

Regionální inovační centrum elektrotechniky
Fakulta elektrotechnická
Západočeská univerzita v Plzni

Uživatelské funkce v Simulinku 3 - Stateflow

Mikroprocesorové řízení pohonů 2

Jakub Talla

1) Úvod do Stateflow

- ▶ Základní prostředí, bloky a koncepce
- ▶ Stavý a stavové diagramy

2) Funkce ve Stateflow

- ▶ Klasické funkce: Grafická, Simulink, Matlab
- ▶ Pravdivostní tabulka

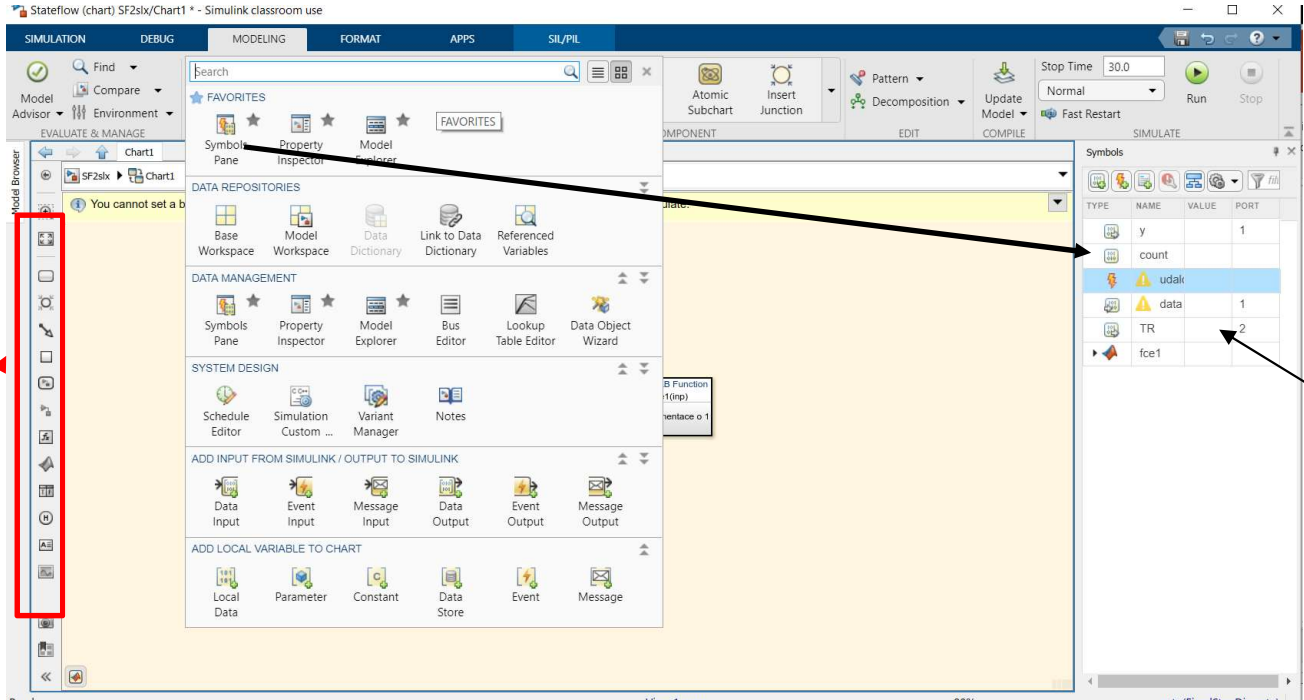
Pro jaké aplikace je vhodný Stateflow?

- Implementace logických a rozhodovacích funkcí
- Implementace stavových automatů
- Tvorba plánů a postupů (sekvencí)
- Detekce chyb
- Implementace událostí řízených systémů

Proč používat Stateflow?

- Stateflow je samostatné grafické prostředí vytvořené pro modelování stavových automatů a logických rozhodování
- Stateflow umožňuje přirozeným způsobem řídit pořadí prováděných operací (viz. globální proměnná přednáška č. 4) pomocí mikrokroků (pořadí operací v rámci jednoho kroku solveru)
- Stateflow umožňuje jednoduchou integraci a spolupráci se zbytkem modelu v Simulinku
- Stateflow zpřehledňuje provádění stavového diagramu pomocí animace a dalších integrovaných analýz
- Umožňuje práci s datovými typy a generování kódu
- Hlavní důvody pro vývojáře: Přehlednost, Testovatelnost, Efektivita

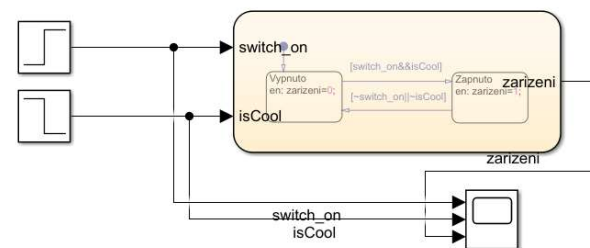
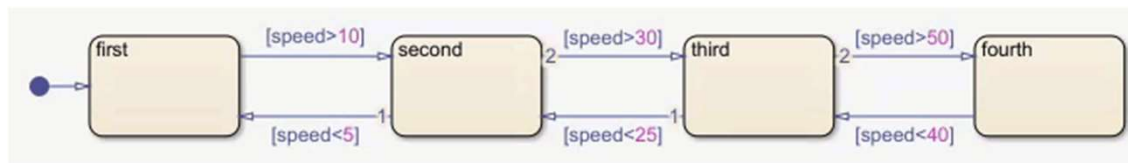
Uživatelské prostředí Stateflow je samostatné grafické prostředí



Statflow bloky

TYPE	NAME	VALUE	PORT
	Y		1
	count		
	udak		
	data	1	
	TR		2
	fce1		

Vstupy, výstupy, události, data


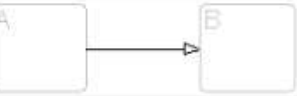






Základní pojmy ve Stateflow

Základní pojmy ve Stateflow

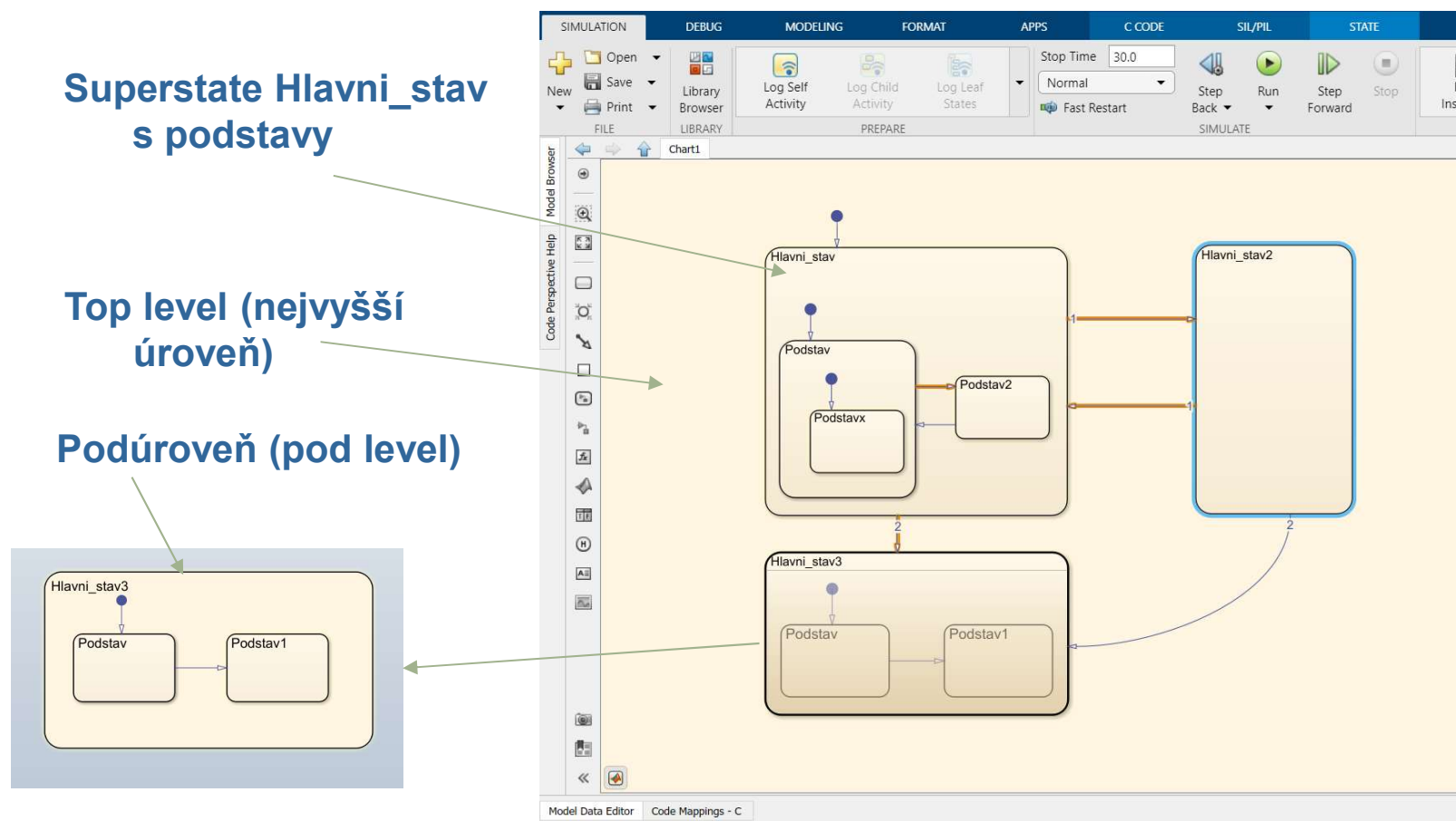
- **CHART/FLOWCHART** (diagram, graf) = je graf obsahující stavové bloky (States) a Flow charts (jednoduché bloky podmínek)
- **LEVEL** = viditelná úroveň Stateflow grafu (Stateflow je hierarchický a umožňuje vytvářet „subsystémy“ subgrafy)
- **STATES** = stavy, základní blok ve Stateflow, tvar bubliny, stav může obsahovat podstavy
- **TRANSITIONS** = přechodové čáry mezi stavy
- **JUNCTIONS** = uzly slouží ke větvení a slučování větví grafu
- **STATEFLOW FUNCTIONS** (funkce) = Stateflow umožňuje volat tři základní typy funkcí Graphical (grafické vytvořené ve Stateflow), Simulink (klasické Simulinkovské pomocí Simulink bloků) a Matlab functions
- Umožňuje práci s datovými typy a generování kódu
- Hlavní cíle: Přehlednost, Testovatelnost, Efektivita

Základní bloky ve Stateflow

State	
Transition	
History Junction	
Default Transition	
Connective Junction	
Truth Table Function	truthtable $y = \text{func}(x)$

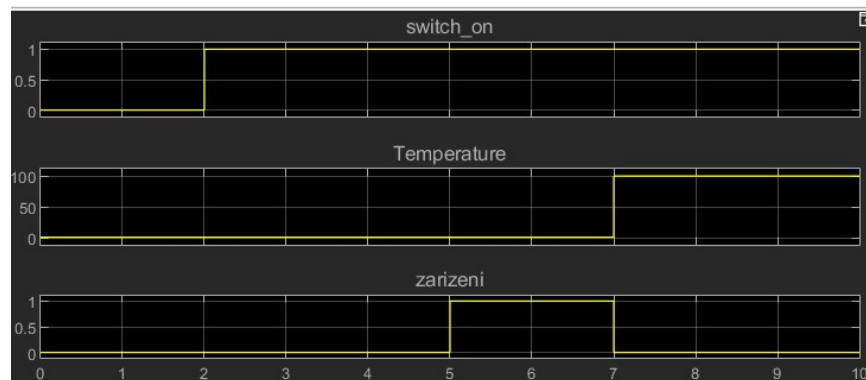
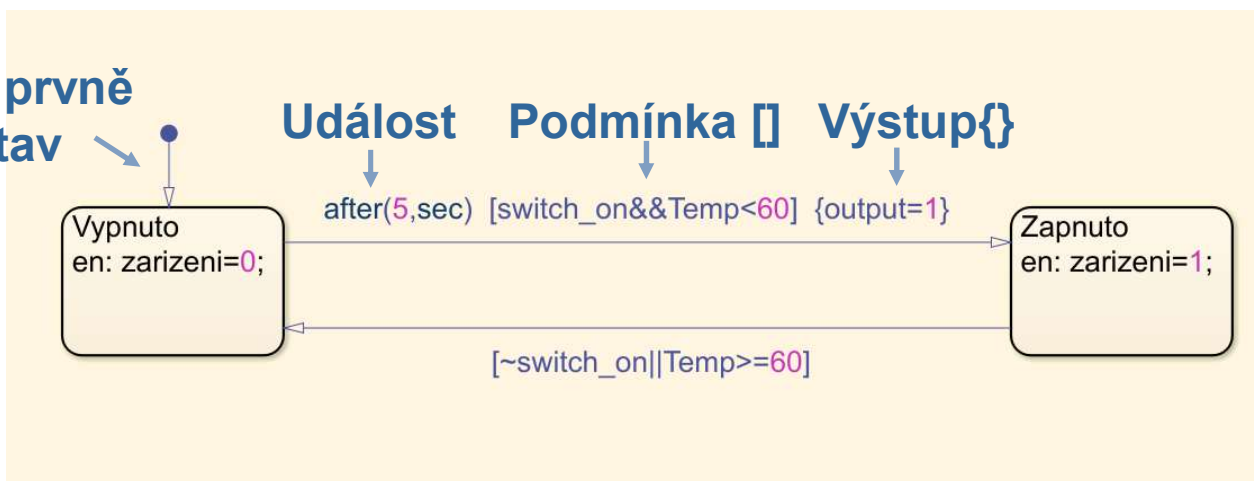
Graphical Function	function $y = \text{func}(x)$
Embedded MATLAB™ Function	eM $y = \text{func}(x)$
Box	

STATES – STAVY jsou znázorněné bublinami a jsou hierarchicky členěny



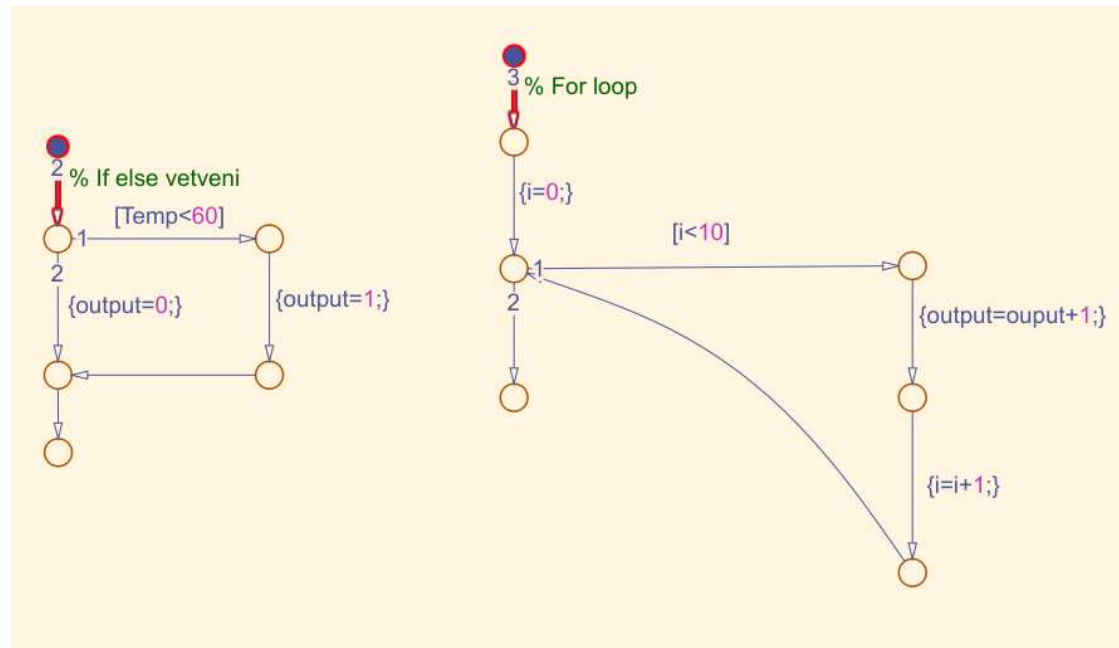
TRANSITIONS – přechodové čáry definují za jakých podmínek či událostí dochází k přechodu z jednoho stavu do druhého. Umožňují i přiřazení.

Definuje prvně
volaný stav



**JUNCTION – Spojení/uzel slouží k větvení či slučování přechodových čar
(Transitions)**

**K větvení dochází podle nastavených priorit. Použití především jako: If Else,
Switch Case, For cyklus, While/Do While apod.**



Stavy a provádění stavů

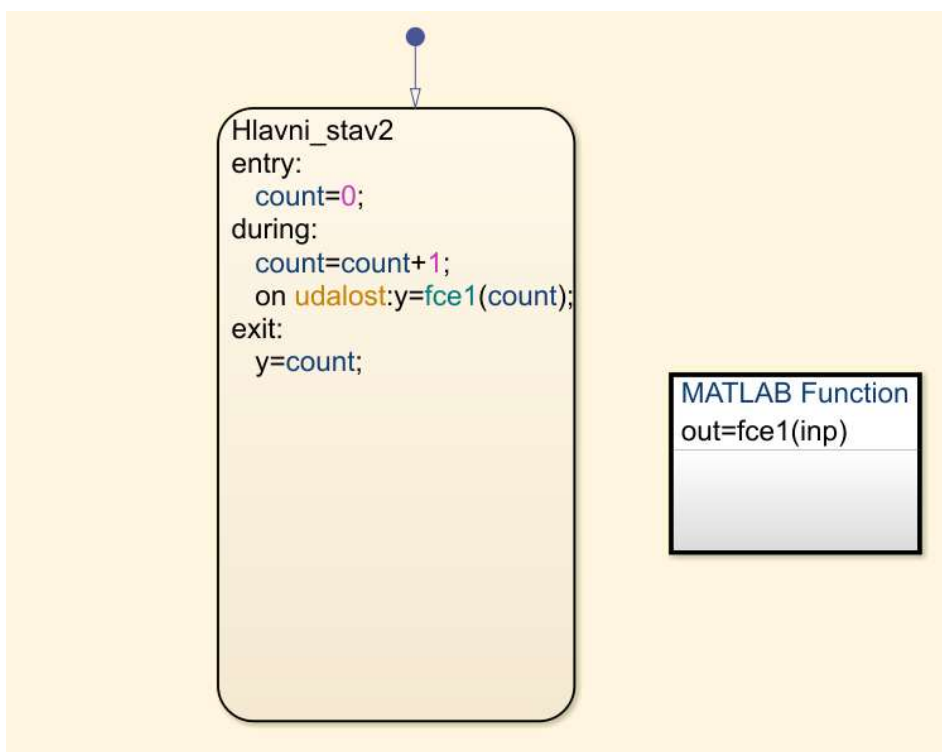
Stavy – kód je možné psát přímo do bloku stavu

Klíčové příkazy pro definování kódu stavů:

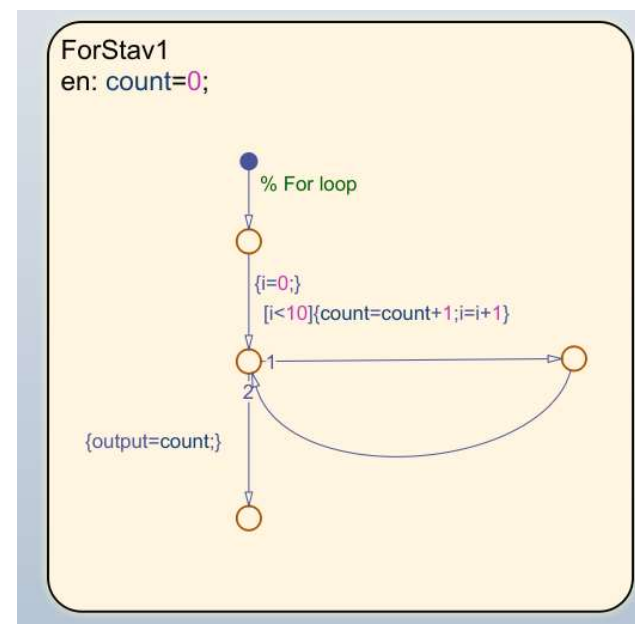
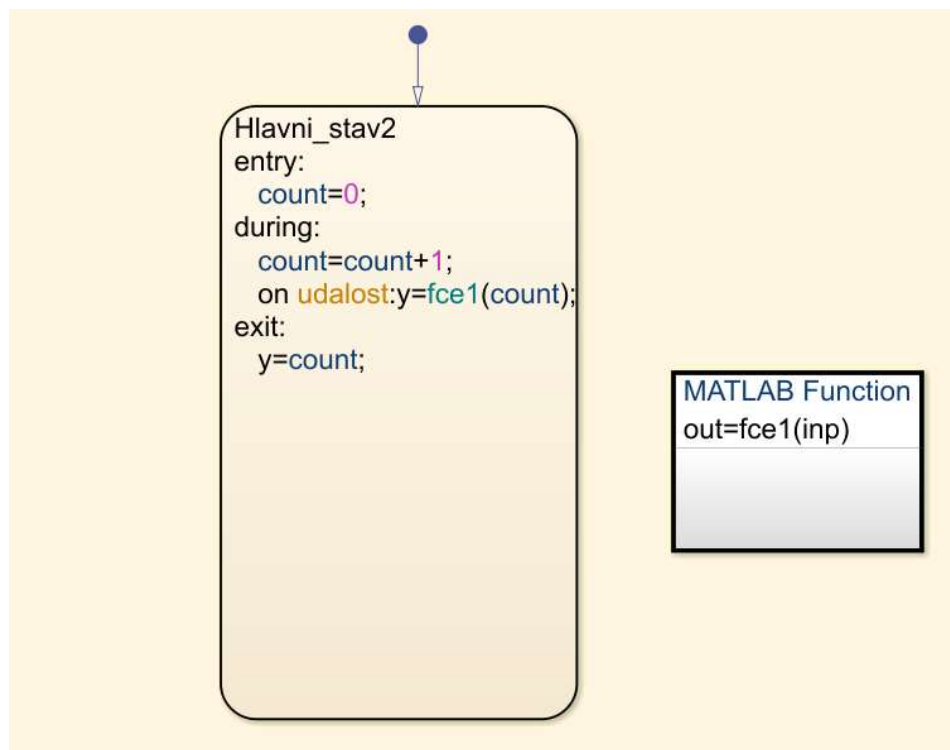
entry/en: Tato část kódu bude provedena pouze po okamžiku vstupu do stavu

during/du: Tato část kódu bude prováděna po dobu platnosti stavu

exit/ex: Tato část kódu bude prováděna po skončení platnosti stavu (tj. před přechodem do jiného stavu)

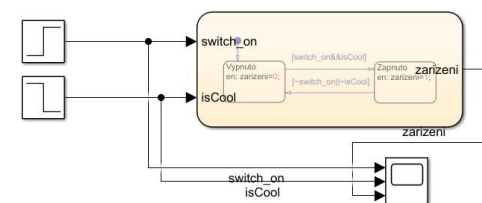
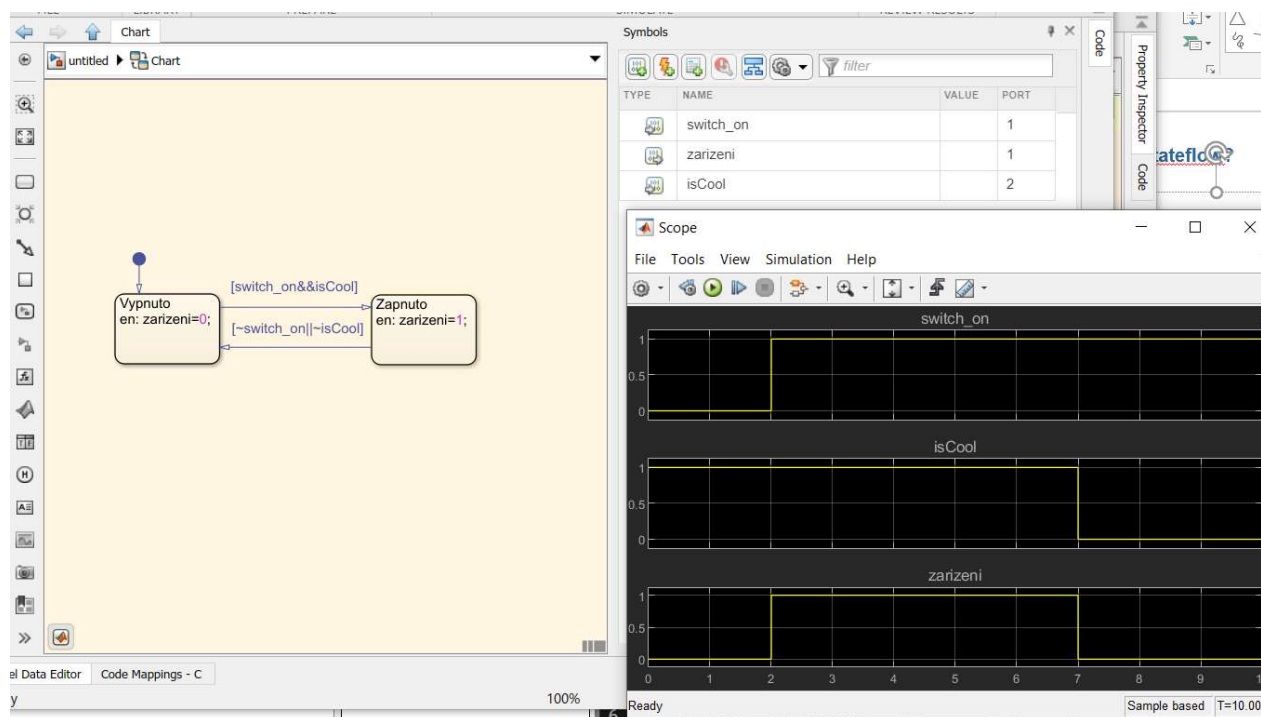


Stav může obsahovat podstav, kód, grafickou funkci, volání funkcí apod.

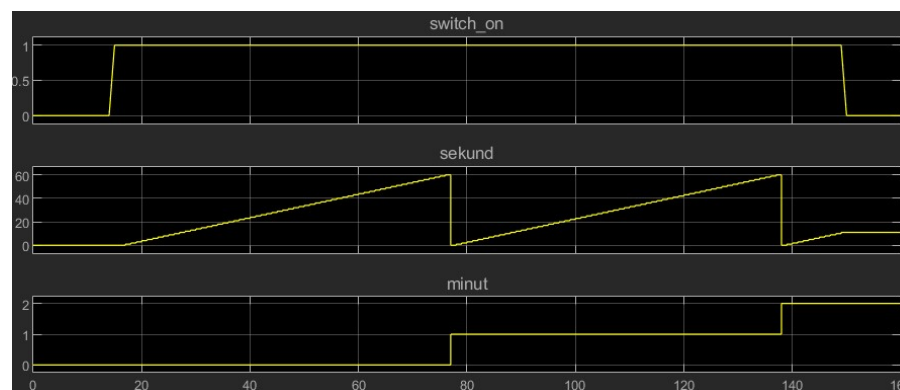
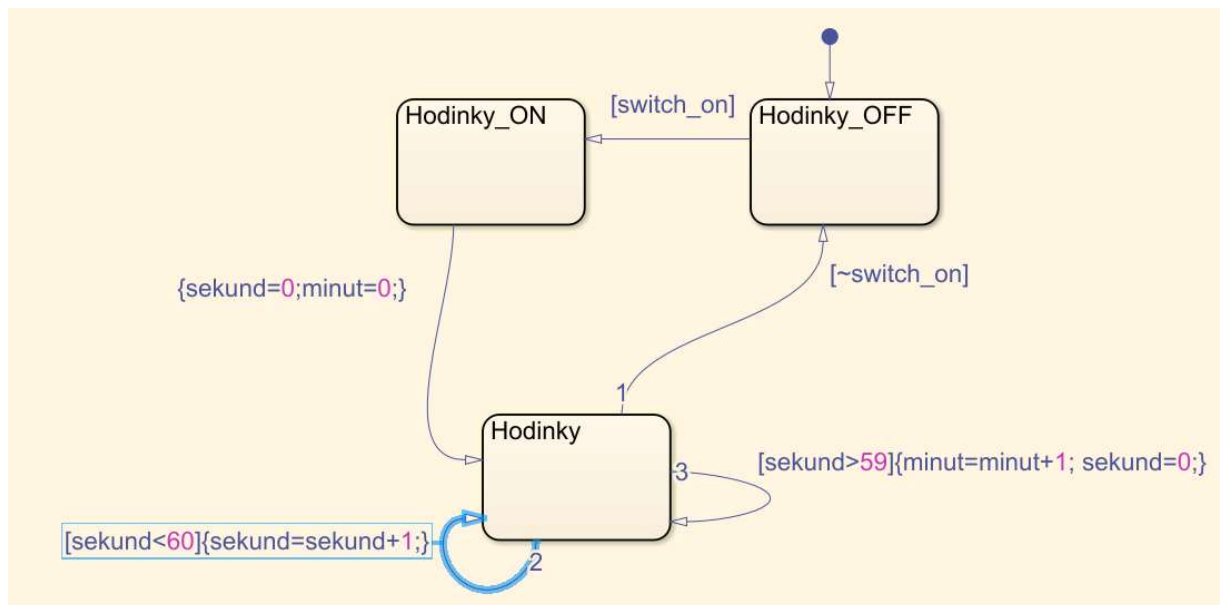


Jednoduchý flowchart (graf)

Příklad jednoduchého grafu se dvěma stavy, dvěma vstupy a jedním výstupem



Jednoduchý model stopky

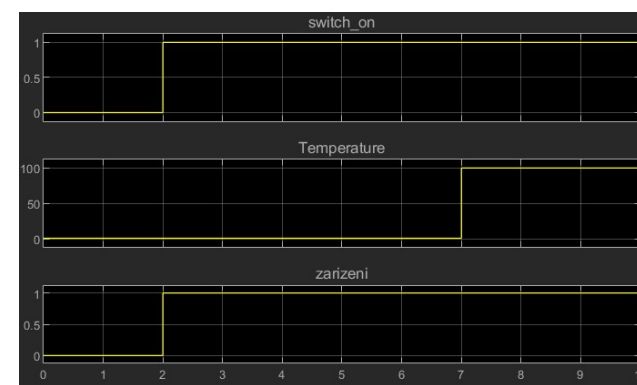
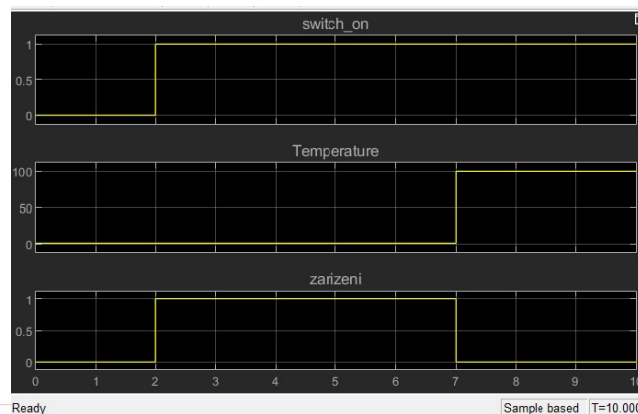
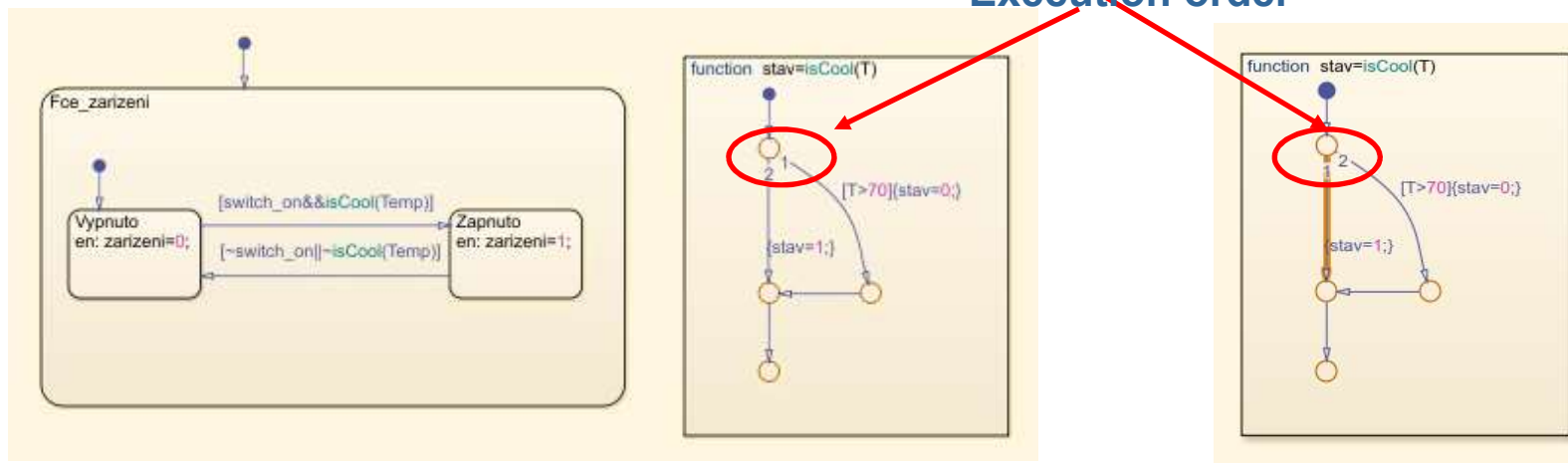


Vliv pořadí provádění stavů a větvení

Důležité je správné nastavení operací v případě grafických rozhodovacích diagramů

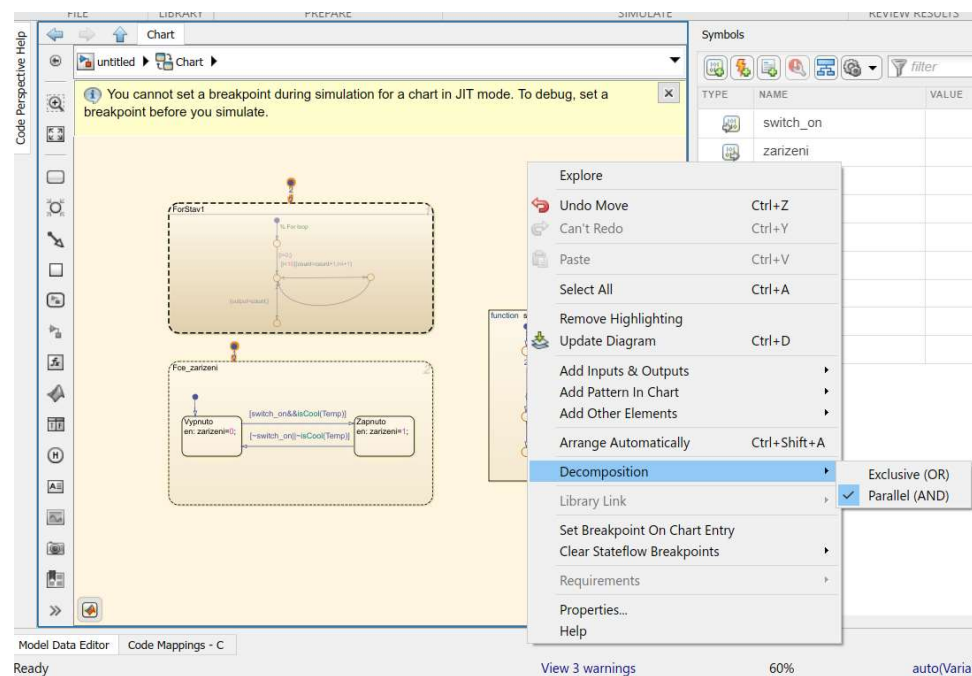
Příklad `if(T>70) stav=0 else stav=1`

Execution order



Provádění stavů

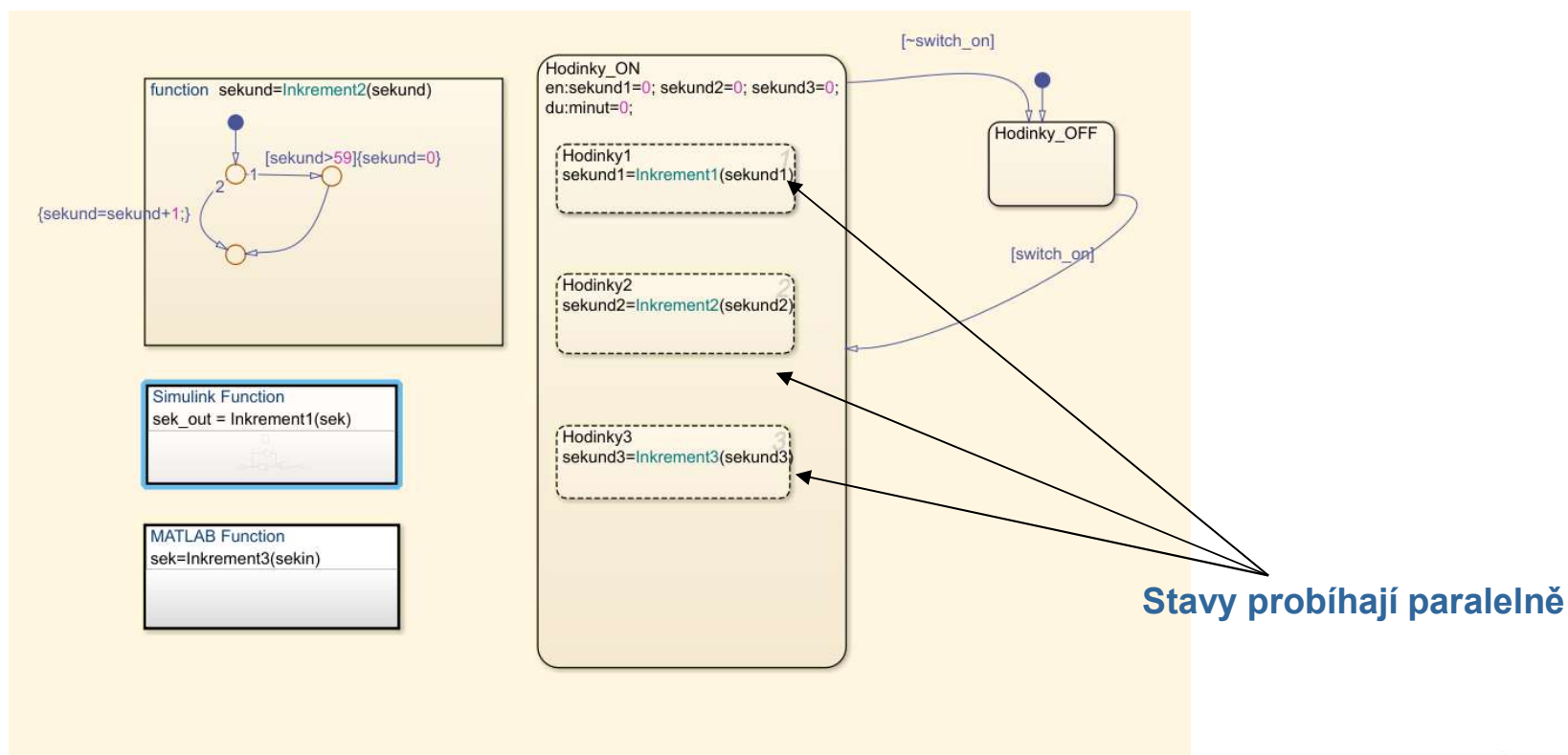
- Stavy jsou implicitně prováděné TOP to BOTTOM a zleva doprava (pokud není upraveno pořadí bloků)
- Mohou být prováděny exkluzivně tj. pouze jeden stav je aktivní (stavové bubliny plnou čarou) nebo paralelně (stavové bubliny čárkované) a poté je nutné hlídat pořadí provádění speciálně pokud sdílí proměnné apod.



Funkce ve Stateflow

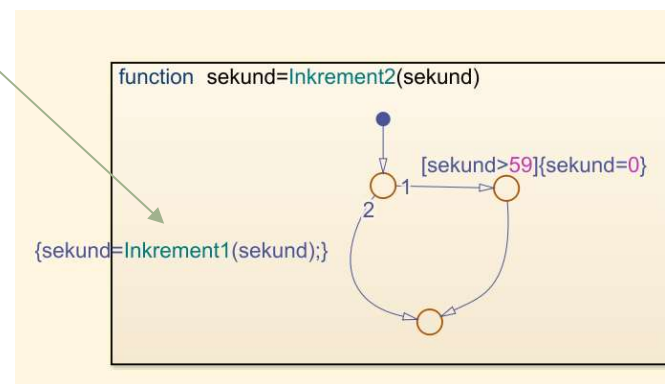
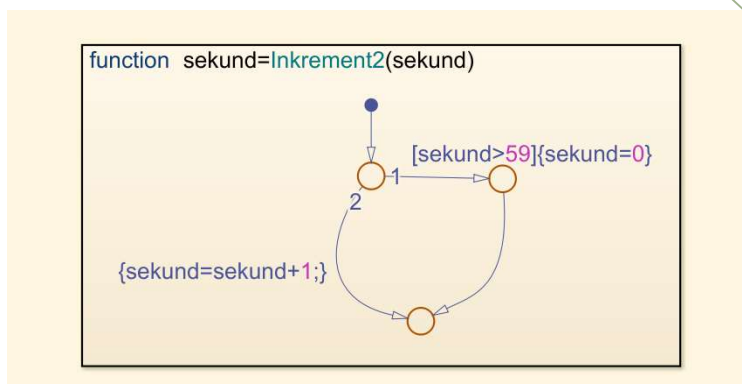
Ve Stateflow je možné dělat 3 druhy funkcí

- Grafické funkce – pomocí Stateflow bloků
- Simulink funkce – pomocí základních Simulinkovských bloků
- Matlab funkce – pomocí Matlab kódu



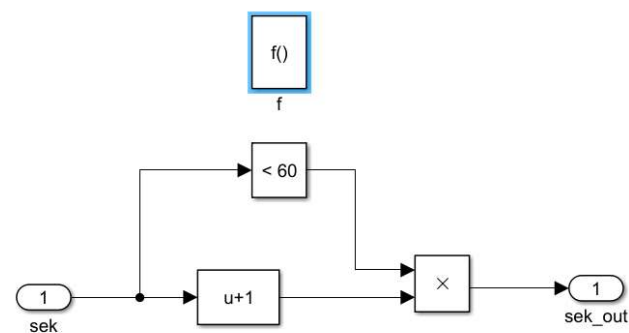
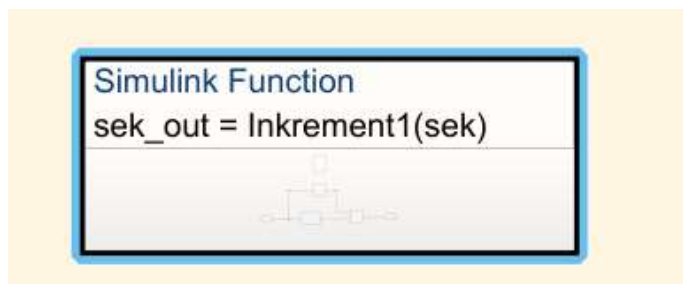
Grafická funkce

- pomocí se pomocí Stateflow bloků nejčastěji s použitím junctions a transitions
- **Funkce může volat funkce**



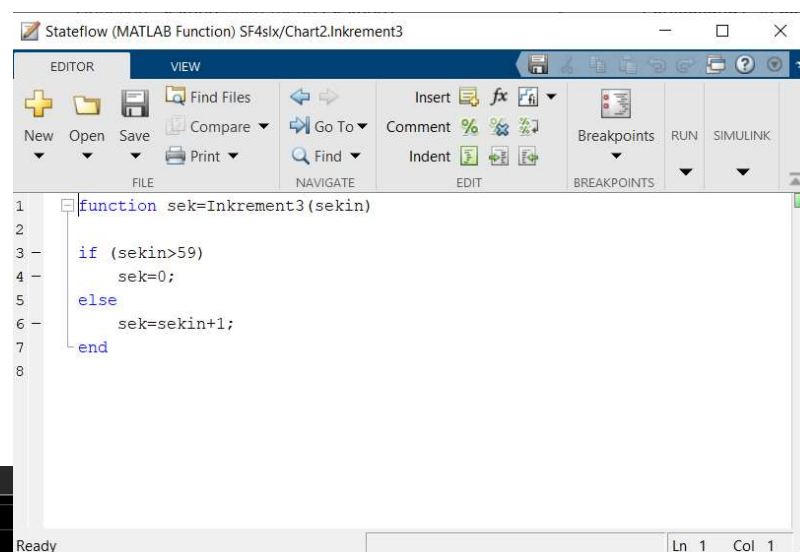
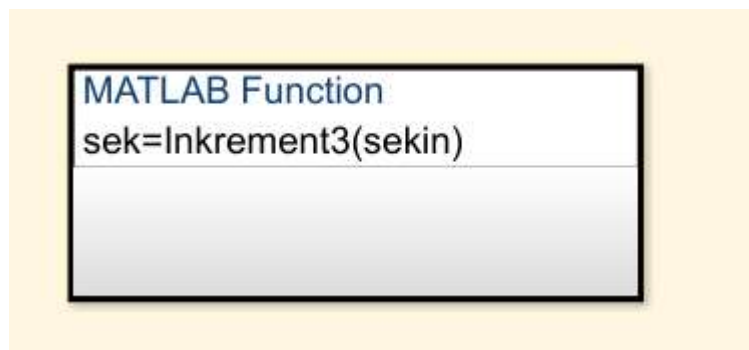
Simulink funkce

- Vhodná pro využití silných stránek Simulinku tj. např. pro dynamické systémy

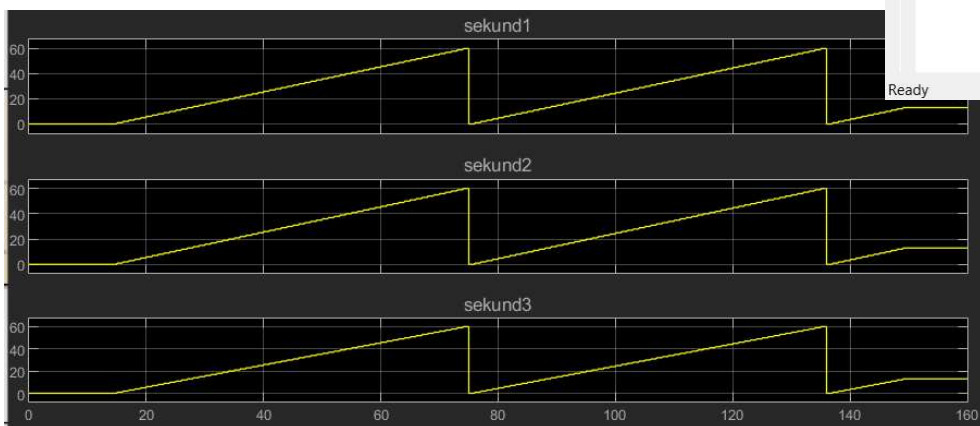


Matlab funkce

- Možnost volání funkcí Matlabu např. FFT apod.



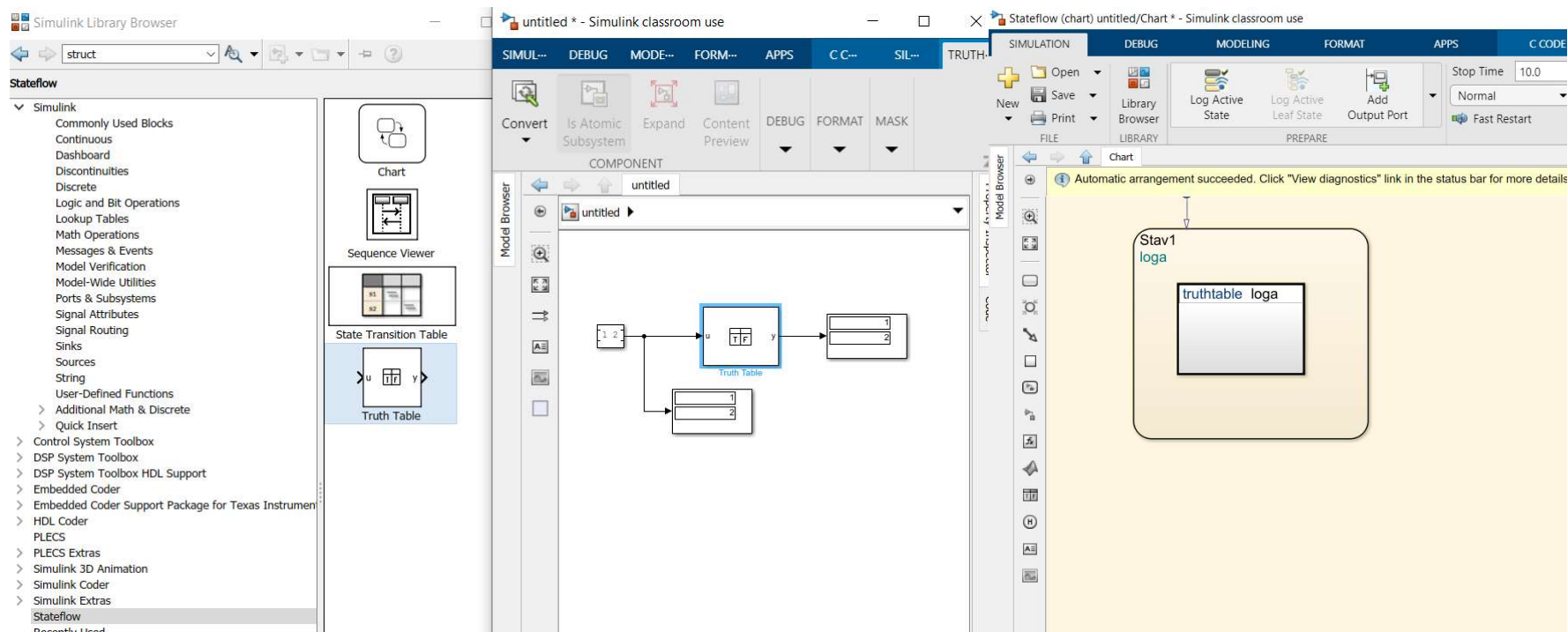
```
function sek=Inkrement3(sekin)
1
2
3 if (sekin>59)
4     sek=0;
5 else
6     sek=sekin+1;
7 end
8
```



Pravdivostní tabulka

Pravdivostní tabulka (Truth table)

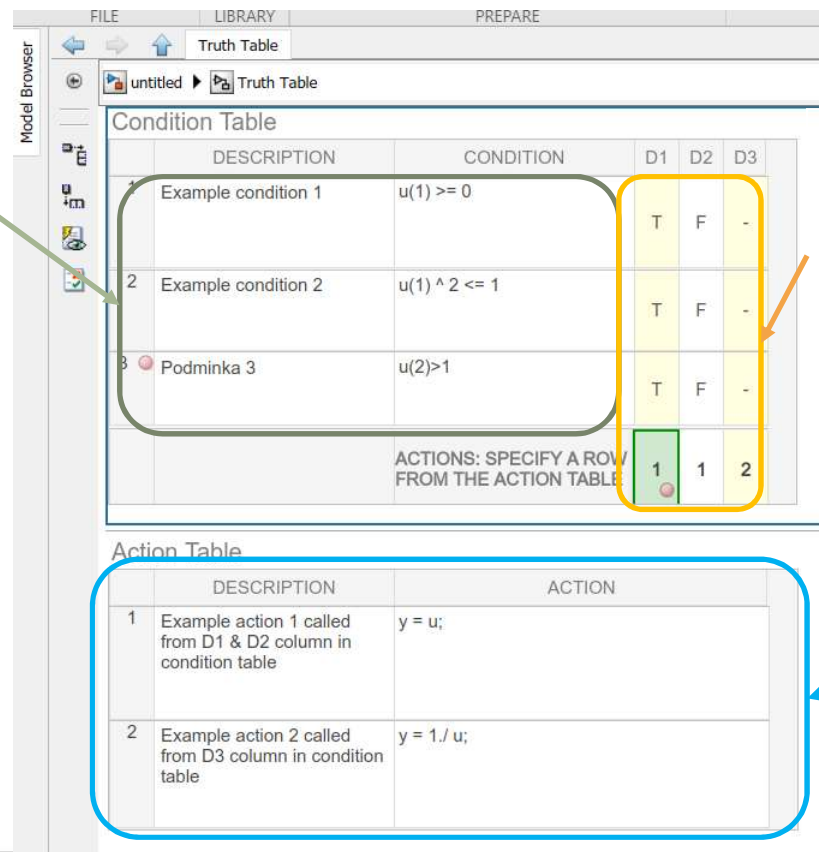
- Slouží k vytvoření logické pravdivostní tabulky ve formě co se má stát když...
- Může být součástí Stateflow diagramu i samostatně v Simulinku



Pravdivostní tabulka

- Jednotlivé dílčí podmínky jsou sloučeny v logické AND formě
- Např. výraz (složená podmínka) $D1=(u(1)>=0)\&\&(u(1)^2<=0)\&\&(u(2)>1)$

Dílčí podmínky



The screenshot shows a software interface with a 'Model Browser' on the left and a main workspace. The workspace contains two tables: a 'Condition Table' and an 'Action Table'.

Condition Table:

	DESCRIPTION	CONDITION	D1	D2	D3
1	Example condition 1	$u(1) \geq 0$	T	F	-
2	Example condition 2	$u(1)^2 \leq 1$	T	F	-
3	Podminka 3	$u(2) > 1$	T	F	-
ACTIONS: SPECIFY A ROW FROM THE ACTION TABLE			1	1	2

Action Table:

	DESCRIPTION	ACTION
1	Example action 1 called from D1 & D2 column in condition table	$y = u;$
2	Example action 2 called from D3 column in condition table	$y = 1./ u;$

Výraz pomocí nastavení stavů dílčích podmínek (T, F, -) a selekce požadované výstupní funkce

Výstupní funkce

Efektivita modelování ve Stateflow

Pravdivostní tabulka

Umožňuje práci s datovými typy a generování kódu do C a Matlabu

DESCRIPTION	CONDITION	D1	D2	D3
1 Example condition 1	$u(1) \geq 0$	T	F	-
2 Example condition 2	$u(1)^2 \leq 1$	T	F	-
3 Podminka 3	$u(2) > 1$	T	F	-

DESCRIPTION	ACTION
1 Example action 1 called from D1 & D2 column in condition table	$y = u;$
2 Example action 2 called from D3 column in condition table	$y = 1/u;$

Contents

Summary

[Subsystem Report](#)

[Code Interface Report](#)

[Traceability Report](#)

[Static Code Metrics Report](#)

[Code Replacements Report](#)

[Coder Assumptions](#)

Generated Code

[-] Main file

[ert_main.c](#)

[-] Model files

[Truth.c](#)

[Truth.h](#)

[Truth_private.h](#)

[Truth_types.h](#)

[+] Utility files (1)

```

33 void Truth_step(void)
34 {
35     boolean_T aVarTruthTableCondition_1;
36     boolean_T aVarTruthTableCondition_2;
37
38     /* Truth Table: '<Root>/Truth Table' incorporates:
39      * Inport: '<Root>/u'
40      */
41     /* Example condition 1 */
42     aVarTruthTableCondition_1 = (Truth_U.u[0] >= 0.0);
43
44     /* Example condition 2 */
45     aVarTruthTableCondition_2 = (Truth_U.u[0] * Truth_U.u[0] <= 1.0);
46
47     /* Podminka 3 */
48     if (aVarTruthTableCondition_1 && aVarTruthTableCondition_2) {
49         /* Output: '<Root>/y' */
50         /* Example action 1 called from D1 & D2 column in condition table */
51         Truth_Y.y[0] = Truth_U.u[0];
52         Truth_Y.y[1] = Truth_U.u[1];
53     } else if ((!aVarTruthTableCondition_1) && (!aVarTruthTableCondition_2)) {
54         if (!(Truth_U.u[1] > 1.0)) {
55             /* Output: '<Root>/y' */
56             /* Example action 1 called from D1 & D2 column in condition table */
57             Truth_Y.y[0] = Truth_U.u[0];
58             Truth_Y.y[1] = Truth_U.u[1];
59         } else {
60             /* Output: '<Root>/y' */
61             /* Default */
62             /* Example action 2 called from D3 column in condition table */
63             Truth_Y.y[0] = 1.0 / Truth_U.u[0];
64             Truth_Y.y[1] = 1.0 / Truth_U.u[1];
65         }
66     } else {
67         /* Output: '<Root>/y' */
68         /* Default */
69         /* Example action 2 called from D3 column in condition table */
70         Truth_Y.y[0] = 1.0 / Truth_U.u[0];
71         Truth_Y.y[1] = 1.0 / Truth_U.u[1];
72     }

```

Block: untitled/Truth Table

EDITOR VIEW

New Open Save Find Files Compare Go To Comment % Find Breakpoints Run Model Stop Model Build Model SIMULINK

FILE NAVIGATE EDIT BREAKPOINTS RUN

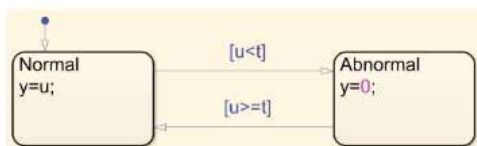
```

1 function y = fcn(u)
2
3     aVarTruthTableCondition_1 = false;
4     aVarTruthTableCondition_2 = false;
5     aVarTruthTableCondition_3 = false;
6
7
8     % Example condition 1
9
10    aVarTruthTableCondition_1 = logical(u(1) >= 0);
11
12    % Example condition 2
13
14    aVarTruthTableCondition_2 = logical(u(1) ^ 2 <= 1);
15
16    % Podminka 3
17
18    aVarTruthTableCondition_3 = logical(u(2) > 1);
19
20    if (aVarTruthTableCondition_1 && aVarTruthTableCondition_2)
21        aFcnTruthTableAction_1();
22    elseif (~aVarTruthTableCondition_1 && ~aVarTruthTableCondition_2 && ~aVarTruthTableCondition_3)
23        aFcnTruthTableAction_1();
24    else % Default
25        aFcnTruthTableAction_2();
26    end
27
28    function aFcnTruthTableAction_1()
29
30        % Example action 1 called from D1 & D2 column in condition table
31
32        y = u;
33
34    function aFcnTruthTableAction_2()

```

MATLAB code expands, while Stateflow chart remains compact

simple logic

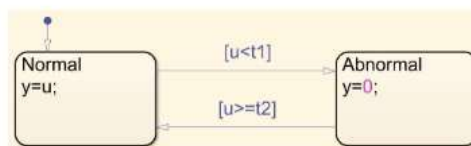


```

for i = 1:length(inData)
    if(inData(i) >= t)
        outData(i) = inData(i);
    else
        outData(i) = 0;
    end
end

```

+ hysteresis

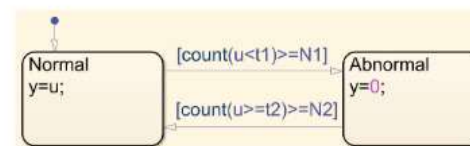


```

inNormalRegion = true;
for i = 1:length(inData)
    if(inNormalRegion && (inData(i)<t1))
        inNormalRegion = false;
    elseif(~inNormalRegion && (inData(i)>=t2))
        inNormalRegion = true;
    end
    if(inNormalRegion)
        outData(i) = inData(i);
    else
        outData(i) = 0;
    end
end
end

```

+ hysteresis + debouncing



```

inNormalRegion = true;
counter = 0;
for i=1:length(inData)
    if(inNormalRegion)
        if(inData(i)<t1)
            counter = counter+1;
            if(counter>=N1)
                inNormalRegion = false;
            end
        else
            counter = 0;
        end
    else
        if(inData(i)>=t2)
            counter = counter+1;
            if(counter>=N2)
                inNormalRegion = true;
            end
        else
            counter = 0;
        end
    end
    if(inNormalRegion)
        outData(i) = inData(i);
    else
        outData(i) = 0;
    end
end
end

```

Regionální inovační centrum elektrotechniky
Fakulta elektrotechnická
Západočeská univerzita v Plzni

Příští přednáška

Automatické generování kódu

Jakub Talla

Regionální inovační centrum elektrotechniky
Fakulta elektrotechnická
Západočeská univerzita v Plzni

Děkuji za pozornost!

Adresa: Univerzitní 26
306 14 Plzeň
Česká republika

Tel: +420 377 634 443
Fax: +420 377 634 402

Email: talic@kev.zcu.cz

www.rice.zcu.cz