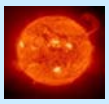
Scheduling and Optimization of Fault-Tolerant Embedded Systems

Viacheslav Izosimov

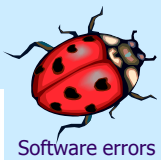
embedded systems lab linköpings universitet

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE


Electromagnetic interference (EMI)


Radiation


Crosstalk


Software errors (Heisenbugs)


Power supply fluctuations


Lightning storms

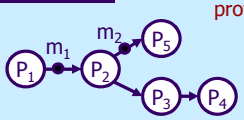

Internal EMI

Transient and Intermittent Faults (Transient Faults)

Fault Occurrences

Processes: Re-execution, Active Replication, Rollback Recovery with Checkpointing

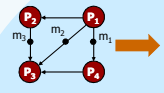
Messages: Fault-tolerant predictable protocol

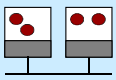


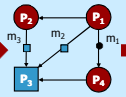
Maximum k transient faults within each application run

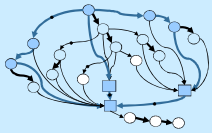
Errors caused by transient faults have to be tolerated before they crash the system or lead to dramatic quality deterioration

We have developed techniques to address fault tolerance aspects during scheduling and design optimization of embedded systems in order to provide efficient design solutions under resource constraints


Application

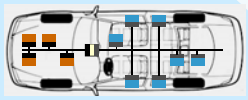

Mapping


Transparency

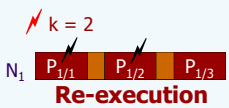

Fault-Tolerant Conditional Process Graph (FT-CPG)

N_1	true	F_{P_i}	$\overline{F}_{P_i}$	$F_{P_i} \wedge F_{P_j}$	$F_{P_i} \wedge \overline{F}_{P_j}$	$F_{P_i} \wedge \overline{F}_{P_j} \wedge F_{P_k}$
P_1	0	35		70		
P_2			30	100	65	90
m_1			31	100	66	
m_2			105	105	105	
m_3				120		120

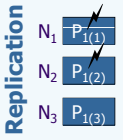
Scheduling Tables

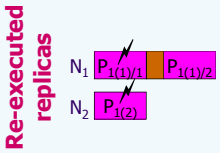

Final Product

Fault-Tolerance Policy Assignment

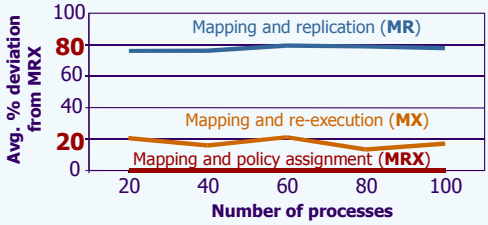


Re-execution





Which fault-tolerance policy should be assigned to which process?



Transparency vs. Performance

Transparency is achieved with frozen processes and messages that are always scheduled at one time independent of external fault occurrences, which results in a much smaller number of system states

Good for debugging and testing but...

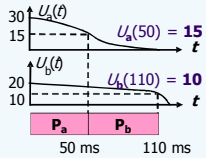
increased transparency

	0%			25%			50%			75%			100%		
	k=1	k=2	k=3	k=1	k=2	k=3	k=1	k=2	k=3	k=1	k=2	k=3	k=1	k=2	k=3
20	24	44	63	32	60	92	39	74	115	48	83	133	48	86	139
40	17	29	43	20	40	58	28	49	72	34	60	90	39	66	97
60	12	24	34	13	30	43	19	39	58	28	54	79	32	58	86
80	8	16	22	10	18	29	14	27	39	24	41	66	27	43	73

leads to the reduction in performance

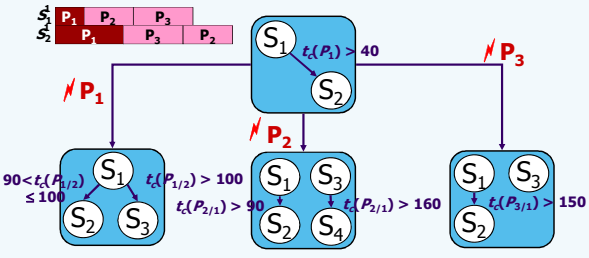
Utility-based Optimization with Soft and Hard Real-Time Constraints

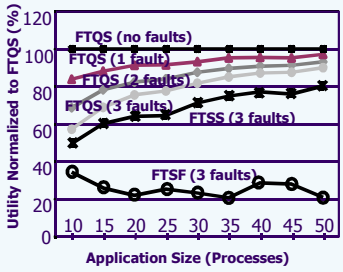
Soft processes with utility functions:



Overall utility:: $U = 15 + 10 = 25$

Hard process with hard deadline:





The quasi-static scheduling generates a set of static schedules that maximize the overall utility for a particular execution and fault scenario while satisfying hard deadlines

FTQS – quasi-static scheduling with fault tolerance
FTSS – static scheduling with fault tolerance
FTSF – quasi-static scheduling with straightforwardly-introduced fault tolerance

+ Global Optimization of Checkpointing

+ Mapping of Embedded Systems with Performance/Transparency Trade-off

Contact info: http://www.ida.liu.se/~viaiz