

# System of Systems Model Building and Acausal Simulation Environment

Peter Menegay – Director, Funded R&D, SynaptiCAD  
pete@syncad.com, 540-953-3390

Dale Burnham – Lead Systems Engineer, AFRL  
dale.burnham@us.af.mil

National Defense Industrial Association (NDIA)  
21<sup>st</sup> Annual Systems Engineering Conference  
Oct. 25, 2018



Promoting National Security Since 1919

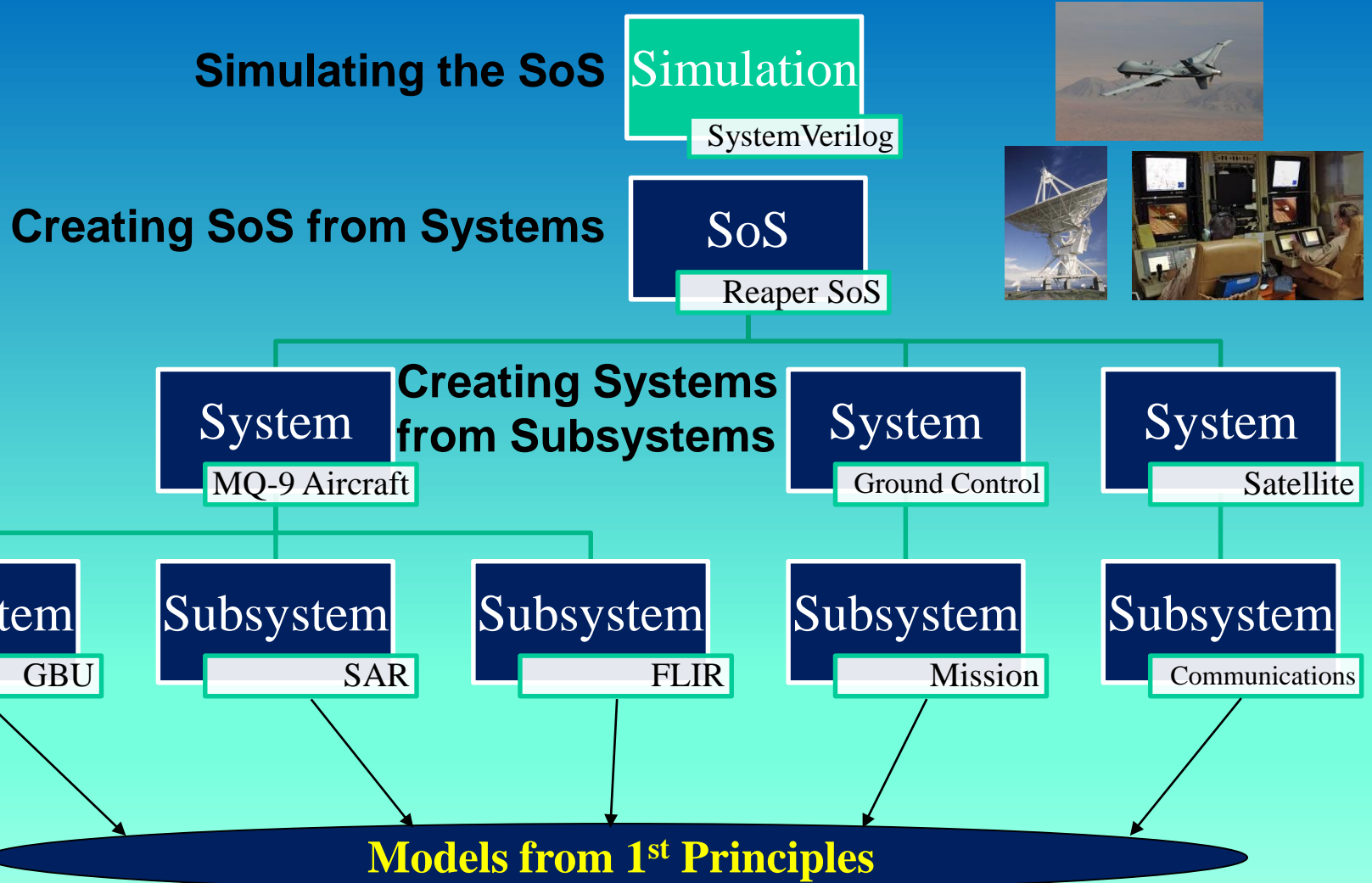


# Technology Motivation

- DoD systems are increasingly complex and challenge human cognitive and organizational abilities. We are now combining those systems into System of Systems.
  - Engineering model flexibility & robustness.
  - Reuse of models.
  - SoS requirements expression and flowdown.
  - Discovering unforeseen behavior through trade studies.
  - Understanding complex results.
  - Accurate simulations, well before we commit.
  - Handling highly scaled simulation problems – “digital twin”.

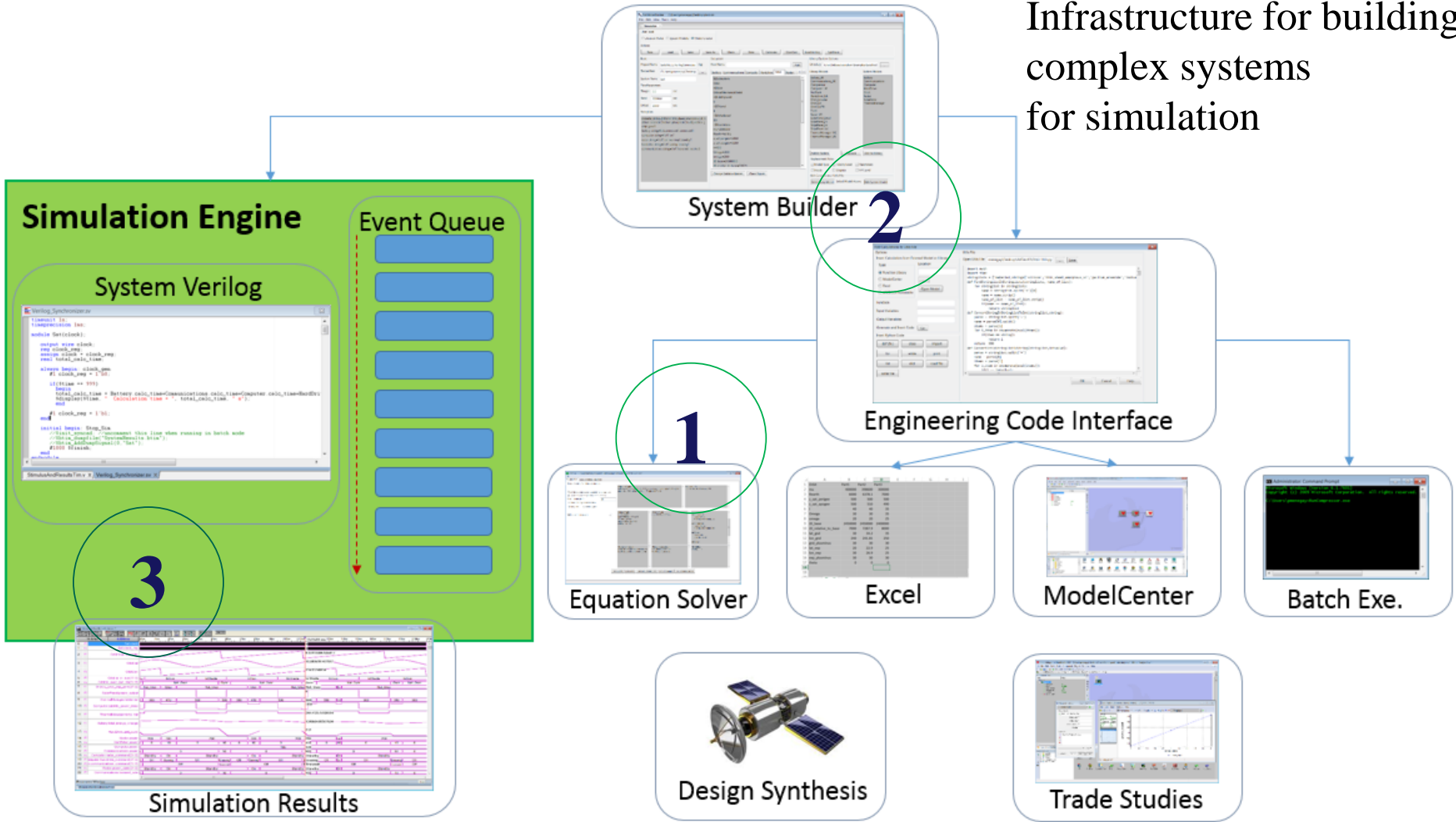


# The Basics



# System Computation Platform

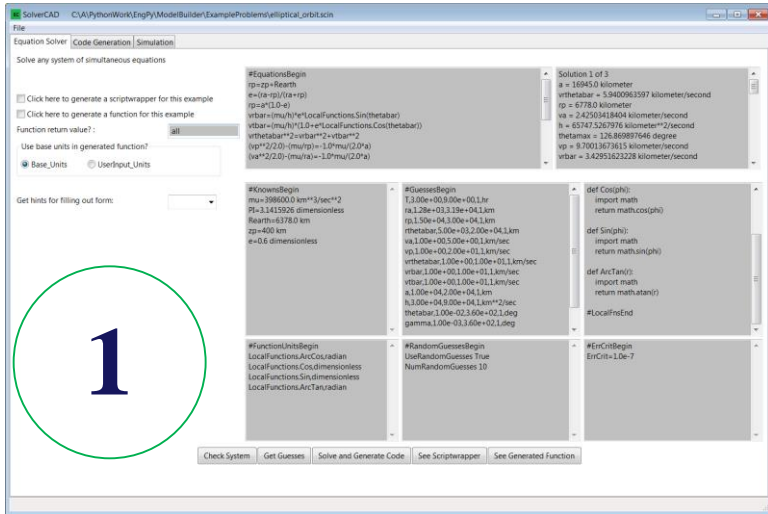
Infrastructure for building complex systems for simulation



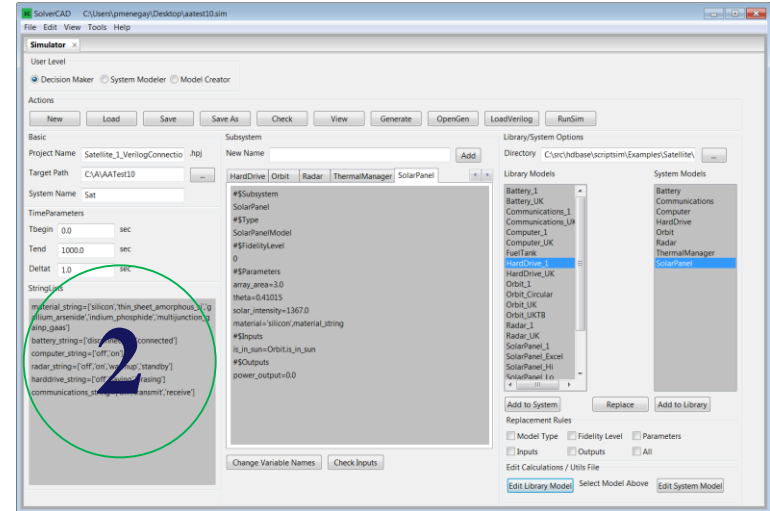
Our approach is to fix weaknesses in the 3 key areas



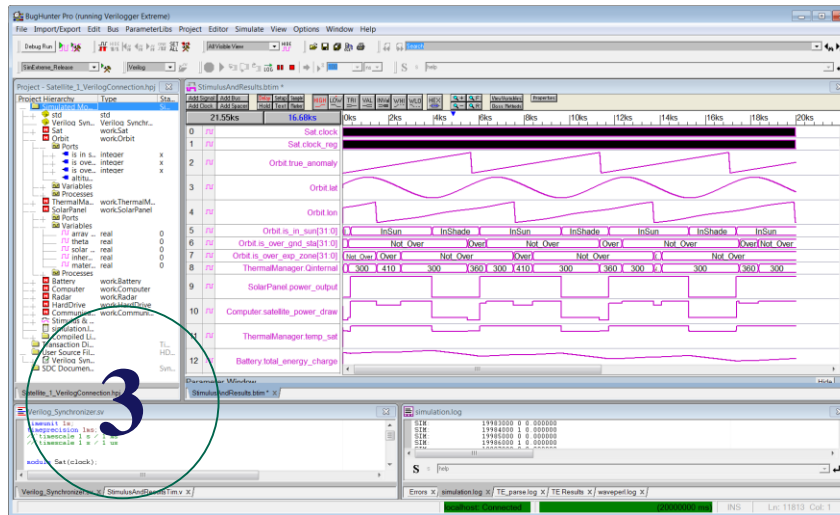
# 3 Core Elements & Workflow



**Eqn. Based Model Creator** -- Simultaneous equation solver for creating new engineering models for subsystems.



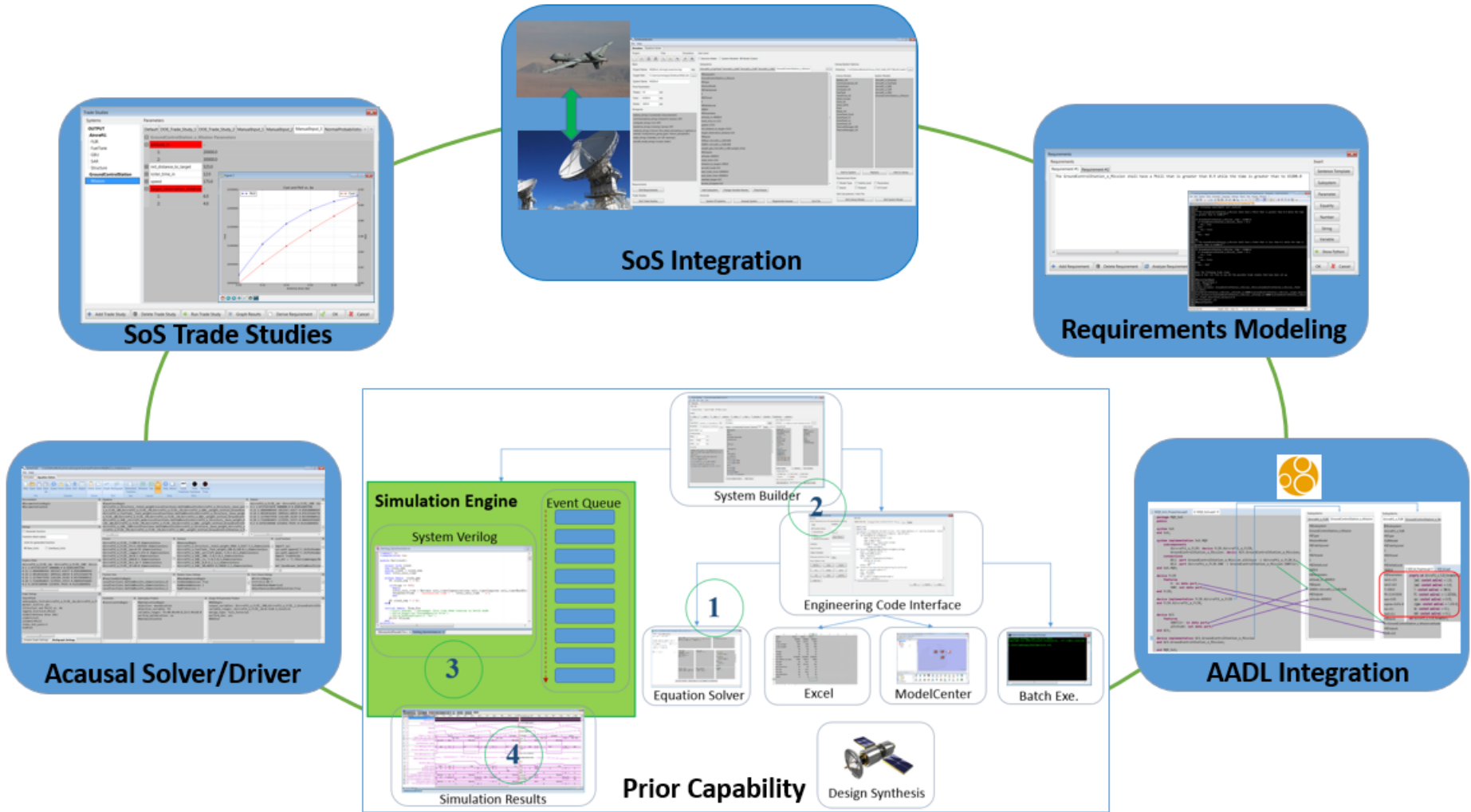
**System Builder** -- User interface for creating new system models.



**System Simulator** -- SystemVerilog simulation engine for system model execution and results.



# New Capability



# Example Problem

An MQ-9 Reaper flies to a target zone and collects sensor data. It identifies a missile site, destroys it and returns to base. The aircraft is equipped with SAR, FLIR, and GBU as part of its system. The ground control station (GCS) controls mission parameters and determines probability of kill. A satellite handles communications between the two.



```
##### Aircraft1_x_FuelTank #####
PAR initial_fuel_weight=3764.0
PAR base_fuel_consumption_rate=0.045
INP None
OUT fuel_weight=2000.0
```

```
##### GroundControlStation_x_Mission #####
PAR altitude_in=40000.0
PAR loiter_time_in=12.0
PAR speed=175.0
PAR init_distance_to_target=525.0
PAR target_observation_distance=6.0
INP SNRsar=Satellite_x_Comm.SNRsar
INP SNRflir=Satellite_x_Comm.SNRflir
INP weight_gbu=Satellite_x_Comm.weight_gbu
OUT altitude=40000.0
OUT loiter_time=13.0
OUT distance_to_target=1000.0
OUT aircraft_mode=0.0
OUT start_loiter_time=200000.0
OUT end_loiter_time=200000.0
OUT reached_target=0.0
OUT bombs_dropped=0.0
OUT Pdetect=0.0
OUT Phit=0.0
OUT Pkill=0.0
OUT Pshot=0.0
OUT ShotDown=0.0
OUT KilledTarget=0.0
```

```
##### Aircraft1_x_FLIR #####
PAR lsm1=2.0
PAR lsm2=6.0
PAR T=500.0
PAR PI=3.1415926
PAR eps=0.95
PAR sigma=5.67e-8
PAR Ae=0.1
PAR Ae0=0.1
PAR N=1.2e-7
PAR AT=20.0
PAR vxref=200.0
PAR beta=6.7e-5
PAR W0=100.0
PAR P0=200.0
PAR C0=1000000.0
INP aircraft_mode_in=Satellite_x_Comm.aircraft_mode
INP h=Satellite_x_Comm.altitude
INP ddt=Satellite_x_Comm.distance_to_target
INP vx=Satellite_x_Comm.speed
OUT aircraft_mode=0.0
OUT C=0.0
OUT F=0.0
OUT P=0.0
OUT R=0.0
OUT Rap=0.0
OUT S=0.0
OUT SNR=0.0
OUT V0=0.0
OUT W=0.0
OUT tau=0.0
```

```
##### Aircraft1_x_Structure #####
PAR base_weight=4900.0
INP fuel_weight=Aircraft1_x_FuelTank.fuel_weight
INP sar_weight=Aircraft1_x_SAR.W
INP flir_weight=Aircraft1_x_FLIR.W
INP gbu_weight=Aircraft1_x_GBU.weight
OUT total_weight=4900.0
```

```
##### Aircraft1_x_GBU #####
PAR weight_initial=1000.0
INP aircraft_mode_in=Satellite_x_Comm.aircraft_mode
INP h=Satellite_x_Comm.altitude
INP ddt=Satellite_x_Comm.distance_to_target
INP vx=Satellite_x_Comm.speed
INP bombs_dropped=Satellite_x_Comm.bombs_dropped
OUT weight=0.0
OUT bombs_dropped_out=0.0
```

```
##### Satellite_x_Comm #####
PAR orbit_altitude_in=117406000.0
INP aircraft_mode_in=GroundControlStation_x_Mission.aircraft_mode
INP altitude_in=GroundControlStation_x_Mission.altitude
INP distance_to_target_in=GroundControlStation_x_Mission.distance_to_target
INP speed_in=GroundControlStation_x_Mission.speed
INP bombs_dropped_in=GroundControlStation_x_Mission.bombs_dropped
INP SNRsar_in=Aircraft1_x_SAR.SNR
INP SNRflir_in=Aircraft1_x_FLIR.SNR
INP weight_gbu_in=Aircraft1_x_GBU.weight_initial
OUT orbit_altitude=117406000.0
OUT aircraft_mode=0.0
OUT altitude=40000.0
OUT distance_to_target=6.0
OUT speed=175.0
OUT bombs_dropped=0.0
OUT SNRsar=5.0
OUT SNRflir=5.0
OUT weight_gbu=1000.0
```

```
##### Aircraft1_x_SAR #####
PAR Pt0=320.0
PAR Daz=44.5
PAR Dh=16.5
PAR rho=1.0
PAR f=16.7
PAR fref=16.7
PAR aw=1.2
PAR lsp=2.0
PAR Lradar=3.16
PAR Fn=1.33
PAR alpha=1.0077
PAR PI=3.14159265
PAR k=1.380648e-23
PAR c=299792458.0
PAR d=0.35
PAR T=200.0
PAR etap=0.5
PAR sigref=3.162e-3
PAR n=2
PAR one=1
PAR Daz0=44.5
PAR Dh0=16.5
PAR Lq0=51.4
PAR W0=29.6
PAR He0=26.7
PAR W0=83.0
PAR C0=1500000.0
INP aircraft_mode_in=Satellite_x_Comm.aircraft_mode
INP h=Satellite_x_Comm.altitude
INP ddt=Satellite_x_Comm.distance_to_target
INP vx=Satellite_x_Comm.speed
OUT aircraft_mode=0.0
OUT SNR=1.00
OUT rs=0.0
OUT R=0.0
OUT V0=0.0
OUT V=0.0
OUT W=0.0
OUT Pt=0.0
OUT C=0.0
```

```
MQ9AGS
tbegin=0.0
tend=82800.0
deltat=3600.0
```



# SoS Integration - 1

- First build independent systems: aircraft, ground control station, and satellite. Aircraft is composed of multiple subsystems (e.g. SAR) which can be modeled in equation solver and connected to overall system:

The screenshot displays the SolverCAD software interface, which is used for solving systems of equations and integrating subsystems. The main window shows the 'Equations' pane with the following equations:

```
1 SNR=Pt*d*etap**2*(Daz*Dh)**2*rhor*(sigref/fref**n)*f**(n-one)/(8.0*PI*aw*c*k*T*vx*Lsp*Lr
2 rs=(dtt**2+h**2)**0.5
3 R=(Daz*Dh)/(Daz0*Dh0)**0.5
4 V0=(Le0+Dh0)*We0*He0
5 V=V0*R**3
6 W=R**3*W0*LocalFunctions.GetK(R)
7 Pt=R**3*Pt0*LocalFunctions.GetK(R)
8 C=C0*R**0.562
9
```

The 'Solution' pane displays the results of the solve operation:

```
1 Solution 1 of 1
2 C = 1500000.0 USD
3 Pt = 320.0 watt
4 R = 1.0 dimensionless
5 SNR = 3.8833034251815737 dimensionless
6 V = 53662.728000000001 centimeter**3
7 W0 = 53662.728000000001 centimeter**3
8 W = 83.00000000000014 pound
9 rs = 9.6639589634190965 mile
```

An 'Add Calculations to Utils File' dialog box is open, showing the 'Utils File' pane with the following code:

```
sigref=passed_in_object.sigref
n=passed_in_object.n
one=passed_in_object.one
Daz0=passed_in_object.Daz0
Dh0=passed_in_object.Dh0
Le0=passed_in_object.Le0
W0=passed_in_object.W0
He0=passed_in_object.He0
W0=passed_in_object.W0
C0=passed_in_object.C0
#call the library function with provided inputs
r = FunctionLibrary.SAR_3413(vx,dtt,h,Pt0,Daz,Dh,rhor,f,fref,aw,Lsp,Lradar,Fn,alpha,PI,k,c,
#get the outputs in a convenient numerical form
SNR = r['SNR'].magnitude
rs = r['rs'].magnitude
R = r['R'].magnitude
V0 = r['W0'].magnitude
W = r['V'].magnitude
W = r['W'].magnitude
Pt = r['Pt'].magnitude
C = r['C'].magnitude
return [SNR,rs,R,V0,V,W,Pt,C]
```

The interface also includes a 'System Models' pane on the right, listing various models such as 'Aircraft1\_x\_Structure', 'Aircraft1\_x\_FuelTank', 'Aircraft1\_x\_SAR', 'Aircraft1\_x\_FLIR', 'Aircraft1\_x\_GBU', and 'GroundControlStation'. The 'SYNAPTICAD' logo is visible in the bottom right corner.



# SoS Integration - 2

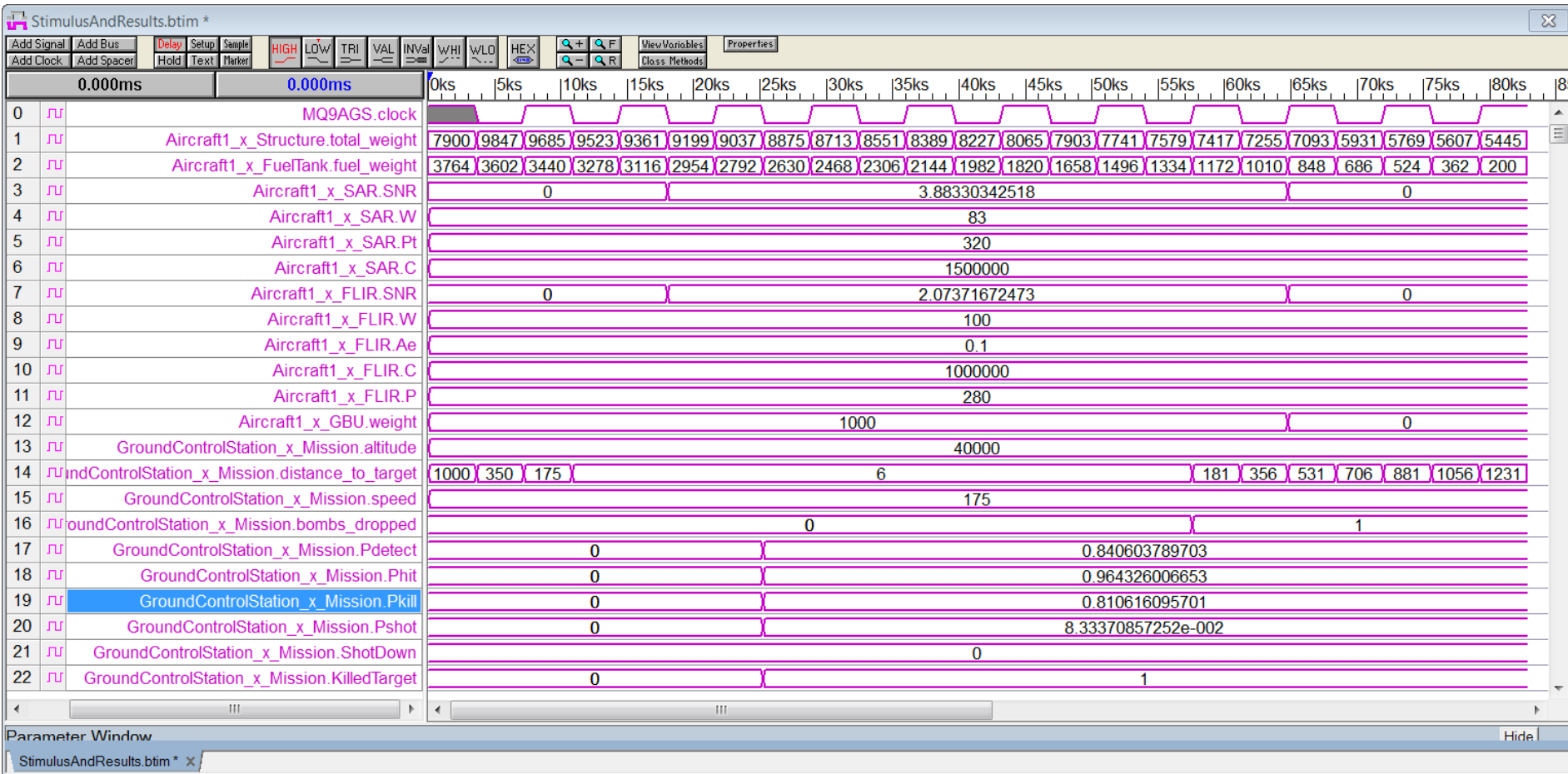
- Next, SoS is built by specifying which systems to unite. A flat hierarchy of all contained subsystems results. Variable links must be created.

The screenshot displays the SimModelBuilder software interface. The main window is titled "SimModelBuilder" and contains several panels:

- Basic:** Project Name: MQ9AGS\_VerilogConnection.hpj, Target Path: C:\Users\pmenegay\Desktop\MQ9\_AGS, System Name: MQ9AGS.
- Time Parameters:** Tbegin: 0.0 sec, Tend: 82800.0 sec, Deltat: 3600.0 sec.
- StringLists:** A list of strings for various components like battery, communications, computer, etc.
- Subsystems:** A list of subsystems including Aircraft1\_x\_SAR, Aircraft1\_x\_FLIR, Aircraft1\_x\_GBU, GroundControlStation\_x\_Mission, and Satellite\_x\_Comm. A green arrow points to the "Variable Links" section within this list.
- Library/System Options:** Directory: C:\PythonWork\AirForce\_DUE\_Feb8\_2017\Work\Code\, Library Models, and System Models.
- Buttons:** "Add Subsystem", "Change Variable Names", "Check Inputs", "System Of Systems" (highlighted with a red circle), "Acausal System", "Regenerate Acausal", "Scin File", "Add to System", "Replace", "Add to Library", "Edit Requirements", "Edit Trade Studies", "Edit Library Model", "Edit System Model".

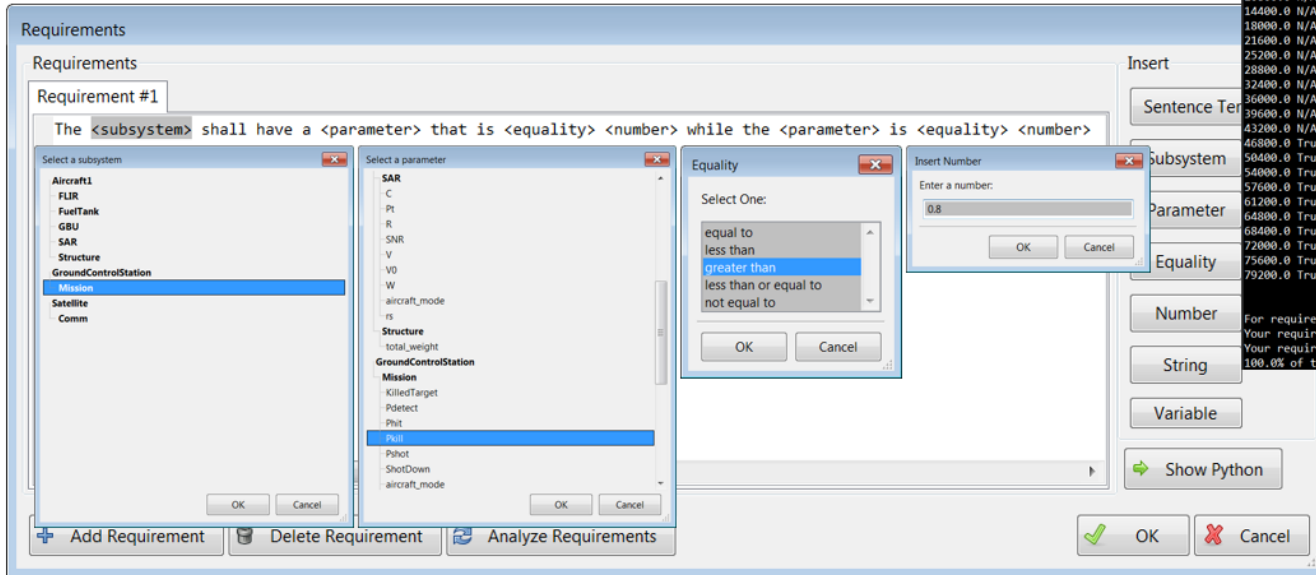
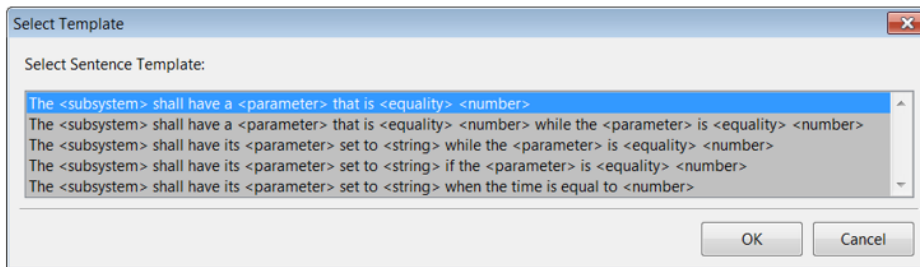
# Simulation Results

- The SoS is simulated using the System Verilog engine and results are shown in a timing diagram:



# Requirements Modeling

- A template is used to create requirements sentences in English which are translated to Python for analysis. Requirements can also be written directly in Python.



```
Requirement #1:
***The GroundControlStation_x_Mission shall have a Pkill that
is greater than 0.8 while the time is greater than to 43200.0***

if GroundControlStation_x_Mission_time > 43200.0:
    if GroundControlStation_x_Mission_Pkill > 0.8:
        req = True
    else:
        req = False
else:
    req = 'N/A'
```

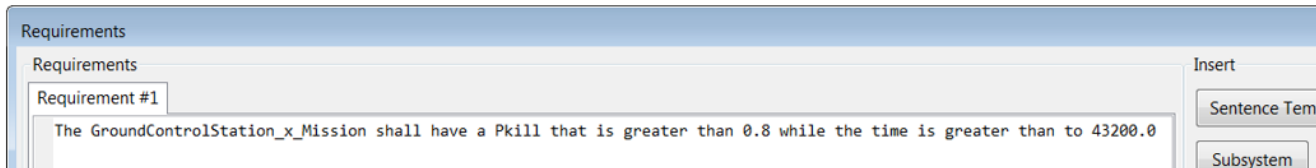
```
Time #1
0.0 N/A
3600.0 N/A
7200.0 N/A
10800.0 N/A
14400.0 N/A
18000.0 N/A
21600.0 N/A
25200.0 N/A
28800.0 N/A
32400.0 N/A
36000.0 N/A
39600.0 N/A
43200.0 N/A
46800.0 True
50400.0 True
54000.0 True
57600.0 True
61200.0 True
64800.0 True
68400.0 True
72000.0 True
75600.0 True
79200.0 True
```

```
name: DOE_Trade_Study_1
times: 43200.0
output_variables:
  GroundControlStation_x_Mission_Pkill,GroundControlStation_x_Mission_Pshot
variable_ranges:
  GroundControlStation_x_Mission_altitude_in=15000.0to50000.0,levels=8
design_type: full_factorial
samples: 3
perform_doe: yes
#DOEEnd

Results over time by combining trade studies for each requirement
Each trade study point is combined with the others using AND.
WARNING: If a point is N/A and another is True, the combined result is
N/A.
```

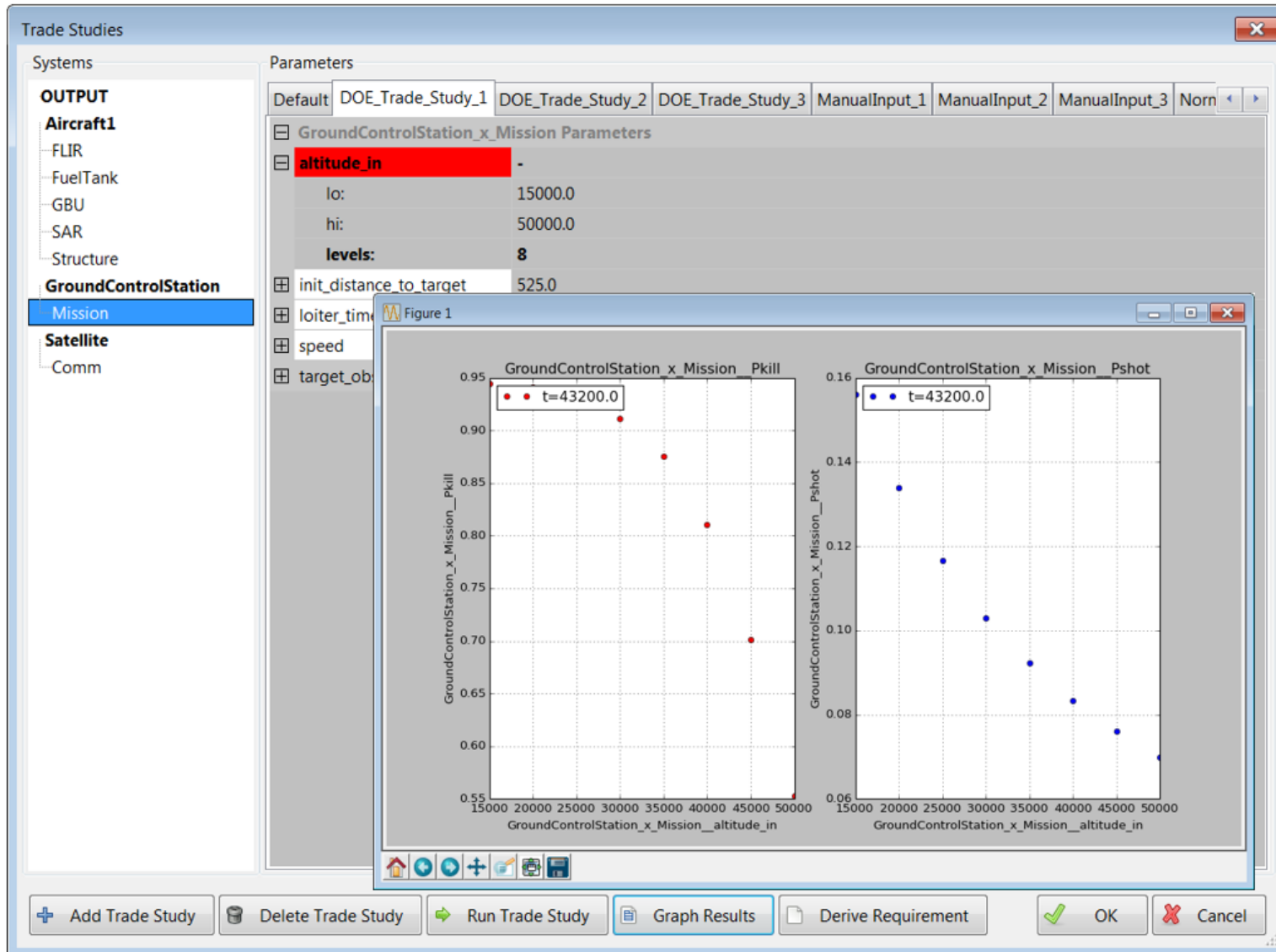
Time	#1_res
0.0	N/A
3600.0	N/A
7200.0	N/A
10800.0	N/A
14400.0	N/A
18000.0	N/A
21600.0	N/A
25200.0	N/A
28800.0	N/A
32400.0	N/A
36000.0	N/A
39600.0	N/A
43200.0	N/A
46800.0	False
50400.0	False
54000.0	False
57600.0	False
61200.0	False
64800.0	False
68400.0	False
72000.0	False
75600.0	False
79200.0	False

```
Requirement 1 across trade studies held true 0.0% of the time
Requirement 1 across trade studies held true 0.0% of the time when
applicable
Requirement 1 across trade studies was N/A 56.217391304% of the time
```



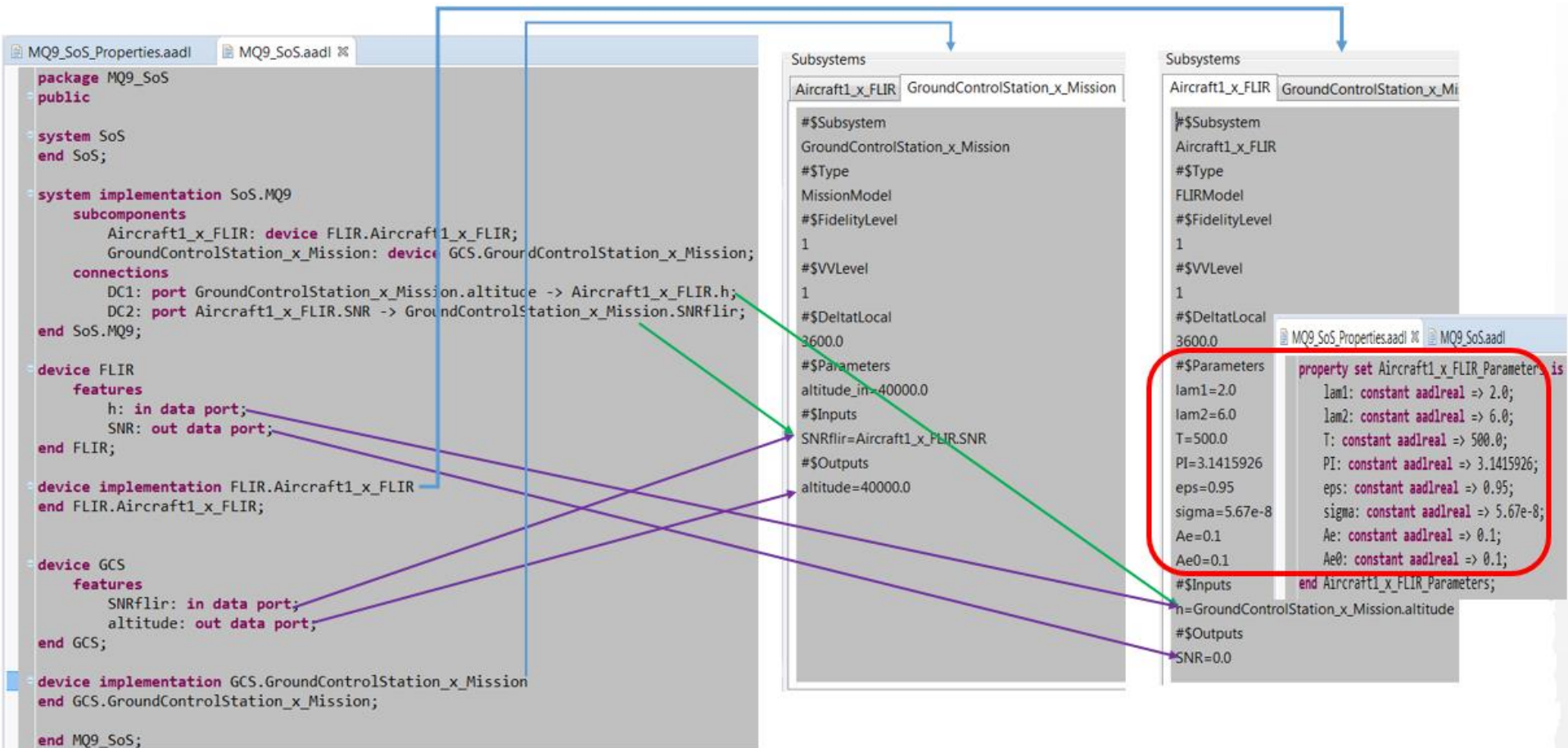
# SoS Trade Studies

- Operate over the entire SoS and perform full simulations for each point. DOE, Monte-Carlo, and single variable optimization is available.



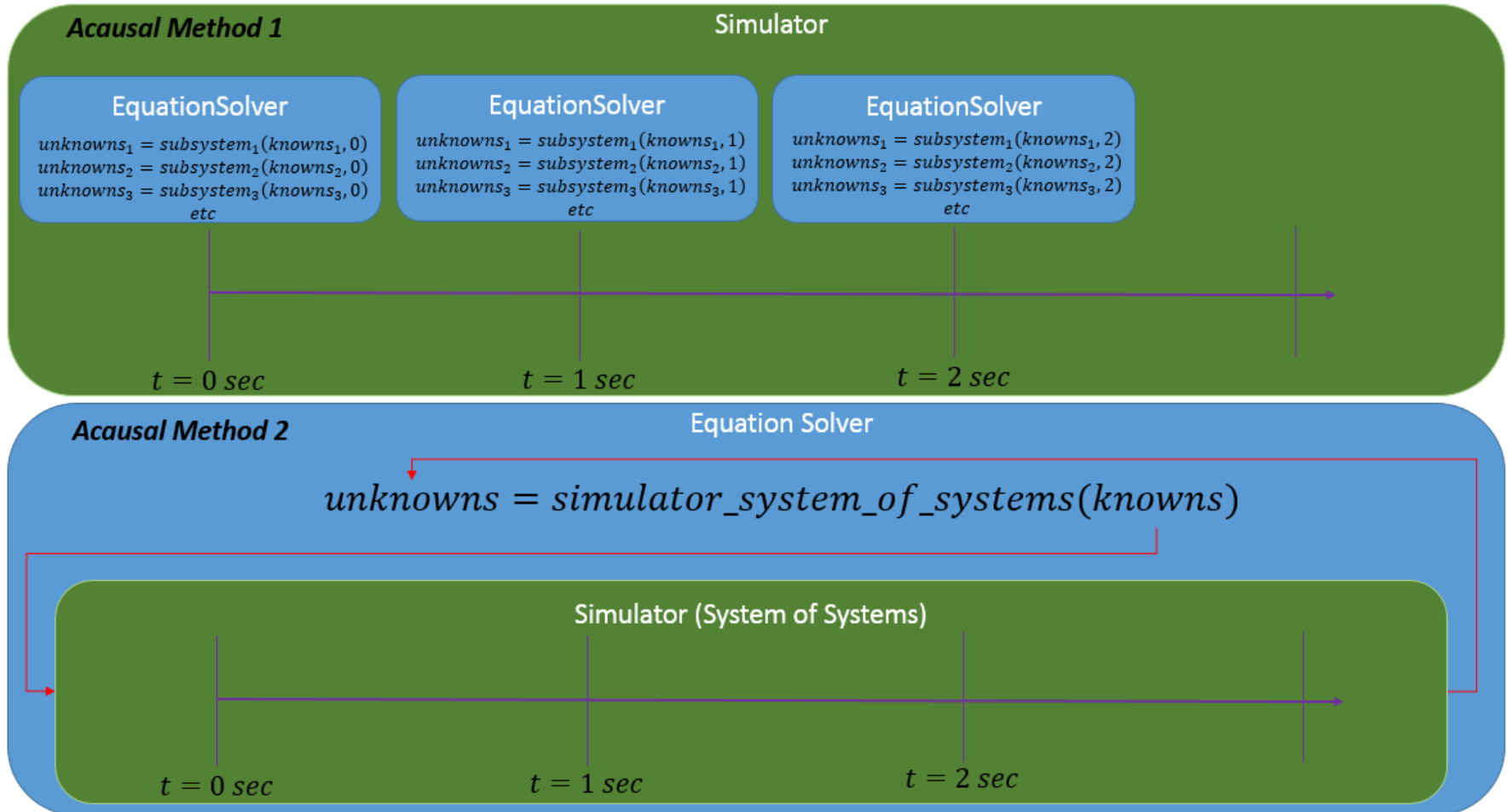
# AADL Integration

- Simulation models are architecturally analogous to AADL language.



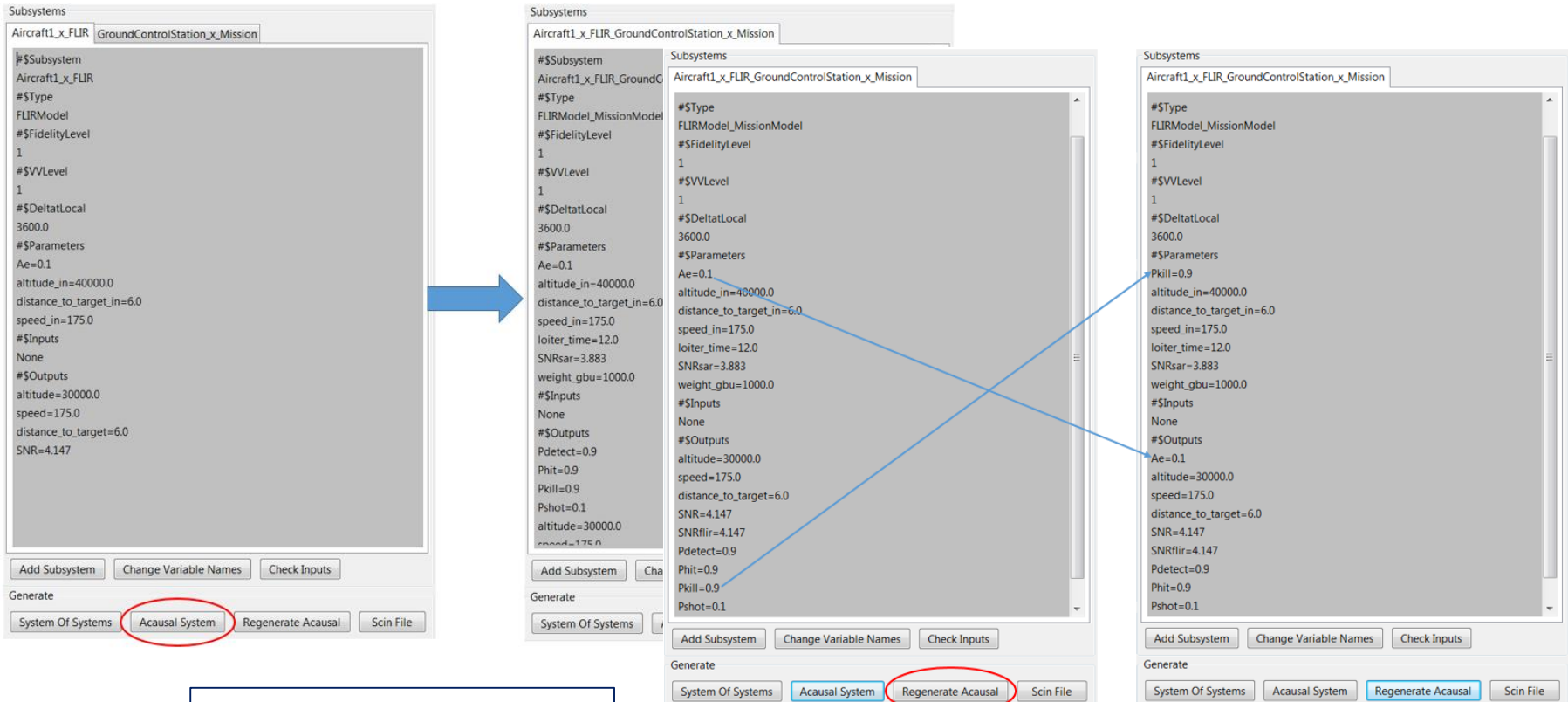
# Acausal Solver/Driver

- Acausal simulation means that what we consider an input or output is not fixed. All models are solved as a system of equations using two methods:

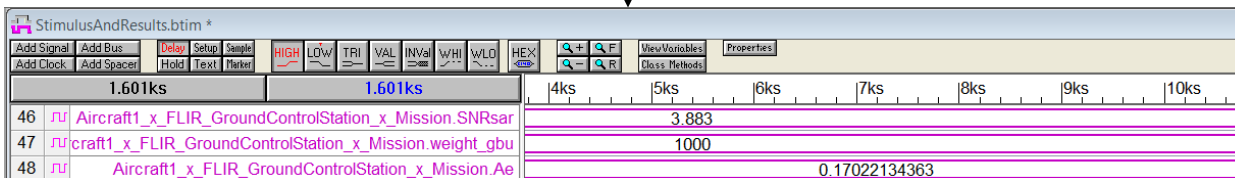


# Acausal Method 1

- Each system is rendered as an equation to be solved at every timestep. The simulator is the driver and the equation solver runs behind the scenes:



**Equation Solver**



# Acausal Method 2

- The entire SoS is viewed as an equation to be solved over the entire span of time of the simulation. The equation solver is the driver and the simulator runs behind the scenes.

The screenshot displays the SolverCAD software interface, specifically the Equation Solver window. The window title is "SolverCAD C:/A/PythonWork/solvercad/engine/ExampleProblems/MQ9AGS\_b.scin". The interface is divided into several panels:

- Equations:** A list of equations defining simulation results, such as `1 Aircraft1_x_Structure_total_weight=LocalFunctions.GetSimResults(Aircraft1_x_Structure_base_weight,Aircraft1_x_FuelTank_fuel_weight=LocalFunctions.GetSimResults(Aircraft1_x_Structure_base_weight,Aircraft1_x_SAR_aircraft_mode=LocalFunctions.GetSimResults(Aircraft1_x_Structure_base_weight,Aircraft1_x_SAR_SNR=LocalFunctions.GetSimResults(Aircraft1_x_Structure_base_weight,Aircraft1_x_SAR_rs=LocalFunctions.GetSimResults(Aircraft1_x_Structure_base_weight,Aircraft1_x_SAR_Re=LocalFunctions.GetSimResults(Aircraft1_x_Structure_base_weight,Aircraft1_x_SAR_V0=LocalFunctions.GetSimResults(Aircraft1_x_Structure_base_weight,Aircraft1_x_SAR_V=LocalFunctions.GetSimResults(Aircraft1_x_Structure_base_weight,Aircraft1_x_SAR_W=LocalFunctions.GetSimResults(Aircraft1_x_Structure_base_weight,Aircraft1_x_SAR_Pt=LocalFunctions.GetSimResults(Aircraft1_x_Structure_base_weight,Aircraft1_x_SAR_C=LocalFunctions.GetSimResults(Aircraft1_x_Structure_base_weight,Aircraft1_x_FLIR_aircraft_mode=LocalFunctions.GetSimResults(Aircraft1_x_Structure_base_weight,Aircraft1_x_FLIR_C=LocalFunctions.GetSimResults(Aircraft1_x_Structure_base_weight,Aircraft1_x_FLIR_F=LocalFunctions.GetSimResults(Aircraft1_x_Structure_base_weight,Aircraft1_x_FLIR_P=LocalFunctions.GetSimResults(Aircraft1_x_Structure_base_weight,`
- Solution:** A list of solved values, such as `1 Solution 1 of 1`, `2 Aircraft1_x_FLIR_Ae = 0.17021165469305916 dimensionless`, `3 Aircraft1_x_FLIR_C = 1161202.5784799999 dimensionless`, `4 Aircraft1_x_FLIR_F = 0.2786674180390003 dimensionless`, `5 Aircraft1_x_FLIR_P = 621.78747828099995 dimensionless`, `6 Aircraft1_x_FLIR_R = 15552.634373999999 dimensionless`, `7 Aircraft1_x_FLIR_Rap = 1.3046518872599999 dimensionless`, `8 Aircraft1_x_FLIR_S = 4.2356490609700001e-07 dimensionless`, `9 Aircraft1_x_FLIR_SNR = 3.52970755081 dimensionless`, `10 Aircraft1_x_FLIR_V0 = 0.023788321751599999 dimensionless`, `11 Aircraft1_x_FLIR_W = 222.06695652900001 dimensionless`, `12 Aircraft1_x_FLIR_aircraft_mode = 1.0 dimensionless`, `13 Aircraft1_x_FLIR_tau = 0.35273913024300002 dimensionless`, `14 Aircraft1_x_FuelTank_fuel_weight = 1820.0 dimensionless`
- Knowns:** A list of known values, such as `44 Aircraft1_x_FLIR_P0=280.0 dimensionless`, `45 Aircraft1_x_FLIR_C0=1000000.0 dimensionless`, `46 Aircraft1_x_GBU_weight_initial=1000.0 dimensionless`, `47 GroundControlStation_x_Mission_altitude_in=40000.0 dimens`, `48 GroundControlStation_x_Mission_loiter_time_in=12.0 dimens`, `49 GroundControlStation_x_Mission_speed=175.0 dimensionless`, `50 GroundControlStation_x_Mission_init_distance_to_target=52`, `51 GroundControlStation_x_Mission_target_observation_distan`, `52 Satellite_x_Comm_orbit_altitude_in=117406080.0 dimensionl`, `53 t=43200.0 dimensionless`, `4 GroundControlStation_x_Mission_Pkill=0.9 dimensionless`
- Guesses:** A list of guessed values, such as `38 GroundControlStation_x_Mission_KilledTarget,0.9,1,1`, `39 Satellite_x_Comm_orbit_altitude,105665472.0,1291466`, `40 Satellite_x_Comm_aircraft_mode,0.9,1,1,1,dimensionl`, `41 Satellite_x_Comm_altitude,36000.0,44000.0,1,dimensi`, `42 Satellite_x_Comm_distance_to_target,5.4,6.6,1,dimen`, `43 Satellite_x_Comm_speed,157.5,192.5,1,dimensionless`, `44 Satellite_x_Comm_bombs_dropped,-5.0,5.0,1,dimension`, `45 Satellite_x_Comm_SNRsar,2.7,3.3,1,dimensionless`, `46 Satellite_x_Comm_SNRflir,2.7,3.3,1,dimensionless`, `47 Satellite_x_Comm_weight_gbu,900.0,1100.0,1,dimensic`, `8 Aircraft1_x_FLIR_Ae,0.1,0.2,1,dimensionless`
- Local Functions:** A list of function definitions, such as `1 import time`, `2 import Utils`, `3 import Globals`, `4 schome,userhome,exhome,pubhome = Globals.g`, `5`, `6 import sys`, `7 sys.path.append('C:/A/PythonWork/AirForce`, `8 sys.path.append('C:/A/PythonWork/solvercad`, `9 import TradeStudy`, `10 res_dir = 'C:/Users/pmeneqay/Desktop/MQ9_`, `11`, `12 def SaveAnswer_GetSimResults(arguments,ca`
- Function Units:** A list of function units, such as `1 LocalFunctions.GetSimResults,dimensionless,0`, `2 LocalFunctions.GetSimResults,dimensionless,1`, `3 LocalFunctions.GetSimResults,dimensionless,2`, `4 LocalFunctions.GetSimResults,dimensionless,3`, `5 LocalFunctions.GetSimResults,dimensionless,4`, `6 LocalFunctions.GetSimResults,dimensionless,5`, `7 LocalFunctions.GetSimResults,dimensionless,6`, `8 LocalFunctions.GetSimResults,dimensionless,7`, `9 LocalFunctions.GetSimResults,dimensionless,8`, `10 LocalFunctions.GetSimResults,dimensionless,9`, `11 LocalFunctions.GetSimResults,dimensionless,10`, `12 LocalFunctions.GetSimResults,dimensionless,11`
- Random Guess Settings:** A list of settings, such as `1 UseRandomGuesses True`, `2 NumRandomGuesses 1`, `3 NumPreGuesses 1`, `4 RandomGuessMethod LatinHypercube`, `5`
- Error Criteria Settings:** A list of error criteria, such as `1 ErrCrit=1.0E-7`, `2 SolveMethod=Numerical`, `3 AdjustGuessesBasedOnSolution=T`, `4 SolutionToUseForAdjustment=1`, `5 PercentSpreadAroundSolution=5`, `6 InputFileToModify=y_junk.scin`, `7`



# Some Observations

- Scale up.
  - Human scalability: our ability to see, understand, and manipulate large SoS models.
  - Computational scalability: Multiple nested engineering models, solved iteratively by equation solver, are possible. Trade studies make this worse.
- “Explosion” of data caused by trade studies.
- English requirements are backed up by requirements expression through a computer language (i.e. Python).
- Requirements modeling can be enforced through equation solver.
- The acausal capability allows for many scenarios without having to rebuild the model. Inputs/outputs can be switched.
- This tool is designed to be linked from 3<sup>rd</sup> party systems engineering tools such as AADL or SysML.



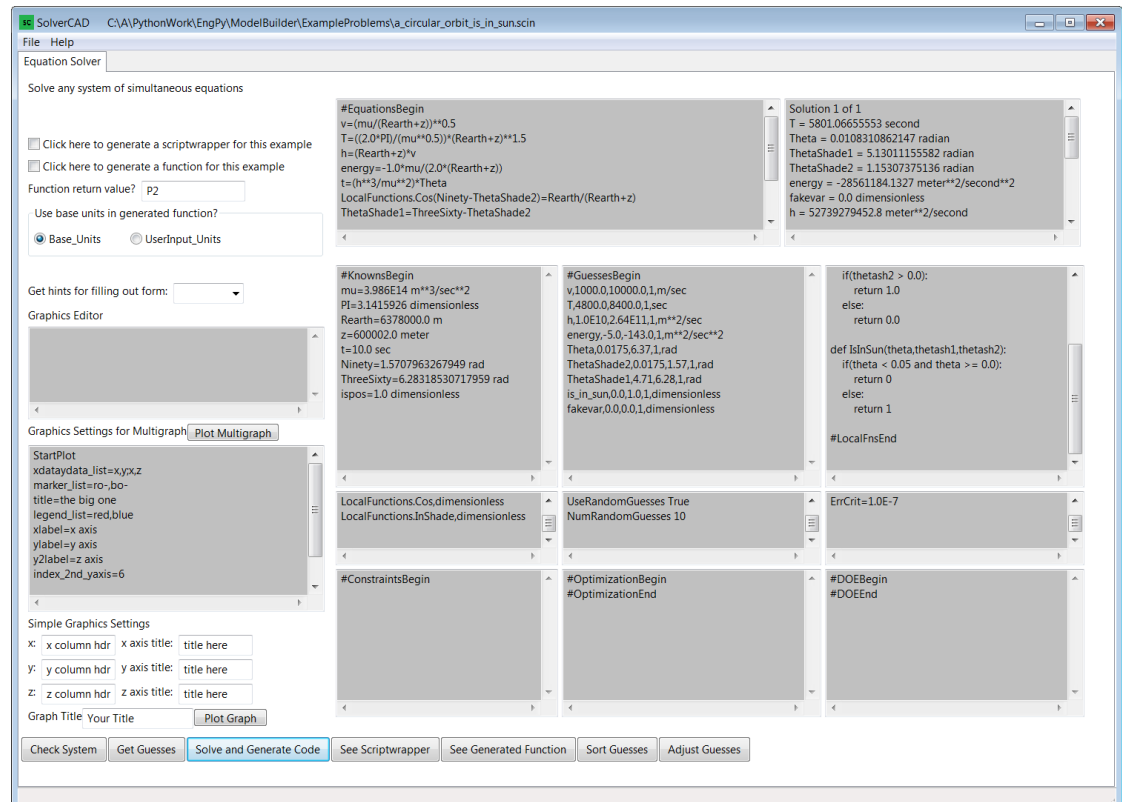
# Backup Slides



# 1. Eqn. Based Model Creator

A general purpose model-creation environment for engineering analysis. Under development.

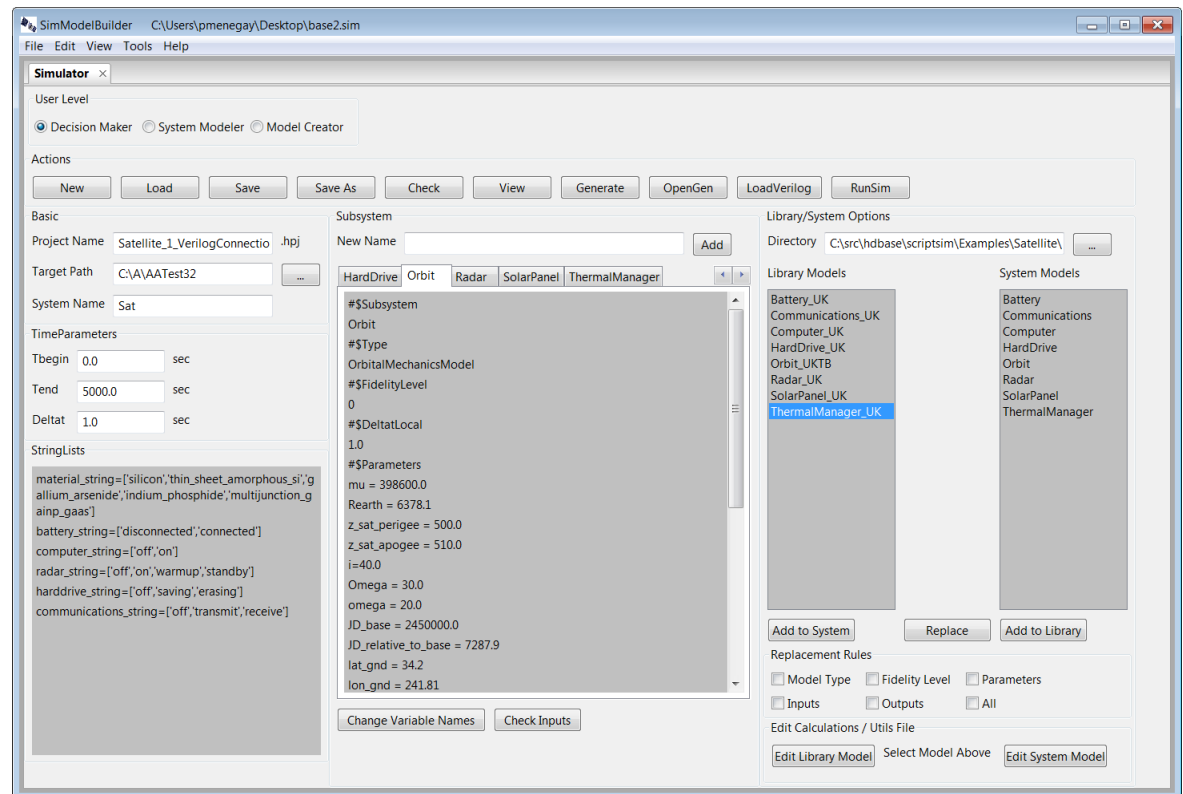
- Solves any system of nonlinear simultaneous equations.
- Manages the core numerical library to achieve robustness.
- Generates Python functions to use in simulation code.
- Uses a library concept for storing functions for later use.



# 2. System Builder

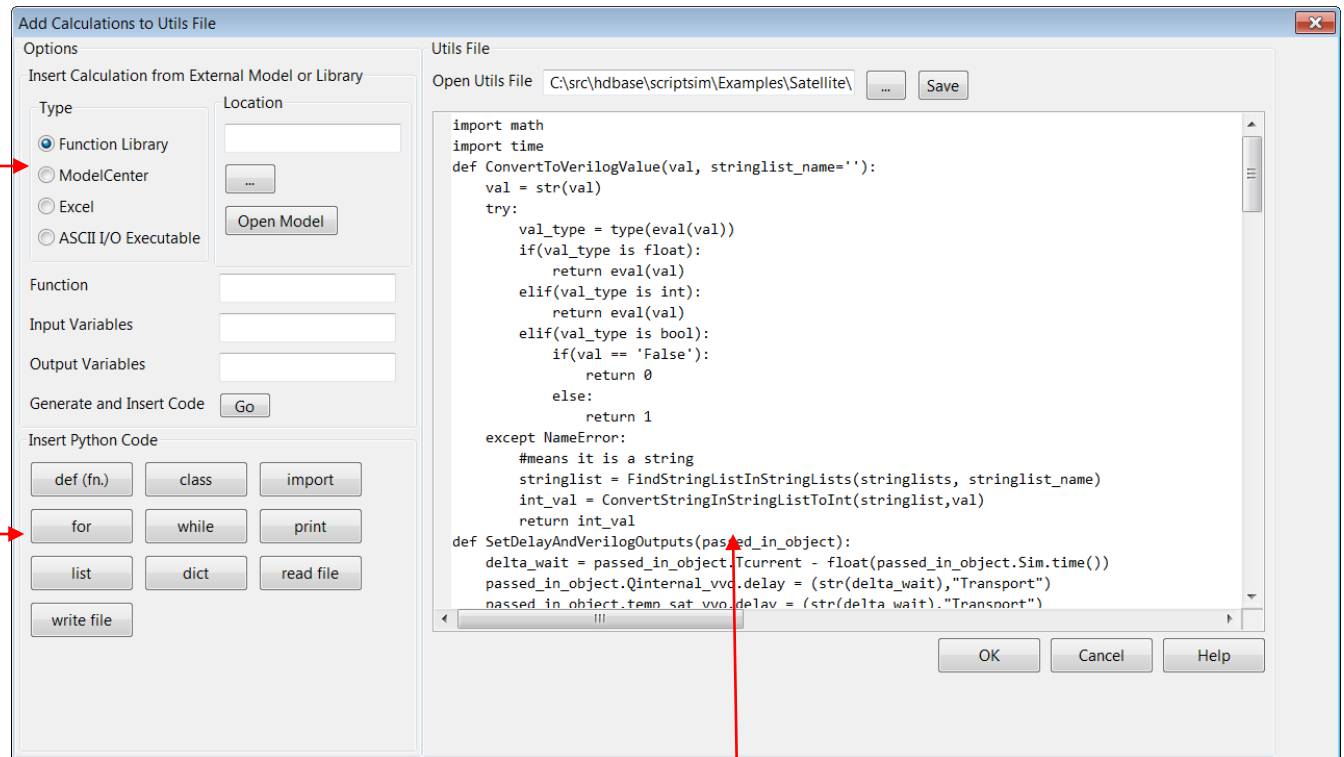
A GUI that helps you build, connect, and modify sub-system models for simulation.

- Create system models from a library of pre-built subsystems.
- Create subsystem models from equation solver and external tools.
- Publish subsystems to library.
- Easily replace subsystems with higher/lower fidelity ones.



## 2. System Builder, cont.

Includes a feature to help you edit calculations or link to them from external tools.



Link to pre-built models in SolverCAD, ModelCenter, Excel, or your own program.

Buttons for quickly creating common Python constructs.

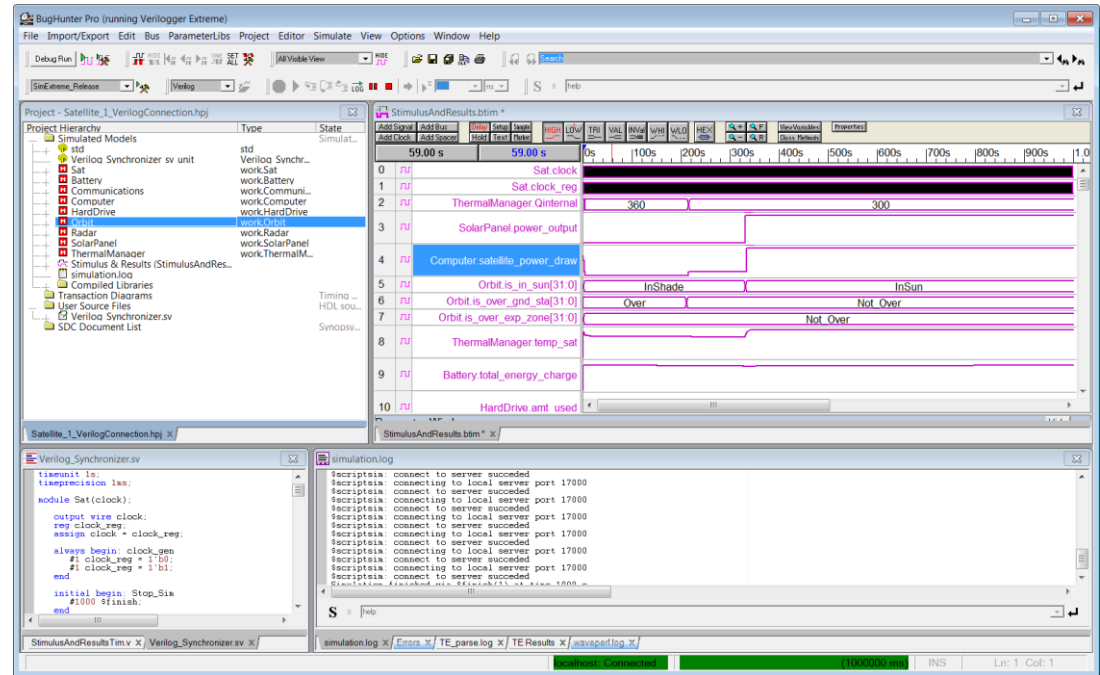
Edit the calculation file for any subsystem in Python.



# 3. System Simulator

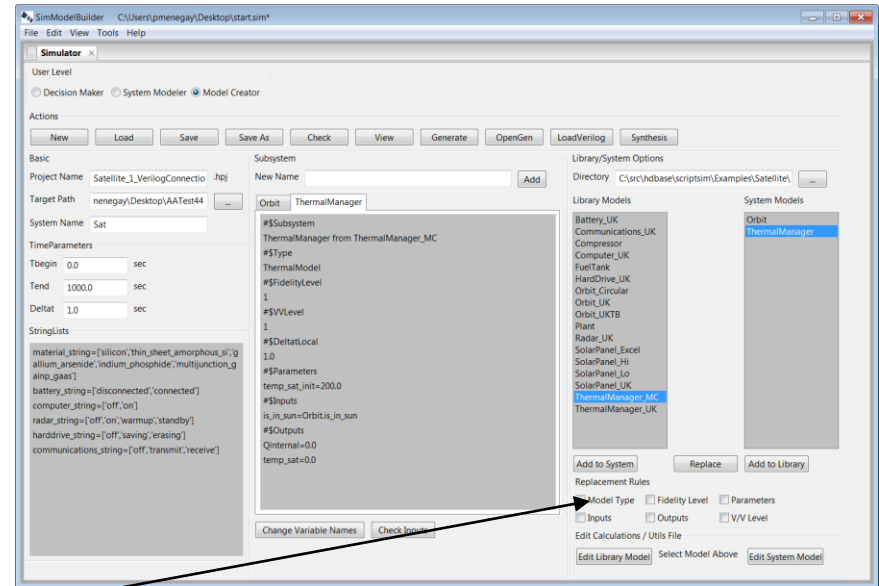
The system simulator combines a high-performance compiled-code SystemVerilog simulator with a Python interpreter to enable engineering level modeling of real world systems.

- Timing diagram with simulation results
- Generated SystemVerilog code.
- Hierarchical view of subsystems and components.
- Full IDE including single step debugging, breakpoints, etc.
- Design browsing & navigation.
- Various output formats



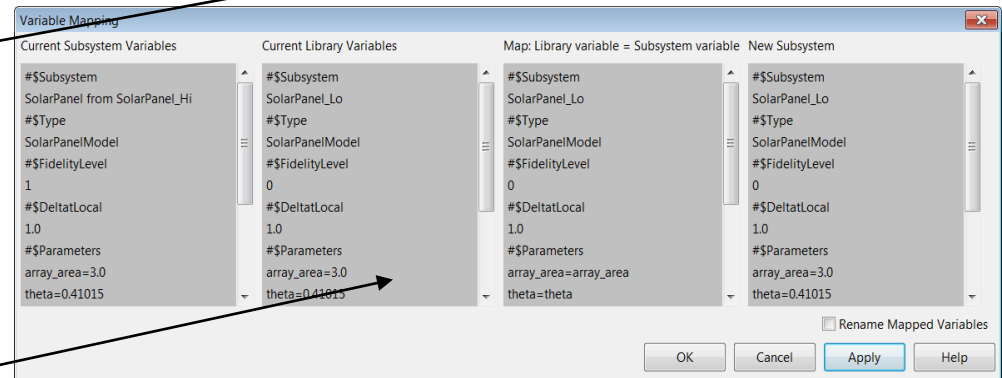
# Multifidelity Modeling

- Models of different fidelity can be switched on the fly.
- As the project advances, the simulation environment remains in place, and maintains connectivity with previous models.



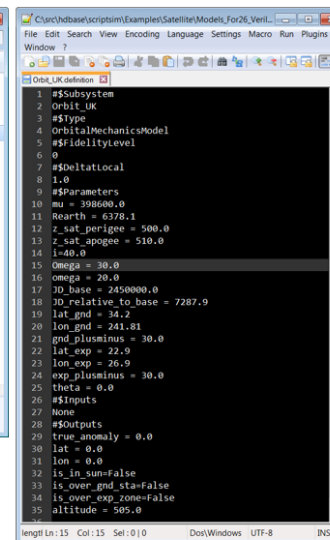
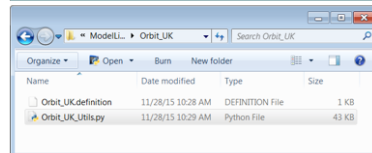
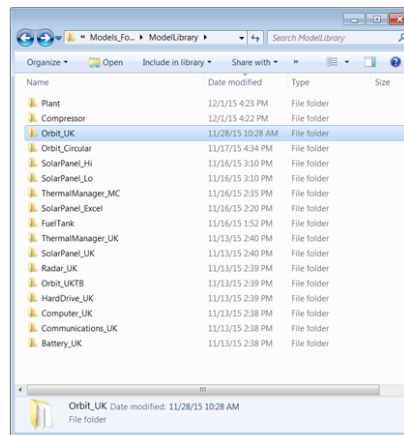
**Replacement rules for switching model fidelity**

**Variable mapping to ensure continuity between models**



# Model Libraries

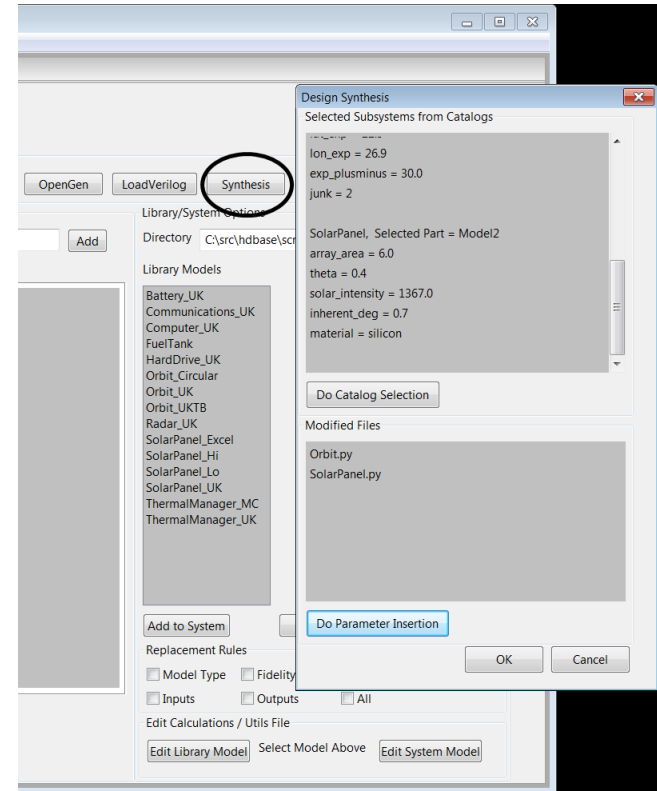
- Subsystem model library.
  - Orbit calculations, solar panel, battery, etc. are publishable and retrievable from library.
- Function library for equation solver.
  - Generated functions can be accessed by System Builder.





# Synthesis

- Compares simulation subsystems with a catalog of parts.
  - User has presumably optimized the subsystem and now wants to select hardware.
  - Software will choose the closest part from catalog and resimulate.



# Satellite Model

Satellite circles the earth in a standard elliptical orbit. It's mission is to collect earth data over an experimental zone and download it to a ground station at another location. It charges a battery in the sun and depletes the battery in the shade. The simulation objective is to understand if the subsystems are sized properly.

##### Orbit #####
PAR mu=398600.0
PAR Rearth=6378.1
PAR z_sat_perigee=500.0
PAR z_sat_apogee=510.0
PAR i=40.0
PAR Omega=30.0
PAR omega=20.0
PAR JD_base=2450000.0
PAR JD_relative_to_base=7287.9
PAR lat_gnd=34.2
PAR lon_gnd=241.81
PAR gnd_plusminus=30.0
PAR lat_exp=22.9
PAR lon_exp=26.9
PAR exp_plusminus=30.0
PAR theta=0.0
INP None
OUT true_anomaly=0.0
OUT lat=0.0
OUT lon=0.0
OUT is_in_sun=False
OUT is_over_gnd_sta=False
OUT is_over_exp_zone=False
OUT altitude=505.0

##### SolarPanel #####
PAR array_area=3.0
PAR theta=0.41015
PAR solar_intensity=1367.0
PAR inherent_deg=0.77
PAR material='silicon',material_string
INP is_in_sun=Orbit.is_in_sun
OUT power_output=0.0

Sat
tbegin=0.0
tend=1000.0
deltat=100.0

##### Computer #####
PAR power_rating=100.0
PAR on_state_init='on',computer_string
INP solar_panel_power_output=SolarPanel.power_output
INP radar_power=Radars.power
INP harddrive_power=HardDrive.power
INP comm_power=Communications.power
INP is_over_gnd_sta=Orbit.is_over_gnd_sta
INP is_over_exp_zone=Orbit.is_over_exp_zone
OUT satellite_power_draw=0.0
OUT on_state='off',computer_string
OUT power=0.0
OUT radar_command='off',radar_string
OUT harddrive_command='off',harddrive_string
OUT communications_command='off',communications_string

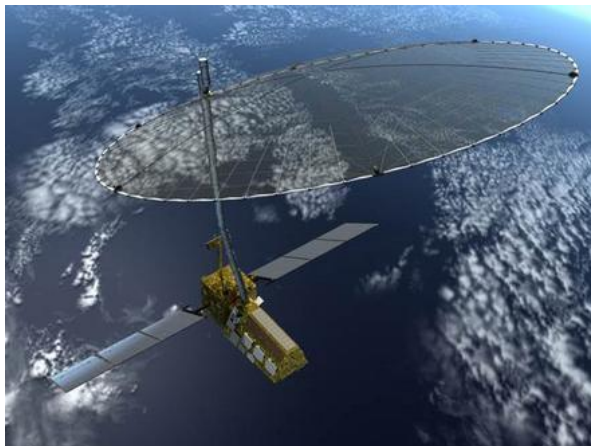
##### Battery #####
PAR capacity_rating=360000.0
PAR voltage_rating=12.0
PAR total_energy_charge_init=2160000.0
PAR con_state='connected',battery_string
INP satellite_power_draw=Computer.satellite_power_draw
OUT total_energy_charge=0.0

##### Radar #####
PAR power_on=300.0
PAR power_standby=200.0
PAR power_warmup=100.0
PAR power_off=0.0
INP command=Computer.radar_command
OUT power_state='off',radar_string
OUT power=0.0

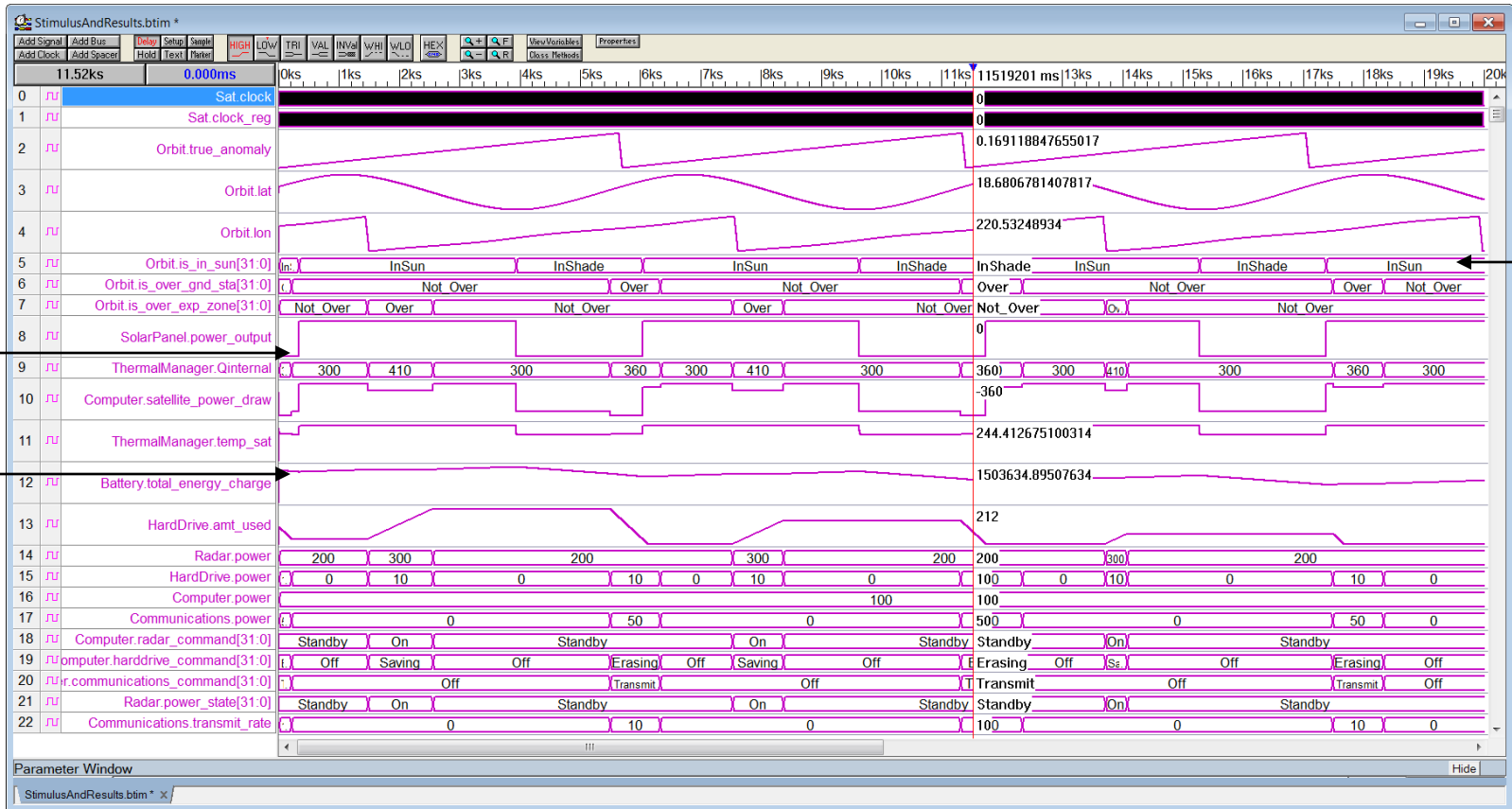
##### HardDrive #####
PAR storage_capacity=650.0
PAR store_speed=0.5
PAR erase_speed=1.0
PAR power_rating=10.0
PAR amt_used_init=300.0
INP command=Computer.harddrive_command
OUT on_state='off',harddrive_string
OUT store_rate=0.0
OUT erase_rate=0.0
OUT power=0.0
OUT amt_used=0.0

##### Communications #####
PAR transmit_speed=10.0
PAR receive_speed=20.0
PAR power_rating=50.0
INP command=Computer.communications_command
OUT on_state='off',communications_string
OUT transmit_rate=0.0
OUT receive_rate=0.0
OUT power=0.0

##### ThermalManager #####
PAR earth_albedo=0.3
PAR solar_intensity=1388.0
PAR stefan_boltzmann_const=5.67e-8
PAR temp_earth=290.0
PAR temp_sun=5800.0
PAR temp_space=2.7
PAR radius_earth=6378.1*1000.0
PAR sat_diameter=1.0
PAR sat_absorptivity=0.7
PAR sat_emissivity=0.9
PAR earth_emissivity=0.5
PAR sat_thermal_capacity=50.0
PAR temp_sat_init=300.0
PAR sat_weight=10.0
INP is_in_sun=Orbit.is_in_sun
INP sat_altitude=Orbit.altitude
INP radar_power=Radars.power
INP computer_power=Computer.power
INP harddrive_power=HardDrive.power
INP comm_power=Communications.power
OUT Qinternal=0.0
OUT temp_sat=0.0



# Results



**Battery slowly drains to 0**

**Solar Panel does not recharge it when exposed to sun  
I.e, the Solar Panel is undersized. Battery is oversized.**



# Results, cont.

- One way to vary the solar panel / battery size is to use constrained randomization.
- Solution was to increase the solar panel area from 3.0 to 4.0 m\*\*2 and decrease the battery capacity from 360,000 to 60,000 amp-sec.

```
Constrain_Batt_SolarPanel.vh
program Constrain_Batt_SolarPanel;
class Constrainer;
rand integer cr_to_aar;
rand integer cr;
rand integer aar;

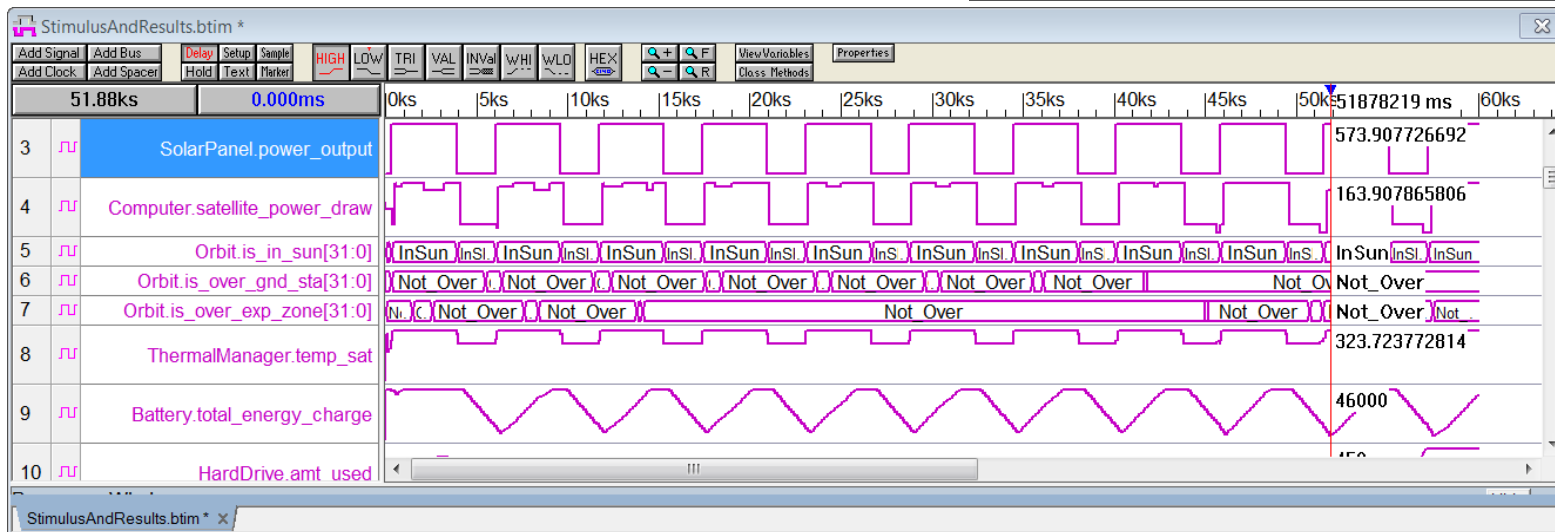
constraint c1 { cr_to_aar < 20000; }
constraint c2 { cr_to_aar > 10000; }

constraint c3 { cr >= 55000; }
constraint c4 { cr <= 200000; }

constraint c5 { aar >= 4; }
constraint c6 { aar <= 10; }

constraint c0 { cr_to_aar == cr / aar; }
endclass

initial
begin
Constrainer obj = new();
int wfile;
wfile = $fopen("c:/users/paenegay/desktop/dae.csv", "w");
$write(wfile, "capacity_rating, array_area \n");
for(int i=0; i<100; i++)
if(obj.randomize())
begin
$write(wfile, "%0d , %0d \n", obj.cr, obj.aar);
$display(" Randomization successful : cr = %0d aar = %0d cr_to_aar = %0d", obj.cr, obj.aar, obj.cr_to_aar);
end
else
$display("Randomization failed");
$display(obj.cr_to_aar);
end
//fclose(wfile);
endprogram
```





# Overall Results

- Once engineering models were made, system integration was fast, 1-2 days for this case.
  - Model libraries were key.
- Provision for multi-fidelity model switching allowed project to remain within a single environment throughout its life.
- Scalability tests on a simple vehicle object lends credence to the SystemVerilog approach.
  - SystemVerilog can simulate up to memory limits of computer. 18 million vehicles for 32-bit and 40 million for 64 bit.
  - SimPy by contrast could simulate 900,000 such objects.
- Runs could be made faster by using event-driven simulation. A 10 fold speed up was achieved this way.
  - Important for long run times over the life of the system.

**We thank the NASA SBIR program for sponsoring this work.  
Contract NNX15CP26P.**

