Karlsruher Institut für Technologie

Preemptable Partial Reconfiguration for Real-Time Computing with FPGAs

Enrico Rossi

Introduction

Computer platforms are evolving towards heterogeneous architectures:

- FPGAs
- SoCs (FPGA + Hardware processor)

These new architectures and their features are attractive for real-time systems. The most attractive feature is:

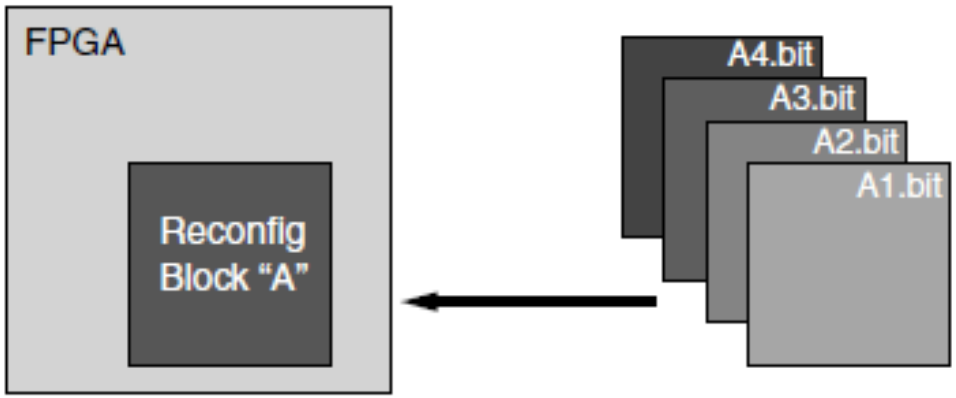
- Dynamic Partial Reconfiguration (DPR)

Introduction

Dynamic partial reconfiguration enables the possibility to reconfigure a portion of the FPGA at runtime, while the rest of the FPGA continues to operate.

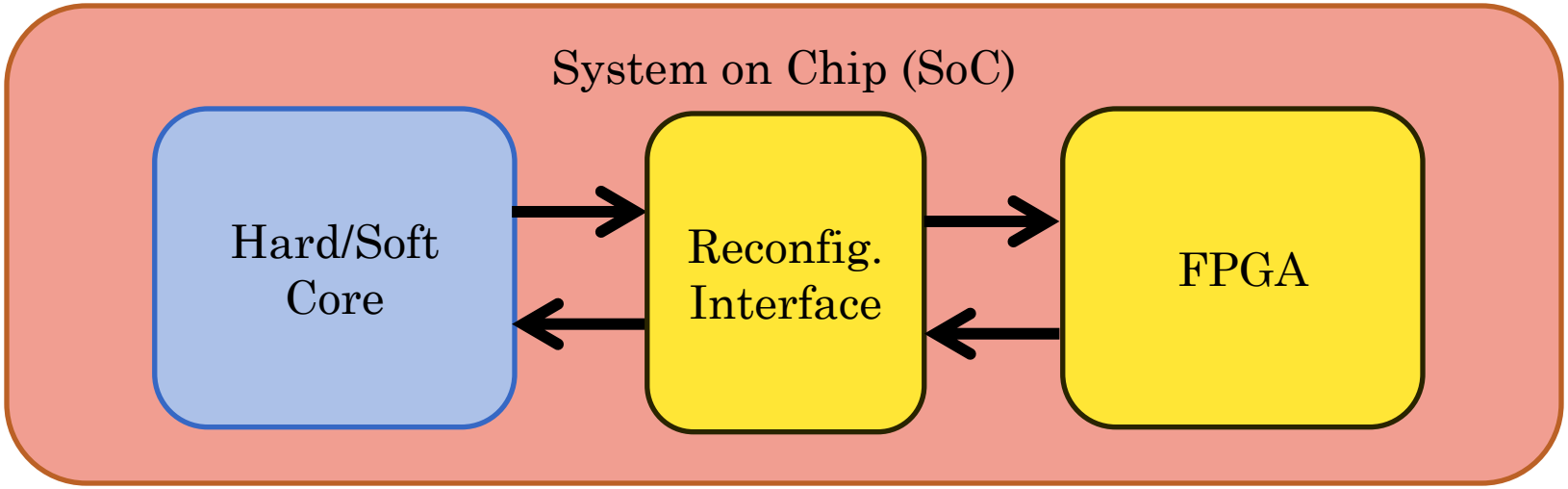
Different kind of systems can benefit from this feature to enhance the overall performance:

- Real-Time Systems
- Mixed-Criticality Systems



Introduction

The FPGA can be seen as a co-processor that exploits the hardware resources to carry out on-demand, computationally intensive tasks.



DPR allows the Core to load inside the FPGA the required hardware module only when it is necessary.

DPR Problems

Currently, the main problem is the Reconfiguration Interface.

- Each reconfiguration process must be finished or aborted.
- No possibility to resume a reconfiguration process.
- There are more than one Reconfig. Interface but only one at a time can be used.

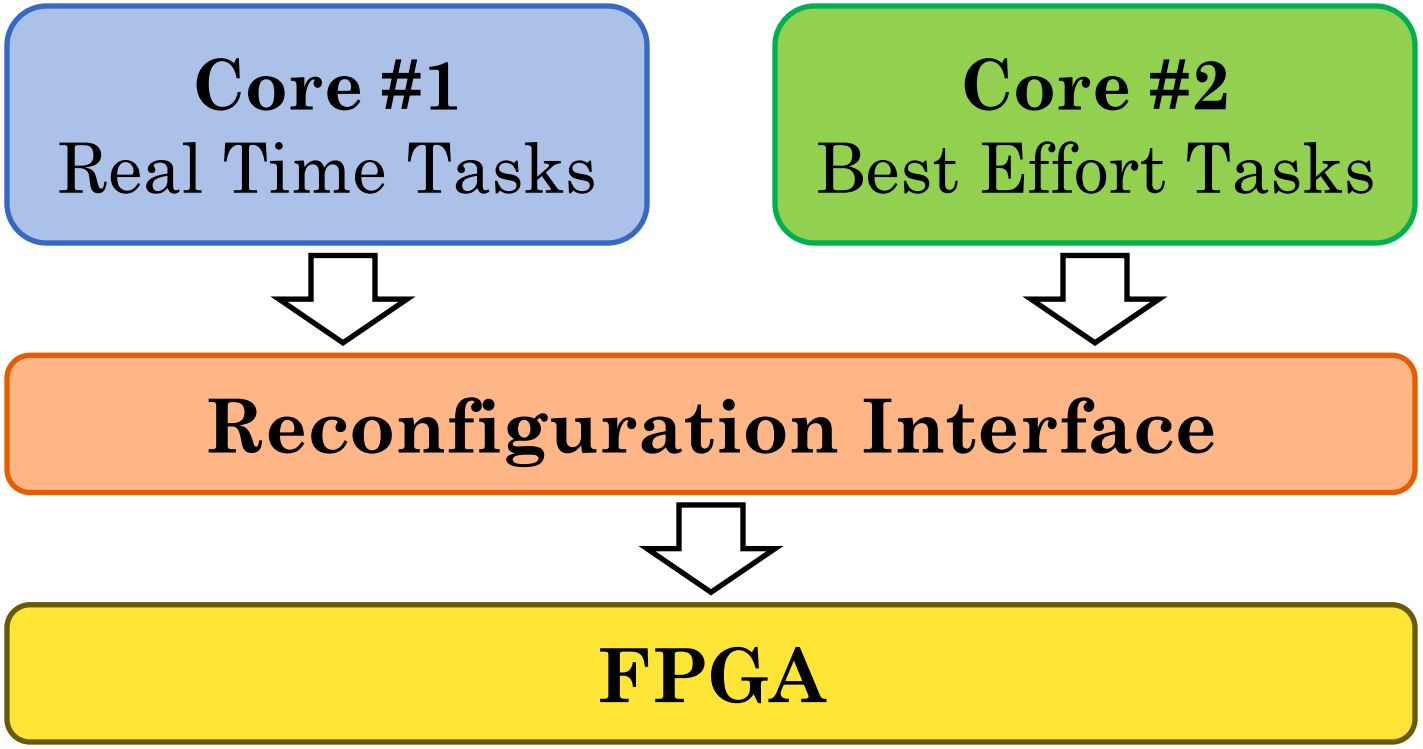
DPR Problems

Currently, the main problem is the Reconfiguration Interface.

- Each reconfiguration process must be finished or aborted.
- No possibility to resume a reconfiguration process.
- There are more than one Reconfig. Interface but only one at a time can be used.

The Reconfiguration Interface is NOT PREEMPTABLE!

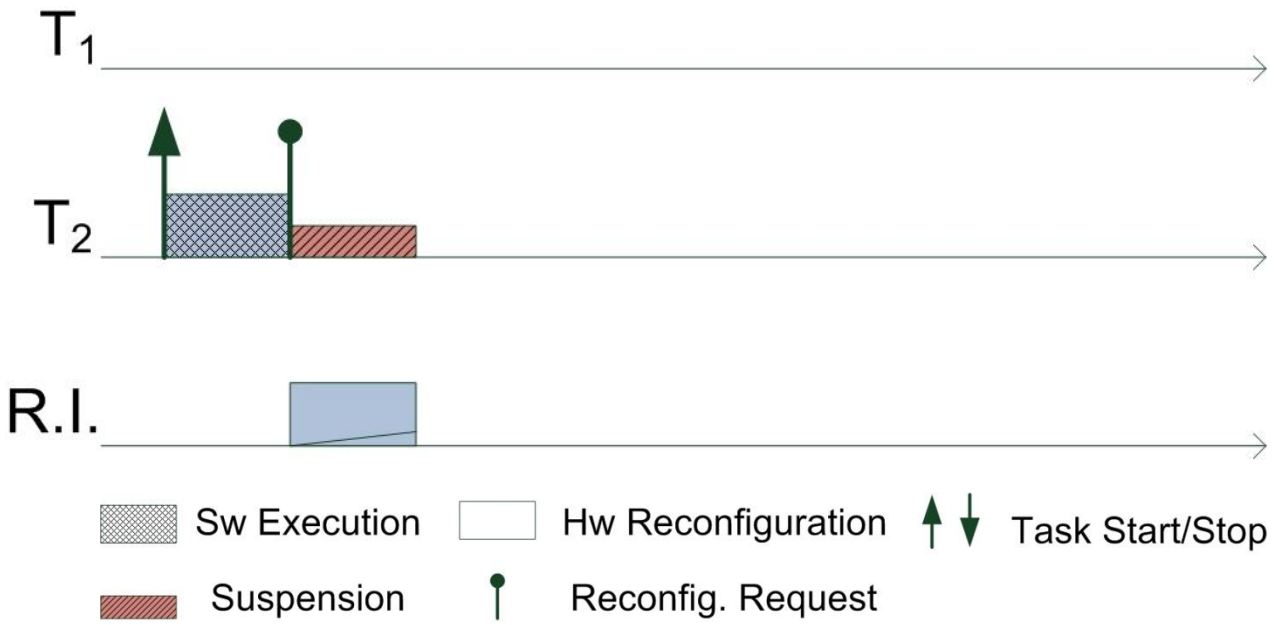
Multi Core Mixed Criticality Systems



Both RT tasks and BE tasks can trigger a hardware reconfiguration. The reconfiguration interface will be a shared resource between the two cores and among all tasks.

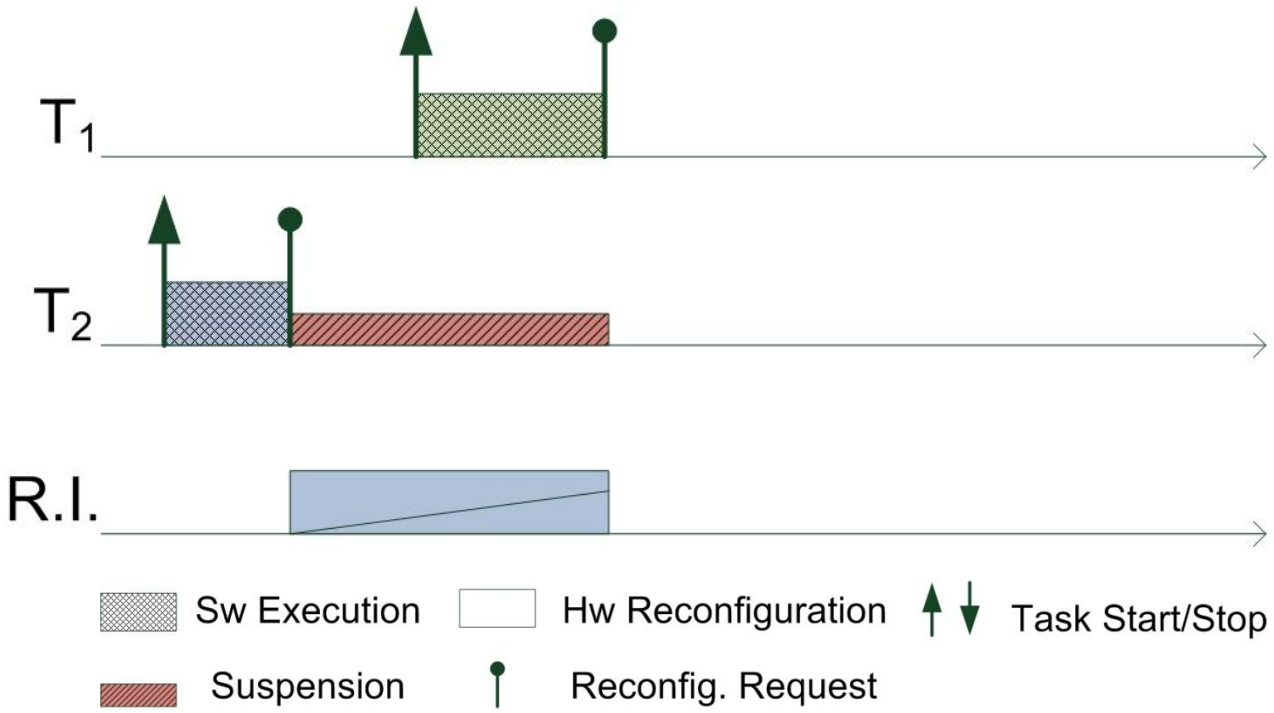
Priority Inversion

- SW tasks with Fixed Priority scheduling
- Each SW task can issue a hardware reconfiguration process
- Reconfiguration process can not be preempted or aborted



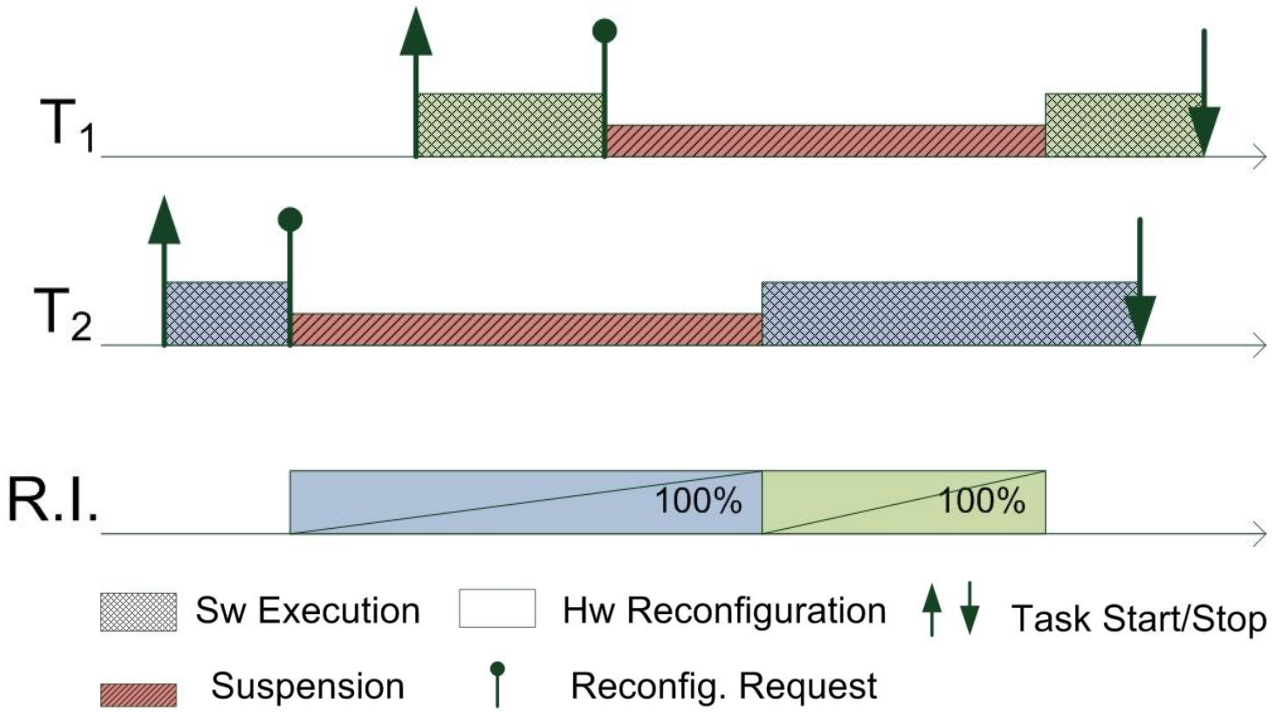
Priority Inversion

- SW tasks with Fixed Priority scheduling
- Each SW task can issue a hardware reconfiguration process
- Reconfiguration process can not be preempted or aborted



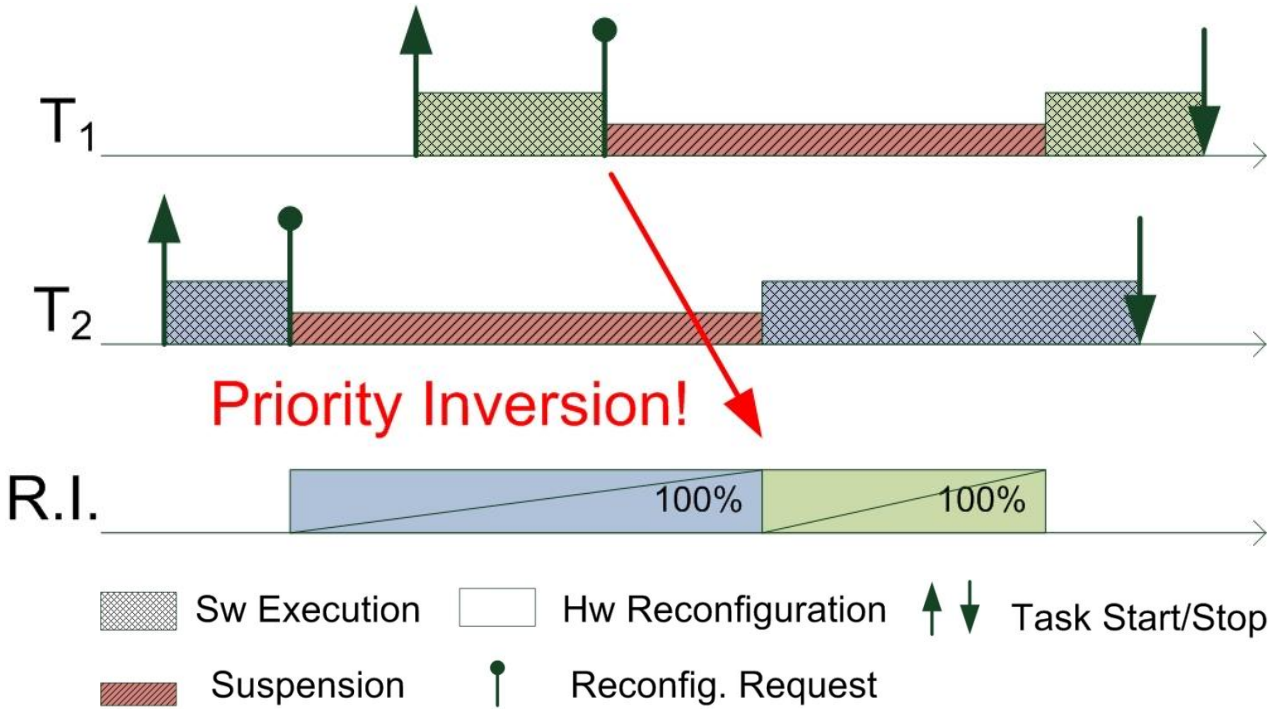
Priority Inversion

- SW tasks with Fixed Priority scheduling
- Each SW task can issue a hardware reconfiguration process
- Reconfiguration process can not be preempted or aborted



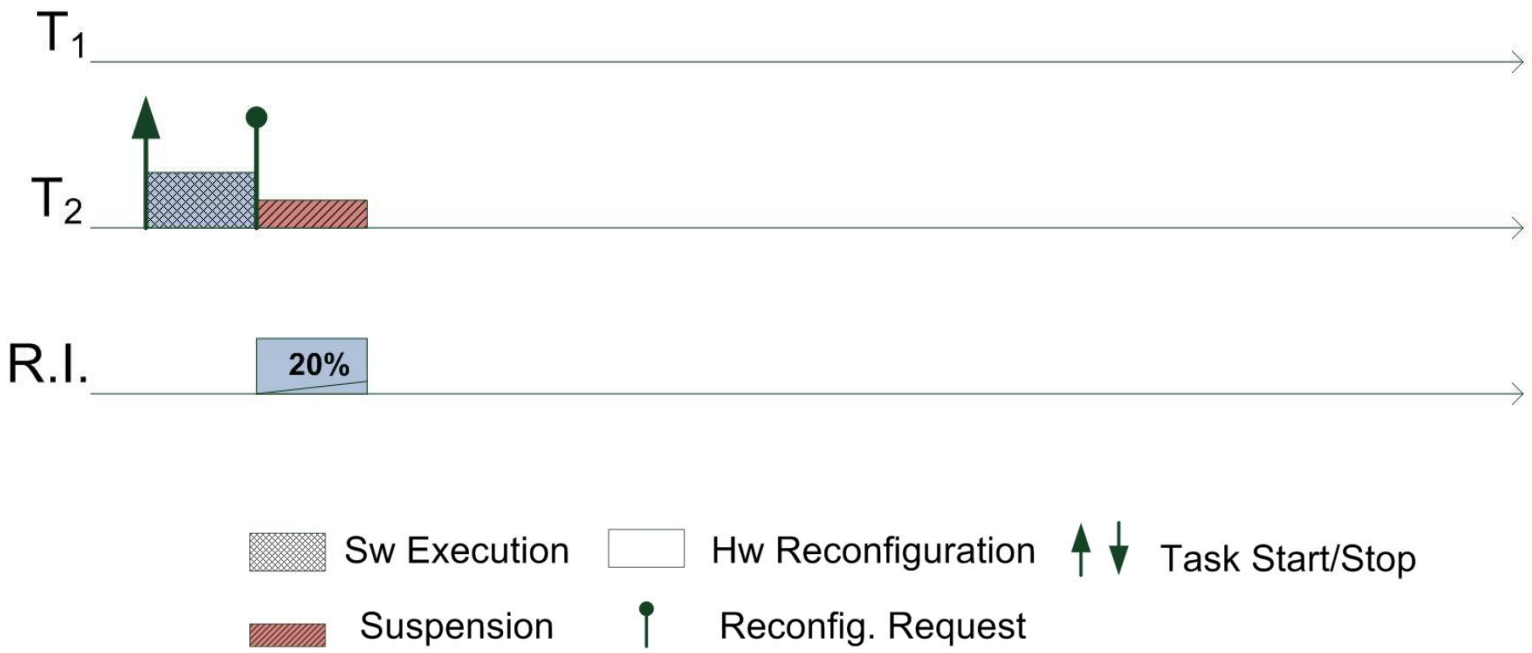
Priority Inversion

- SW tasks with Fixed Priority scheduling
- Each SW task can issue a hardware reconfiguration process
- Reconfiguration process can not be preempted or aborted



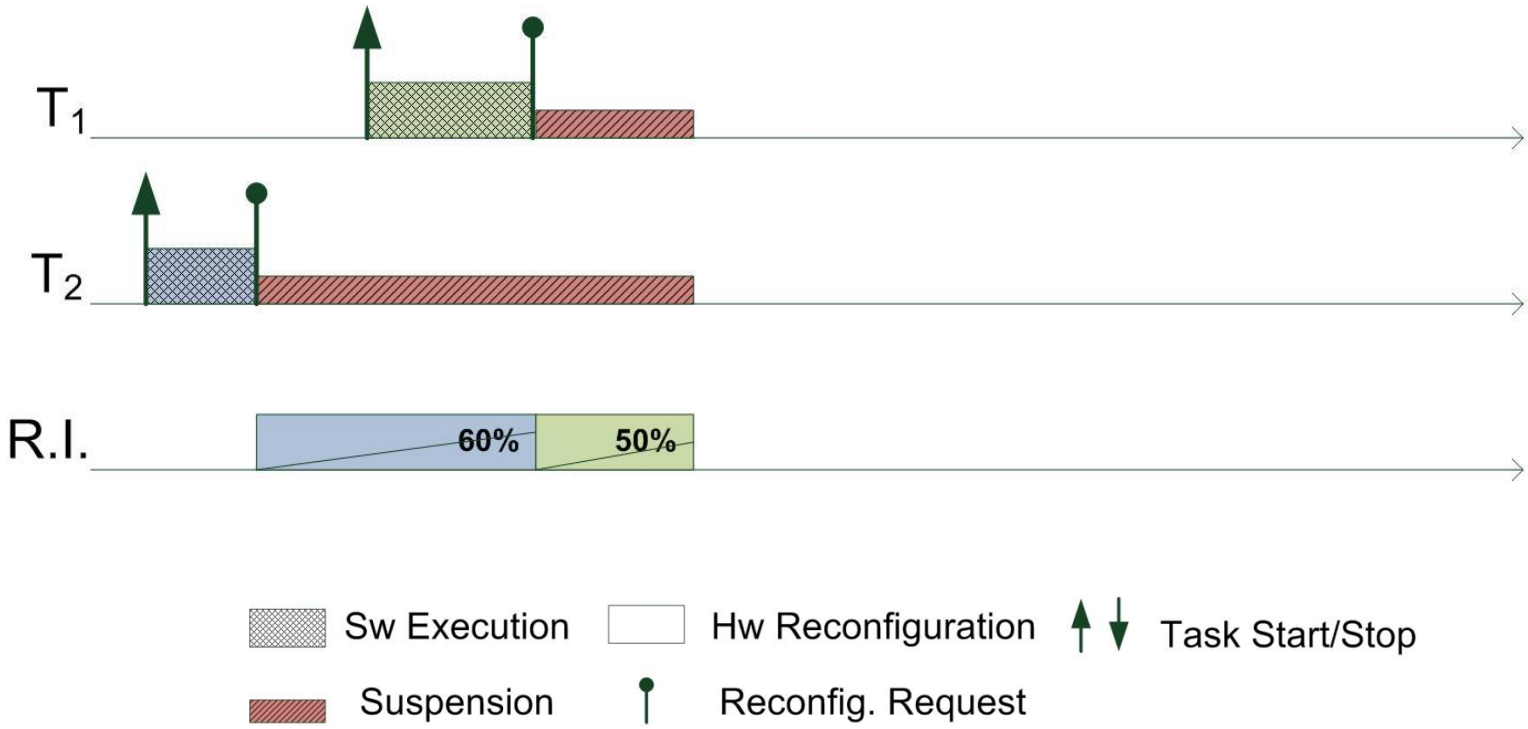
Starvation

- Real Time task (T1) has priority over Best Effort task (T2)
- The reconfiguration interface is not preemptive
- The reconfiguration process can be aborted
  - The reconfiguration process must be resumed from scratch



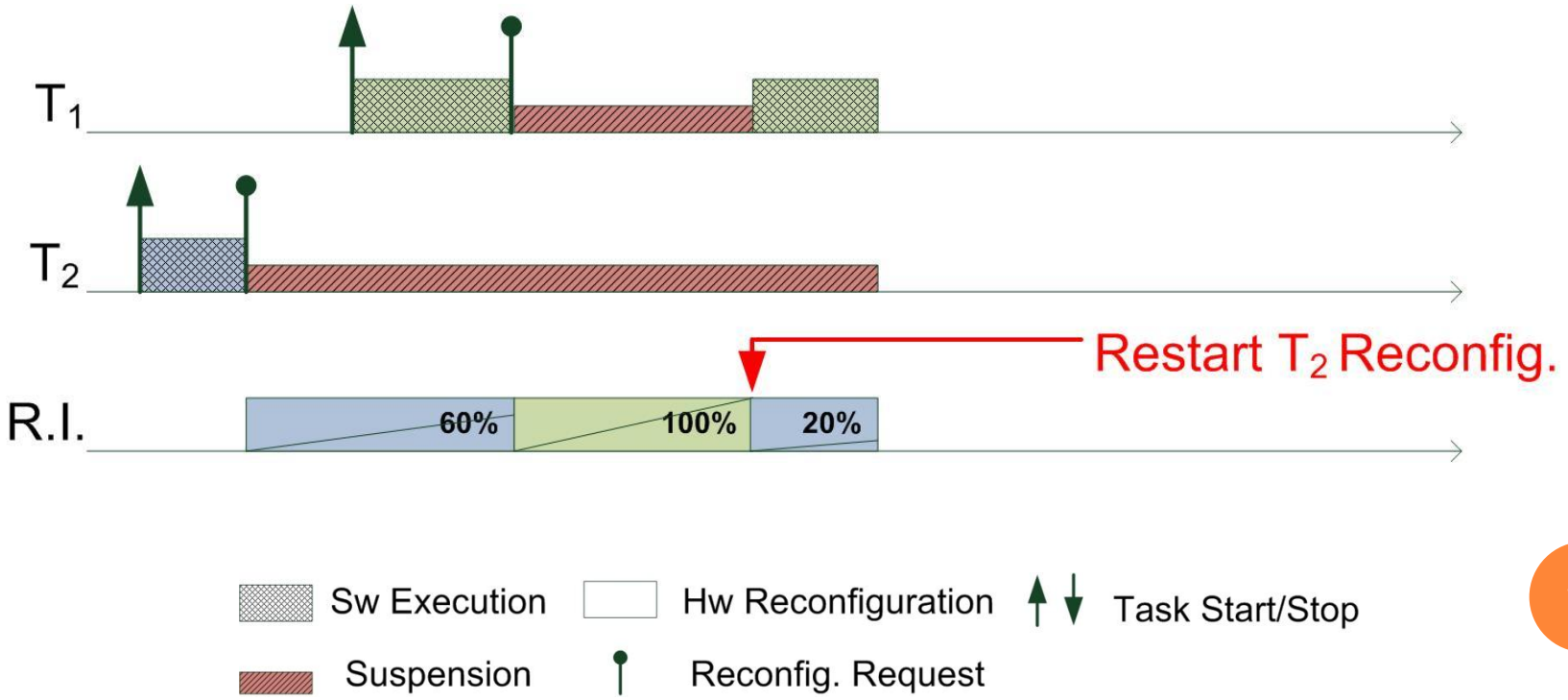
STARVATION

- Real Time task (T1) has priority over Best Effort task (T2)
- The reconfiguration interface is not preemptive
- The reconfiguration process can be aborted
  - The reconfiguration process must be resumed from scratch



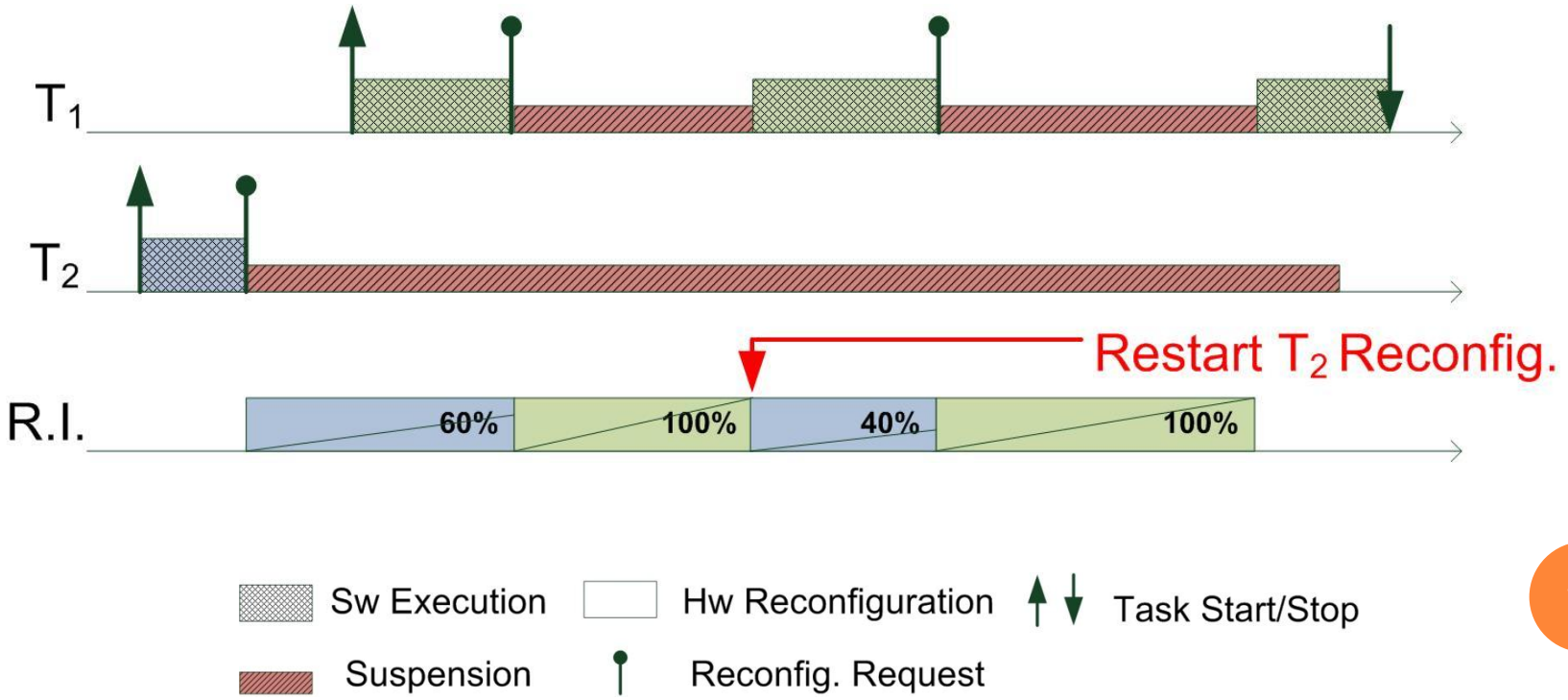
STARVATION

- Real Time task (T1) has priority over Best Effort task (T2)
- The reconfiguration interface is not preemptive
- The reconfiguration process can be aborted
  - The reconfiguration process must be resumed from scratch



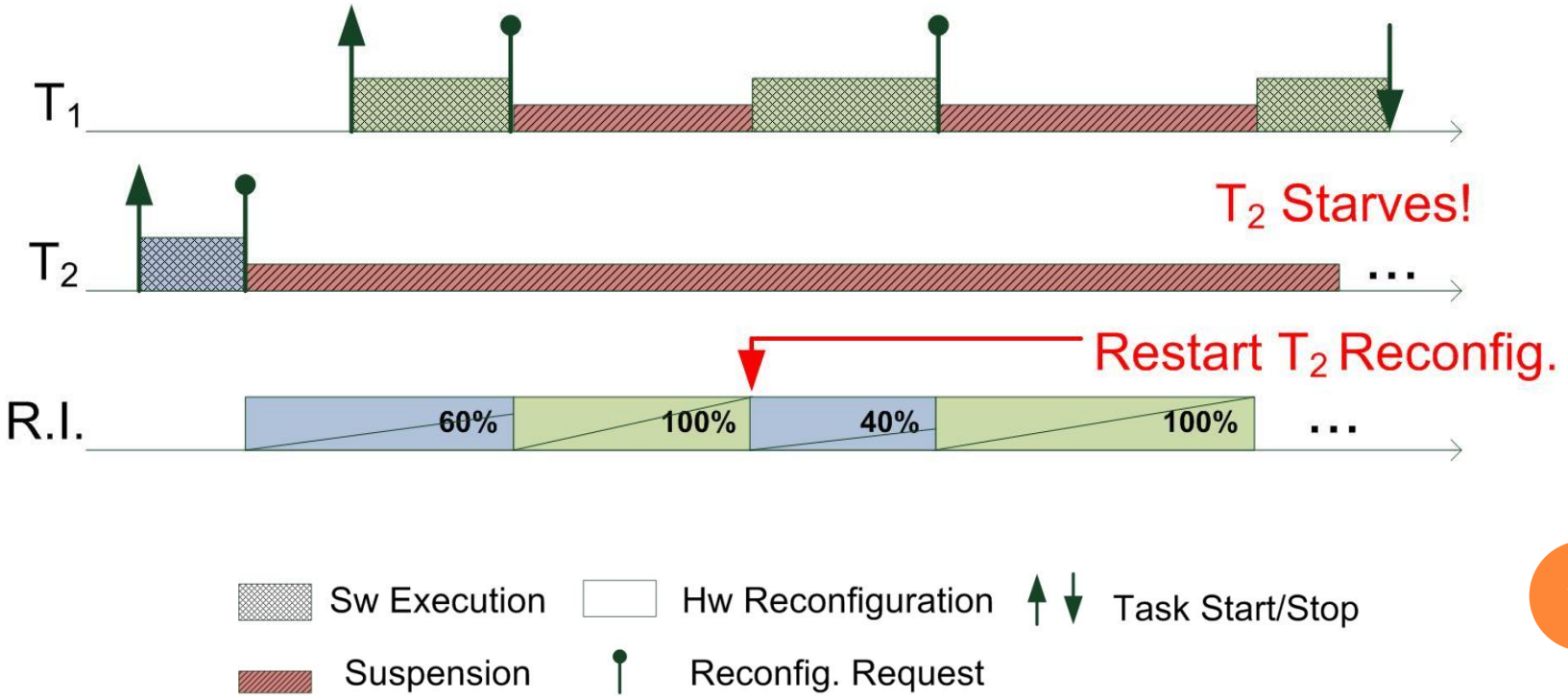
STARVATION

- Real Time task (T1) has priority over Best Effort task (T2)
- The reconfiguration interface is not preemptive
- The reconfiguration process can be aborted
  - The reconfiguration process must be resumed from scratch



Starvation

- Real Time task (T1) has priority over Best Effort task (T2)
- The reconfiguration interface is not preemptive
- The reconfiguration process can be aborted
  - The reconfiguration process must be resumed from scratch



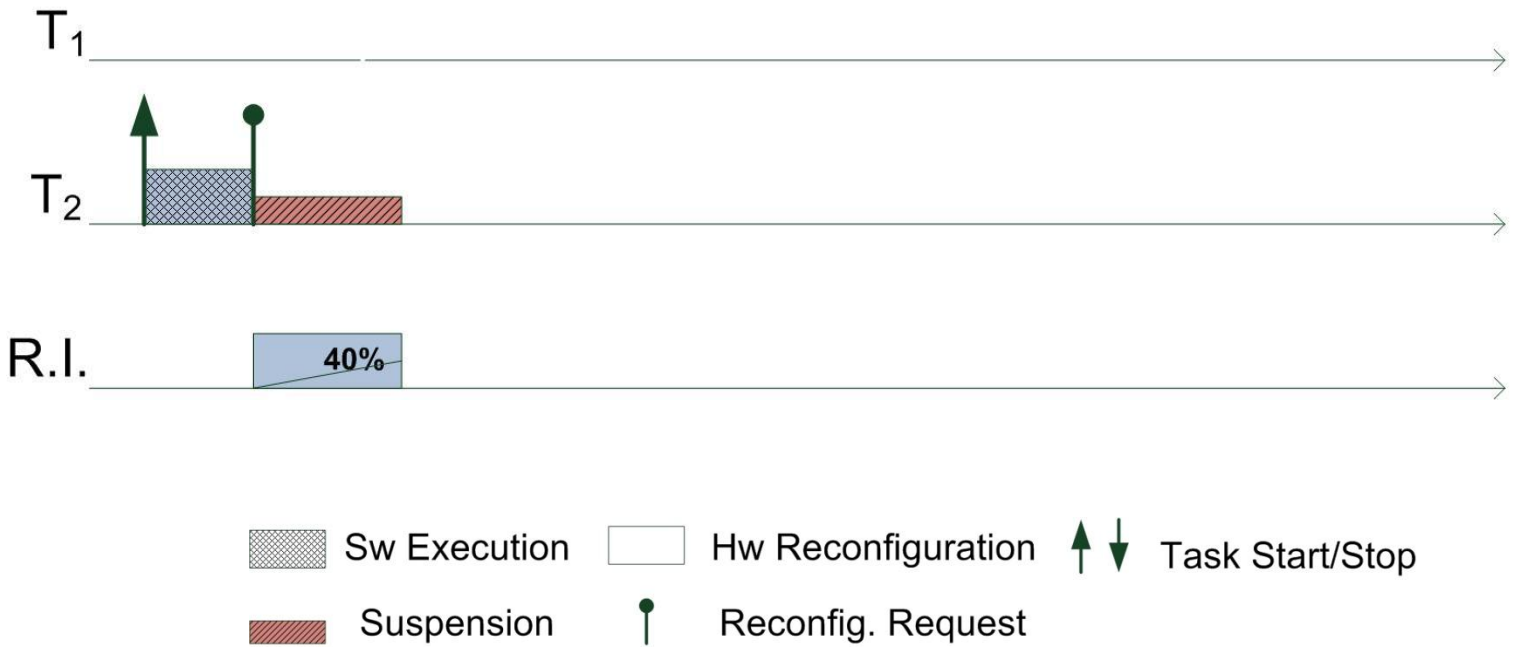
Priority Invertion and Starvation

Without a preemptable interface:

- The delay experienced by the higher priority task could be very high.
- High priority tasks could miss their deadline!
- Upper bound of the delay experienced by the Best Effort task cannot be found.

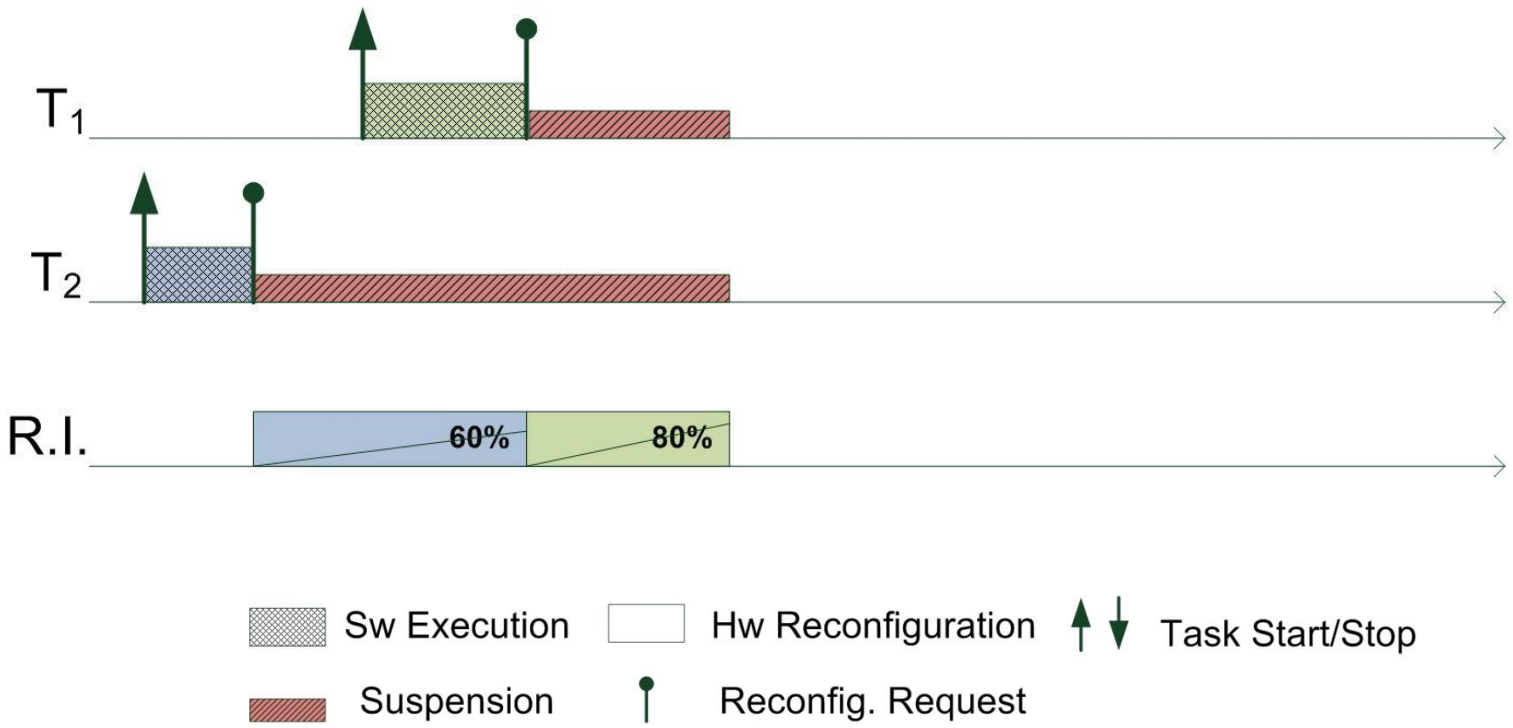
Preemptable Reconfiguration

- Real Time task (T1) has priority over Best Effort task (T2)
- The reconfiguration process is preemptable
  - Hardware reconfiguration can be resumed



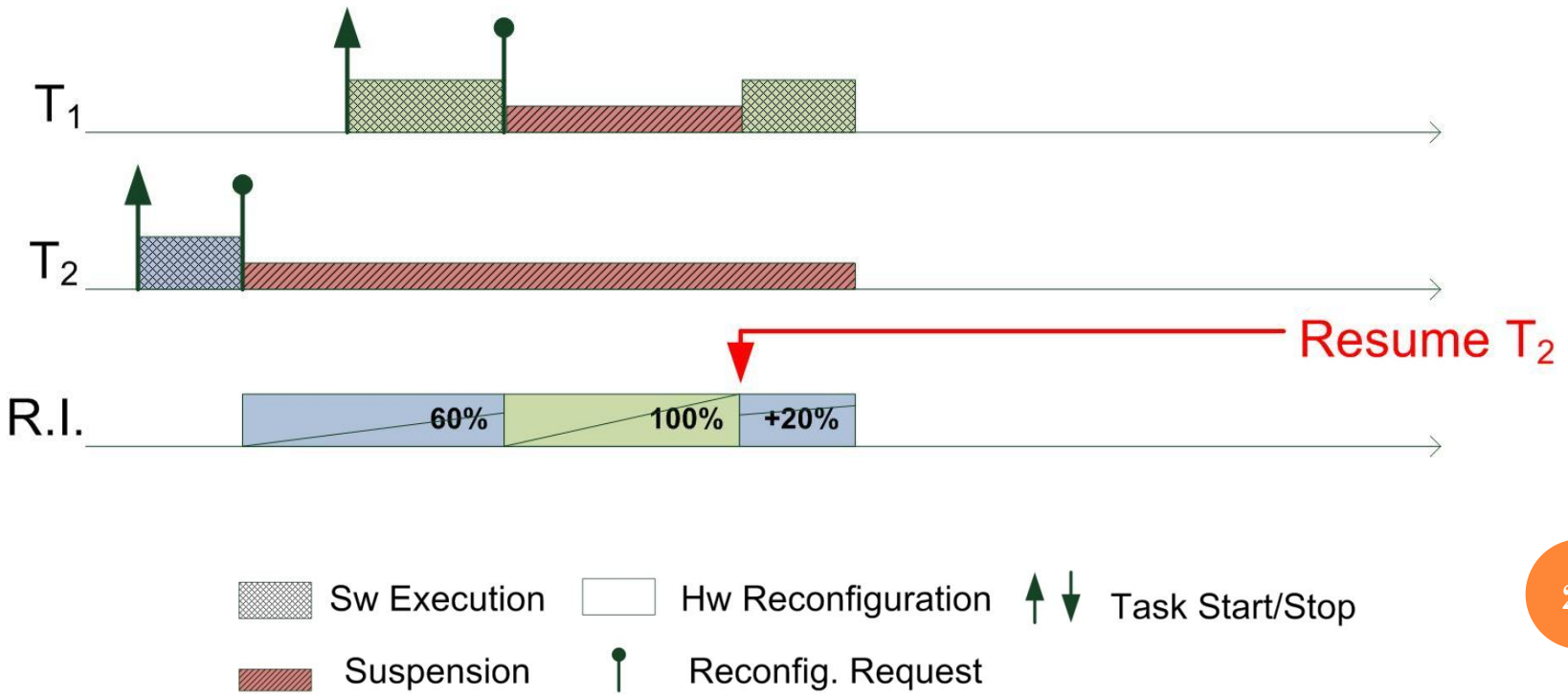
Preemptable Reconfiguration

- Real Time task (T1) has priority over Best Effort task (T2)
- The reconfiguration process is preemptable
  - Hardware reconfiguration can be resumed



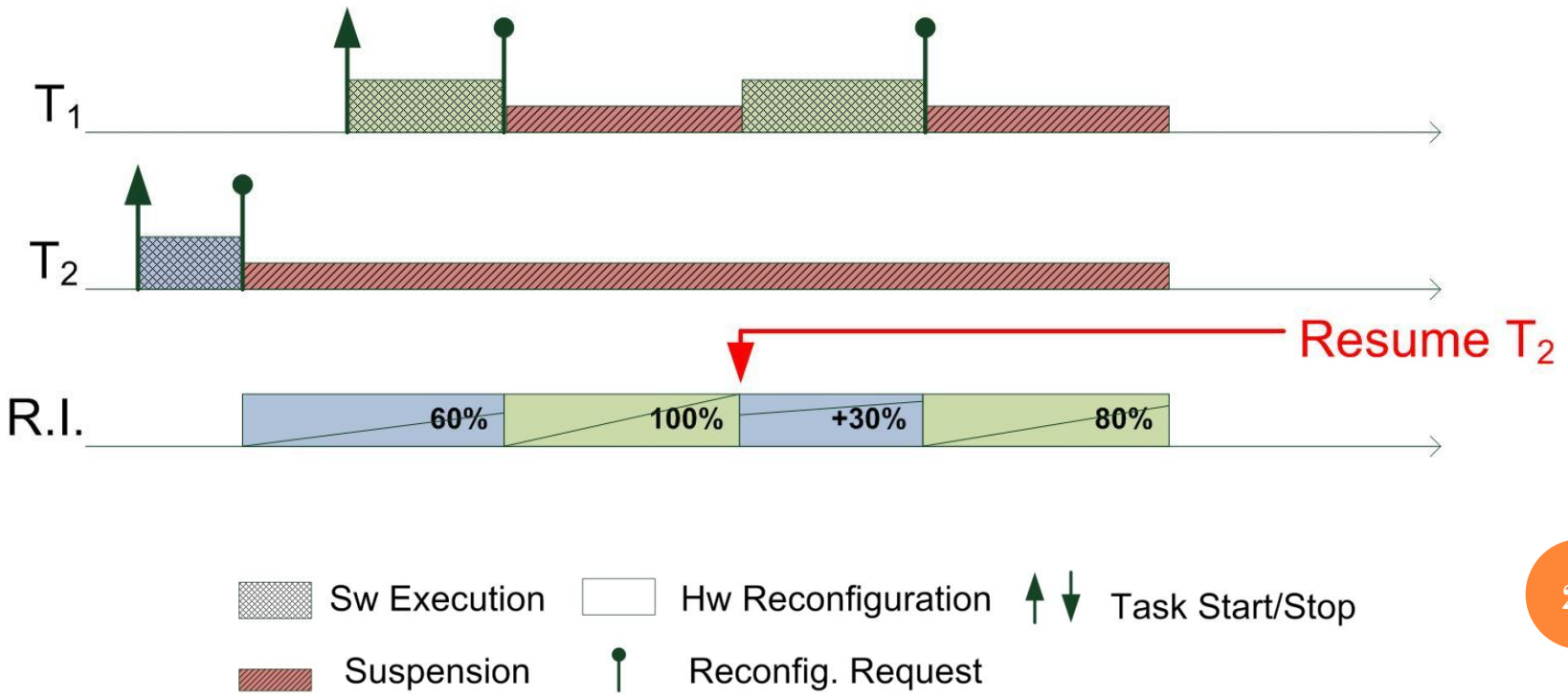
Preemptable Reconfiguration

- Real Time task (T1) has priority over Best Effort task (T2)
- The reconfiguration process is preemptable
  - Hardware reconfiguration can be resumed



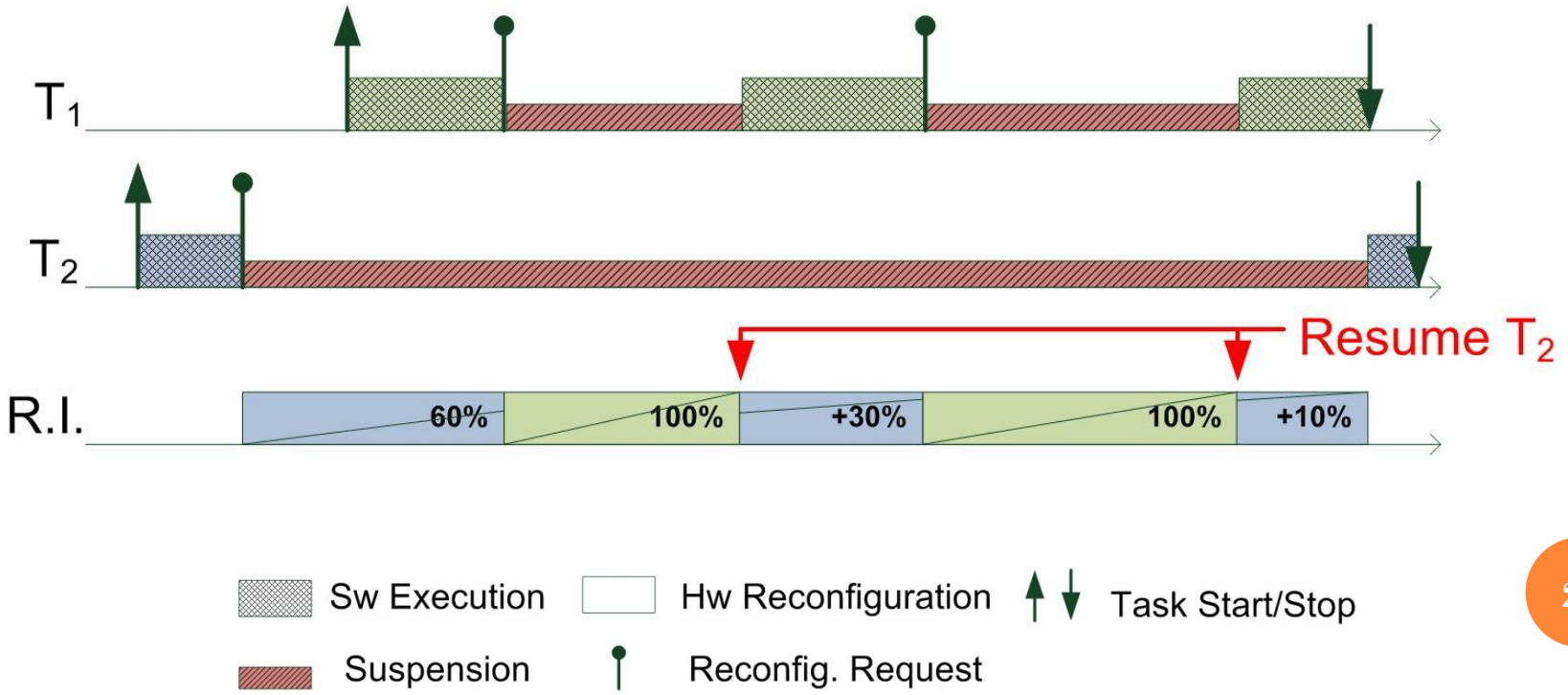
Preemptable Reconfiguration

- Real Time task (T1) has priority over Best Effort task (T2)
- The reconfiguration process is preemptable
  - Hardware reconfiguration can be resumed



Preemptable Reconfiguration

- Real Time task (T1) has priority over Best Effort task (T2)
- The reconfiguration process is preemptable
  - Hardware reconfiguration can be resumed



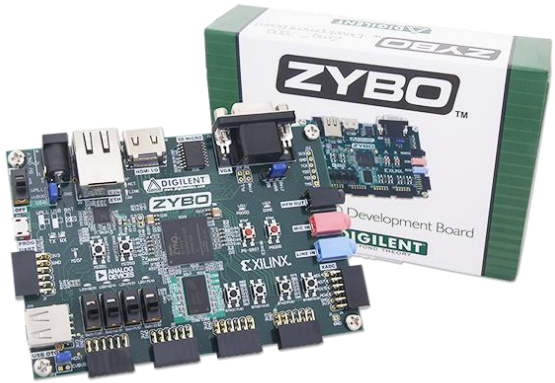
Benefits

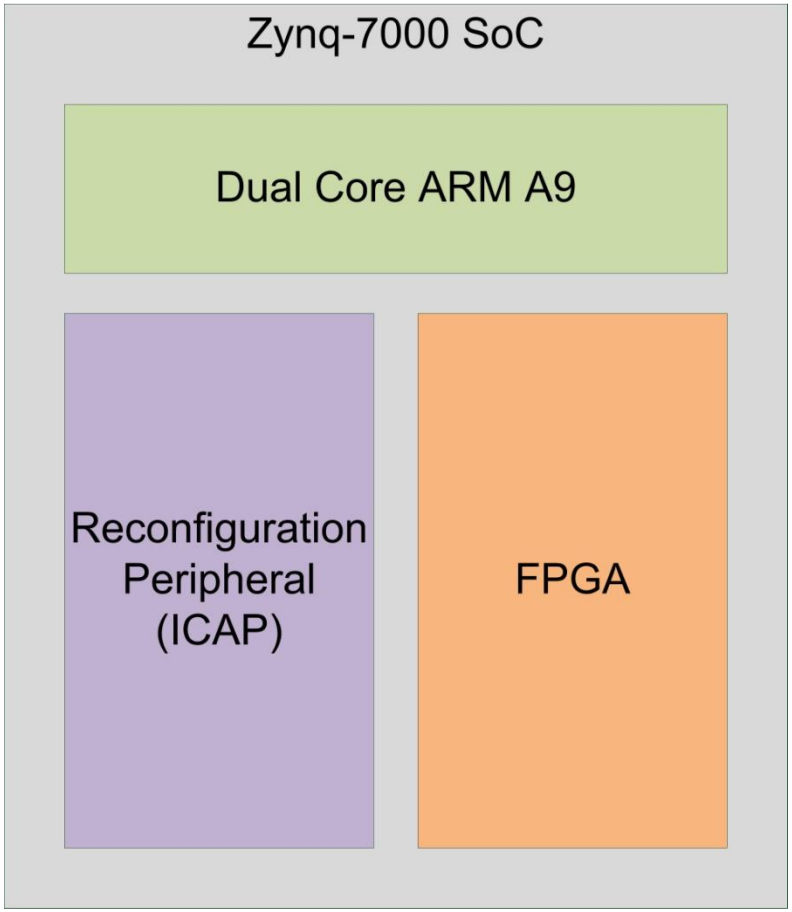
With a preemptable interface:

- Performance of both Real Time and Best Effort tasks are improved.
- The delay experienced by the higher priority task can be reduced.
- The delay experienced by Best Effort tasks can be bounded.

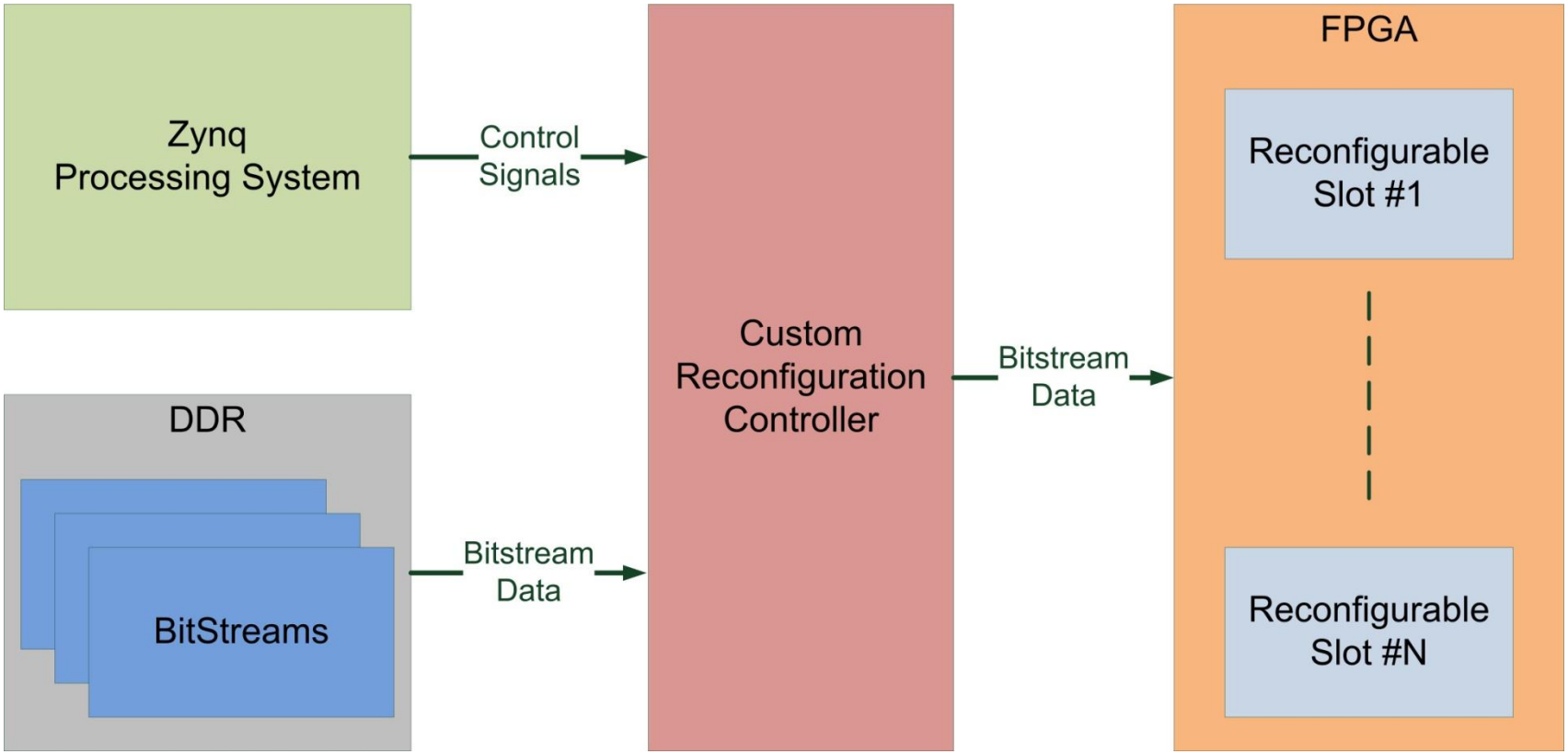
Implementation

Preemptable reconfiguration has been realized on a Xilinx Zynq-7000 platform.





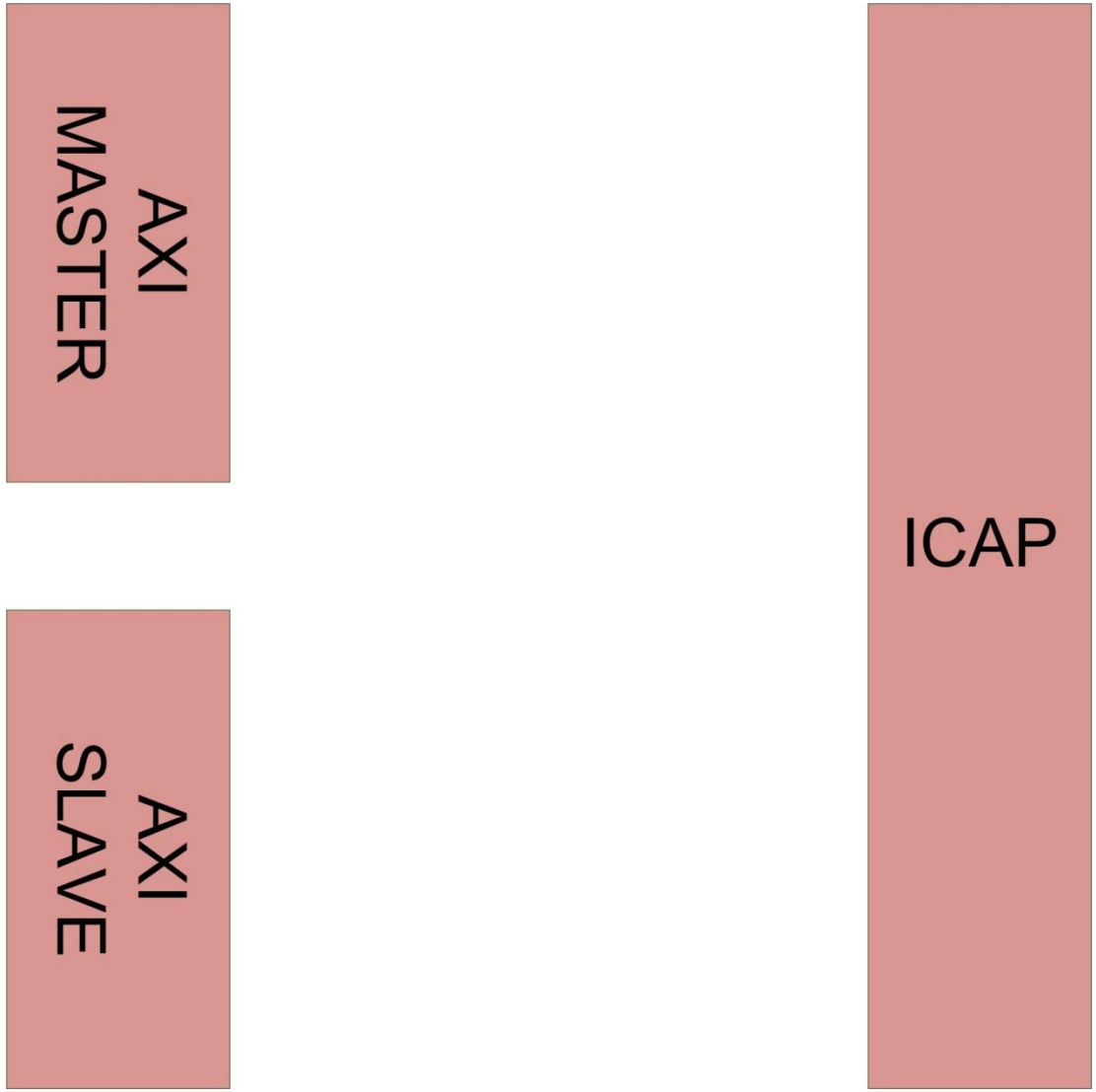
Base System Overview



- Zynq Processing System can trigger a hardware reconfiguration by sending the bitstream to configure to the Reconfiguration Controller.

Custom Reconfiguration Controller

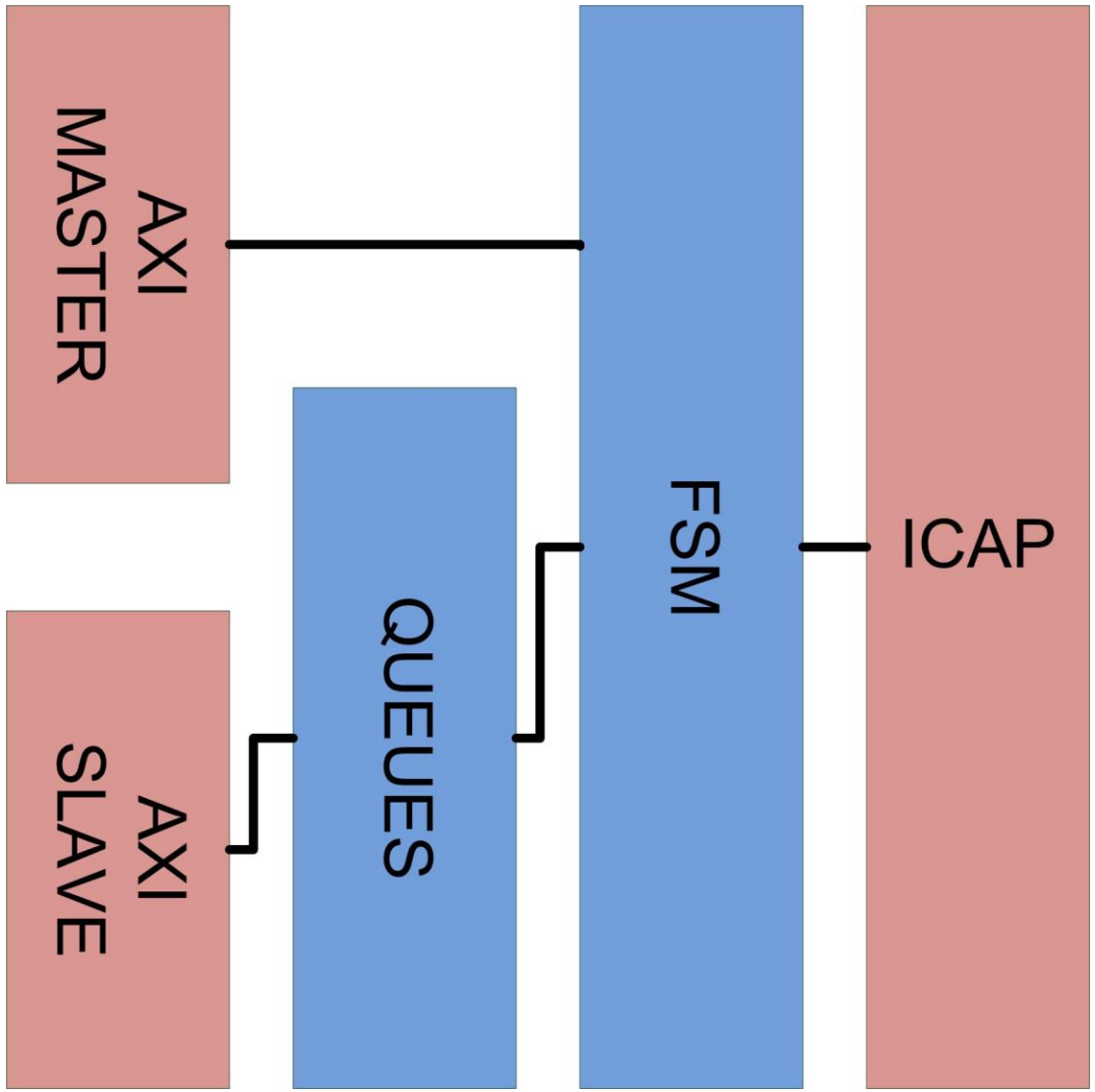
Xilinx IP Cores



Custom Reconfiguration Controller

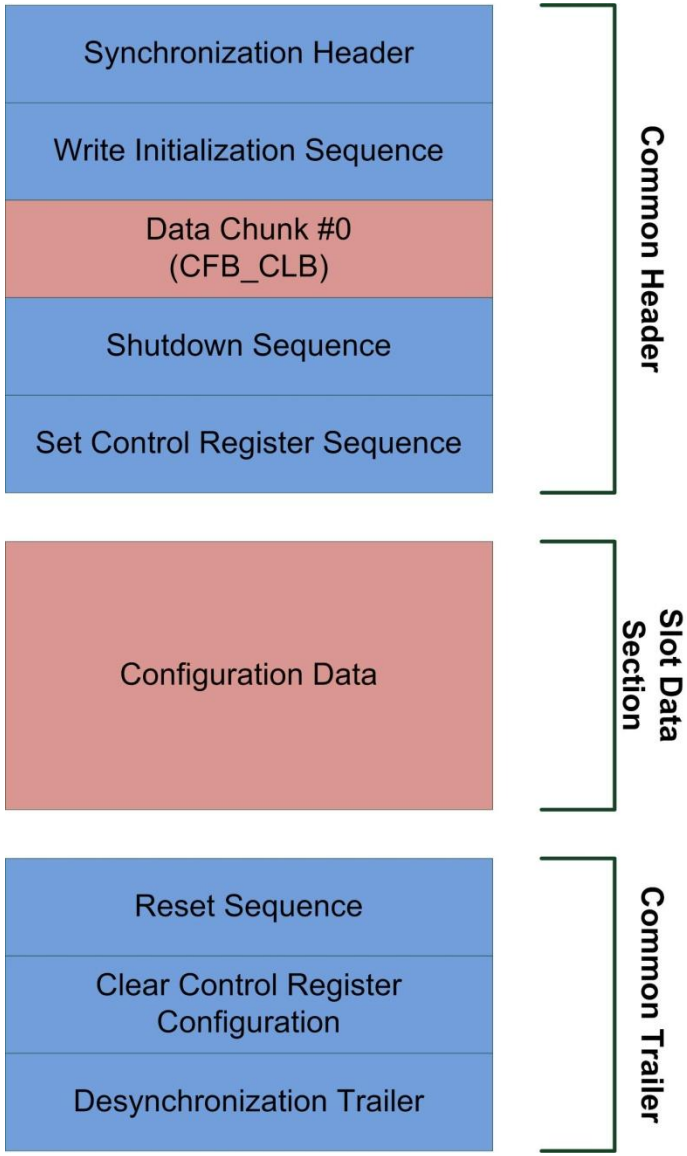
Xilinx IP Cores

Custom IP Cores

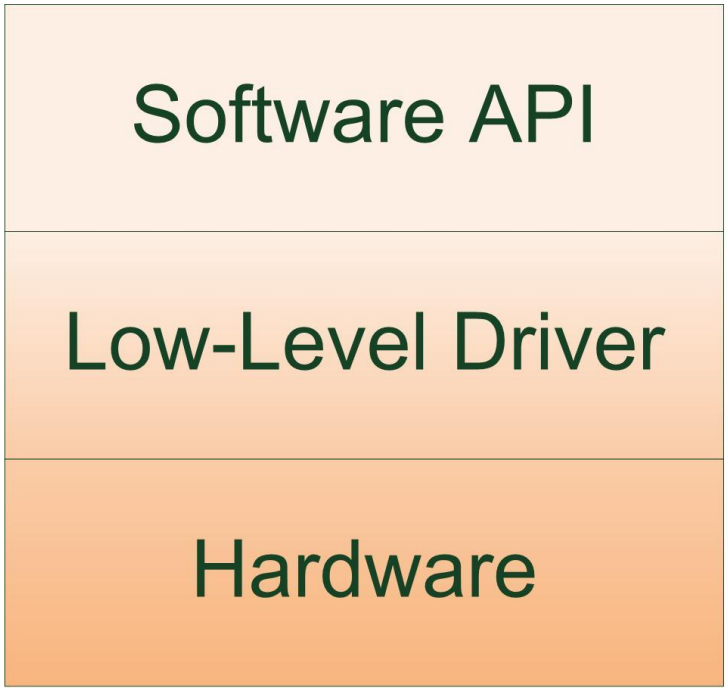


Valid Resumption Points

- Bitstream contain configuration data and commands.
- Commands control the Xilinx reconfiguration port (ICAP)
- The ABORT can be performed anywhere in the bitstream.
- Resumption point of reconfiguration must be carefully calculated.

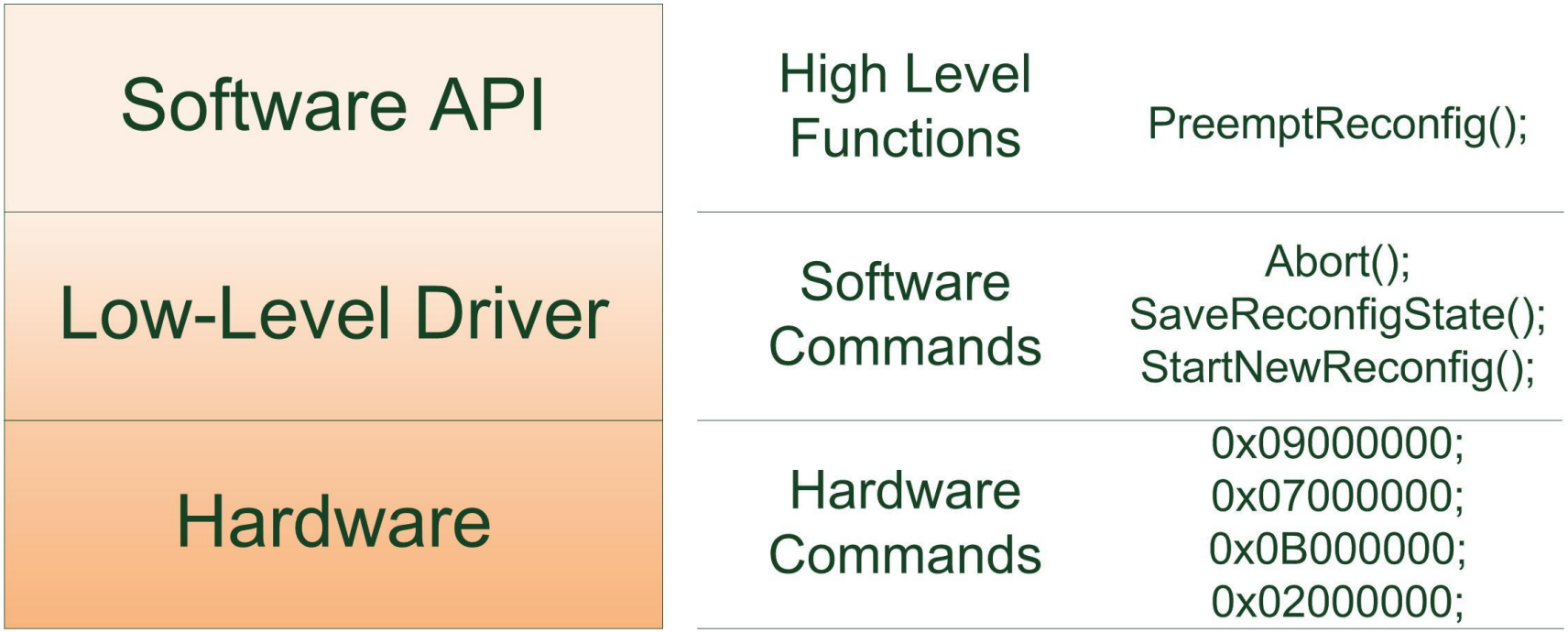


Custom Reconfiguration Controller



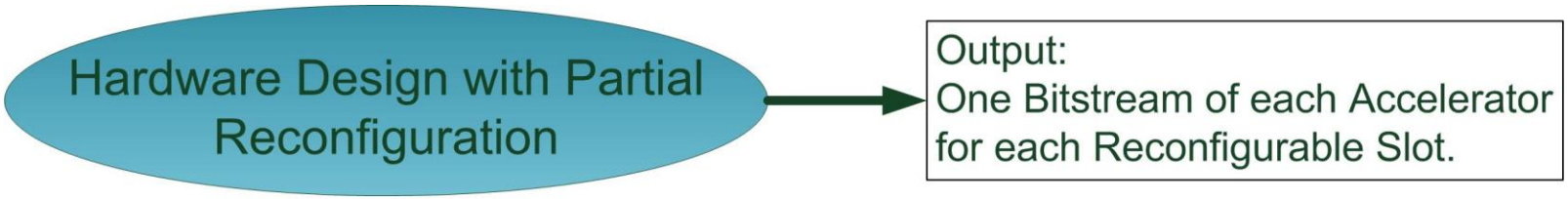
High Level Functions	PreemptReconfig();
Software Commands	Abort(); SaveReconfigState(); StartNewReconfig();
Hardware Commands	0x09000000; 0x07000000; 0x0B000000; 0x02000000;

Custom Reconfiguration Controller

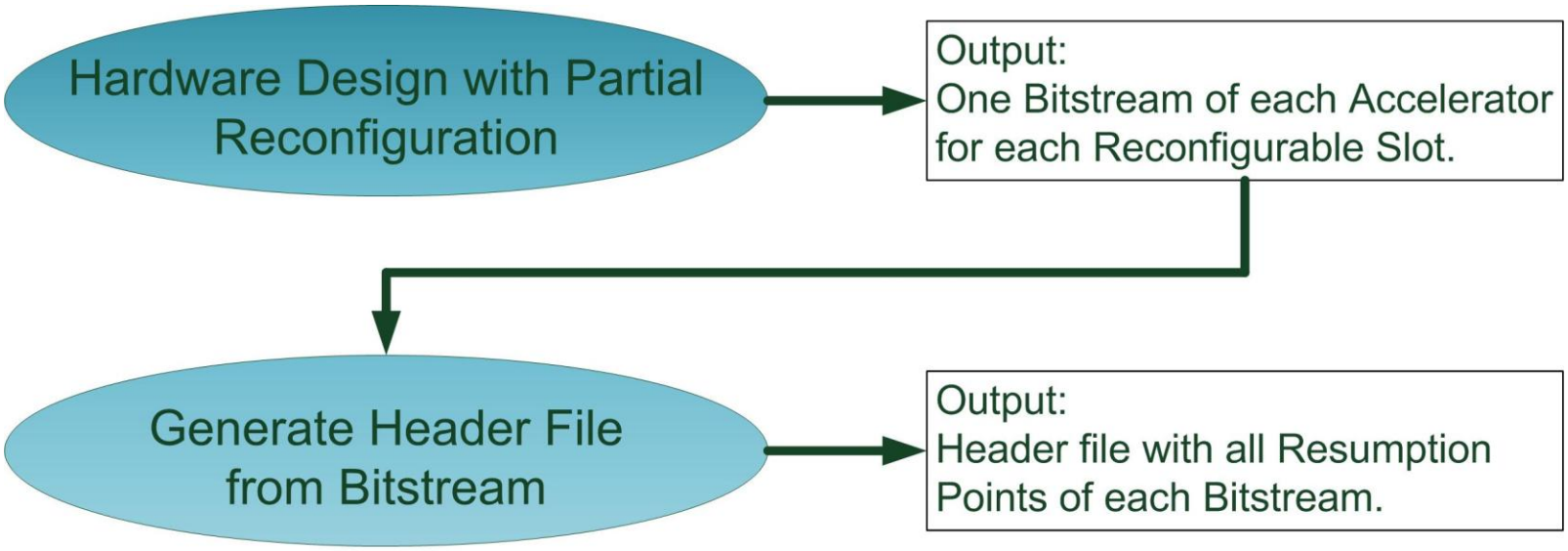


- The hardware layer guarantees worst-case latency bounds on the commands it processes, even on the reconfiguration itself when a memory bandwidth is guaranteed.

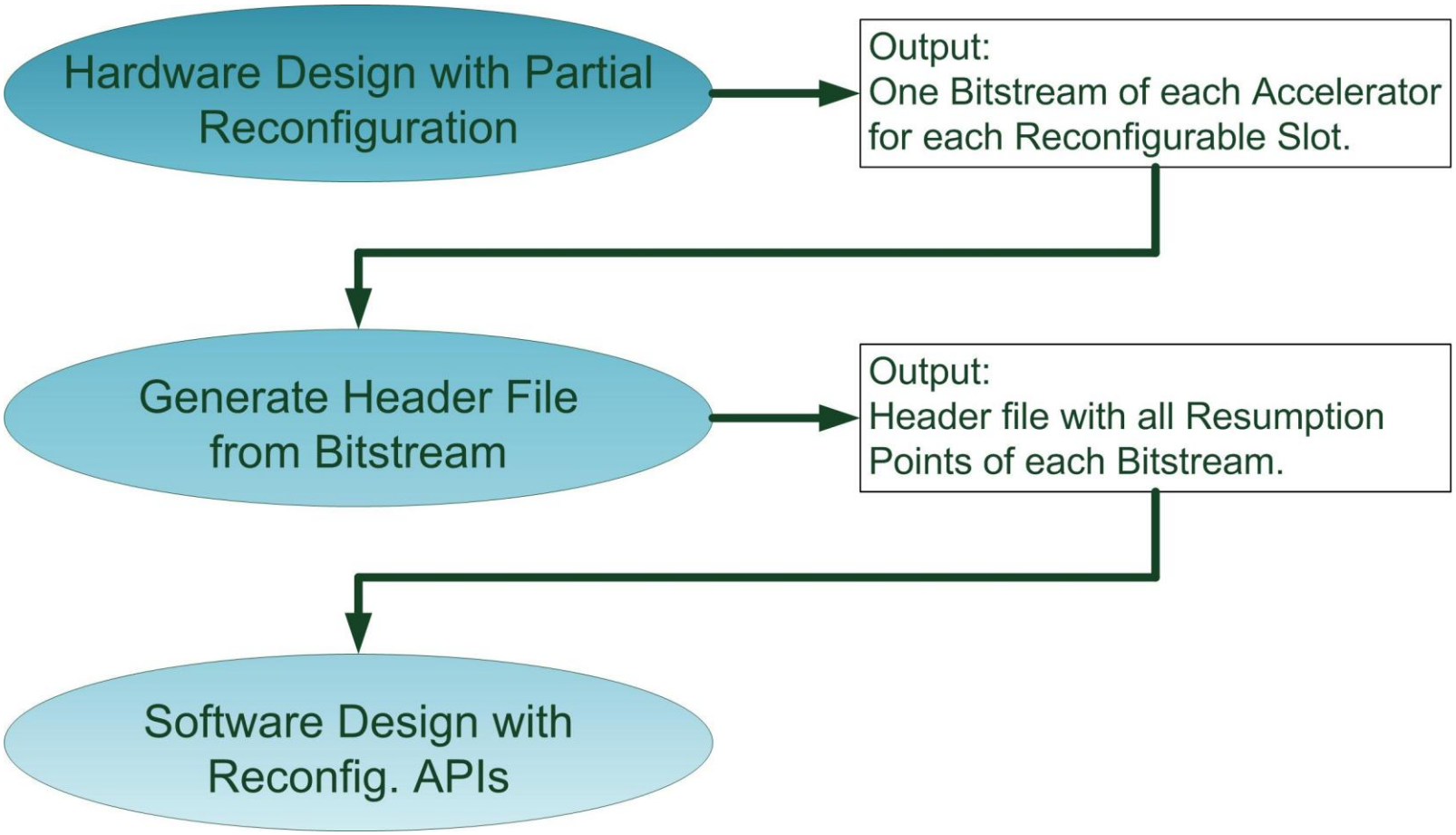
Project Flow



Project Flow



Project Flow



Measured Results

- High priority task
  - Period: from 5ms to 44ms (1ms step)
  - Reconfig. time: 0.79ms
- Low priority task
  - Period: 50ms
  - Reconfig. time: 32.63ms

	Low Priority Task		**High Priority Task**	
	Max. Observed Exec. Time	Avg. Observed Exec. Time	Max. Observed Exec. Time	Avg. Observed Exec. Time
No Abort	33.46 ms	32.68 ms	2.45 ms	0.805 ms
Abort	1664.12 ms	33.16 ms	0.811 ms	0.808 ms
Preemption	42.34 ms	32.68 ms	0.810 ms	0.805 ms

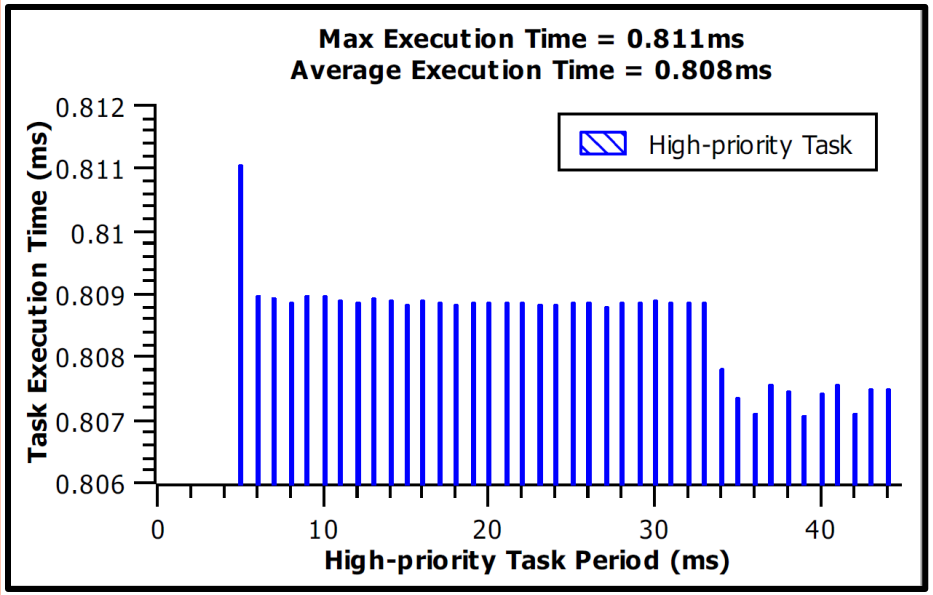
Measured Results

- High priority task
  - Period: from 5ms to 44ms (1ms step)
  - Reconfig. time: 0.79ms
- Low priority task
  - Period: 50ms
  - Reconfig. time: 32.63ms

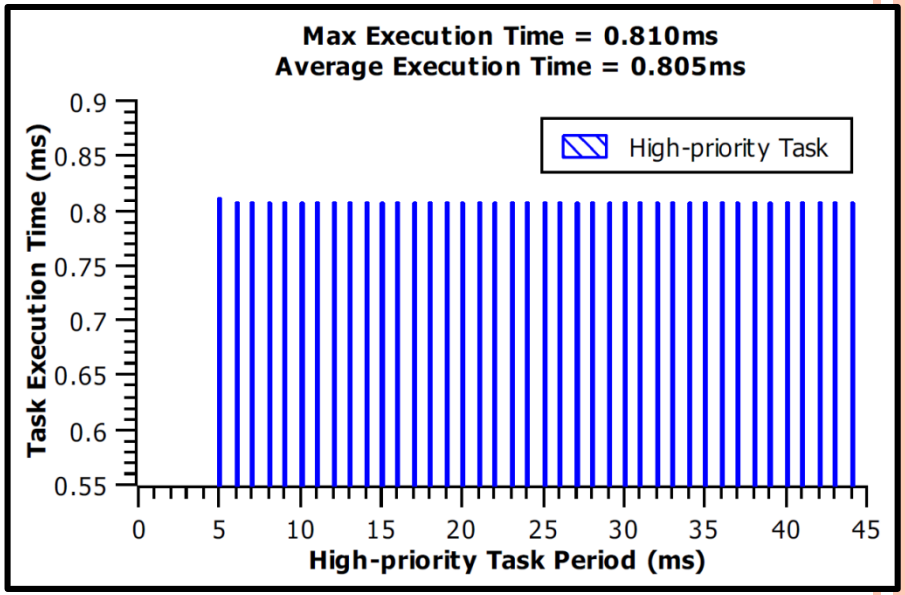
	Low Priority Task		High Priority Task	
	Max. Observed Exec. Time	Avg. Observed Exec. Time	Max. Observed Exec. Time	Avg. Observed Exec. Time
No Abort	33.46 ms	32.68 ms	2.45 ms	0.805 ms
Abort	1664.12 ms	33.16 ms	0.811 ms	0.808 ms
Preemption	42.34 ms	32.68 ms	0.810 ms	0.805 ms

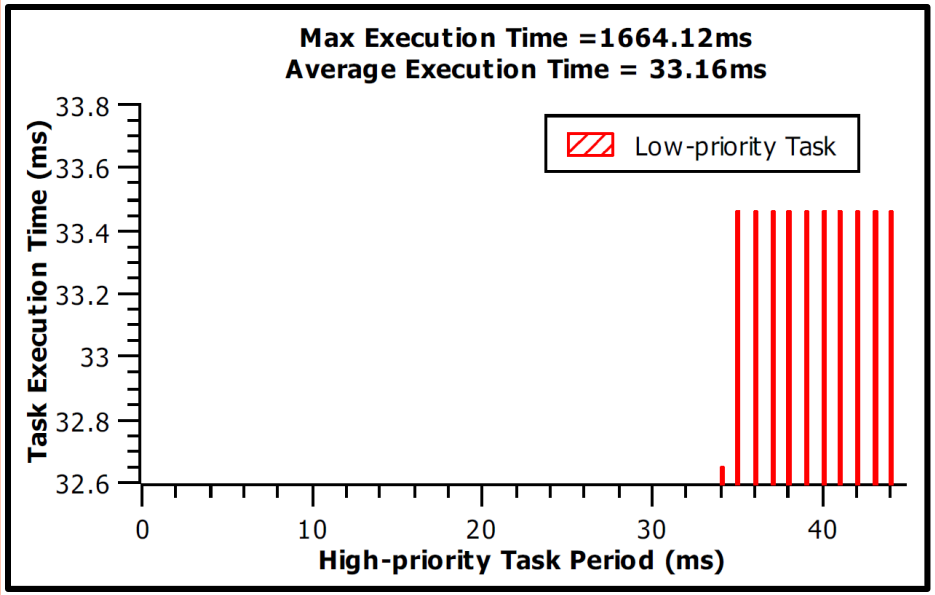
Measured Results

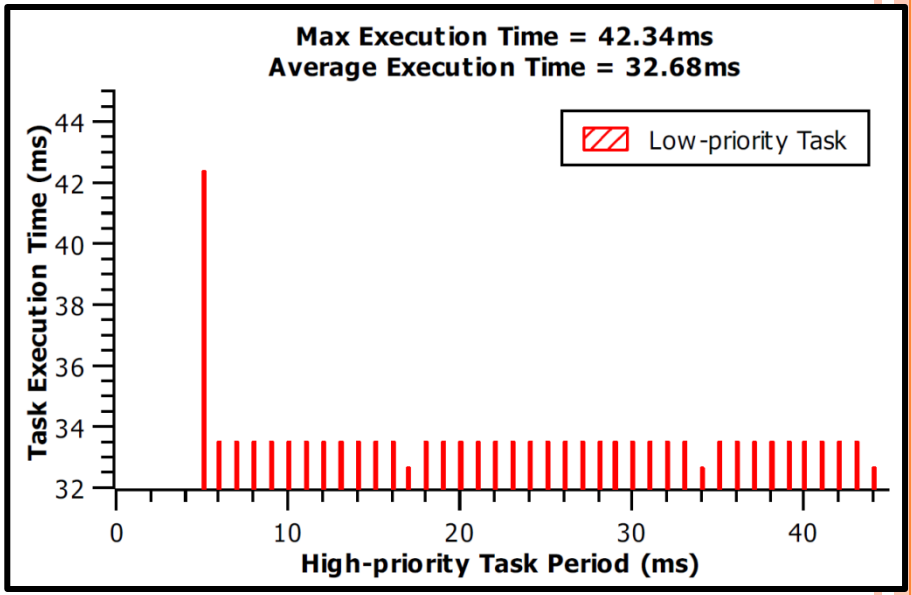
Starvation Experiment



Preemption Experiment







Conclusions

- Design and realization of Preemptable Reconfiguration with guaranteed latencies
  - Custom Reconfiguration Controller
  - Software driver, API
  - Tools to analyze and manipulate bitstreams
- Base application developed on a RealTime OS (FreeRTOS) that benefits from having Preemptable Reconfiguration


рахмат
danke
謝謝
ngiyabonga
teşekkür ederim
Баярлалаа
спасибо
faafetai lava
mersi
barka
welalin
tack
spas
kia ora
dank je
misaotra
paldies
grazzi
mahalo
vinaka
спасибі
blagodaram
matondo
thank you
gracias
tapadh leat
хвала
kiitos
dankie
nanni
nandri
dhanyavad
maururu
hvala
köszönöm
bedankt
gracie
bayarlalaa
aciu
dankon
akun
asante
manana
obrigada
tenki
enkosi
djiere dieuf
tau
mochchakkeram
murakoze
дякую
mamnun
dziękuję
chnorakaloutioun
gratias ago
gracies
sulpáy
go raibh maith agat
shokrane
obrigado
sobodi
dėkuji
sagolun
sukriya
kop khun krap
taiku
arigatô
takk
dakujem
trugarez
mèsi
madloba
didi
najis tuke
ありがとう
tanemirt
grazie
rahmet
diolch
dhanyavadagalu
shukriya
merce
мерси
kam sah hamnida
rahmat
terima kasih
তোমাকে ধন্যবাদ
감사합니다
xiexie
ευχαριστώ
merci