Trans1 vs. Trans2

(without further optimizations)

Example	Version	|StateVector| (byte)	#stored states	#matched states	#transitions	depth	|totalMemory| (MByte)	|time| (sec)
HalfAdder	Trans1	44	14	0	**14**	**17**	2.501	0
	Trans2	**40**	22	0	22	25	2.501	0
-DNOREDUCE	Trans1\POR	44	14	0	14,stop	17	2.501	0
FullAdder	Trans1	128	97	0	**97**	**130**	2.501	0.01
	Trans2	**116**	178	0	178	211	2.501	0.01
-DNOREDUCE	Trans1\POR	128	97	0	97,stop	130	2.501	0
Adder [_X]	Trans1	224	9083	517	**9600**	**2336**	3.868	0.07
	Trans2	**192**	11067	517	11584	2864	**3.770**	**0.05**
-DNOREDUCE	Trans1\POR	224	12222	5919	18141	2336	4.258	**0.1**
RedundantBatterySystem2	Trans1	164	**13438**	**4921**	**18359**	**9999ok**	**4.649**	**0.12**
	~~Trans2~~	~~128~~	~~7833+~~	~~1+~~	~~7834+~~	~~9999err~~	~~3.379+~~	~~0.05+~~
pan -m1000000	Trans2+	**128**	1007794	13585	1021379	526019	136.815	3.14
-DNOREDUCE	Trans1\POR	164	13438	6216	19654	9999ok	4.649	0.11
TransitionChangesDataport	Trans1	224	148	0	**148**	**192**	2.501	0.01
	Trans2	**136**	201	0	201	243	2.501	0.01
-DNOREDUCE	Trans1\POR	224	148	0	148,stop	192	2.501	0

Trans2 has a little smaller state vector but induces much more states (Why? Is one global variable less efficient than two local copies?).

- May be the best solution: combination of Trans1 + Trans2, i.e. global variables in Trans2 naming schema for LTL property but with declarator “local” and data exchange between process via messages.