A METHODOLOGY FOR THE SIMPLIFICATION OF TABULAR DESIGNS IN MODEL-BASED DEVELOPMENT

FormaliSE 2015

Monika Bialy, Mark Lawford, Vera Pantelic, and Alan Wassyng

May 18, 2015

McMaster Centre for Software Certification
Department of Computing and Software
McMaster University

OVERVIEW

- Introduction
- Methodology
- Industrial Case Studies
- Conclusions

MOTIVATION

- Model-Based Development (MBD) is increasingly used for embedded control software
- Complex decision logic in Simulink is often implemented with *Stateflow truth tables*[1]
- Automotive partner has flagged some as particularly problematic
  - Hard to understand, test, and trace to requirements
  - ISO 26262 ASIL D stresses low complexity, MC/DC testing, unambiguous language constructs

▷ Thus, more effective tabular constructs and reliable refactoring techniques are needed

[1] Classically known as *decision tables*

Stateflow truth table

		Decisions			
#	Conditions	D_1	D_2	D_3	D_4
1	*Condition*$_1$	T	T	T	-
2	*Condition*$_2$	T	-	F	-
3	*Condition*$_3$	T	T	-	-
	Actions	1	2	3	4

#	Actions
1	*Action*$_1$
2	*Action*$_2$
3	*Action*$_3$
4	*Action*$_4$

- Diagnostic tools do not check disjointness
- Implicit left-to-right semantics
- Completeness commonly forced with *else* catch-all case
- Non-Boolean conditions expressed unintuitively
- Readability does not scale well

Tabular Expressions

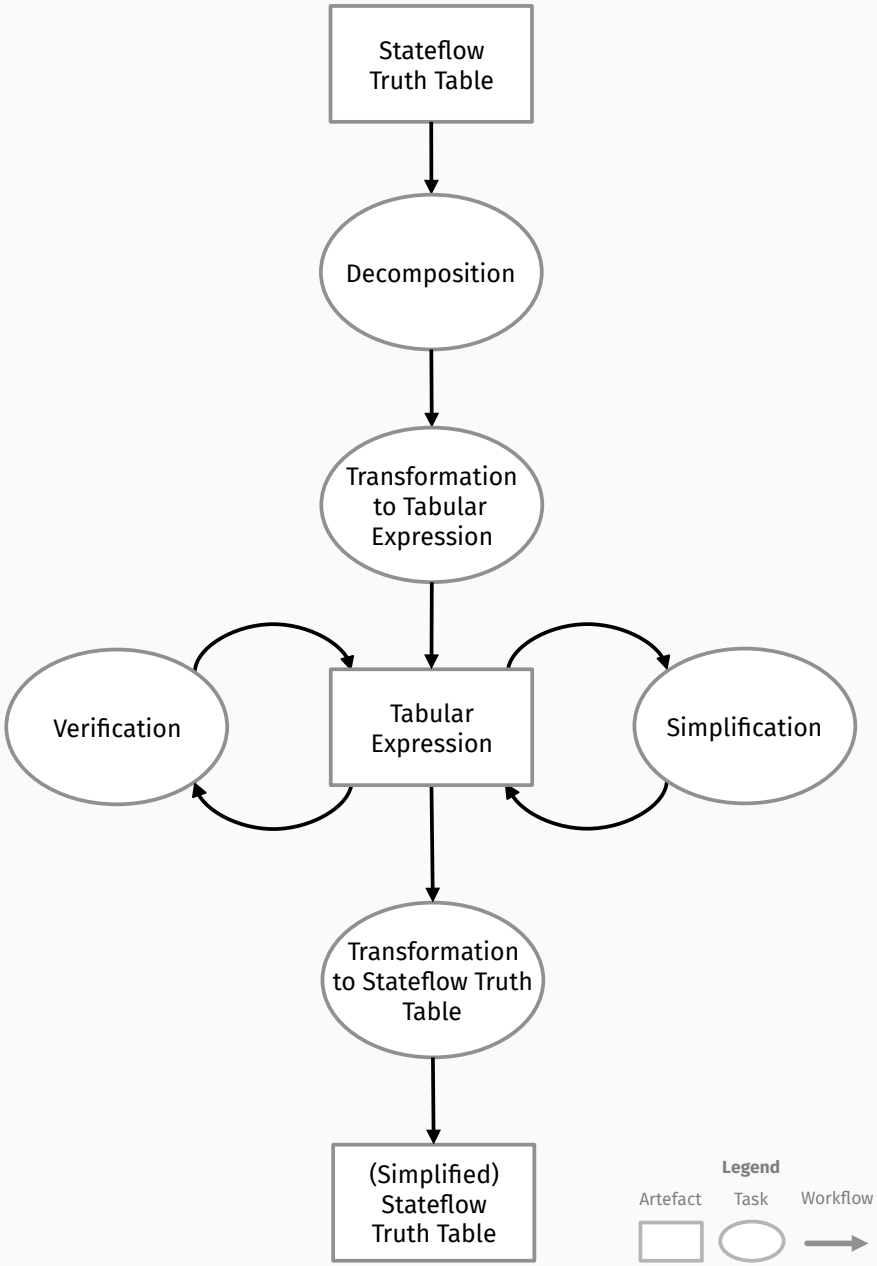
Horizontal Condition Table (HCT)

		Result
Conditions		*Var*
Condition$_1$	*Condition*$_2$	*Result*$_1$
	¬*Condition*$_2$	*Result*$_2$
¬*Condition*$_1$		*Result*$_3$

- Disjointness required
- No prescribed row evaluation order
- Completness is explicit
- More intuitive and concise syntax
- Easily readable and traceable to requirements

METHODOLOGY

- Thus, we leverage the use of tabular expressions to remedy the deficiencies of Stateflow truth tables
- Guided refactoring is facilitated as logical simplifications are easier to detect and apply
- Heuristics were designed to be an easy-to-follow process of performing guided refactoring



AUTOMOTIVE CASE STUDY 1

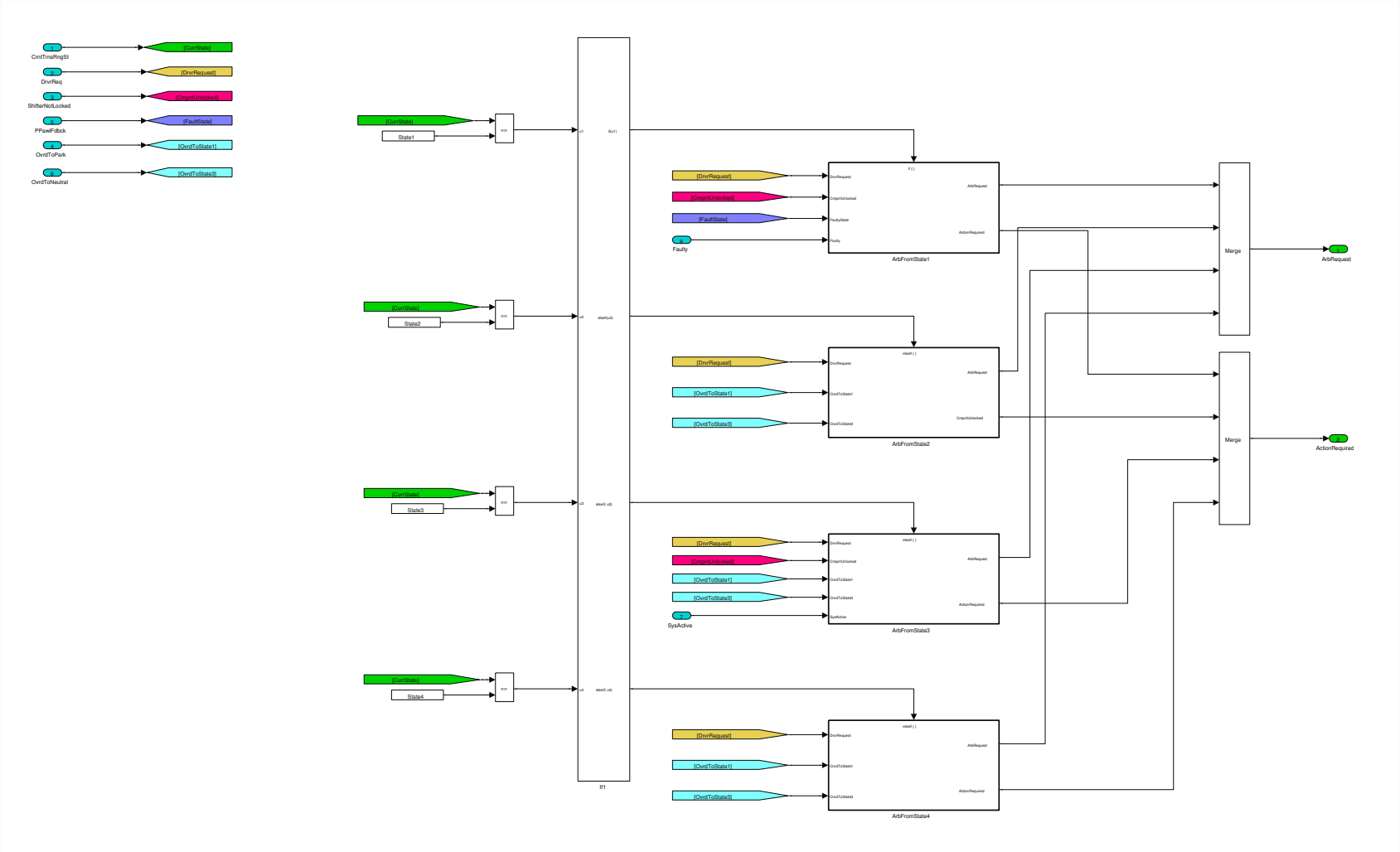


Figure 1: Subsystem for driver request arbitration

ORIGINAL STATEFLOW TRUTH TABLE

		Decisions										
#	Conditions	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}
1	*eDrvrRequest* == $cState_1$	T	F	F	F	F	F	F	F	F	F	-
2	*eDrvrRequest* == $cState_2$	F	T	F	F	T	F	F	T	F	F	-
3	*eDrvrRequest* == $cState_3$	F	F	T	F	F	T	F	F	T	F	-
4	*eDrvrRequest* == $cState_4$	F	F	F	T	F	F	T	F	F	T	-
5	*bCmpntUnlocked*	-	T	T	T	-	-	-	-	-	-	-
6	*bFaulty*	-	F	F	F	T	T	T	-	-	-	-
	Actions	1	2	3	4	5	5	5	5	5	5	1

#	Actions
1	*eArbRequest*=$cState_1$; *bActionRequired=false*
2	*eArbRequest*=$cState_2$; *bActionRequired=false*
3	*eArbRequest*=$cState_3$; *bActionRequired=false*
4	*eArbRequest*=$cState_4$; *bActionRequired=false*
5	*eArbRequest*=$cState_1$; *bActionRequired=true*

Table 1: Driver request arbitration from $cState_1$

DECOMPOSITION

- If the table computes multiple outputs, decompose into multiple tables each computing a single output
  - Increases modularity, requirements traceability, greater reductions during simplification

		Decisions										
#	Conditions	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}
1	*eDrvrRequest* == $cState_1$	T	F	F	F	F	F	F	F	F	F	-
2	*eDrvrRequest* == $cState_2$	F	T	F	F	T	F	F	T	F	F	-
3	*eDrvrRequest* == $cState_3$	F	F	T	F	F	T	F	F	T	F	-
4	*eDrvrRequest* == $cState_4$	F	F	F	T	F	F	T	F	F	T	-
5	*bCmpntUnlocked*	-	T	T	T	-	-	-	-	-	-	-
6	*bFaulty*	-	F	F	F	T	T	T	-	-	-	-
	Actions	1	2	3	4	5	5	5	5	5	5	1

#	Actions
1	*eArbRequest*=$cState_1$; *bActionRequired*=*false*
2	*eArbRequest*=$cState_2$; *bActionRequired*=*false*
3	*eArbRequest*=$cState_3$; *bActionRequired*=*false*
4	*eArbRequest*=$cState_4$; *bActionRequired*=*false*
5	*eArbRequest*=$cState_1$; *bActionRequired*=*true*

(a) Original

		Decisions										
#	Conditions	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}
1	*eDrvrRequest* == $cState_1$	T	F	F	F	F	F	F	F	F	F	-
2	*eDrvrRequest* == $cState_2$	F	T	F	F	T	F	F	T	F	F	-
3	*eDrvrRequest* == $cState_3$	F	F	T	F	F	T	F	F	T	F	-
4	*eDrvrRequest* == $cState_4$	F	F	F	T	F	F	T	F	F	T	-
5	*bCmpntUnlocked*	-	T	T	T	-	-	-	-	-	-	-
6	*bFaulty*	-	F	F	F	T	T	T	-	-	-	-
	Actions	1	2	3	4	1	1	1	1	1	1	1

#	Actions
1	*eArbRequest*=$cState_1$
2	*eArbRequest*=$cState_2$
3	*eArbRequest*=$cState_3$
4	*eArbRequest*=$cState_4$

(b) Decomposed

STATEFLOW TRUTH TABLE → HCT

1. Augment decisions with conditions and actions
2. Transpose
3. Group related conditions
4. Ensure disjointness and completeness
5. Formatting

		Decisions										
#	Conditions	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}
1	*eDrvrRequest == cState*$_1$	T	F	F	F	F	F	F	F	F	F	-
2	*eDrvrRequest == cState*$_2$	F	T	F	F	T	F	F	T	F	F	-
3	*eDrvrRequest == cState*$_3$	F	F	T	F	F	T	F	F	T	F	-
4	*eDrvrRequest == cState*$_4$	F	F	F	T	F	F	T	F	F	T	-
5	*bCmpntUnlocked*	-	T	T	T	-	-	-	-	-	-	-
6	*bFaulty*	-	F	F	F	T	T	T	-	-	-	-
	Actions	1	2	3	4	1	1	1	1	1	1	1

#	Actions
1	*eArbRequest=cState*$_1$
2	*eArbRequest=cState*$_2$
3	*eArbRequest=cState*$_3$
4	*eArbRequest=cState*$_4$

(a) Stateflow truth table

Conditions			Result
			eArbRequest
eDrvrRequest == cState$_1$			*cState*$_1$
eDrvrRequest == cState$_2$	¬*bFaulty*	*bCmpntUnlocked*	*cState*$_2$
		¬*bCmpntUnlocked*	*cState*$_1$
	bFaulty		*cState*$_1$
eDrvrRequest == cState$_3$	¬*bFaulty*	*bCmpntUnlocked*	*cState*$_3$
		¬*bCmpntUnlocked*	*cState*$_1$
	bFaulty		*cState*$_1$
eDrvrRequest == cState$_4$	¬*bFaulty*	*bCmpntUnlocked*	*cState*$_4$
		¬*bCmpntUnlocked*	*cState*$_1$
	bFaulty		*cState*$_1$

(b) HCT

A. Condition Ordering

- Vertical
  - Move decisions with the most “don’t cares” to upper rows to increase efficiency and speed of evaluation

Conditions			Result *eArbRequest*
eDrvrRequest == $cState_1$			$cState_1$
eDrvrRequest == $cState_2$	¬*bFaulty*	*bCmpntUnlocked*	$cState_2$
		¬*bCmpntUnlocked*	$cState_1$
	bFaulty		$cState_1$
eDrvrRequest == $cState_3$	¬*bFaulty*	*bCmpntUnlocked*	$cState_3$
		¬*bCmpntUnlocked*	$cState_1$
	bFaulty		$cState_1$
eDrvrRequest == $cState_4$	¬*bFaulty*	*bCmpntUnlocked*	$cState_4$
		¬*bCmpntUnlocked*	$cState_1$
	bFaulty		$cState_1$

(a) Pre-simplification

Conditions			Result *eArbRequest*
eDrvrRequest == $cState_1$			$cState_1$
eDrvrRequest == $cState_2$	*bFaulty*		$cState_1$
	¬*bFaulty*	*bCmpntUnlocked*	$cState_2$
		¬*bCmpntUnlocked*	$cState_1$
eDrvrRequest == $cState_3$	*bFaulty*		$cState_1$
	¬*bFaulty*	*bCmpntUnlocked*	$cState_3$
		¬*bCmpntUnlocked*	$cState_1$
eDrvrRequest == $cState_4$	*bFaulty*		$cState_1$
	¬*bFaulty*	*bCmpntUnlocked*	$cState_4$
		¬*bCmpntUnlocked*	$cState_1$

(b) Vertical reordering of *bFaulty* for consistency

A. Condition Ordering, *Continued*

- Horizontal
  - Nest conditions to minimize evaluation
  - Visual means of representing the dominance of conditions

Conditions			Result *eArbRequest*
$eDrvrRequest == cState_1$			$cState_1$
$eDrvrRequest == cState_2$	$bFaulty$		$cState_1$
	$\neg bFaulty$	$bCmpntUnlocked$	$cState_2$
		$\neg bCmpntUnlocked$	$cState_1$
$eDrvrRequest == cState_3$	$bFaulty$		$cState_1$
	$\neg bFaulty$	$bCmpntUnlocked$	$cState_3$
		$\neg bCmpntUnlocked$	$cState_1$
$eDrvrRequest == cState_4$	$bFaulty$		$cState_1$
	$\neg bFaulty$	$bCmpntUnlocked$	$cState_4$
		$\neg bCmpntUnlocked$	$cState_1$

(a) Pre-simplification

Conditions			Result *eArbRequest*
$bFaulty$			$cState_1$
$\neg bFaulty$	$eDrvrRequest == cState_1$	$bCmpntUnlocked$	$cState_1$
		$\neg bCmpntUnlocked$	$cState_1$
	$eDrvrRequest == cState_2$	$bCmpntUnlocked$	$cState_2$
		$\neg bCmpntUnlocked$	$cState_1$
	$eDrvrRequest == cState_3$	$bCmpntUnlocked$	$cState_3$
		$\neg bCmpntUnlocked$	$cState_1$
	$eDrvrRequest == cState_4$	$bCmpntUnlocked$	$cState_4$
		$\neg bCmpntUnlocked$	$cState_1$

(b) Horizontal reordering of *bFaulty* to show dominance

B. Granted State Simplification

- Particularly useful for systems which arbitrate operational modes
- A condition check is not required when the condition's value is granted and passed through
- The mode variable can be directly placed in the result column

Conditions			Result
			eArbRequest
bFaulty			$cState_1$
$\neg bFaulty$	$eDrvrRequest == cState_1$	*bCmpntUnlocked*	$cState_1$
		$\neg bCmpntUnlocked$	$cState_1$
	$eDrvrRequest == cState_2$	*bCmpntUnlocked*	$cState_2$
		$\neg bCmpntUnlocked$	$cState_1$
	$eDrvrRequest == cState_3$	*bCmpntUnlocked*	$cState_3$
		$\neg bCmpntUnlocked$	$cState_1$
	$eDrvrRequest == cState_4$	*bCmpntUnlocked*	$cState_4$
		$\neg bCmpntUnlocked$	$cState_1$

(a) Pre-simplification

Conditions			Result
			eArbRequest
bFaulty			$cState_1$
$\neg bFaulty$	*bCmpntUnlocked*		*eDrvrRequest*
	$\neg bCmpntUnlocked$	$eDrvrRequest == cState_1$	$cState_1$
		$eDrvrRequest == cState_2$	$cState_1$
		$eDrvrRequest == cState_3$	$cState_1$
		$eDrvrRequest == cState_4$	$cState_1$

(b) Granted state simplification of *eDrvrRequest*

C. Removal of “Don’t Care” Conditions

- When a condition does not affect the outcome of a decision, it is treated as a “don’t care”
- Identify multiple instances of the same output in the results
  - If the paths are the same except for one condition, and can be combined to cover the range of the condition’s type

Conditions			Result *eArbRequest*
bFaulty			$cState_1$
¬*bFaulty*	*bCmpntUnlocked*		*eDrvrRequest*
	¬*bCmpntUnlocked*	*eDrvrRequest* == $cState_1$	$cState_1$
		eDrvrRequest == $cState_2$	$cState_1$
		eDrvrRequest == $cState_3$	$cState_1$
		eDrvrRequest == $cState_4$	$cState_1$

(a) Pre-simplification

Conditions		Result *eArbRequest*
bFaulty		$cState_1$
¬*bFaulty*	*bCmpntUnlocked*	*eDrvrRequest*
	¬*bCmpntUnlocked*	$cState_1$

(b) Simplifying *eDrvrRequest* to a “don’t care”

SIMPLIFICATIONS, CONT'D

D. Grouping

- A subset of rows which lead to the same output can be grouped
- Used when removing mode-centric conditions in their entirety is not achievable, or unnecessary

E. Compound Simplification

- Expand already simplified rows to enable further table simplification through the use of the newly introduced rows
- Useful when altering an existing table to display a design in a specific manner that corresponds to a requirement

HCT $\rightarrow$ STATEFLOW TRUTH TABLE

1. Remove tabular expression formatting
2. Transpose
3. Construct condition section
4. Construct action section

Conditions		Result
		eArbRequest
bFaulty		$cState_1$
$\neg$*bFaulty*	*bCmpntUnlocked*	*eDrvrRequest*
	$\neg$*bCmpntUnlocked*	$cState_1$

(a) HCT

		Decisions		
#	Conditions	D_1	D_2	D_3
1	*bFaulty*	T	F	F
2	*bCmpntUnlocked*	-	T	F
	Actions	1	2	1

#	Actions
1	*eArbRequest*=$cState_1$
2	*eArbRequest*=*eDrvrRequest*

(b) Stateflow truth table

- Formal proof of equivalence between original and refactored tabular expressions with *Prototype Verification System* (PVS)
  - The equivalence between steps is easy to see
  - Verifying equivalence between other intermediate steps is also possible

- Methodology applied to the remaining tables of the subsystem
- Refactored tables replaced original tables in the model

CASE STUDY 1: RESULTS

- Compare original & refactored designs w.r.t. testing and complexity
- *Simulink Design Verifier* (SDV) was used

	Original	Refactored
Tests	7	9
Test Steps	97	48
Test Time (s)	18	7.8
Number of Objectives	1016	371
Objectives Satisfied	797 (78%)	311 (84%)
Objectives Proven Unsatisfiable	219 (22%)	60 (16%)
Cyclomatic Complexity	274	107

Table 2: Comparison of test suite generation and complexity

	Original			Refactored		
	Satisfied	Total	Percentage	Satisfied	Total	Percentage
Condition	368	452	81%	110	140	79%
Decision to	112	112	100%	95	95	100%
MC/DC	141	226	62%	44	70	63%

Table 3: Comparison of test coverage

CASE STUDY 1: RESULTS

- Can be used directly in documentation
- HCTs are generally more readable
- Requirements presented more evidently and thus are more traceable

Conditions		Result
		eArbRequest
bFaulty		$cState_1$
$\neg$*bFaulty*	*bCmpntUnlocked*	*eDrvrRequest*
	$\neg$*bCmpntUnlocked*	$cState_1$

Requirement: "remain in $cState_1$ when there are no faults, but the component is locked"

2 Stateflow truth tables + 1 calibration matrix

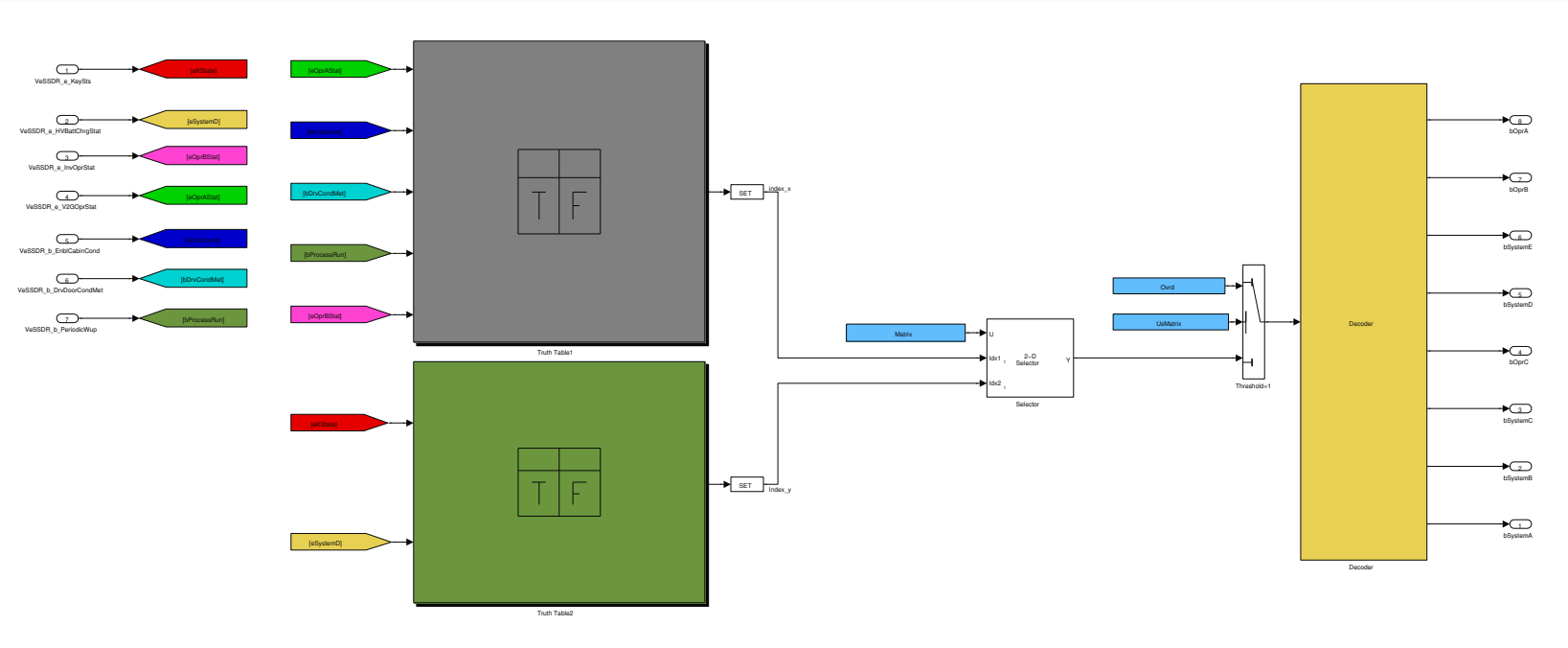


Figure 9: Subsystem for determining overall system status

Conditions	Decisions																																																																								
$ePowerSource == NotConnected$	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	-
$ePowerSource == Connected\_NotOpr$	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	-
$ePowerSource == Connected\_Opr$	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	-
$eComponent == NotPluggedIn$	T	T	T	F	F	F	F	F	F	T	T	T	F	F	F	F	F	F	T	T	T	F	F	F	F	F	F	T	T	T	F	F	F	F	F	F	T	T	T	F	F	F	F	F	F	T	T	T	F	F	F	F	F	F	T	T	T	F	F	F	F	F	F	T	T	T	F	F	F	F	F	F	-
$eComponent == PluggedIn\_NotOpr$	F	F	F	T	T	T	F	F	F	F	F	F	T	T	T	F	F	F	F	F	F	T	T	T	F	F	F	F	F	F	T	T	T	F	F	F	F	F	F	T	T	T	F	F	F	F	F	F	T	T	T	F	F	F	F	F	F	T	T	T	F	F	F	F	F	F	T	T	T	F	F	F	-
$eComponent == PluggedIn$	F	F	F	F	F	F	T	T	T	F	F	F	F	F	F	T	T	T	F	F	F	F	F	F	T	T	T	F	F	F	F	F	F	T	T	T	F	F	F	F	F	F	T	T	T	F	F	F	F	F	F	T	T	T	F	F	F	F	F	F	T	T	T	F	F	F	F	F	F	T	T	T	-
$bSysEnabled$	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	-
$bPeriodicOpr$	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	T	T	T	T	T	F	F	F	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	-
$bCompartmentOpen$	F	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T	T	T	T	F	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T	-
Actions:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73

#	Action
1	$Action_1$
2	$Action_2$
3	$Action_3$
4	$Action_4$
5	$Action_5$
6	$Action_6$
7	$Action_7$
8	$Action_8$
9	$Action_9$
10	$Action_{10}$
11	$Action_{11}$
12	$Action_{12}$
13	$Action_{13}$
14	$Action_{14}$
15	$Action_{15}$
16	$Action_{16}$
17	$Action_{17}$
18	$Action_{18}$
19	$Action_{19}$
20	$Action_{20}$
21	$Action_{21}$
22	$Action_{22}$
23	$Action_{23}$
24	$Action_{24}$
25	$Action_{25}$
⋮	⋮
72	$Action_{72}$
73	$Action_{73}$

(a) Stateflow truth table (1 of 2)

Conditions	Result $bOpr_A$
$eOprAStat == cNotPlgIn \land eOprBStat == cNoTools \land \neg bEnblCond \land \neg bProcessRun \land eSystem_D == cNotPlgdIn \land eKState == cPos5$	*true*
$eOprAStat \sim= cNotPlgIn \land eOprBStat \sim= cNoTools \land bEnblCond \land bProcessRun \land eSystem_D \sim= cNotPlgdIn \land eKState \sim= cPos5$	*false*

(b) Tabular Expression (1 of 8)

	Original	Refactored
Tests	23	6
Test Steps	1214	24
Test Time (s)	100	3.6
Number of Objectives	1954	498
Objectives Satisfied	1591 (82%)	445 (89%)
Objectives Proven Unsatisfiable	202 (10%)	53 (10%)
Objectives Undecided	161 (8%)	0
Cyclomatic Complexity	935	248

Table 4: Comparison of test suite generation and complexity

	Original			Refactored		
	Satisfied	Total	Percentage	Satisfied	Total	Percentage
Condition	1309	1672	78%	363	418	87%
Decision	297	300	99%	84	84	100%
MC/DC	473	836	57%	154	209	74%

Table 5: Comparison of test coverage

Conclusions

- Proposed a methodology for the refactoring of complex tabular designs
  - Shown to increase testability and decrease complexity
- Applied to two industrial models
- **Refactored designs incorporated into production code**

Future Work

- Simulink lacks model refactoring tools
- Tool support and automation