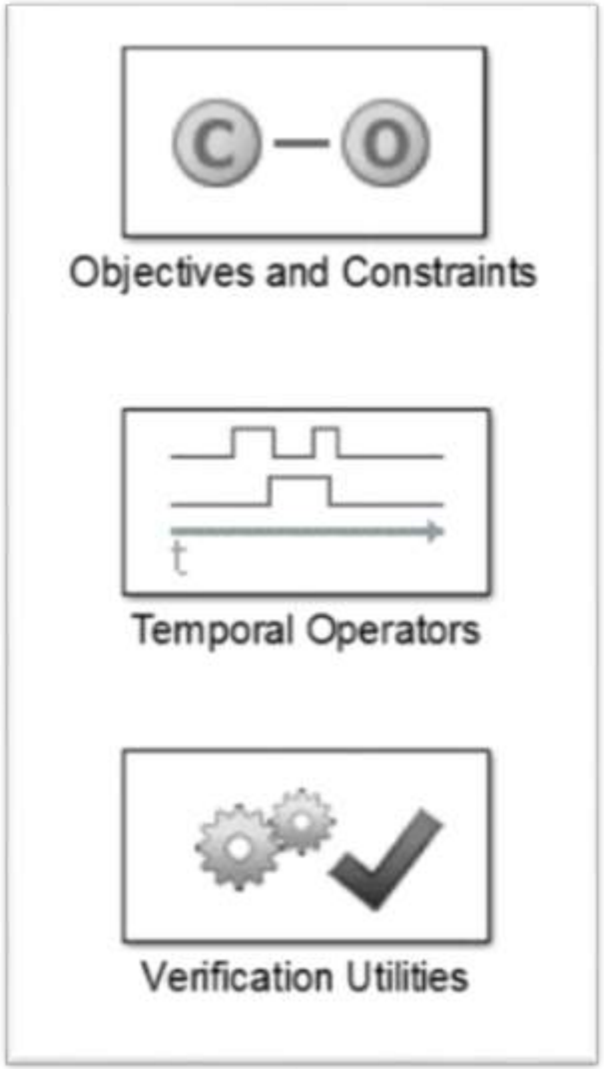


Exploring SLDV - 04

Natasha Y Jeppu

Yogananda Jeppu

Voter Logic

- A voter does not avoid the occurrence of sensor faults. It detects them and prevents their propagation to the systems that employ the voted signals like a safety critical controller.
- There are many types of voting logic (refer to the taxonomy of voters)
- Flight control systems use voting logic. A standard method is to look at signals and their validity. Compare two valid signals against a threshold. If they miscompare flag them. Do this for other sensors. This gives an indication of erroneous sensor.
- A 3 sensor voter is formally validated here using SLDV. It is in lines with a Arduino code available on the net.

http://forum.arduino.cc/index.php?topic=106354.0

Voter Logic Description

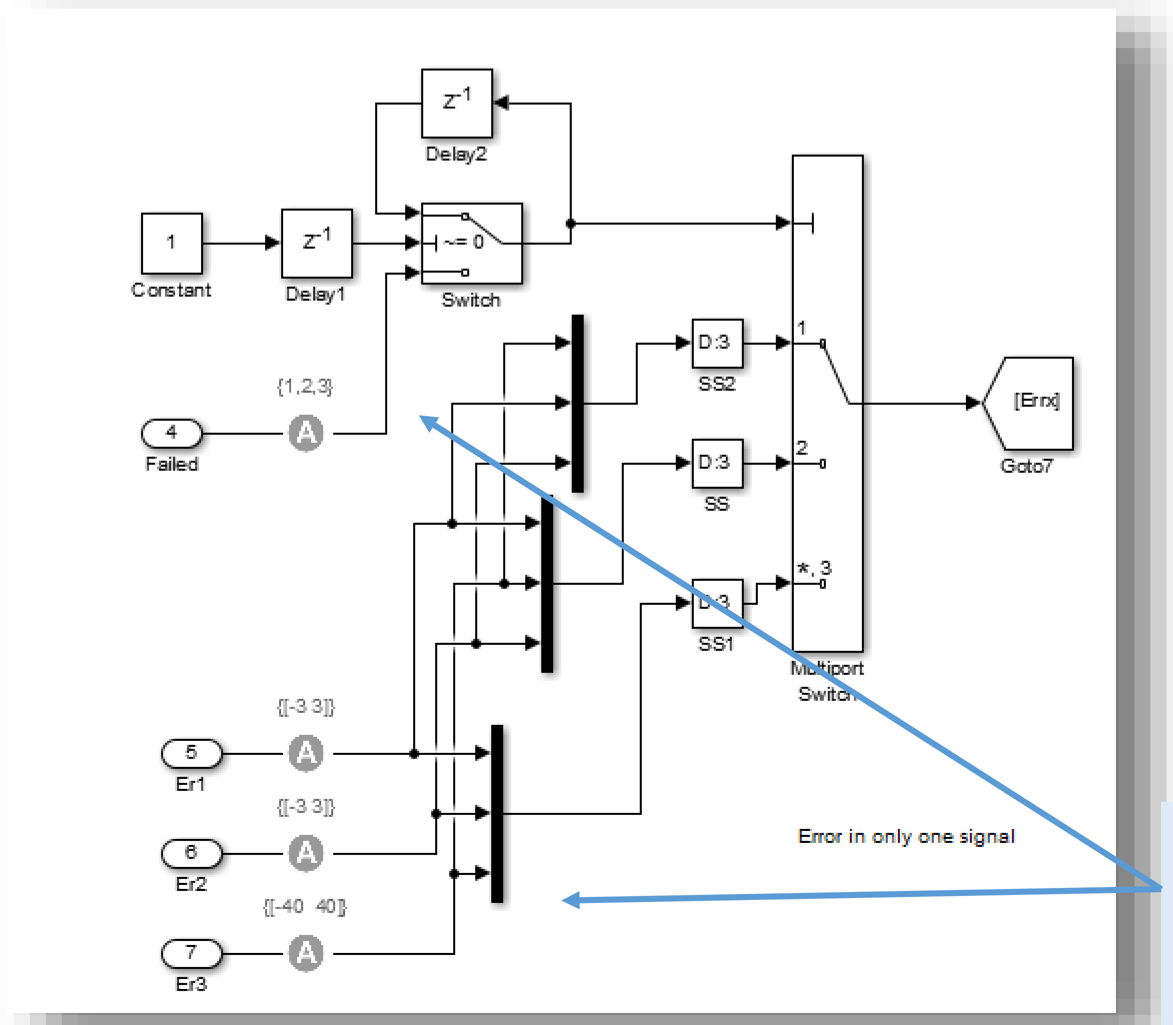
- There are 3 sensors A, B and C. Their validities are AV, BV and CV respectively.
- Sensor combination A and B is valid if
  - AV AND BV AND (abs(A-B) <= Threshold) (similarly for others AC and BC)
- All three sensor pairs are thus validated and a three input truth table is created for AB, AC and BC.
- If any signal is valid select that as output. Is any sensor is failed average the other two. If all sensor fail hold the previous value and if all pass then average all three.

```
% AB   AC   BC
%  0    0    0      (0) All fail
%  0    0    1      (1) A fail
%  0    1    0      (2) B fail
%  0    1    1      (3) C valid
%  1    0    0      (4) C fail
%  1    0    1      (5) B valid
%  1    1    0      (6) A valid
%  1    1    1      (7) All valid
```

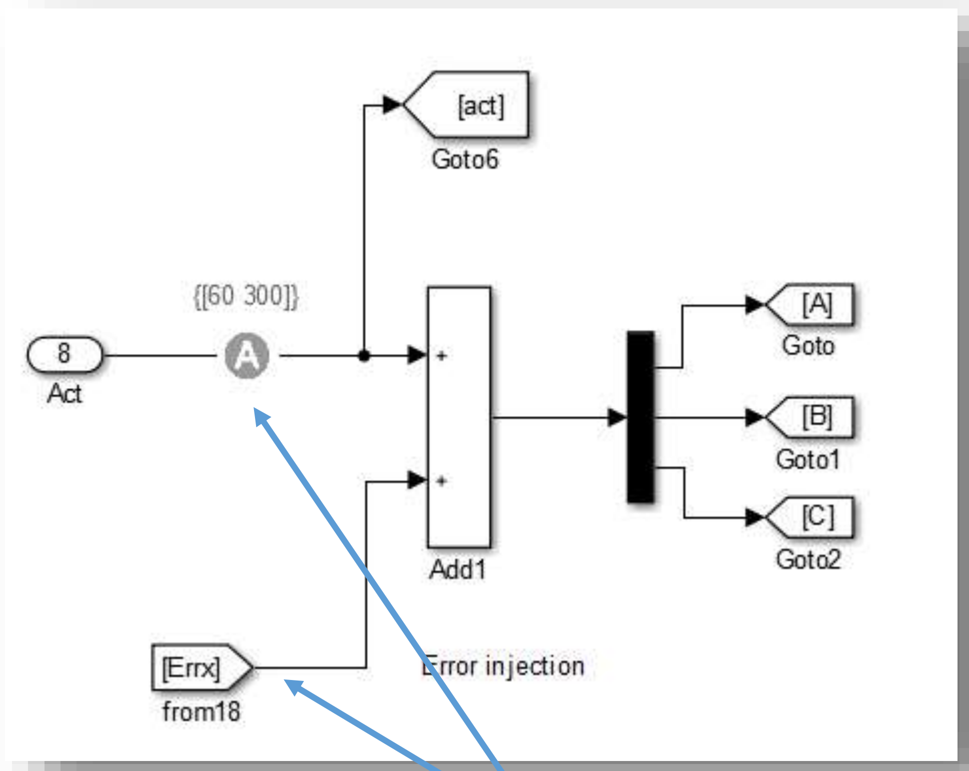
Voter Assumption

- Sensor A, B and C measure a physical parameter (perhaps Calibrated Airspeed) which can vary from 60 to 300 (knots).
- Each sensor has a random noise component with a maximum value of 3. We need to pass this as it is and not trip a sensor for this.
- Only one sensor fails with a large transient. This is to be detected and isolated.
- If any two sensor are valid the voted value shall be very close to the actual – an error of ±6 is tolerated. The voted output shall be declared as valid.
- Let us prove this to be correct in our design using Simulink Design Verifier

Implementing the assumption



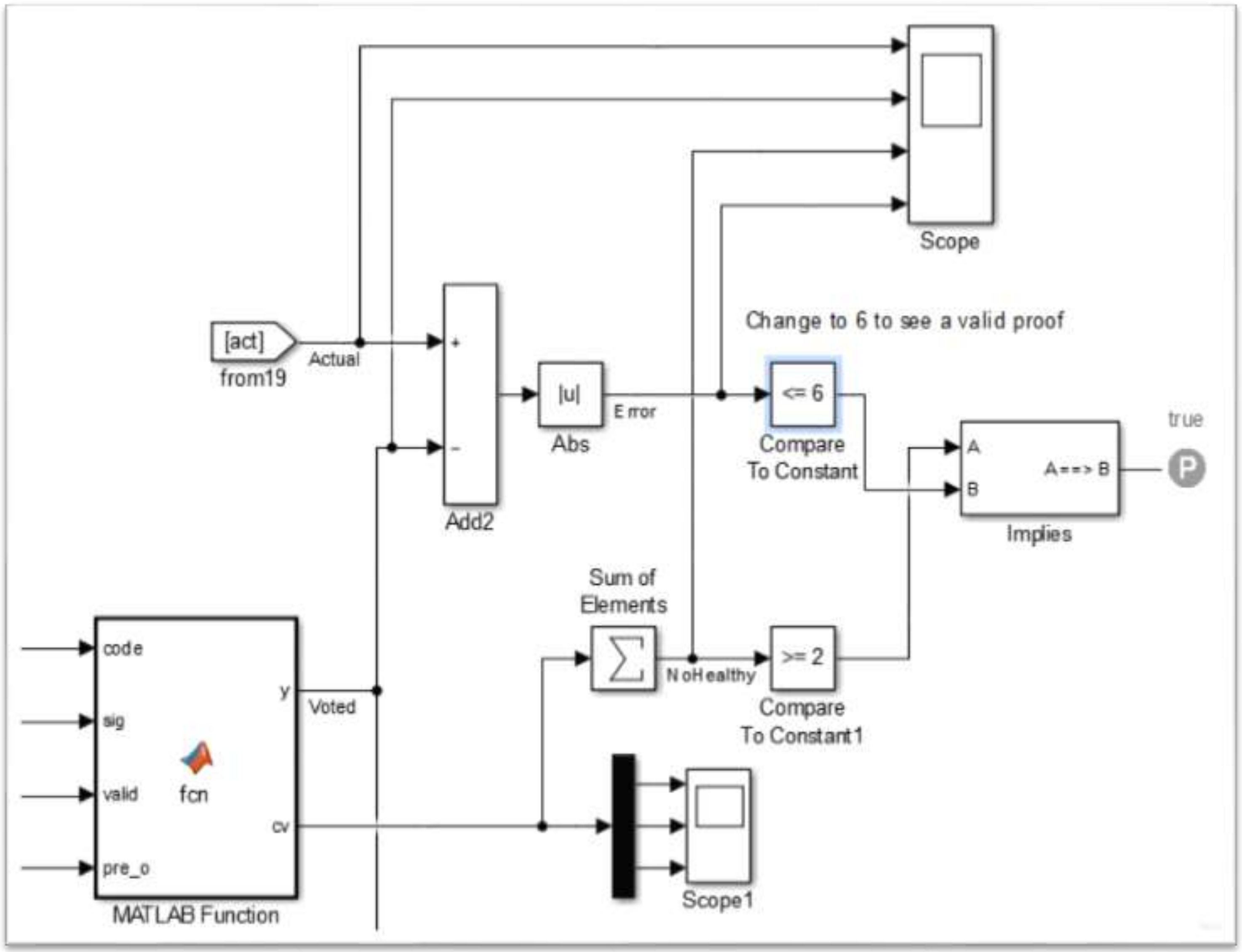
Errors are assumed and based on "Failed" signal. Only one sensor is failed at a time



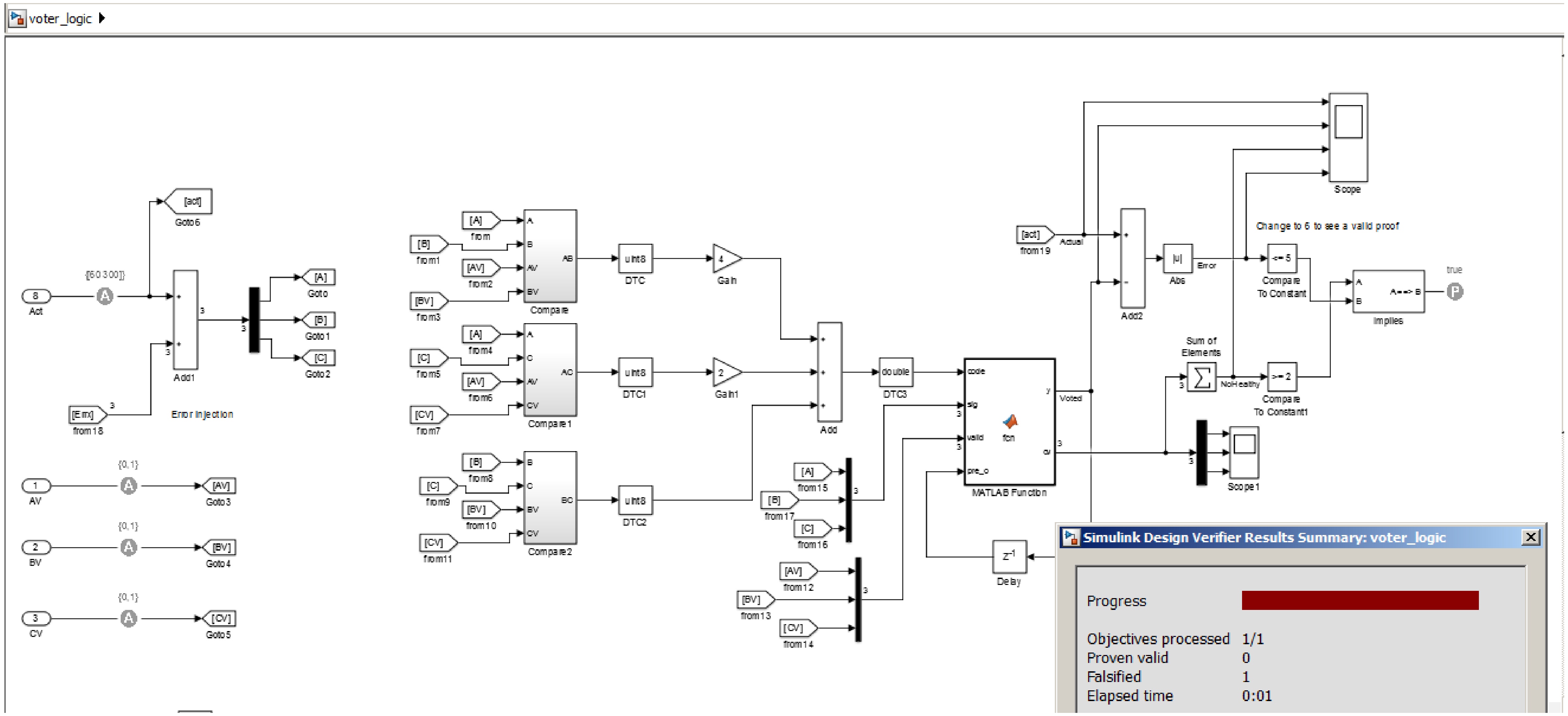
An actual signal is assumed with assumption block A {60, 300}. Error is added to this.

SLDV Assertion

- If sum(cumulative valid signals) >=2 IMPLIES that abs(Voted – Actual) <= 6.
- This is proved valid for the design



Voter proved false for <= 5

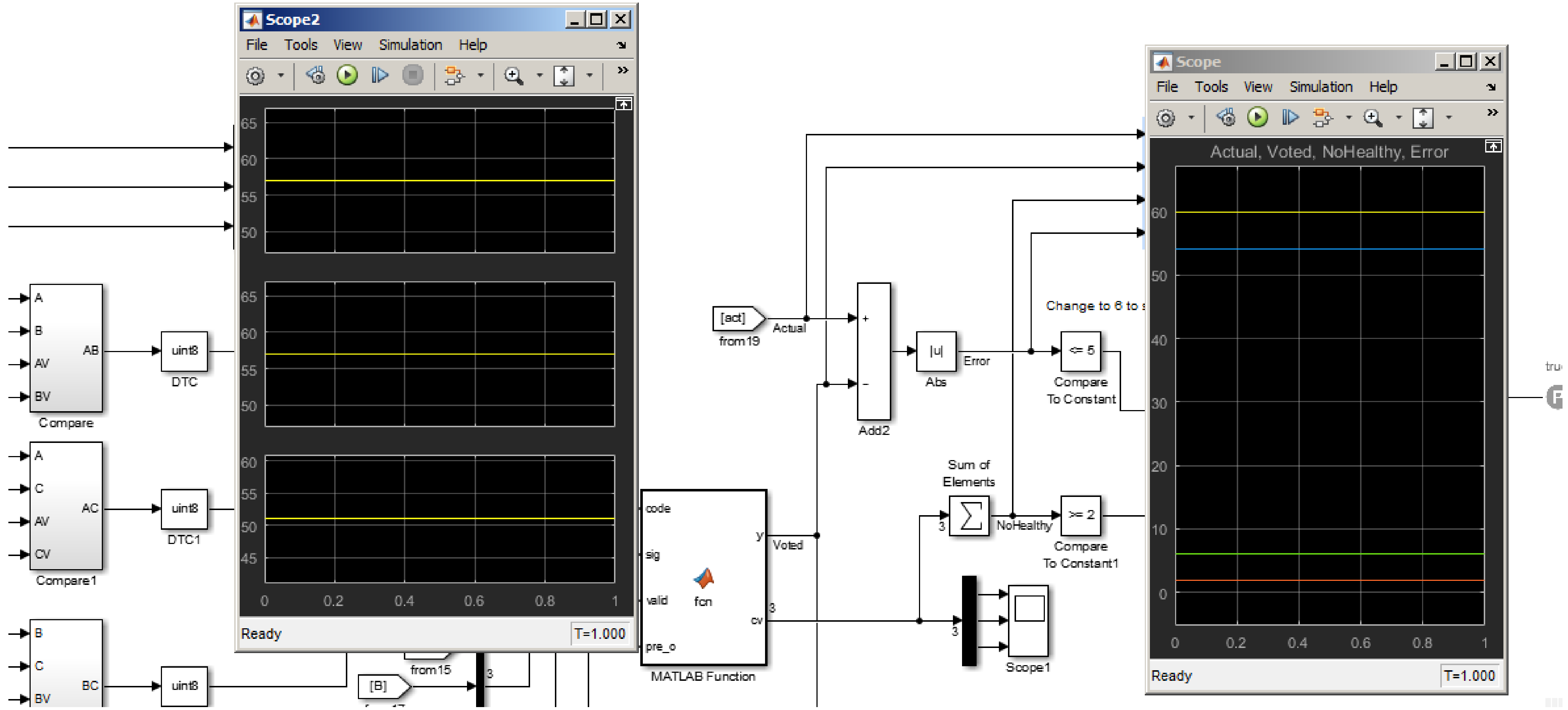


Simulink Design Verifier Results Summary: voter_logic

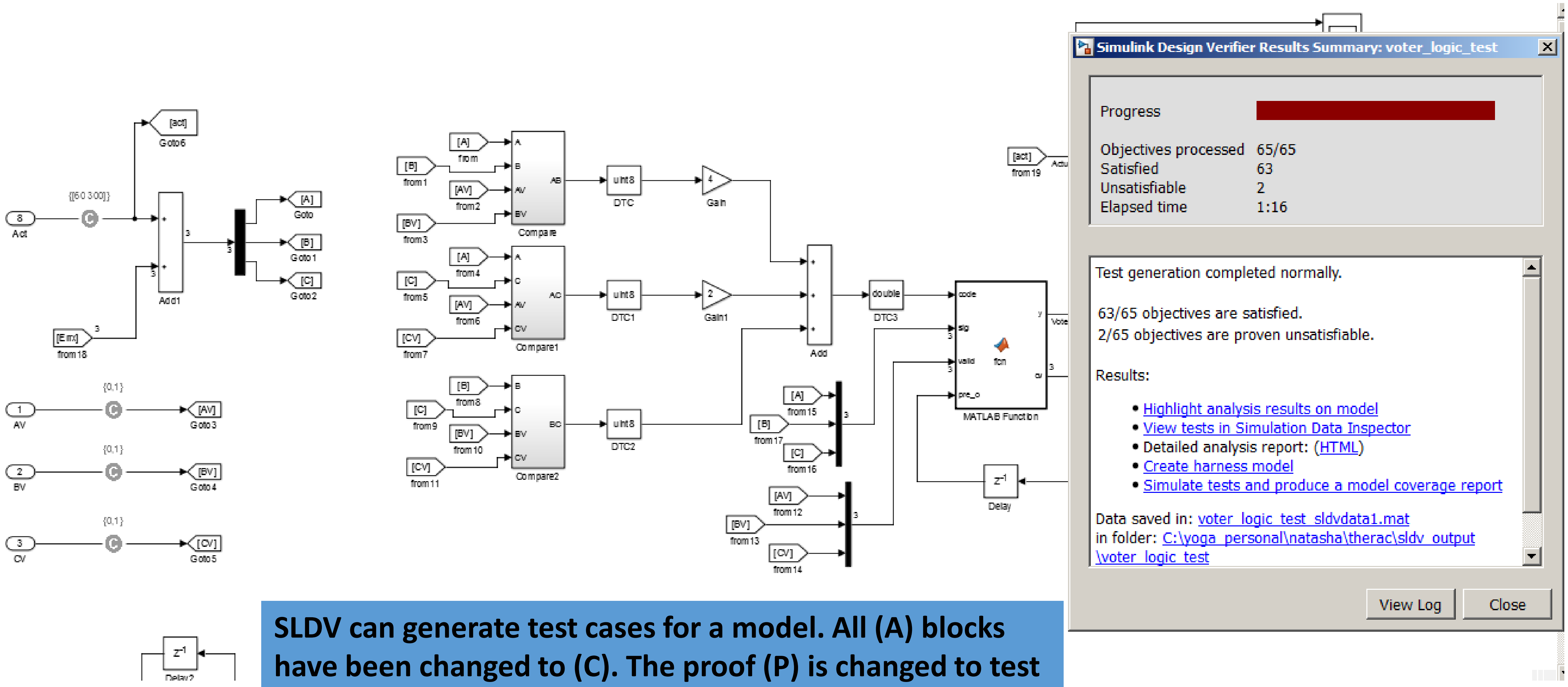
Progress

Objectives processed 1/1
Proven valid 0
Falsified 1
Elapsed time 0:01

Counterexample



Voter Logic Test Case Generation



Simulink Design Verifier Results Summary: voter_logic_test

Progress

Objectives processed 65/65
Satisfied 63
Unsatisfiable 2
Elapsed time 1:16

Test generation completed normally.

63/65 objectives are satisfied.
2/65 objectives are proven unsatisfiable.

Results:

- Highlight analysis results on model
- View tests in Simulation Data Inspector
- Detailed analysis report: (HTML)
- Create harness model
- Simulate tests and produce a model coverage report

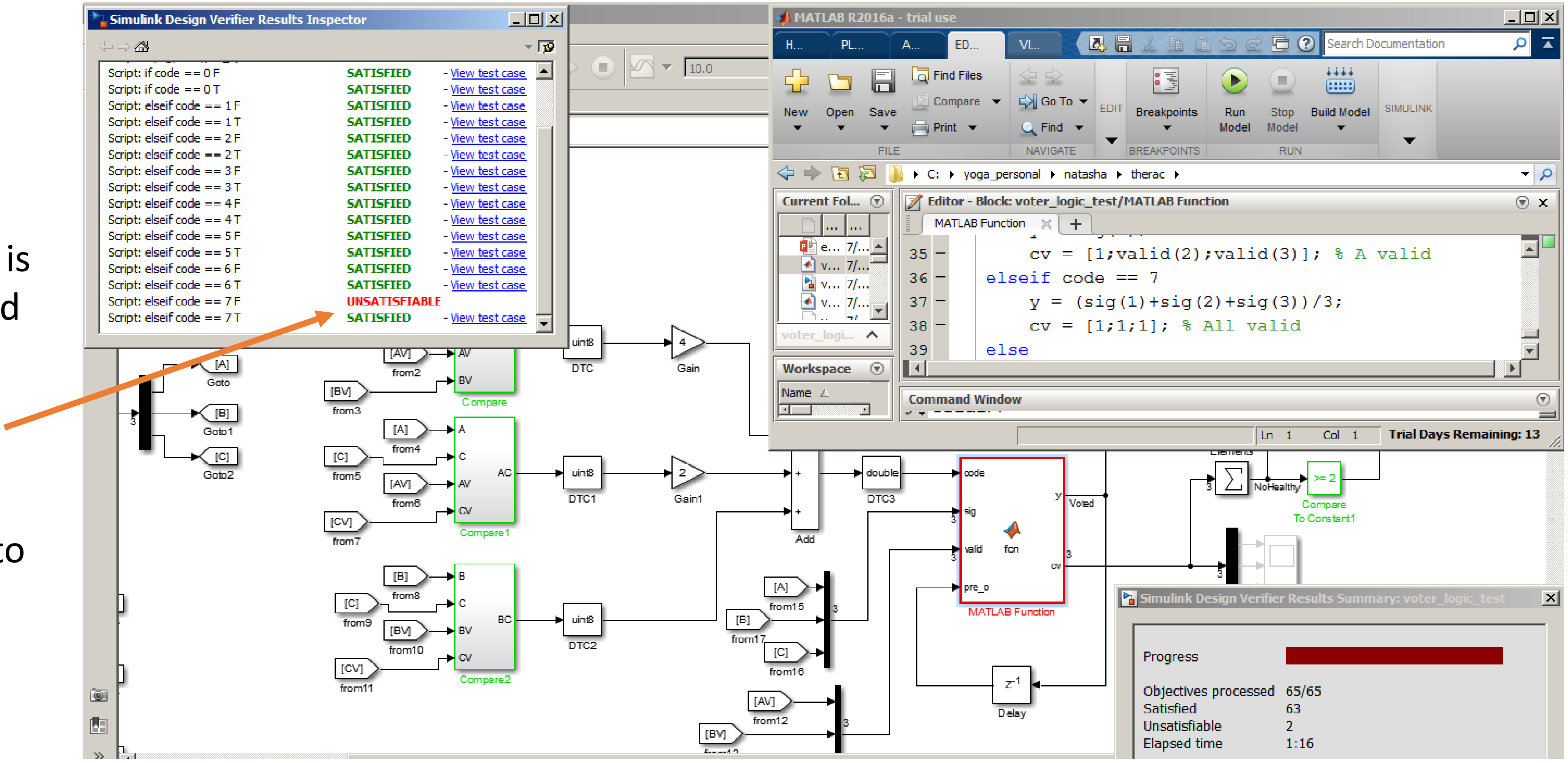
Data saved in: voter_logic_test_sldvdata1.mat
in folder: C:\yoga_personal\natasha\therac\sldv_output\voter_logic_test

View Log Close

SLDV can generate test cases for a model. All (A) blocks have been changed to (C). The proof (P) is changed to test objective (O)

Indicates unsatisfiable condition

The test case is generated and the report indicates the uncovered zone. This is not possible to cover.


Simulink Design Verifier Results Inspector
MATLAB R2016a - trial use
Editor - Block: voter_logic_test/MATLAB Function
MATLAB Function
Command Window
Trial Days Remaining: 13
Simulink Design Verifier Results Summary: voter_logic_test

Script: if code == 0 F	SATISFIED	- View test case
Script: if code == 0 T	SATISFIED	- View test case
Script: elseif code == 1 F	SATISFIED	- View test case
Script: elseif code == 1 T	SATISFIED	- View test case
Script: elseif code == 2 F	SATISFIED	- View test case
Script: elseif code == 2 T	SATISFIED	- View test case
Script: elseif code == 3 F	SATISFIED	- View test case
Script: elseif code == 3 T	SATISFIED	- View test case
Script: elseif code == 4 F	SATISFIED	- View test case
Script: elseif code == 4 T	SATISFIED	- View test case
Script: elseif code == 5 F	SATISFIED	- View test case
Script: elseif code == 5 T	SATISFIED	- View test case
Script: elseif code == 6 F	SATISFIED	- View test case
Script: elseif code == 6 T	SATISFIED	- View test case
Script: elseif code == 7 F	UNSATISFIABLE	
Script: elseif code == 7 T	SATISFIED	- View test case

```
        cv = [1;valid(2);valid(3)]; % A valid
    elseif code == 7
        y = (sig(1)+sig(2)+sig(3))/3;
        cv = [1;1;1]; % All valid
    else
```

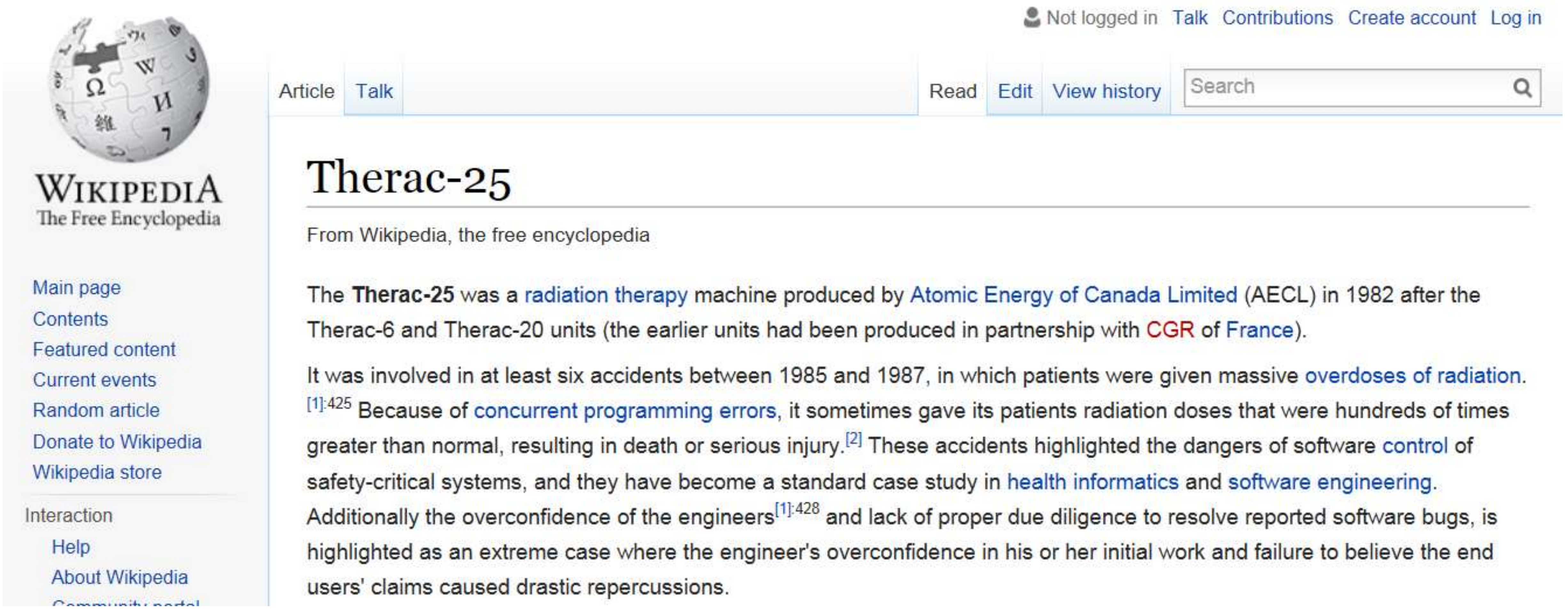
```
Progress
Objectives processed  65/65
Satisfied             63
Unsatisfiable         2
Elapsed time          1:16
```

Therac - 25

Revisited as SLDV

http://malomaal.com/technology/10-modern-radiation-accidents-involving-civilians/

Therac - 25


Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia

Therac-25

From Wikipedia, the free encyclopedia

The **Therac-25** was a radiation therapy machine produced by Atomic Energy of Canada Limited (AECL) in 1982 after the Therac-6 and Therac-20 units (the earlier units had been produced in partnership with CGR of France).

It was involved in at least six accidents between 1985 and 1987, in which patients were given massive overdoses of radiation.[1]:425 Because of concurrent programming errors, it sometimes gave its patients radiation doses that were hundreds of times greater than normal, resulting in death or serious injury.[2] These accidents highlighted the dangers of software control of safety-critical systems, and they have become a standard case study in health informatics and software engineering. Additionally the overconfidence of the engineers[1]:428 and lack of proper due diligence to resolve reported software bugs, is highlighted as an extreme case where the engineer's overconfidence in his or her initial work and failure to believe the end users' claims caused drastic repercussions.

Death by Software

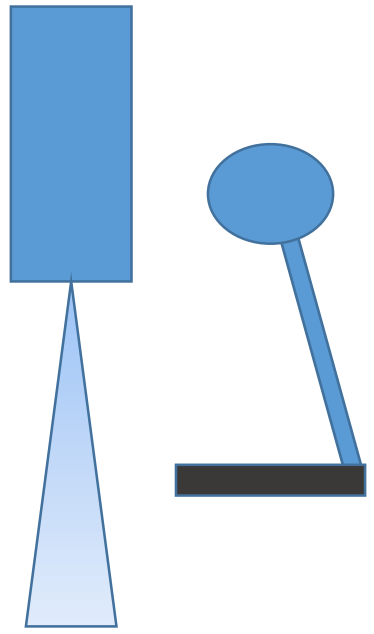
- Due to a programming error the radiation therapy machine overdosed patients and caused death is some cases. (refer Wiki and Nancy Leveson’s paper for details)
- A patient could be administered with a low intensity electron beam or with an X-ray therapy. The X-ray was generated by the same machine by bombarding a target with high intensity electron beam.
- The software error could bombard a patient with high intensity beam when commanded even if the target was not in place.
- The earlier mechanical system had interlocks that was removed in the Therac 25 because software was “so much better”

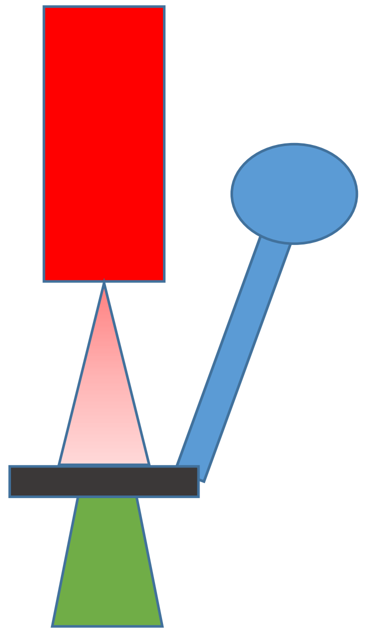
The Problem

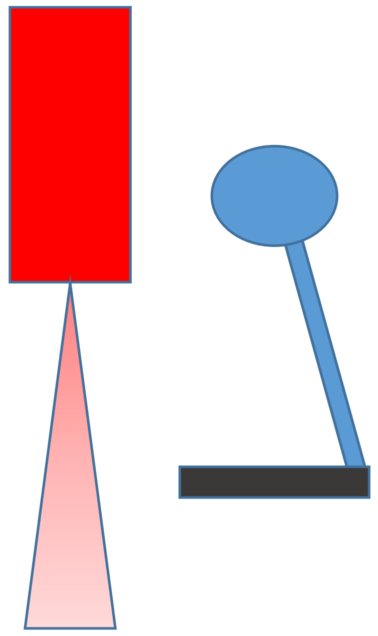
Direct beam low doses of high-energy (5 MeV to 25 MeV)

X-rays - high-energy (25 MeV) electrons into a "target"

A high beam without the “target” caused the radiation deaths







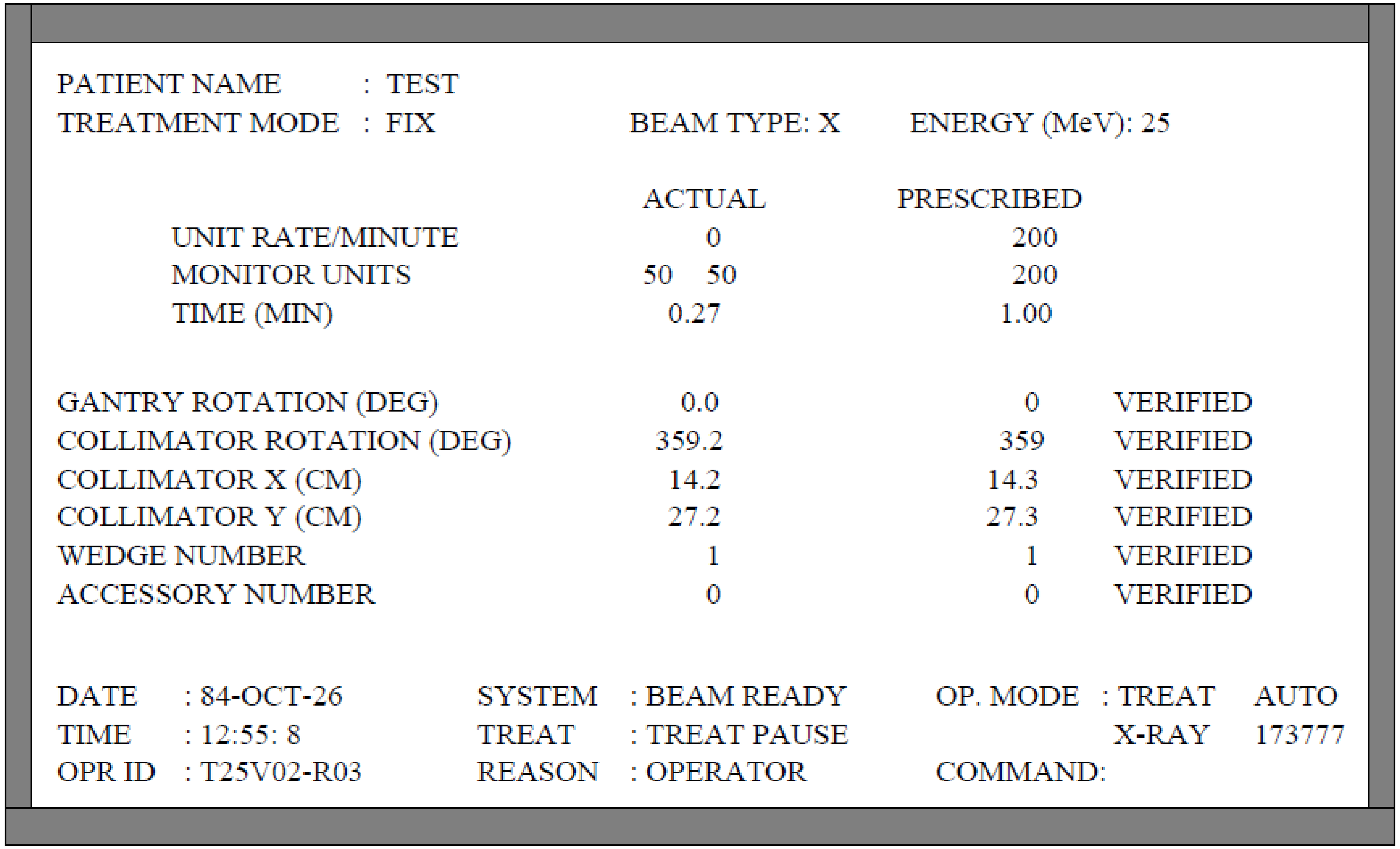
http://radonc.wikidot.com/radiation-accident-therac25

Therac – 25 Screen

The dosage and the X-ray or Electron Beam was selected by a DEC terminal

Medical Devices: The Therac-25*

Nancy Leveson
University of Washington



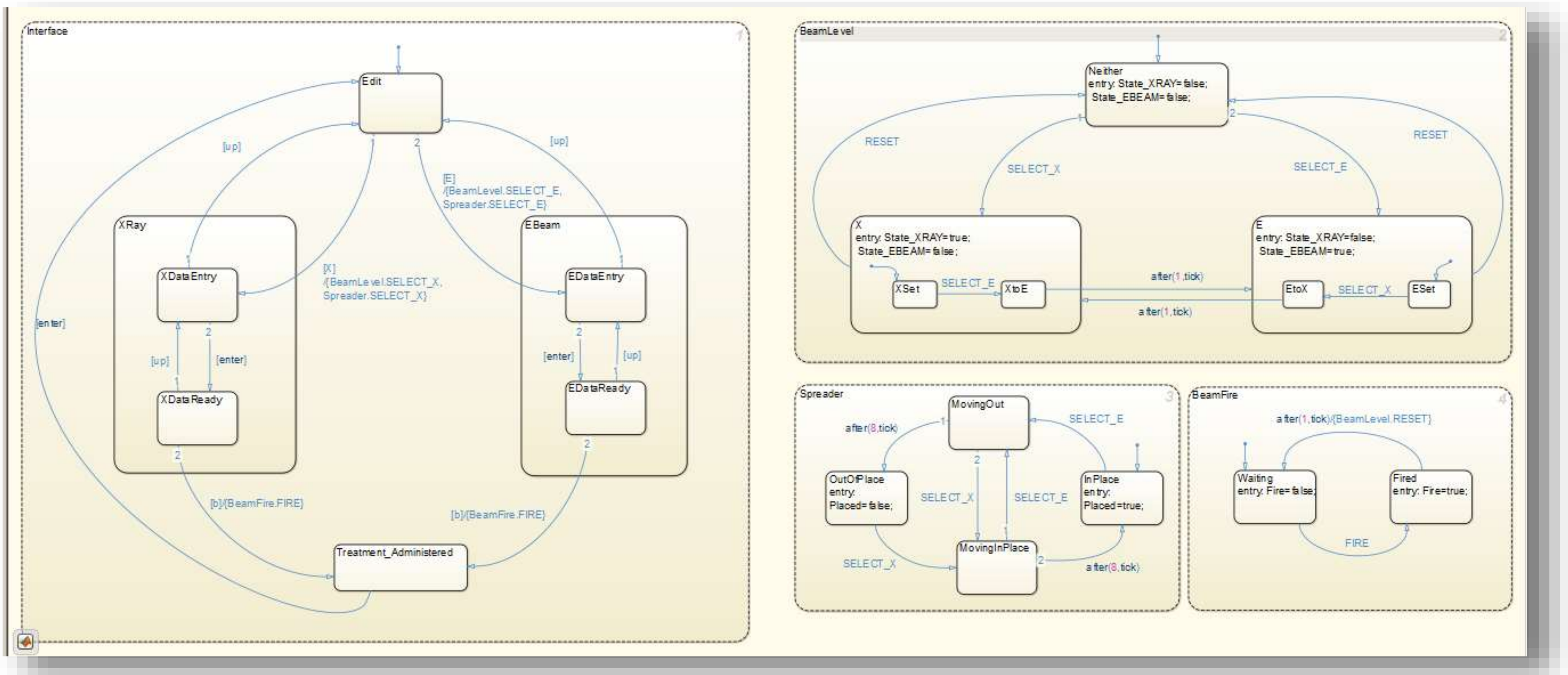
```
PATIENT NAME        :  TEST
TREATMENT MODE      :  FIX             BEAM TYPE: X      ENERGY (MeV): 25

                                        ACTUAL           PRESCRIBED
          UNIT RATE/MINUTE                 0                200
          MONITOR UNITS                 50    50            200
          TIME (MIN)                      0.27              1.00

GANTRY ROTATION (DEG)                    0.0                  0     VERIFIED
COLLIMATOR ROTATION (DEG)              359.2                359     VERIFIED
COLLIMATOR X (CM)                       14.2               14.3     VERIFIED
COLLIMATOR Y (CM)                       27.2               27.3     VERIFIED
WEDGE NUMBER                               1                  1     VERIFIED
ACCESSORY NUMBER                           0                  0     VERIFIED

DATE     : 84-OCT-26      SYSTEM   : BEAM READY      OP. MODE  : TREAT    AUTO
TIME     : 12:55: 8       TREAT    : TREAT PAUSE                 X-RAY    173777
OPR ID   : T25V02-R03     REASON   : OPERATOR        COMMAND:
```

Algorithm (simplified)

- The operator (hospital employee) could set the parameters using the keyboard and scroll through the different field using the up keys. Enter was used to confirm a value and move down to lower field.
- The operator could select E or X for Electron or X ray beam.
- The operator could radiate the patient with the B (fire) key
- Selecting the E or X set the intensity levels of the machine
- Selecting X moved the target “In Place” and selecting E moved the target “Out of Place”. This took about 8 seconds.

The operator could fire the beam from the keyboard if he was ready. There was no interlock for “Fire” with the “In Place”. This was a major problem.

Stateflow model



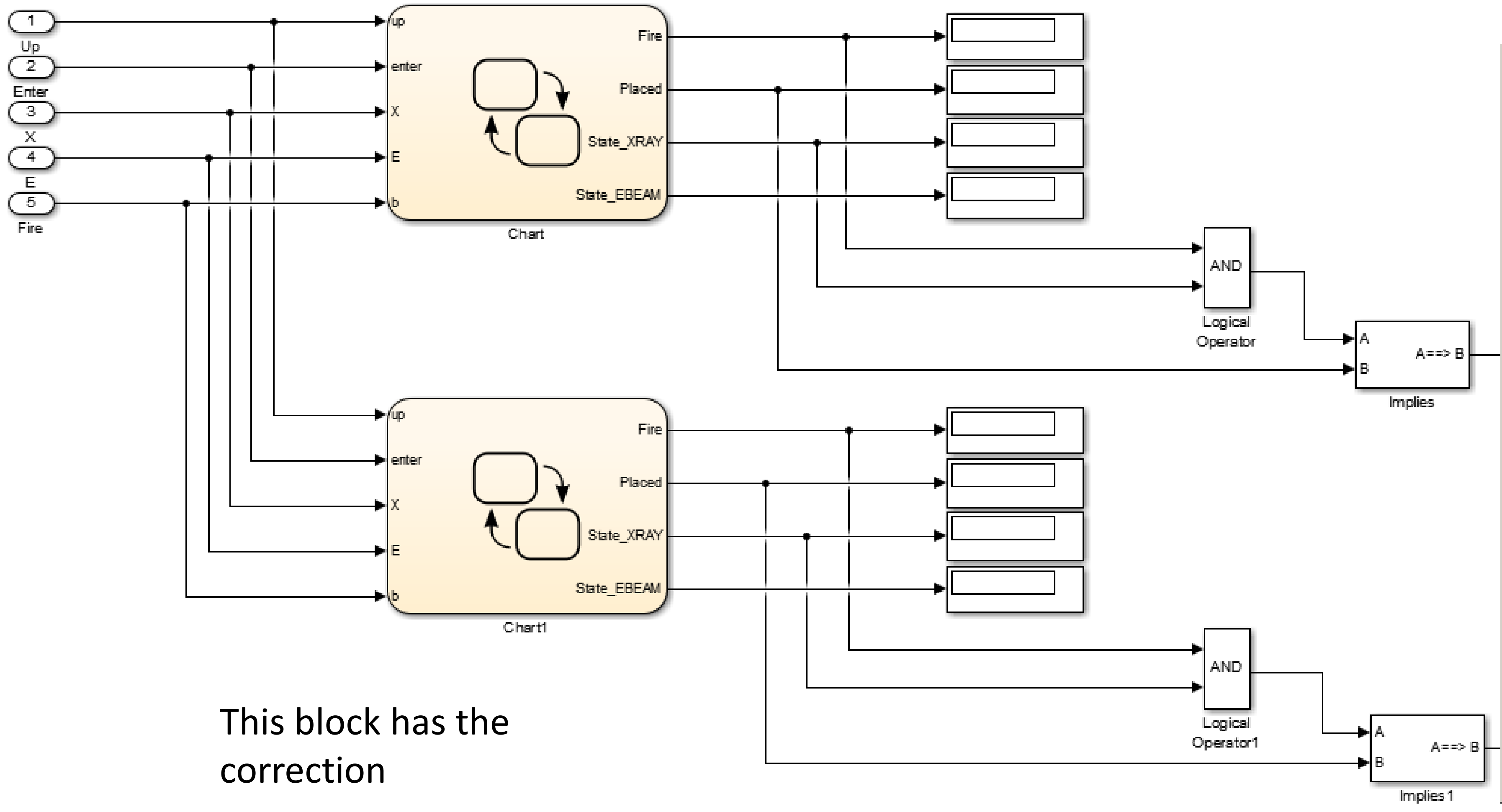
Stateflow

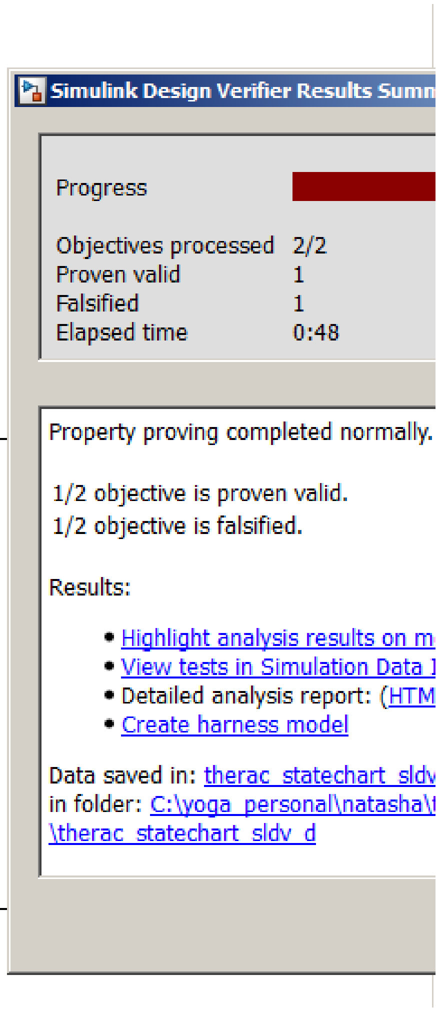
- The interface State has the initial sub state as Edit. X or E takes it to X data entry or E data entry.
- Up arrow takes it back to edit and Enter takes it to X data ready or E data ready. From this sub state “B” sends it to Treatment Administered
- In the Beam Level State the initial sub state is Neither. From here based on E or X it transits to X beam or E beam. If in X if E is pressed a temporary sub state X to E is entered. Similarly from E, E to X when X is pressed. This happens in the next cycle of computation.
- A reset signals resets the Beam Level back to Neither.

Stateflow

- The Spreader State is the mechanical device. Initially it is “In Place”. An E makes it transit to Moving Out. After 8 seconds it transits to Out Of Place. Similarly, an X makes it move back to In Place with a intermediate state Moving In.
- The Beam Fire State has two sub states Fire and Waiting. Initially it is in Waiting and Transits to Fire when B is pressed.
- Once fired, it comes back to Waiting in the next execution frame and sends the reset signal also in the process.

Therac-25 Stateflow proof




Simulink Design Verifier Results Summ

Progress

Objectives processed 2/2
Proven valid 1
Falsified 1
Elapsed time 0:48

Property proving completed normally.

1/2 objective is proven valid.
1/2 objective is falsified.

Results:

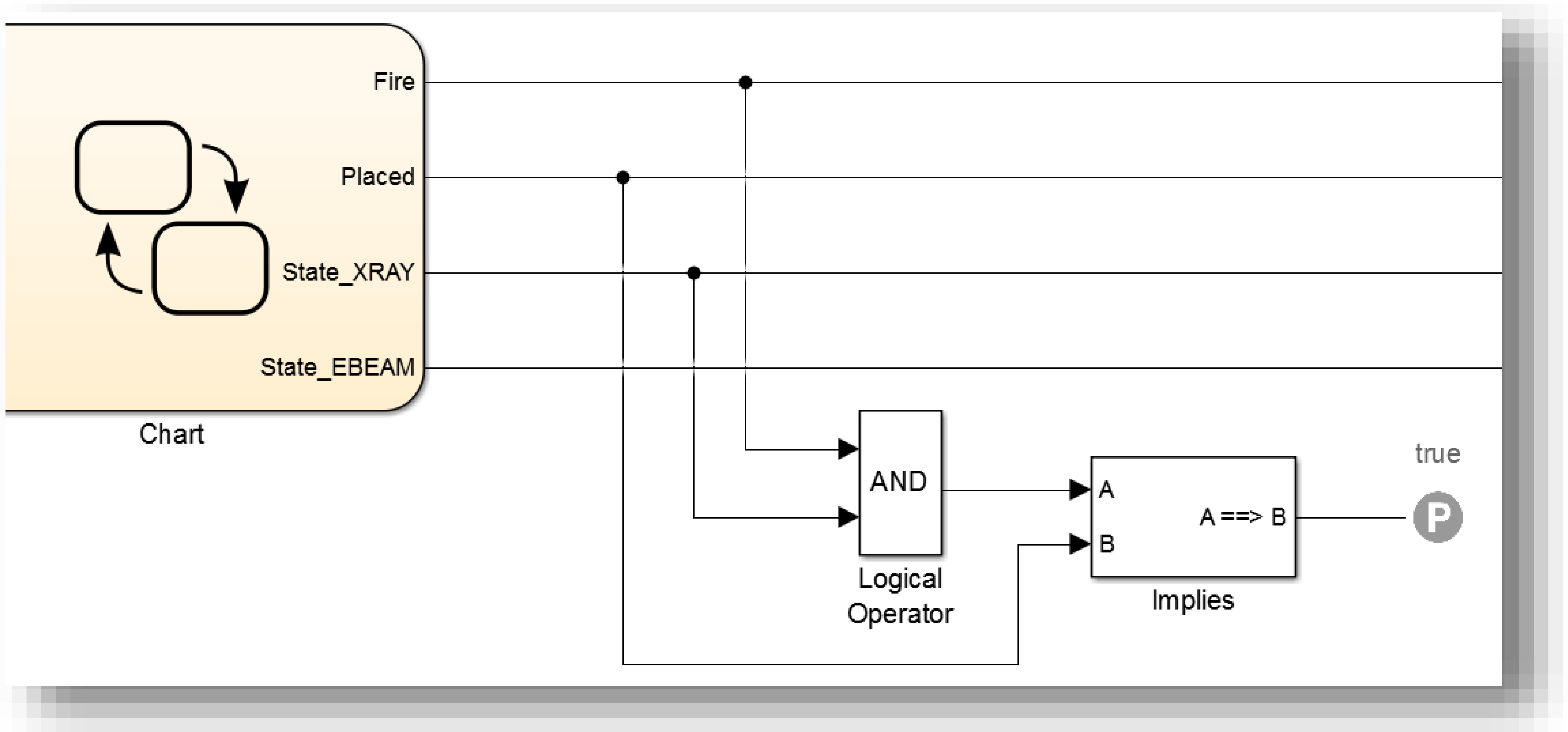
- Highlight analysis results on m
- View tests in Simulation Data I
- Detailed analysis report: (HTM
- Create harness model

Data saved in: therac_statechart_sldv
in folder: C:\yoga_personal\natasha\t
\therac_statechart_sldv_d

This block has the correction

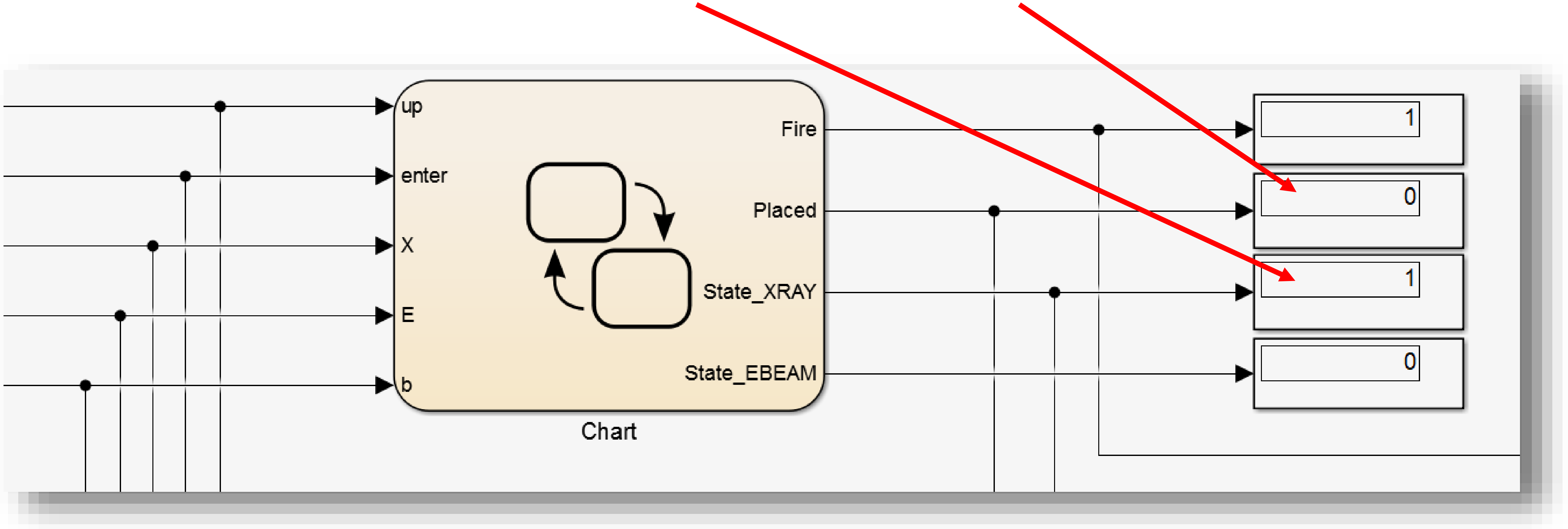
Assertion

- The assertion is very simple
- When the Beam Level is X ray and Beam is “Fired” IMPLIES that the Spreader is “In Place”



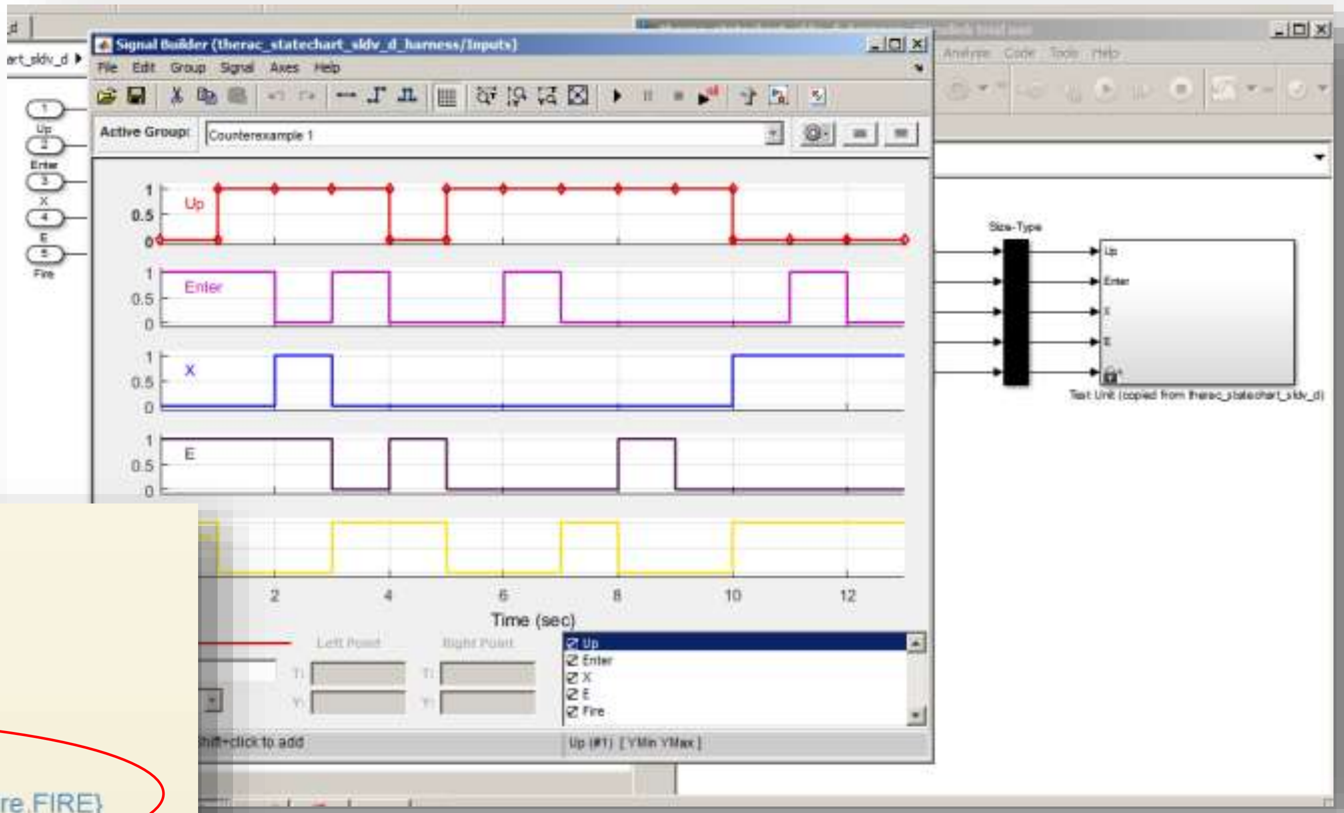
The error

Fire = 1, State_XRAY = 1 but Placed = 0 (False)



Counterexample

There are two stateflow models. The bottom one has a correction to the problem. This is so simple – ensure "InPlace" when Firing.


Signal Builder (therac_statechart_sldv_d_harness/Inputs)
File Edit Group Signal Axes Help
Active Group: Counterexample 1
Up
Enter
X
E
Time (sec)
Up
Enter
X
E
Fire
Size-Type
Test Unit (copied from therac_statechart_sldv_d)

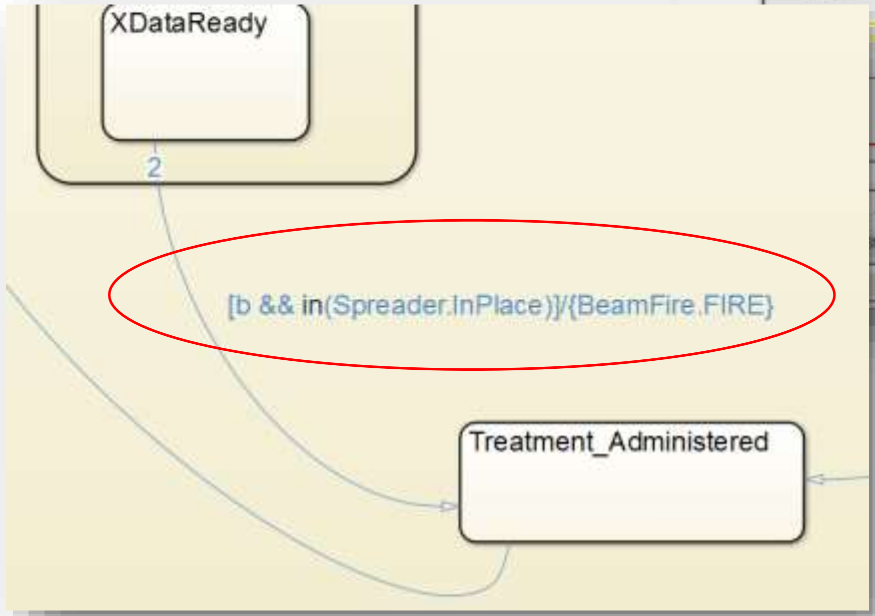
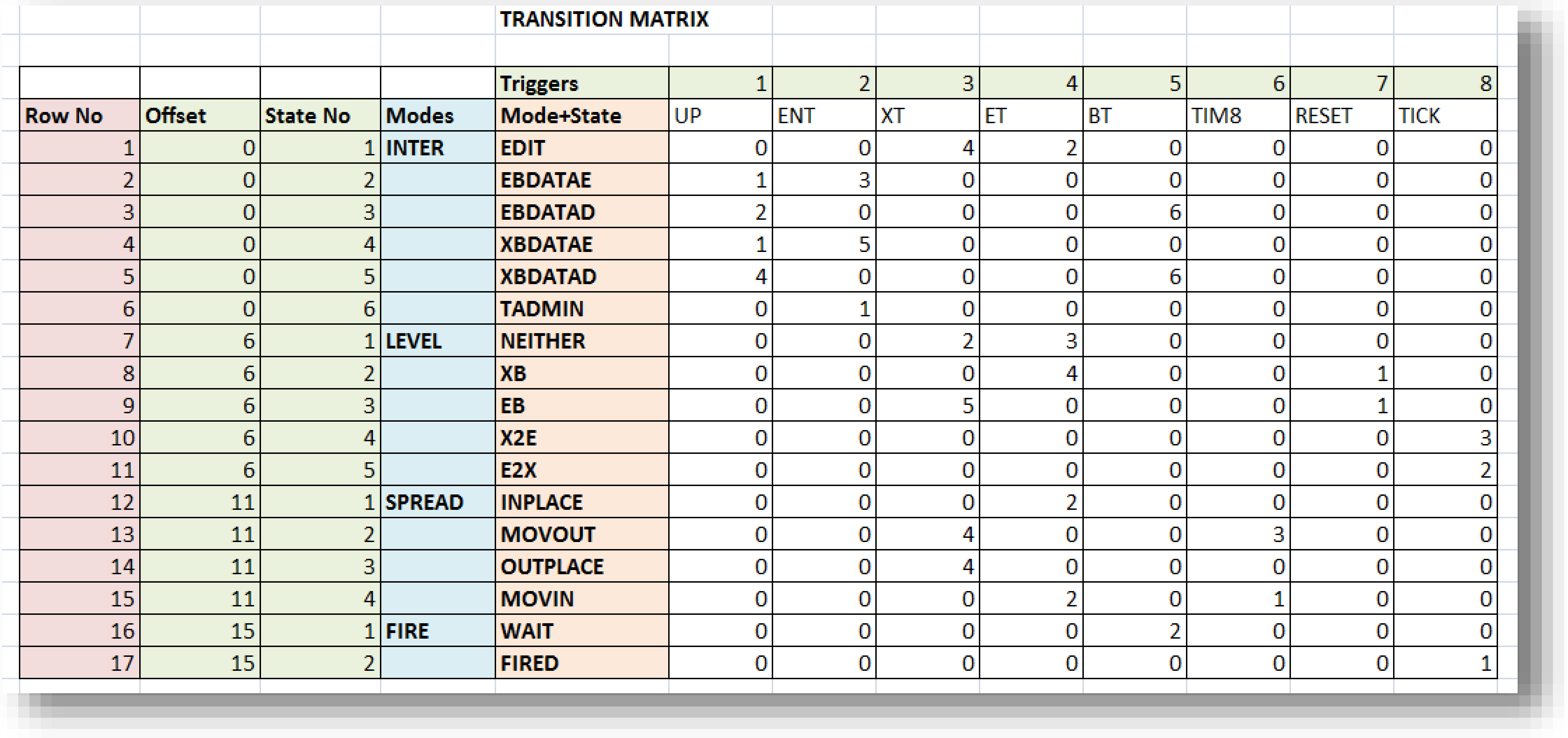


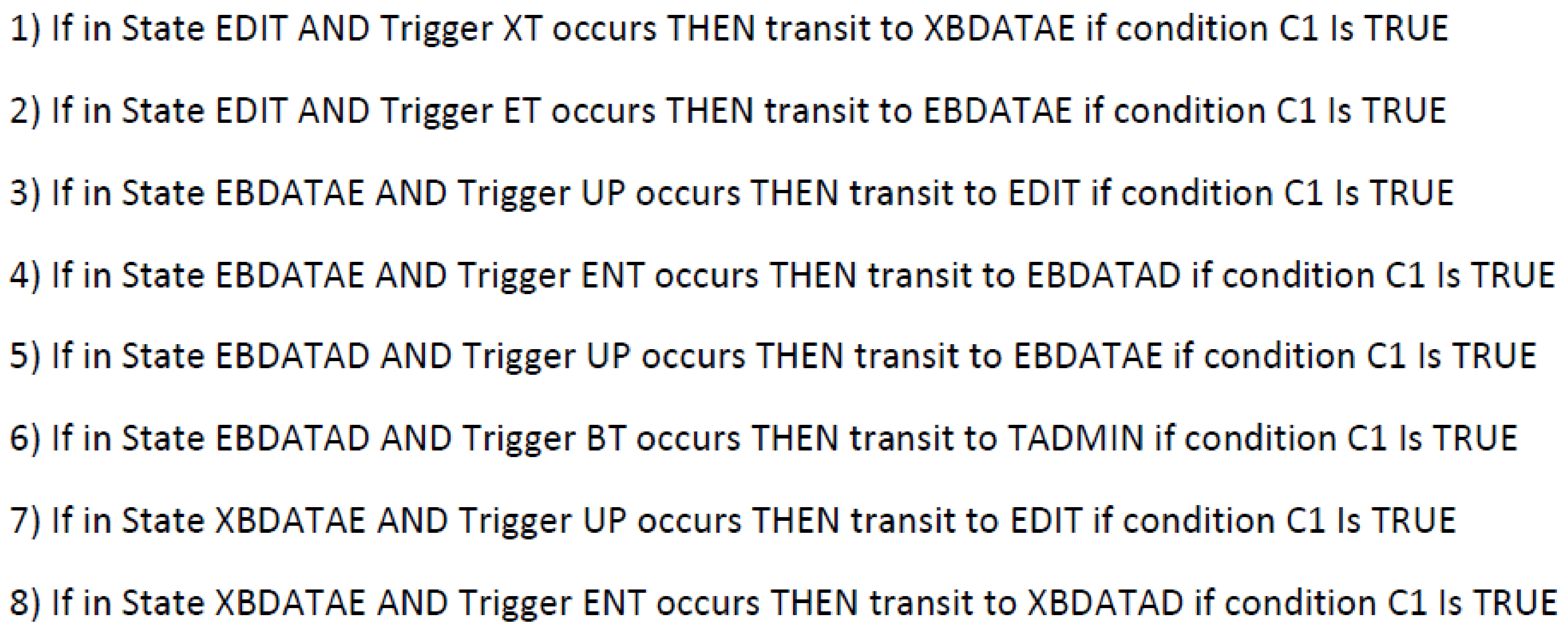
Table for Mode Transition



TRANSITION MATRIX

				Triggers	1	2	3	4	5	6	7	8
Row No	Offset	State No	Modes	Mode+State	UP	ENT	XT	ET	BT	TIM8	RESET	TICK
1	0	1	INTER	EDIT	0	0	4	2	0	0	0	0
2	0	2		EBDATAE	1	3	0	0	0	0	0	0
3	0	3		EBDATAD	2	0	0	0	6	0	0	0
4	0	4		XBDATAE	1	5	0	0	0	0	0	0
5	0	5		XBDATAD	4	0	0	0	6	0	0	0
6	0	6		TADMIN	0	1	0	0	0	0	0	0
7	6	1	LEVEL	NEITHER	0	0	2	3	0	0	0	0
8	6	2		XB	0	0	0	4	0	0	1	0
9	6	3		EB	0	0	5	0	0	0	1	0
10	6	4		X2E	0	0	0	0	0	0	0	3
11	6	5		E2X	0	0	0	0	0	0	0	2
12	11	1	SPREAD	INPLACE	0	0	0	2	0	0	0	0
13	11	2		MOVOUT	0	0	4	0	0	3	0	0
14	11	3		OUTPLACE	0	0	4	0	0	0	0	0
15	11	4		MOVIN	0	0	0	2	0	1	0	0
16	15	1	FIRE	WAIT	0	0	0	0	2	0	0	0
17	15	2		FIRED	0	0	0	0	0	0	0	1

Requirements (make_mtl.m)



1) If in State EDIT AND Trigger XT occurs THEN transit to XBDATAE if condition C1 Is TRUE

2) If in State EDIT AND Trigger ET occurs THEN transit to EBDATAE if condition C1 Is TRUE

3) If in State EBDATAE AND Trigger UP occurs THEN transit to EDIT if condition C1 Is TRUE

4) If in State EBDATAE AND Trigger ENT occurs THEN transit to EBDATAD if condition C1 Is TRUE

5) If in State EBDATAD AND Trigger UP occurs THEN transit to EBDATAE if condition C1 Is TRUE

6) If in State EBDATAD AND Trigger BT occurs THEN transit to TADMIN if condition C1 Is TRUE

7) If in State XBDATAE AND Trigger UP occurs THEN transit to EDIT if condition C1 Is TRUE

8) If in State XBDATAE AND Trigger ENT occurs THEN transit to XBDATAD if condition C1 Is TRUE

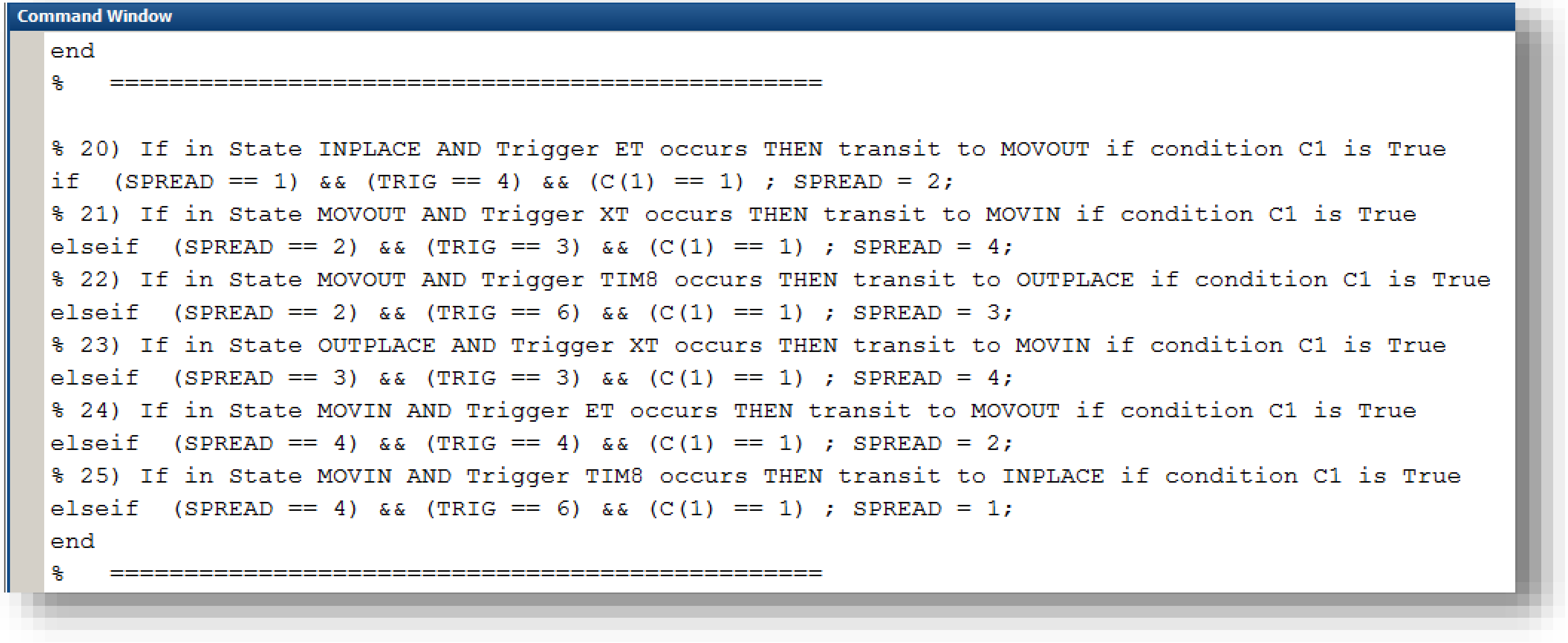
This is automatically generated by the M File. You can review and check for correctness.

NuSMV Code (make_mtl_code_SMV_gen.m)

```
(SPREAD = MOVOUT) & (XT = TRUE) : 0;
(SPREAD = MOVIN) & (ET = TRUE) : 0;
(SPREAD = MOVOUT) & (countx < 8) : countx+1;
(SPREAD = MOVIN) & (countx < 8) : countx+1;
(countx >= 8): 0;
TRUE : 0;
esac;

next(INTER):=case
--1) If in State EDIT AND Trigger XT occurs THEN transit to XBDATAE if condition C1 Is TRUE
INTER = EDIT & (TRIG = 3) & C1 : XBDATAE;
--2) If in State EDIT AND Trigger ET occurs THEN transit to EBDATAE if condition C1 Is TRUE
INTER = EDIT & (TRIG = 4) & C1 : EBDATAE;
--3) If in State EBDATAE AND Trigger UP occurs THEN transit to EDIT if condition C1 Is TRUE
INTER = EBDATAE & (TRIG = 1) & C1 : EDIT;
--4) If in State EBDATAE AND Trigger ENT occurs THEN transit to EBDATAD if condition C1 Is TRUE
INTER = EBDATAE & (TRIG = 2) & C1 : EBDATAD;
--5) If in State EBDATAD AND Trigger UP occurs THEN transit to EBDATAE if condition C1 Is TRUE
INTER = EBDATAD & (TRIG = 1) & C1 : EBDATAE;
```

This is automatically generated by the M File. Additional code for the 8 second timer is added.

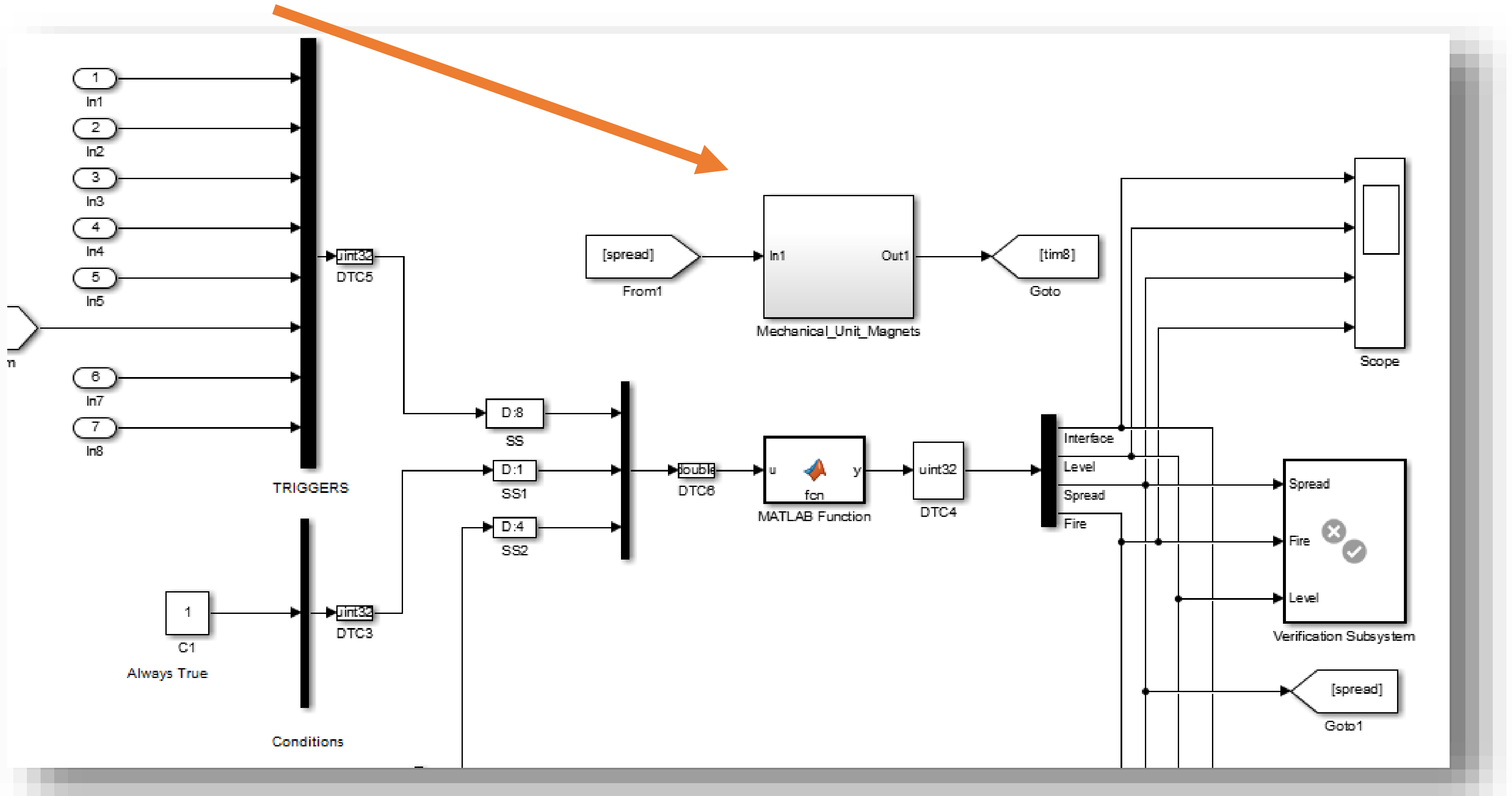
A M code (make_mtl_code_M_gen.m)



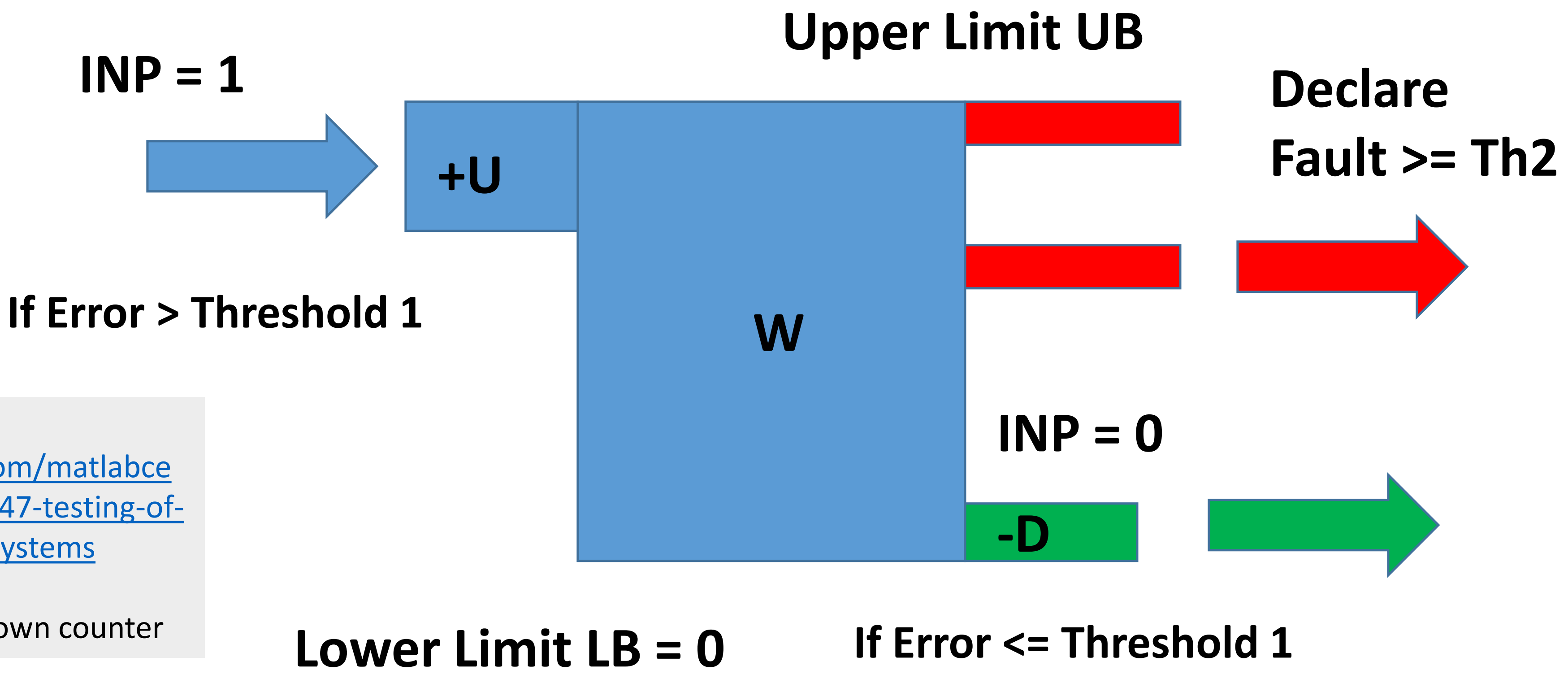
```
end
%   ================================================

% 20) If in State INPLACE AND Trigger ET occurs THEN transit to MOVOUT if condition C1 is True
if  (SPREAD == 1) && (TRIG == 4) && (C(1) == 1) ; SPREAD = 2;
% 21) If in State MOVOUT AND Trigger XT occurs THEN transit to MOVIN if condition C1 is True
elseif  (SPREAD == 2) && (TRIG == 3) && (C(1) == 1) ; SPREAD = 4;
% 22) If in State MOVOUT AND Trigger TIM8 occurs THEN transit to OUTPLACE if condition C1 is True
elseif  (SPREAD == 2) && (TRIG == 6) && (C(1) == 1) ; SPREAD = 3;
% 23) If in State OUTPLACE AND Trigger XT occurs THEN transit to MOVIN if condition C1 is True
elseif  (SPREAD == 3) && (TRIG == 3) && (C(1) == 1) ; SPREAD = 4;
% 24) If in State MOVIN AND Trigger ET occurs THEN transit to MOVOUT if condition C1 is True
elseif  (SPREAD == 4) && (TRIG == 4) && (C(1) == 1) ; SPREAD = 2;
% 25) If in State MOVIN AND Trigger TIM8 occurs THEN transit to INPLACE if condition C1 is True
elseif  (SPREAD == 4) && (TRIG == 6) && (C(1) == 1) ; SPREAD = 1;
end
%   ================================================
```

Simulink model with M Code

This Simulink model uses the M code and a mechanical model of the target magnets. This is modelled as a rate limiter. The timer trigger is set by the mechanical device when it reaches “in place” or “out of place”.



Up/Down Counter



Refer to
http://in.mathworks.com/matlabcentral/fileexchange/39047-testing-of-safety-critical-control-systems

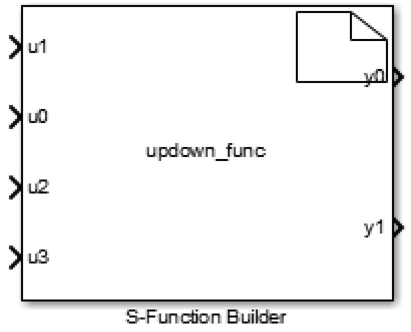
For details of the up/down counter

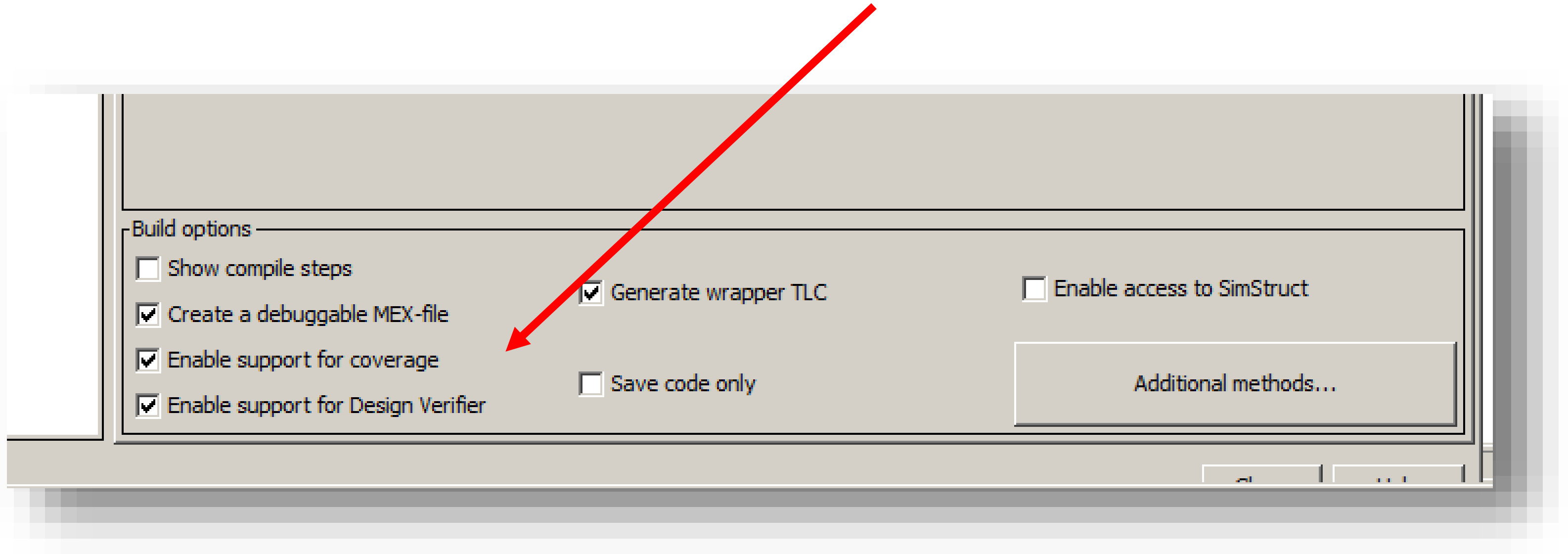
If U > (UB−LB) and −D = −U then the up-down counter will declare a fault if the residual exceeds Threshold 1 for one time step.

S Function

While compiling remember to enable the support for coverage and Design verifier.

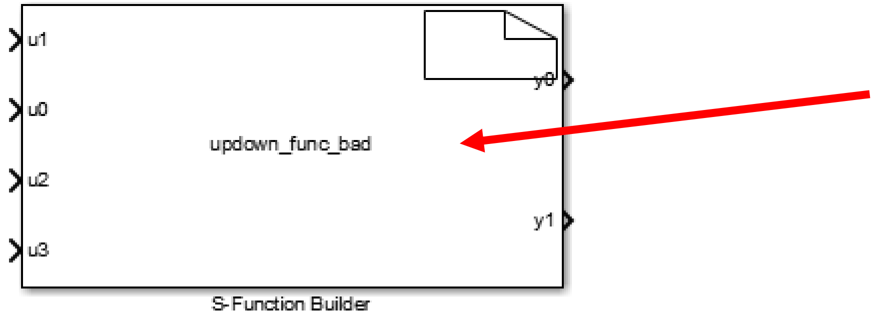
Use the S function builder for the C code example





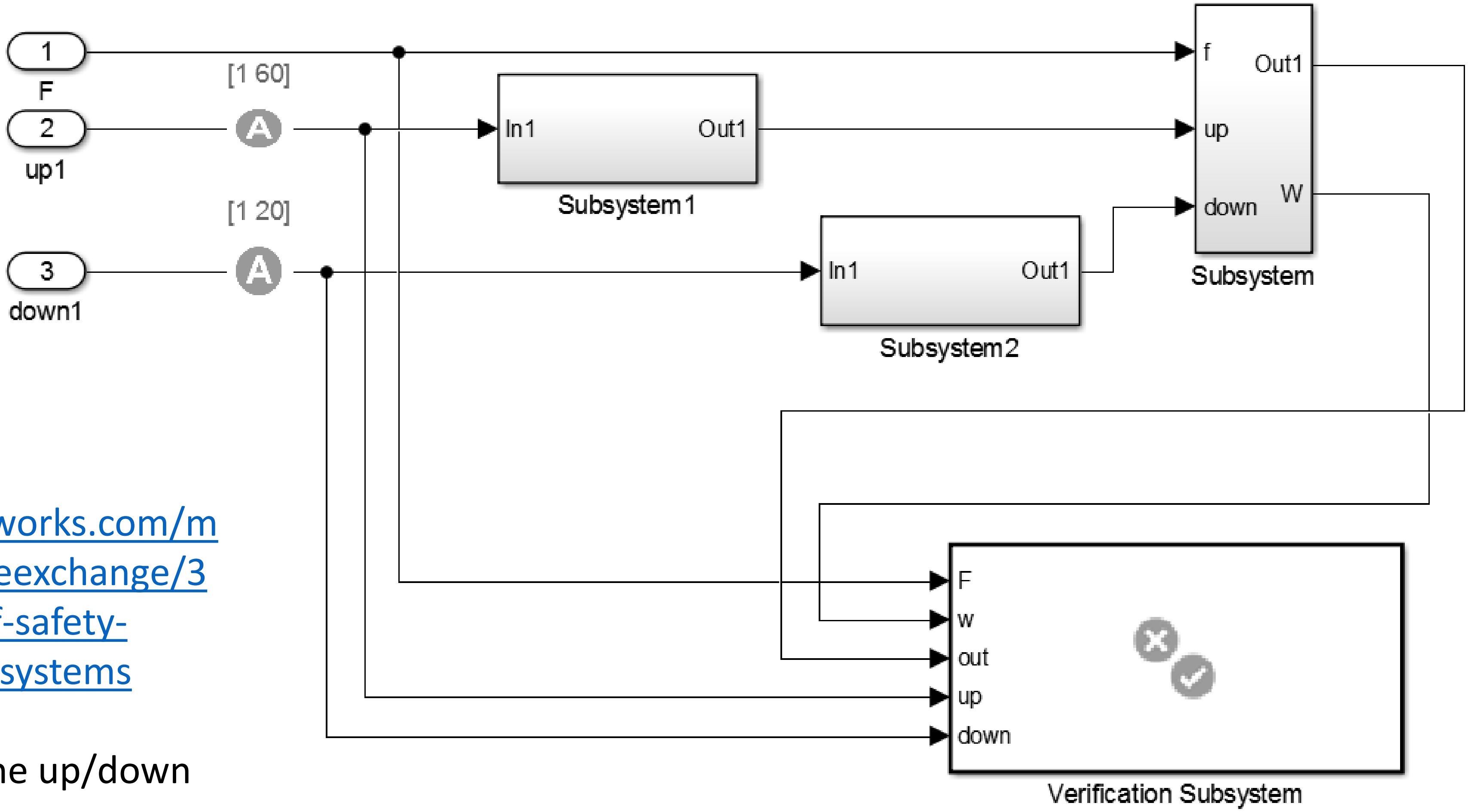
Build options

- [ ] Show compile steps
- [x] Create a debuggable MEX-file
- [x] Enable support for coverage
- [x] Enable support for Design Verifier
- [x] Generate wrapper TLC
- [ ] Save code only
- [ ] Enable access to SimStruct



This S function has the error

Verification subsystem



Refer to http://in.mathworks.com/matlabcentral/fileexchange/39047-testing-of-safety-critical-control-systems

For details of the up/down counter

References

- Lívia Camargos Rodrigues de Oliveira, Roberto Kawakami Harrop Galvão, "A Four-signal Voting Algorithm For Aircraft Redundant Sensors", 21st Brazilian Congress of Mechanical Engineering, COBEM 2011, October 24-28, 2011, Natal, RN, Brazil, http://www.abcm.org.br/app/webroot/anais/cobem/2011/PDF/125301.pdf
- Kumar N.Anil, Jeppu Yogananda V., Chandramouli Krishna, "A Worst Case Benchmark Problem to Validate Voter Logic," Computing and Communication Technologies (WCCCT), 2014 World Congress on , vol., no., pp.271,274, Feb. 27 2014-March 1 2014, doi: 10.1109/WCCCT.2014.17
- Nancy Leveson, Medical Devices: The Therac-25*, http://sunnyday.mit.edu/papers/therac.pdf
- Abbas Karimi, Faraneh Zarafshan, S. A. R. Al-Haddad, and Abdul Rahman Ramli, "A Novel -Input Voting Algorithm for -by-Wire Fault-Tolerant Systems," The Scientific World Journal, vol. 2014, Article ID 672832, 9 pages, 2014. doi:10.1155/2014/672832

- Natasha.Jeppu@gmail.com , YVJeppu@gmail.com