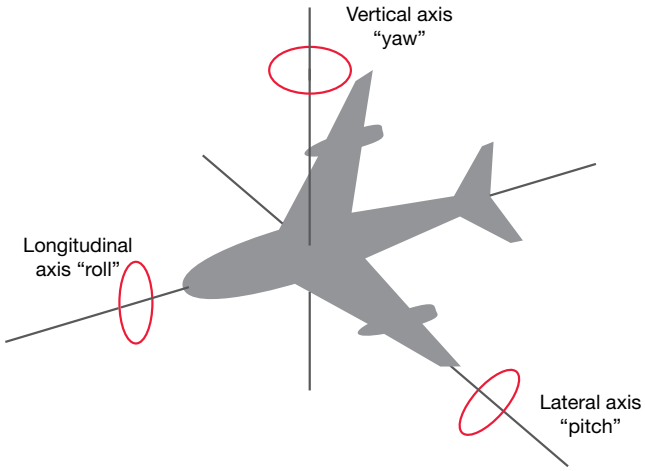

Figure 6. Here, the orientation of the antenna is specified using yaw, pitch and roll. Photo credit: http://en.wikipedia.org/wiki/Flight_dynamics_(fixed-wing_aircraft)

In this case, the antenna is fixed to the ship. By specifying the orientation of the ship using yaw, pitch and roll we can specify the orientation of the antenna.

The specification for the Radar receiving antenna is similar to that of the transmitting antenna. In the case of mono-static Radar they are the same. For bi-static and multi static Radar, the position, orientation and motion will be different.

Next, consider the target parameters. Many of the parameters for a target like the position, orientation and motion are the same as those for the transmitting and receiving antenna. In addition, the Radar Cross Section (RCS) has to be specified for a target. If the target is an extended object, then various parts of the object will reflect differently; either because of their reflective properties or the angle they make with the incoming beam. We call these various parts of the object, scatterers. The location of each scatterer can be specified with respect to the centroid of the object. The RCS value of each of the scatterers can also be specified individually. The model used for this purpose is shown in Figure 7.

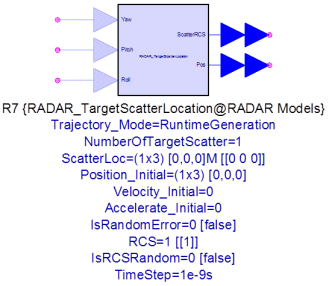

Figure 7. Shown here is a model of the Radar target parameters.

As can be seen from Figure 7, the Position_initial is the position of the centroid of the object in [L L A] format for RuntimeGeneration mode. The positions of scatterers are set with the parameter ScatterLoc. The format of ScatterLoc is [x0, y0, z0, x1, y1, z1,...], the unit is meter, the reference point is the centroid, and the frame is the body frame. If the ScatterLoc is set to [0 0 0], that means the scatter location is at the centroid. The parameter RCS then specifies the RCS value of each scatterer. The orientation of the target is specified with yaw, pitch and roll input pins. If the target is not static, the Velocity_Initial and Accelerate_Initial parameters can be used to set the initial velocity (m/s) and accelerate (m/s2) in the body frame. The random value of jerk is provided when IsRondomError = true. A completed trajectory layer for a mono-static Radar and a single target is shown in Figure 8.

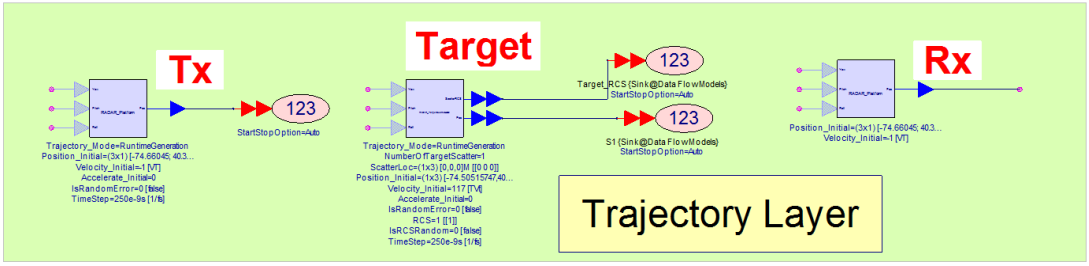

Figure 8. Shown here is a completed trajectory layer for a mono-static Radar with a single target.

Antenna Layer: In the trajectory layer the location of the target is specified with respect to the center of the earth. It is convenient to convert the target coordinates so that they are located in the antenna beams of the transmitter and receiver, that is, as azimuth and elevation angles made with the bore site of the antenna. This information is then used in the signal layer so that the phased-array antenna can steer the beam in that direction to detect the target. In this layer, there are no user-defined parameters except that multiple instances have to be placed for the Radar receiver. A typical antenna layer is shown in Figure 9.

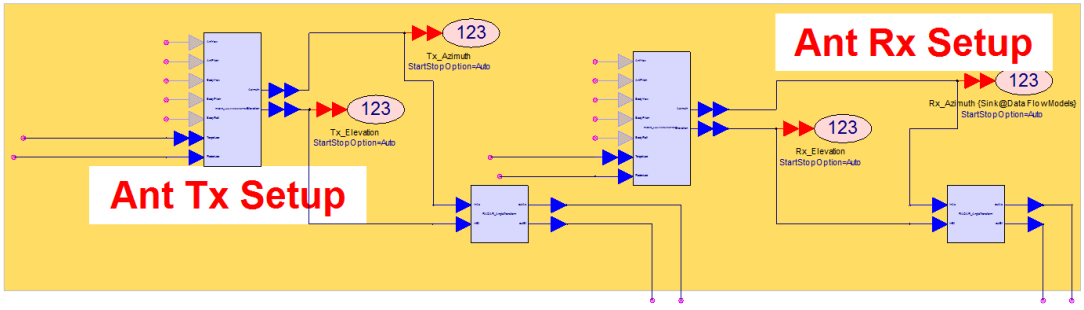

Figure 9. A typical antenna layer is shown here.

The pins protruding at the top of Figure 9 bring the information about the positions of the transmitter and receiver antennae and the target. The pins going out at the bottom of the figure supply the location of the target in both the transmitter and receiver antenna frames to the signal layer.

Signal Layer: The signal layer describes waveform generation, beam formation, target return, clutter, jamming, interferer and noise addition, and receiver signal processing.

Waveform Generation: The waveform used in a Radar can potentially dictate the outcome and hence, is a top-level system parameter. By changing the waveforms, one can resolve ambiguities, increase the unambiguous range and velocity, increase the resolution, counter certain types of deception jamming, and increase the probability of detection. Since the Radar we chose to model is pulse Doppler the appropriate waveform is a LFM signal.

The model for waveform generation is a flexible source called Radar_SignalX (Figure 10). Most modern Radars are migrating toward multi-functional (MFR) operation. In a MFR, it is required that the waveform type, carrier frequency, pulse width, PRI, and CPI among others, be changed on the fly. The input pins serve this purpose effectively. The data read from a waveform library can be fed to these pins dynamically. A pulse Doppler Radar also employs pulse compression, which is very sensitive to jitter in the waveform. A jitter parameter on this model serves this purpose. As mentioned in the introduction, the model is attempting to be very versatile but its depth is limited by not going down to the details of the hardware implementation.

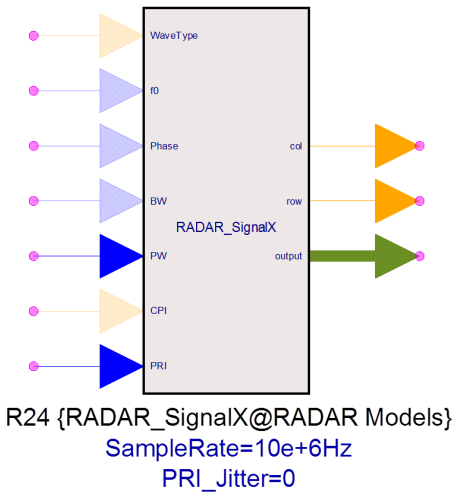

Figure 10. This waveform generation model is called Radar_SignalX

Beam formation: To keep the model fidelity simple we are not going to model the transmitter elaborately. We are modelling a 16-element linear array. The model supports any size array, one or two dimensional. The 16 baseband channels are generated by the Radar_TX_DBS_2D and Radar_MulltiCh_Tx parts. These 16 inputs flow into the Radar_PhasedArrayTx part, which combines the information coming from the antenna layer to generate a beam in the direction of the target. In this mode, it will be tracking the target. This part also can be operated in scanning mode by sweeping the elevation and azimuth angles externally. The beam forming is done according to:

$$F(\theta) = \cos^{\frac{EF}{2}}\theta \cdot \sum_{m=1}^{M} a_m e^{j\left(\frac{2\pi}{\lambda}x_m \sin\theta - \frac{2\pi}{\lambda}x_m \sin\theta_o\right)}$$

$$EP = \cos^{\frac{EF}{2}}\theta$$

$$F(\theta) = EP \cdot AF = \cos^{\frac{EF}{2}}\theta \cdot \sum_{m=1}^{M} A_m e^{j\frac{2\pi}{\lambda}x_m \sin\theta}$$

For grating lobe free operation $d = \frac{\lambda}{1+\sin\theta_o}$

And for a two dimensional array we need to specify x and y spacing.

$$F(\theta,\phi) = \cos^{\frac{EF}{2}}\theta \cdot \sum_{l=1}^{M\cdot N} c_l e^{j\left(\frac{2\pi}{\lambda}x_l \sin\theta\cos\phi + \frac{2\pi}{\lambda}y_l \sin\theta\sin\phi\right) - \left(\frac{2\pi}{\lambda}x_l \sin\theta_o\cos\phi_o - \frac{2\pi}{\lambda}y_l \sin\theta_o\sin\phi_o\right)}$$

The corresponding model where the above algorithm is implemented is RADAR_PhasedArrayTx (Figure 11).

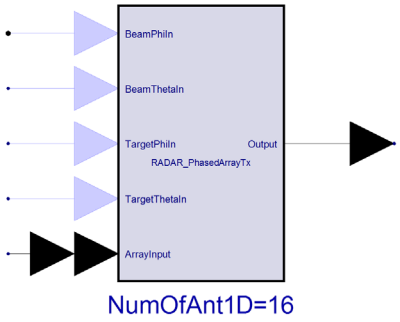

Figure 11. The 2-dimensional array algorithm is implemented in this model.

There are several useful parameters that make this model highly versatile.

- An antenna array can be two dimensional and any size can be specified
- Spacing can be uniform or non-uniform
- The antenna grid can be rectangular or circular
- A mask_array can be defined to turn on/off any selection of antenna elements
- It can model the failure of a percentage of antenna elements randomly to simulate the graceful degradation of the phased array

Target return, jamming, clutter and noise: The scheme for modeling the target reflection, jamming signal, clutter, and noise is shown in Figure 12.

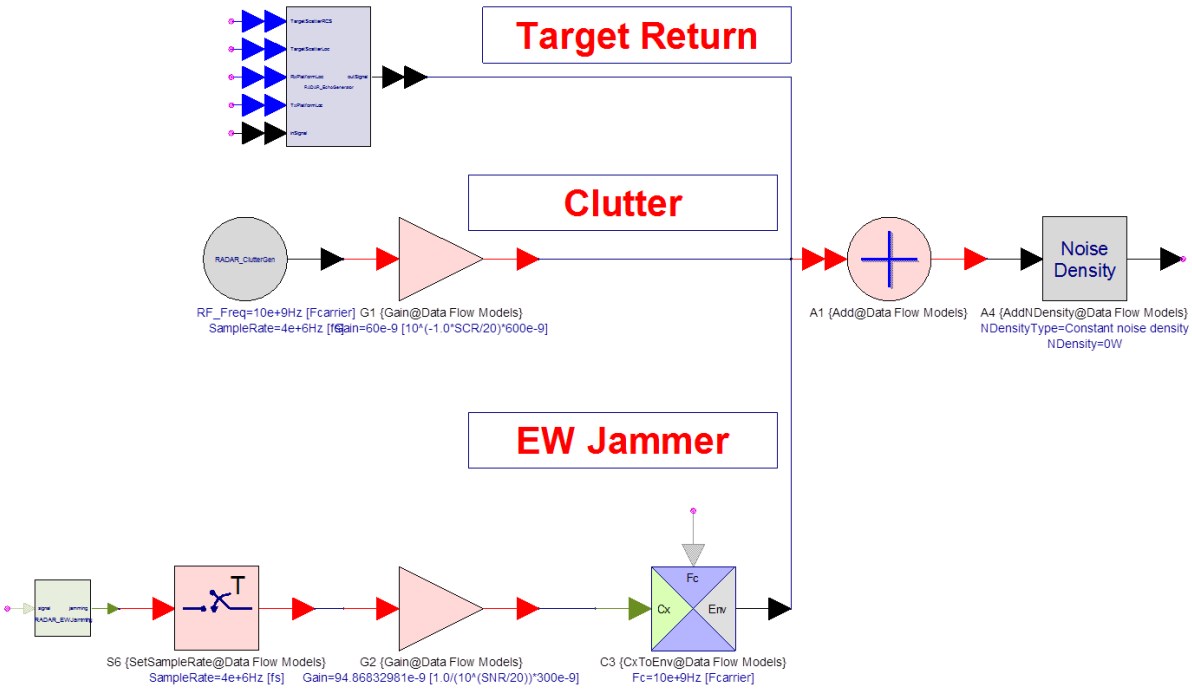

Figure 12. Shown here is the scheme for modeling the target reflection, jamming signal, clutter, and noise.

The inputs to the target return model—the target position, target RCS, transmitting antenna position, receiving antenna position, and waveform from the source—come from the antenna layer. The output from this model is computed using the Radar equation.

The Jammer model is versatile and allows both cover and deceptive jamming to be modelled. There are four types of cover jamming: barrage, spot, multi-spot, and swept spot. For barrage jamming, the model generates the Additive White Gaussian Noise (AWGN) with mean and standard deviation values determined by the parameters Mean and Stdev in the full bandwidth. For spot jamming, the model generates the band-limited AWGN noise. The normalized bandwidth is determined by the parameter Bandwidth. Also, the FilterTapsLength is used to set up the maximum length of low-pass filter. For multi-spot jamming, the model generates band-limited AWGN noise in the sub-bands determined by the parameter MultiSpotBand, which can be used to set up the normalized start and cut-off frequencies of the passbands. For swept-spot jamming, the band-limited AWGN noise sweeps in the scope of the full band, with the sweep rate decided by the parameter SweepFreqStep.

The inputs to the target return model—the target position, target RCS, transmitting antenna position, receiving antenna position, and waveform from the source—come from the antenna layer. The output from this model is computed using the Radar equation.

The Jammer model is versatile and allows both cover and deceptive jamming to be modelled. There are four types of cover jamming: barrage, spot, multi-spot, and swept spot. For barrage jamming, the model generates the Additive White Gaussian Noise (AWGN) with mean and standard deviation values determined by the parameters Mean and Stdev in the full bandwidth. For spot jamming, the model generates the band-limited AWGN noise. The normalized bandwidth is determined by the parameter Bandwidth. Also, the FilterTapsLength is used to set up the maximum length of low-pass filter. For multi-spot jamming, the model generates band-limited AWGN noise in the sub-bands determined by the parameter MultiSpotBand, which can be used to set up the normalized start and cut-off frequencies of the passbands. For swept-spot jamming, the band-limited AWGN noise sweeps in the scope of the full band, with the sweep rate decided by the parameter SweepFreqStep.

NOTE: The idea here is not to absorb all of the details of the model, but to emphasize that in MBE the models have to be made as wide as possible.

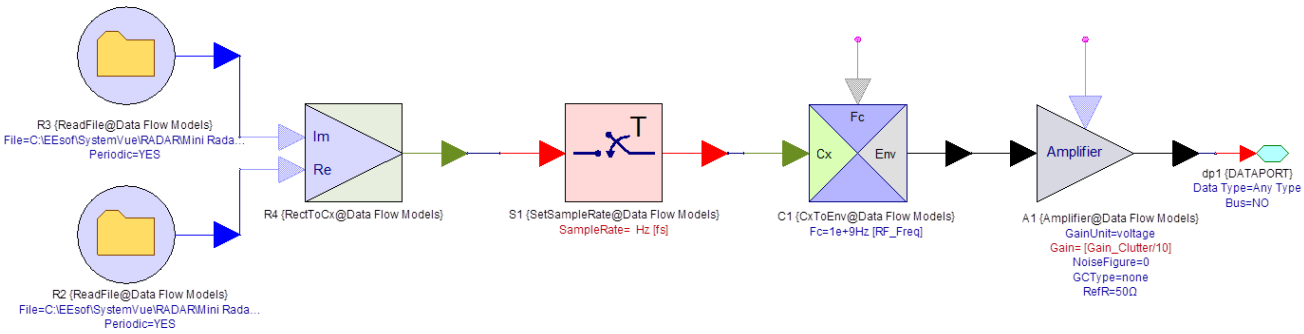

Figure 13. Measured or simulated clutter files can be used for a given terrain.

The measured clutter files, in terms of I and Q, are read from the external files into the simulation. This process is both fast and accurate.

After the addition of noise, the return signal goes into the Radar receiving antenna. The receiving antenna is very similar to the transmitting antenna in that it takes input from the antenna layer and forms a beam in the direction of the target. The output of the receiving antenna is passed through an RF down-converter into the baseband processor. Adaptive digital beamforming techniques are used to combine the outputs of various phased-array antenna elements to form a signal input for the pulse compression block. The pulse compression may be implemented either in the time or frequency domain, as follows:

Time Domain	Frequency Domain
Matched Filter $h(t) = \alpha x^*(T_M - t)$ - s(t): received signal - Output:y(t) = s(t) convolve h(t)	Filtering using DFT/IDFT - Y(w)=S(w) H(w) - y(t)=IDFT(Y(w)) Windowing - Y(W)=S(w) H(w) W(w) - Control side-lobes - Main-lobe broadens

The frequency domain implementation is much faster than the time domain. Conforming to the MBE, frequency-domain pulse compression is implemented in the main system model. After pulse compression, moving target indication (MTI) and moving target detection (MTD) are implemented. MTI is based on the fact that from pulse-to-pulse the return from a stationery object does not change. These static signatures in the return can be canceled from pulse-to-pulse. If the process is done over two pulses, it is called a two-pulse canceler, while if it is done over three pulses it is called a three-pulse canceler. Both of these are implemented in the model and the user can either switch between them or bypass them altogether.

NOTE: While the model is versatile, it can take a long time to simulate and this is contrary to the MBE main system model philosophy. Hence, a better way is to use measured or simulated clutter files for the given terrain. One such implementation is shown in Figure 13.

MTD is where the samples on the received pulses (the number of pulses is determined by system design considerations) are arranged in a matrix. The columns represent the time samples of individual pulses and the rows indicate the time samples on each successive pulse. Thus, we can see that the rows represent the slow time and the columns represent the fast time. After arranging the samples into a matrix, a two-dimensional Fast Fourier Transform (FFT) is performed. The output of the FFT can be plotted in a 3-dimensional graph where the X-axis is the range axis and the Y-axis is the velocity axis. The process is represented in Figure 14.

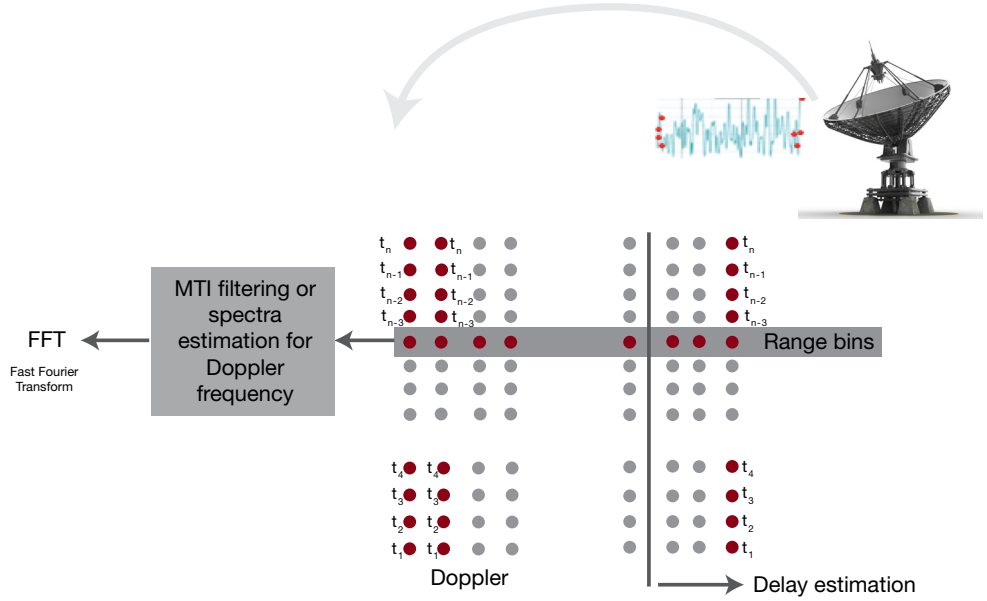

Figure 14. These steps are required for MFD.

The 3-dimensional plot of the FFT output shows a clear peak at every moving target (Figure 15). The coordinates of the peak are the range and velocity of the target. In addition to finding the range and velocity, most Radar simulation and measurements include the probability of detection. For this purpose, additional signal processing blocks are added to the main system model (Figure 16).

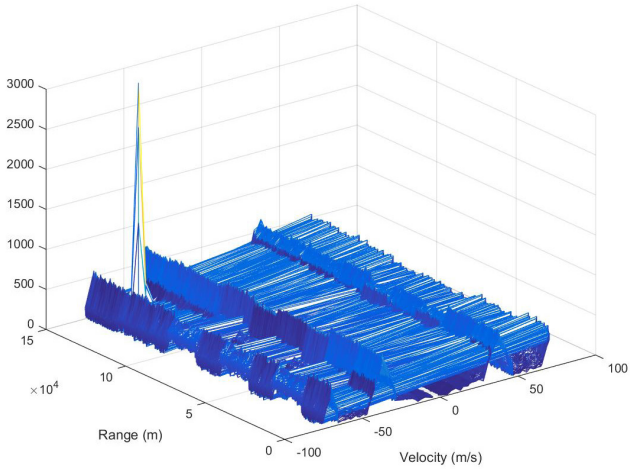

Figure 15. Shown here is a 3-dimensional plot of the FFT output.

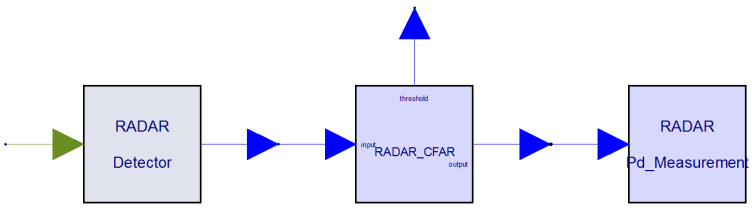

Figure 16. Additional signal processing blocks are added to the main system model.

A completed main system model has the following description:

Specified number of targets, Radar platforms, their positions, orientation in space, velocities, accelerations, number of scatterers on each target, location of each scatterer, RCS value of each scatterer.

Transformed the target(s) coordinates into the antenna frame for each of the Radar transmitter and radar receivers.

Modeled the Radar that includes the waveform generation, antenna beam formation, target return including clutter, jamming, interference and noise, signal processing of the received waveform and finally computation of the target Range, velocity and image.

The corresponding schematic in SystemVue is shown in Figure 17.

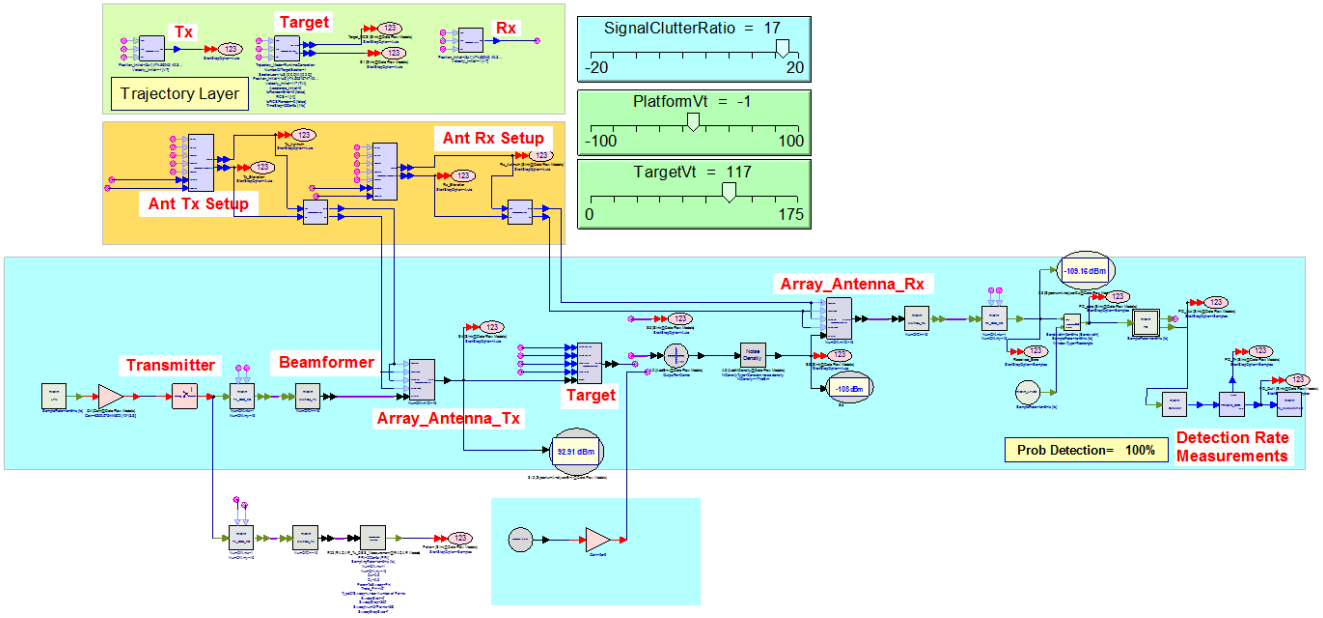

Figure 17. The SystemVue schematic of a completed main system model is shown here. Note that jamming is not shown.

Testing the Main System Model

Now that the model is built, a scenario must be created to test it. This is done by identifying the points along the trajectory of the target and Radar platform. To keep it simple, we chose to create a scenario where a mono-static Radar is stationary and a plane is flying around it. The coordinates of the Radar and the flying object can be extracted from Google maps at http://www.nuclearprojects.com/ins/mapserver/ as shown in Figure 18.

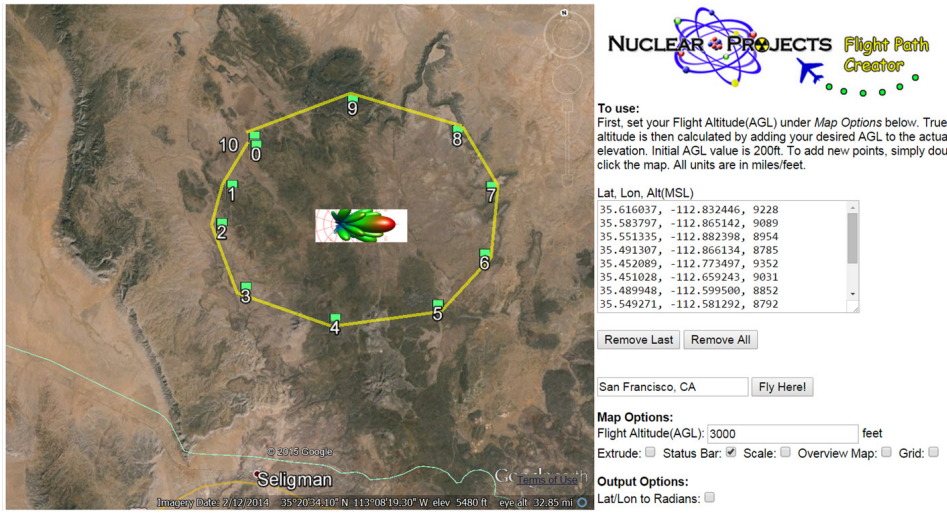

Figure 18. This Google maps image shows a Radar with a plane flying around it.

By entering the altitude (5000 feet, for example) and double clicking on the map, the latitude, longitude and altitude loads on the right hand side. By continuing to click along the desired trajectory we can obtain the LLA coordinates. These can be cut and pasted into a file. After capturing the trajectory into a file the position of the Radar and target can be read into the SystemVue workspace through parameters. Finally, we can compute the distance between the Radar and target. This serves as the reference data that will later be compared to simulation. A sample simulation, with velocity set to 117 meters/sec, yielded the results in the Table.

Set Range Meters	Simulated Range Meters	Simulated Velocity Meters/sec
13522	13500	-29.3
12204	12187	-93.7
14517	14512	-117.2
12971	12975	-102.5
14729	14737	-70.3
10752	10762	-32.2
13938	13912	87.9
13549	13537	108.4
13602	13575	117.2
13277	13275	105.5
13593	13575	74.2
14334	14325	43.94

Table 1. Results of a sample simulation.

Notice that the range has excellent agreement with the set range, indicating that the Radar waveform's pulse width, PRI and sample rate parameters are appropriate. One may think that velocity is not in good agreement with the set velocity. To gauge the agreement, we have to properly interpret the data. To do this, we set the input pins, [YAW PITCH ROLL], to zero in the models for the Radar antenna and target. This makes both the Radar antenna and target orient in the direction of north. Hence, the velocity of the target at every point on the trajectory is actually in the direction of north. We know that Radar measures velocity in the radial direction. This means that the measured velocity is actually the component of the set velocity in the radial direction. If we see the 4th point when the target is at 3, both Radar and target are aligned in the direction of north. The set velocity and simulated velocity should agree. Since the target is moving away from the Radar, the simulated velocity will be negative. This is exactly what the simulation above produced.

How do we use this main system model to derive the top-level system parameters? Even when all building blocks are ideal, the system architecture dictates the performance of the system. For example, the accuracy of velocity depends on the FFT order along the rows in Figure 15. This is nothing but the number of pulses collected for signal processing. We used 128 pulses to achieve adequate accuracy in the prediction of velocity. Similarly, one can experiment with the sample rate (number of samples per pulse) to achieve range accuracy.

Many other attributes of the scenario can be experimented with to study the performance of the Radar for a given concept of operation. Various parameters set in the workspace are:

```
%-------------------------------------------
% System Architectural parameters
%-------------------------------------------
C=3e8                                   % velocity of light
Fcarrier = 2e9                          % carrier frequency
PRI = 200e-6;
fs = 4e6;                           % sampling frequency
PulseWidth=10e-6
CPI_Num = 128;                                  % Number of samples in one CPI(coherent
processing interval), this is also the FFT Size
Pf =1e-6                                        % Probability of false alarm
%-------------------------------------------
% System level derived parameters
%-------------------------------------------
PRF = 1 / PRI;
BandWidth=fs/2;
PRI_Sample_Num = PRI * fs;               % Number of samples in a PRI

range_max=C*PRI/2                   % maximum unambiguous range
fd_max=1/PRI/2;                     % maximum unambiguous Doppler frequency
velocity_max=C*fd_max/2/Fcarrier  % maximum unambiguous velocity M/sec.  1M/s = 3.6 KM/hour
wavelength = C/Fcarrier;
%-----------------------------------------------------
% Parameters to create various conditions
%-----------------------------------------------------
SignalClutterRatio = ?18;         % signal-to-clutter ratio (dB). Controlled by slider
SCR = SignalClutterRatio;
PlatformVt = ?-1;                 % Radar platform speed in the direction of North
VT = PlatformVt;
TargetVt = ?117;                  % Target speed in the direction towards North
SNR = 0;                          % signal-to-noise ratio (in dB), decides the jammer level
rcs_index=?59                     % index to sweep rcs values read from the file
NoiseDensity = ?-170              % Noise Density in dBm/Hz
```

Command and Control of the Main System Model

Once the main system model is created, as in the SystemVue workspace in Figure 17, the next step in MBE is to gain control of the model in a higher level environment like another simulation tool or programming language.

With SystemVue, this is a straightforward process. SystemVue has two Data Flow simulation engines: a built-in one that is used when running simulations inside the SystemVue environment and a standalone one called SystemVue Engine. The SystemVue Engine can be called from the command line to run a Data Flow Analysis in a SystemVue workspace. Advanced users can also write C++ programs through the programming API to control the SystemVue Engine. Some of the popular programming environments and simulation tools from which the SystemVue Engine can be called are C#, MATLAB, LabView, Visual Basic, and STK.

A detailed and well documented procedure for command and control of a main system model is available in the SystemVue Manual3. A graphical depiction of this process is shown in Figure 19. Using SystemVue, it is possible to invoke a SystemVue workspace, set the parameters for simulation and run the analysis. SystemVue can then be placed in standby as the output is read. Finally, the parameters can be resent for another simulation.

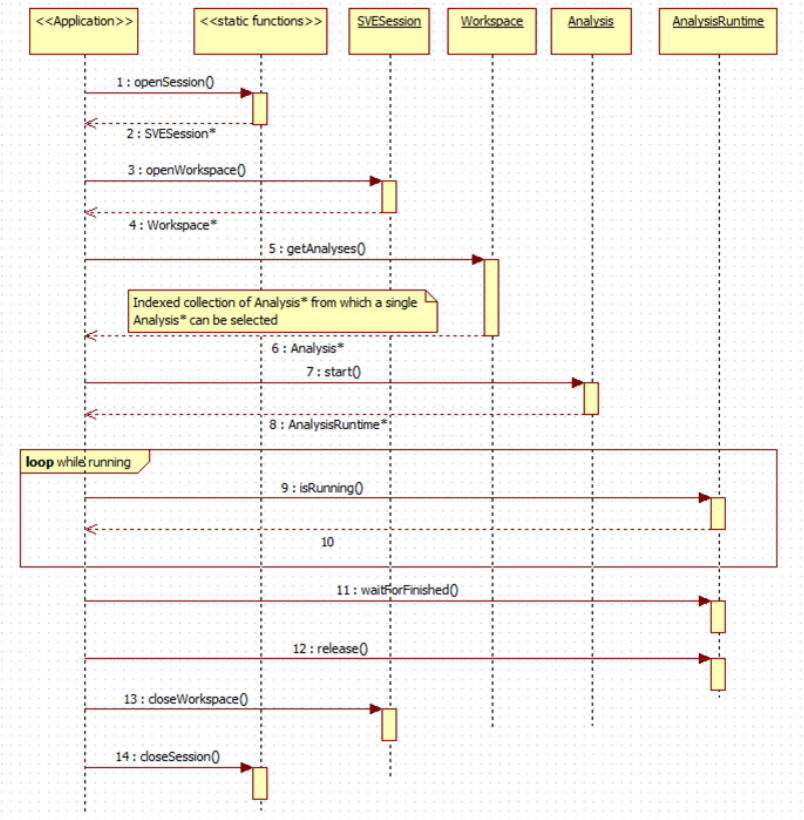

Figure 19. This graphical image shows how a main system model can be called and controlled from within a SystemVue workspace.

Increasing Subsystem Model Fidelity

While the main system model is useful for the quick top-level simulation, it should also support verification of the subsystems designed with higher fidelity models. As an example, the power amplifier (PA) in the transmitter might be designed by a circuit specialist and it may be desirable to see how the actual PA circuit performs in the top-level model (Figure 20).

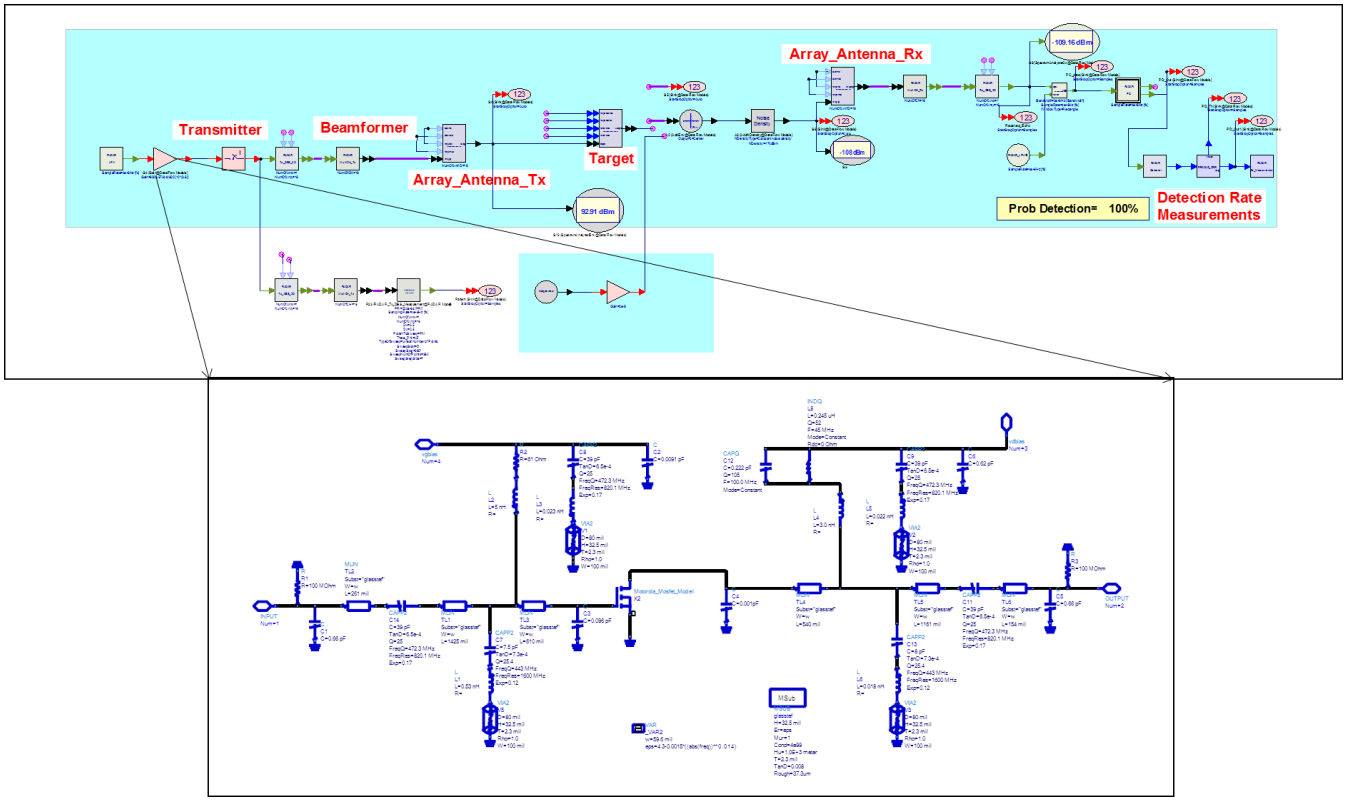

Figure 20. To see how the actual PA circuit in the transmitter performs, ADS can be used to simulate it. SystemVue enables co-simulation with ADS.

A circuit simulator like Advanced Design System (ADS) can simulate the PA circuit. SystemVue can co-simulate with ADS. It is also possible to scale the fidelity of the PA model. To do this, we extract the X-parameters* of the PA and place it as a model for the PA in SystemVue. Note that while X-parameters are excellent in describing the PA's nonlinearity, they cannot completely characterize the memory effects.

An example can also be provided for baseband models. We simply substitute HDL code for a building block that takes into account the finite precision accounting for the quantization effects. Another example is to read in the measured antenna patterns for the antenna elements rather than assuming an analytical antenna factor. **The choice of varying fidelity for the building block models is a key requirement of MBE.**

Bringing a More Dynamic Environment into the Main System Model

SystemVue can co-simulate with the STK simulation tool from Analytic Graphic Incorporated (AGI). STK is a physics-based software geometry engine that accurately displays and analyzes land, sea, air, and space assets in real or simulated time.

First, users model the time-dynamic position and orientation of vehicles. Given these dynamic positions and orientations, users can model the characteristics and pointing of sensors, communications and other payloads aboard the asset. STK then determines the spatial relationships (e.g., line-of-sight) between the assets of interest and all of the objects under consideration. These relationships can also be modeled across multi-hop links or over regions of interest. STK assesses the quality of these relationships through a wide array of constraining conditions (e.g., payload capability, unique user algorithms, etc.), while also incorporating environmental effects such as terrain, lighting and weather conditions on sensor visibility or communication link quality.

When SystemVue is co-simulating with STK, the inputs are brought from STK directly into SystemVue's signal layer. For more details on this process, please refer to reference 4.

Conclusion

Any tool utilized to build a system using the model-based engineering technique requires three key characteristics. It must be able to create the wide and versatile models needed to build a main system model. It must support scalable fidelity for the models. And, it must have the ability for an external program to command and control the main system model. This application note has successfully demonstrated that together, SystemVue, ADS, MATLAB, STK and Google maps fit these requirements.

References

- Final report of the Model Based Engineering (MBE) subcommittee, NDIA systems engineering division, M&S committee, February 10, 2011.
- Ron Williams, INCOSE Model Based Systems Engineering, Integration and verification scenario, Raytheon, INCOSE IW12 MBSE Workshop, January 21-22, 2012.
- SystemVue Programmer's Guide, Controlling SystemVue from external Programs.
- Virtual flight testing using SystemVue and STK, 5991-1254EN, Keysight Technologies.

Related Literatures

- W1461 SystemVue Comms Architect
  http://www.keysight.com/find/eesof-systemvue-comms-architect
- W1905 Radar Model Library
  http://www.keysight.com/find/eesof-systemvue-radar-library

Learn more at: www.keysight.com

For more information on Keysight Technologies' products, applications or services, please contact your local Keysight office. The complete list is available at: www.keysight.com/find/contactus

KEYSIGHT TECHNOLOGIES

This information is subject to change without notice. © Keysight Technologies, 2015-2020, Published in USA, October 11, 2020, 5992-0544EN