MPSoC'16

Efficient Hardware Context-Switch for Task Migration between Heterogeneous FPGAs

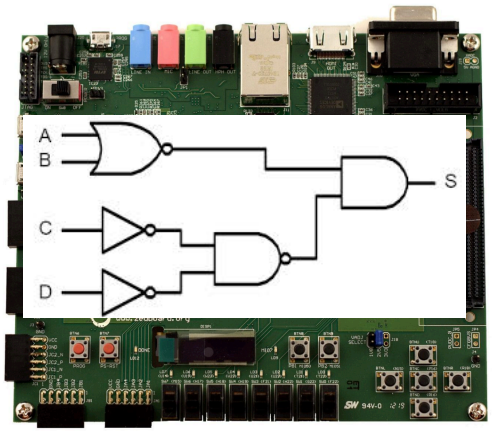
Frédéric Rousseau

TIMA lab – University of Grenoble Alpes

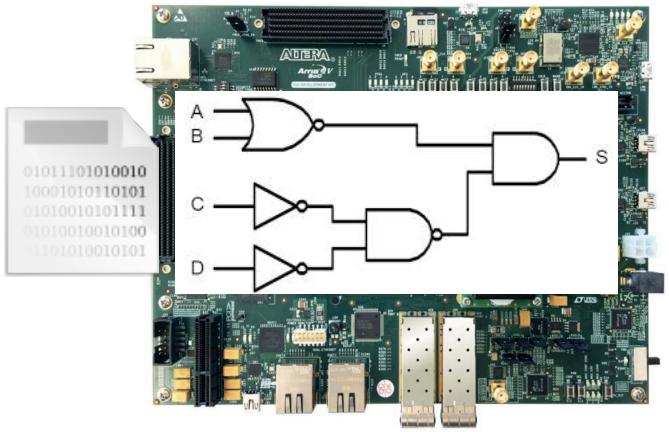
A join work with Alban Bourge, Arief Wicaksana and Olivier Muller

How to pause, migrate, and resume a task between Heterogeneous FPGAs ?

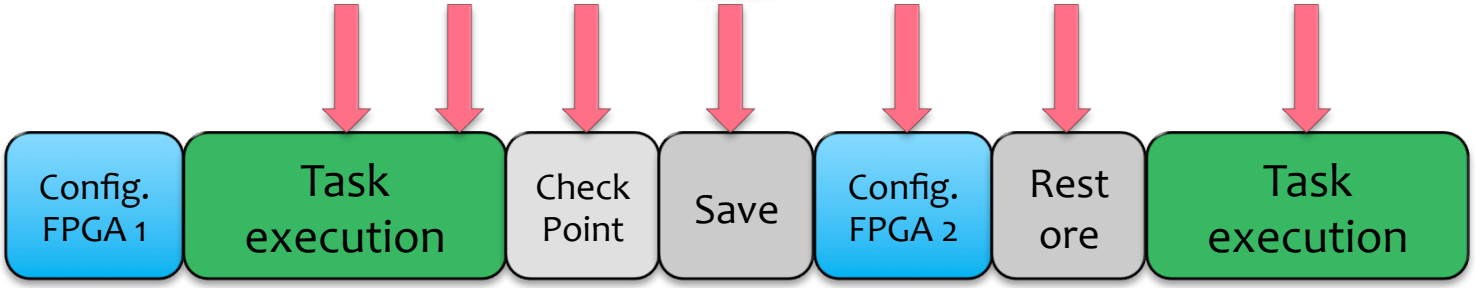
XilinX Zynq



Altera SoC



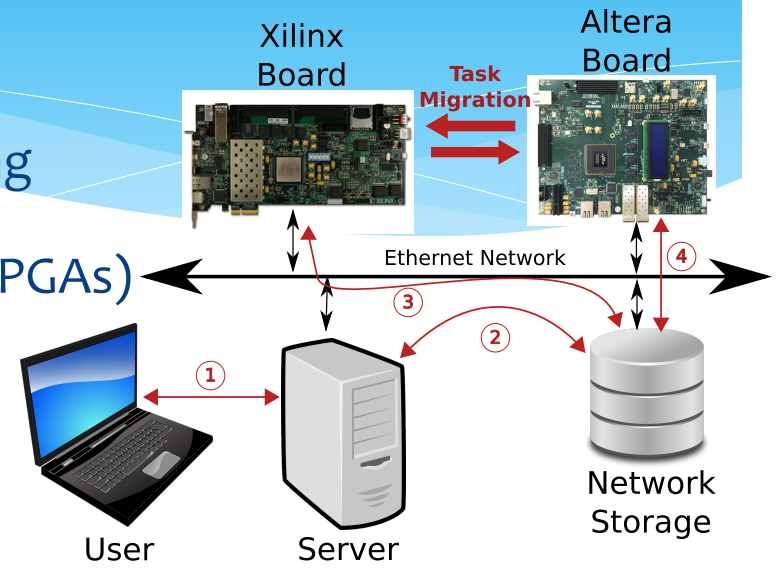
Migration request



Agenda

1. Definition, motivation, and challenges
2. Efficient Context-Switch Circuits
3. Design Flow and Validation Environment
4. Results
5. Conclusion and Future Work

Background and Definitions



* Overall system
  * A control part responsible for managing the whole system
  * 2 different reconfigurable resources (FPGAs) connected by a communication media
* Hardware task
  * A classical ***FSM/datapath*** model
* Hardware context
  * The content of the memory elements of the running circuit
    * Set of ***live variables*** related to the current state of the task
* Hardware context-switch
  * Capability of a system to pause, migrate and resume a circuit
* Hardware checkpoint
  * A task state where context-switch operations are allowed

Challenges

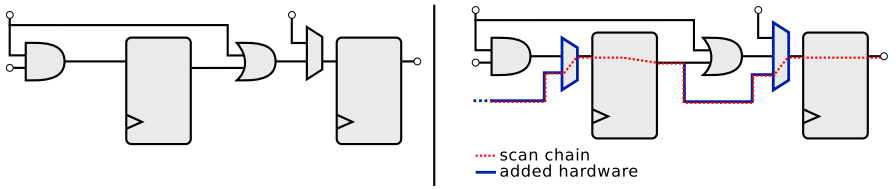
* Environnement for (automatic) management of bitstreams and context

* Support of task migration
  * Hardware context-switch support
    * Ensures an answer to system ***preemption*** demands within a given ***latency***
    * Is FPGA-***technology independant***
    * Is ***effortless*** for designers and users
    * The ***hardware overhead*** is ***low***
    * The ***hardware overhead*** has a ***negligeable*** impact on ***performance***

Motivations and Related Work

* Hardware context-switch for **Task migration**
  * ***Multi-user*** capability
  * Task relocation, FPGA defragmentation

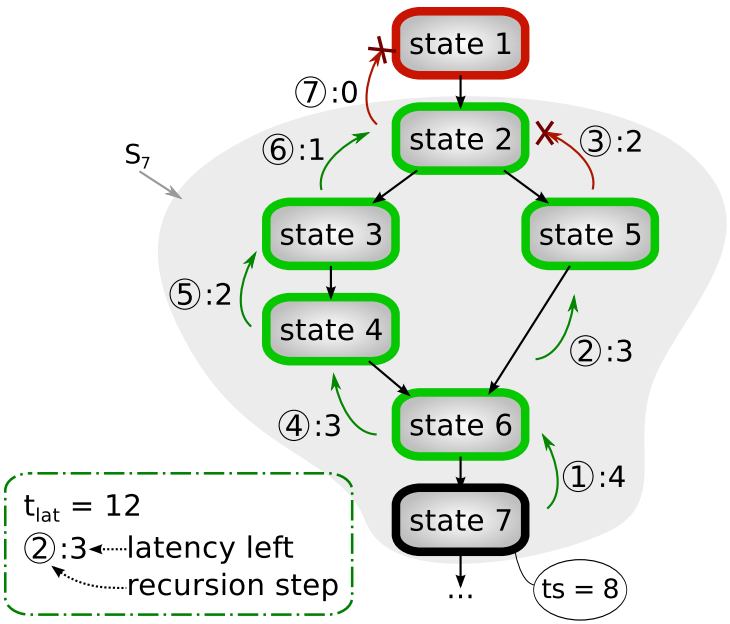
* Related work on context extraction techniques
  * ***Readback*** method: extraction of the FPGA components (LUT, FF, routing) through the configuration mechanism
    * Technology dependant (not yet supported by Altera, used of ICAP technology on Xilinx Virtex technology)
  * ***Scan-chain***: the circuit embeds the mechanisms that allow reading and writing – serial link
    * Smaller memory footprint
    * Extraction time reduced
    * But extra design effort



* Related work on hardware task migration
  * [Koch2013]: HW checkpoint with scan-chain, back-end flow modifications

Efficient Context-Switch Circuit

* Checkpoint selection (greedy algorithm)
  * Minimal sets of live variables within a given latency



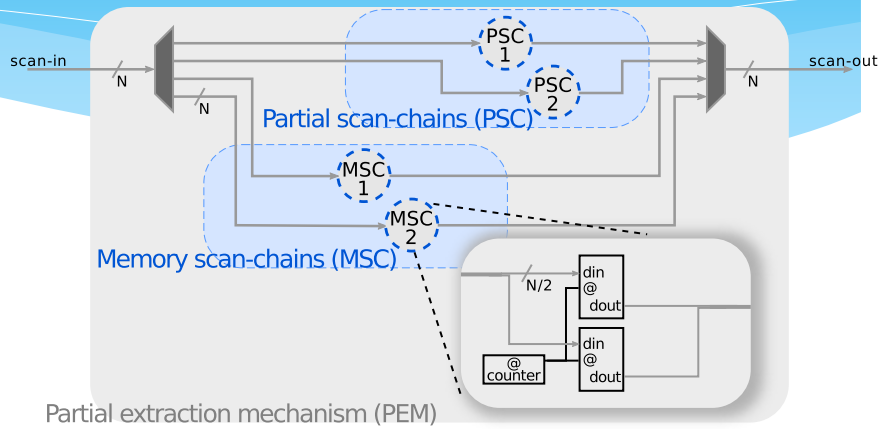
Checkpointing technique:
Coverage computing of
"state 7"
ts = extraction time = 8

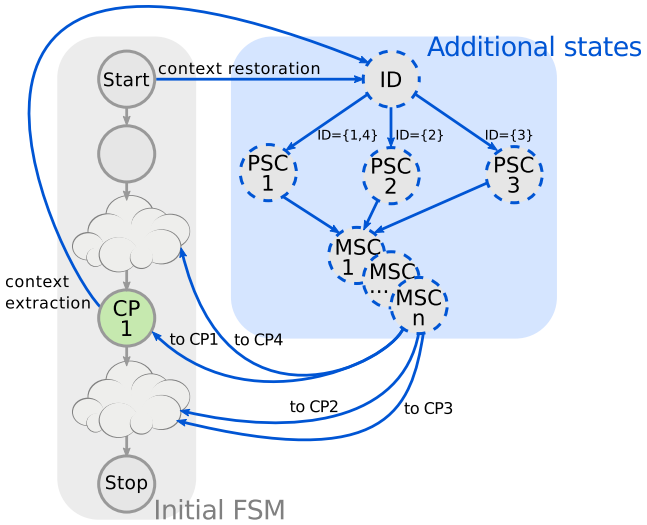
Latency of 1 for all states

A loop with dynamic iteration numbers is not covered by any outer state: the worst-case scenario

Efficient Context-Switch Circuit

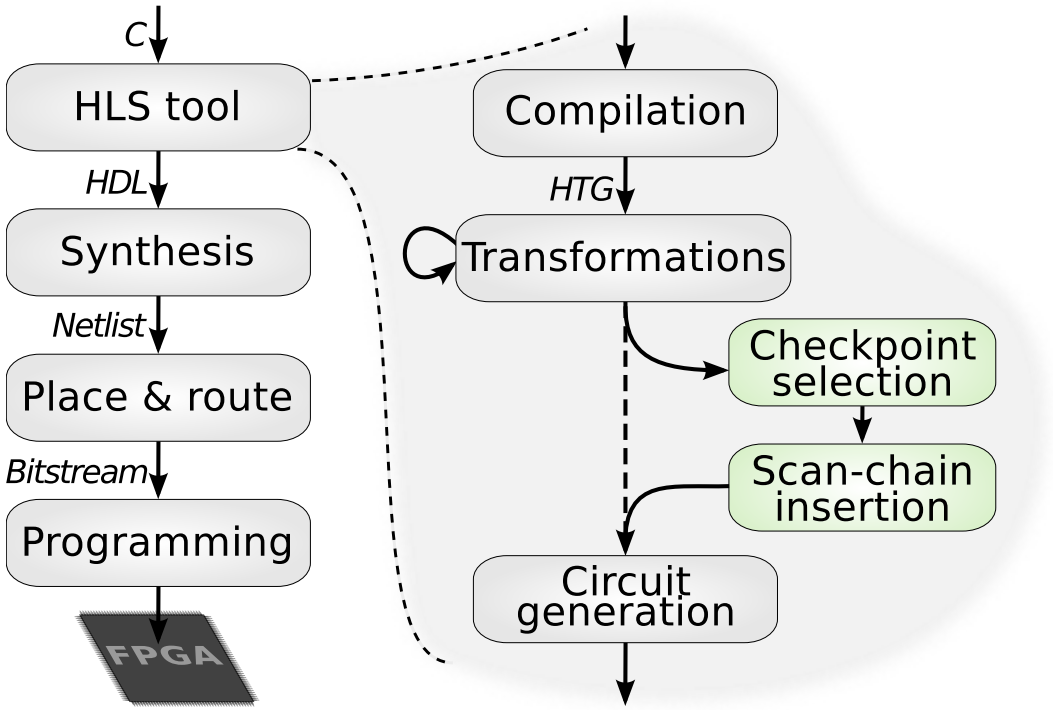
* Scan-chain insertion
  * Addition of hardware resources (mainly multiplexors and routing)
    * Partial scan-chain are shared by checkpoints with the same set of live variables (data in register)
    * Specific scan-chain for data in memory (LUTRAM or BRAM)
  * The FSM needs additional states to enter in context switching mode





Design Flow based on HLS

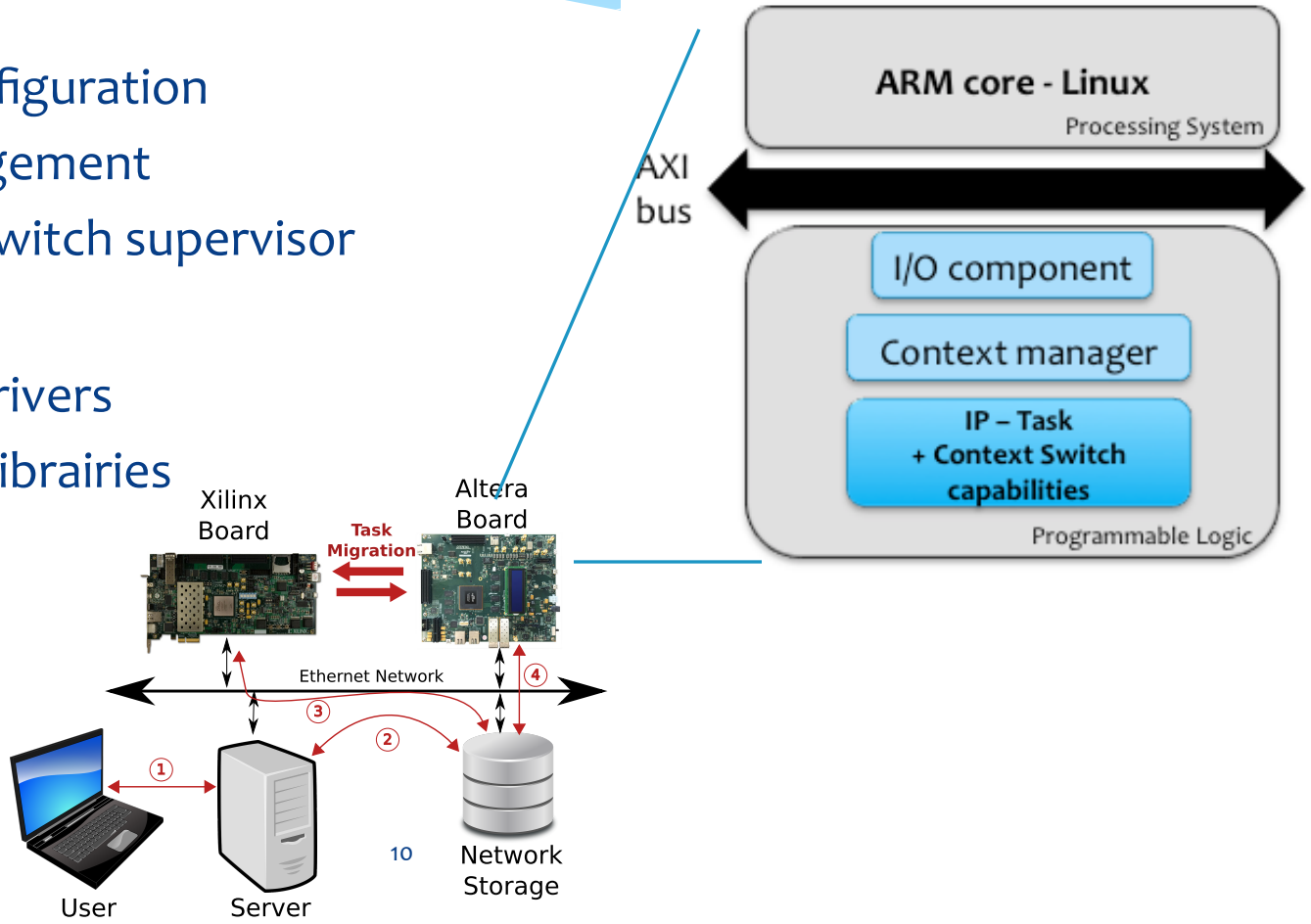
* Use of AUGH High Level Synthesis tool
  * Developed in TIMA Lab
  * Free and open-source



CheckPoint PinPoint (CP3) Plug-in

Task Migration Capability

* Thanks to CPU on FPGA boards
  * Its role
    * FPGA configuration
    * I/O management
    * Context switch supervisor
  * Add-ons
    * Specific drivers
    * Function librairies



Interesting Results

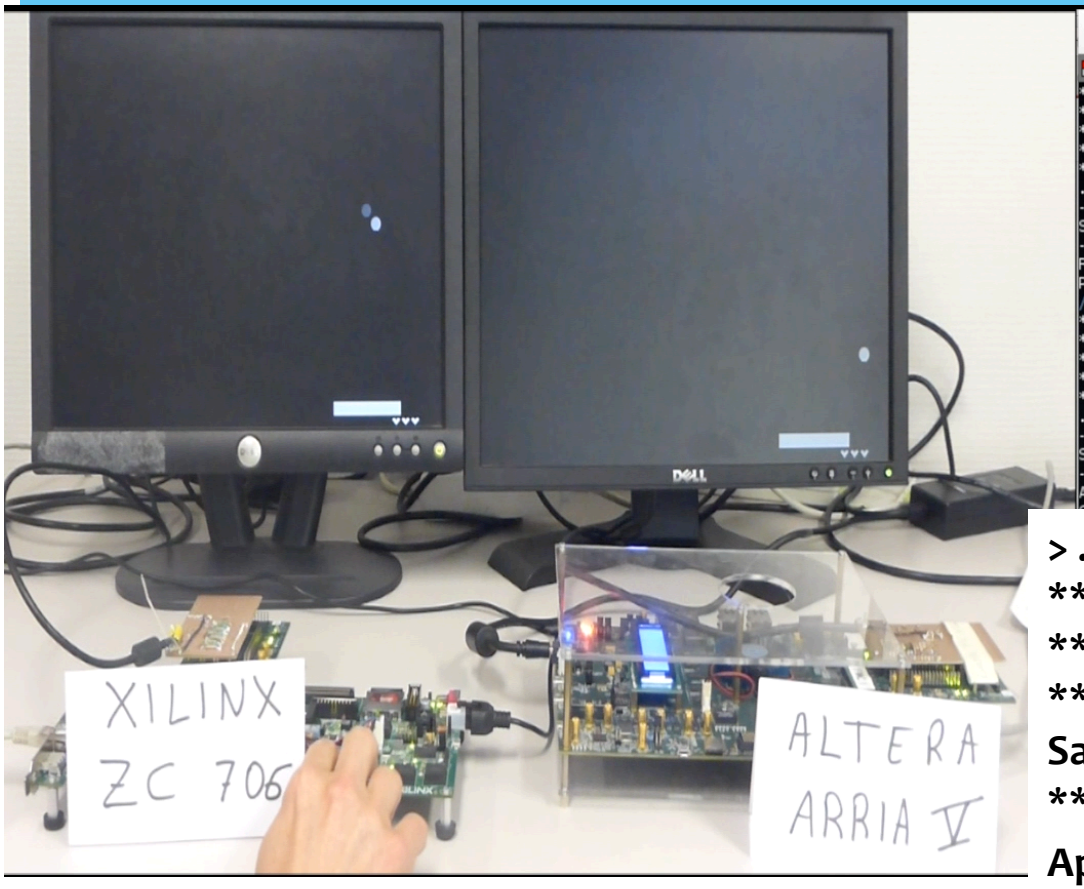
* CHStone benchmark suite app. + MJPEG dec. + IDCT

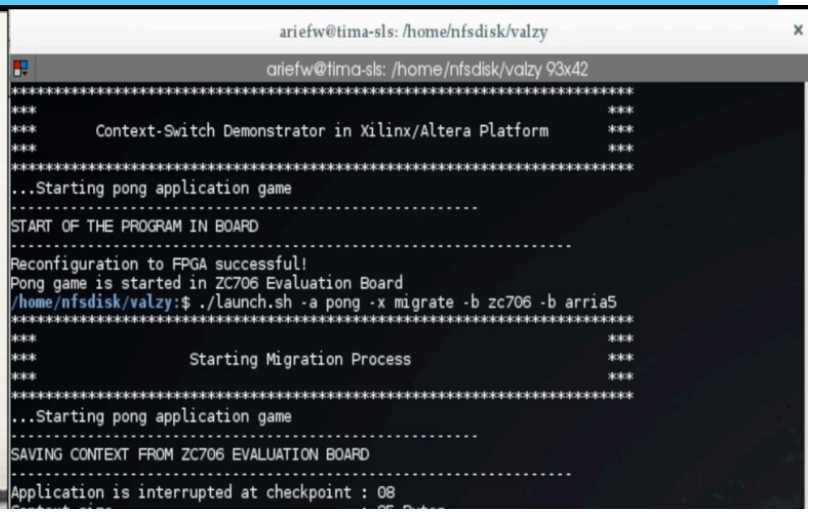
* Context Switch validation
  * 11.5 times reduction of extracted bits compared to full scan chain
  * Between 3 % and 32 % of HW resources (LUT, FF) overhead
  * Loss of about 2.6 % of running frequency

A. Bourge, O. Muller, F. Rousseau, *Automatic High-Level Hardware Checkpoint Selection for Reconfigurable Systems*, International Symposium on Field-Programmable Custom Computing Machines (IEEE FCCM 2015), pp. 155 - 158, May. 3-15, 2015, Vancouver (BC), Canada.

A. Bourge, O. Muller, F. Rousseau, *Generating Efficient Context-Switch Capable Circuits Through Autonomous Design Flow*, submitted to ACM Transactions on Reconfigurable Technology and Systems, submitted in Nov. 2015

Exemple of visual simple demo





> ./launch.sh pong migrate arria5 zc706

****

***** starting Migration Process *****

****

Saving context from ARRIA5

****

Application interrupted at *checkpoint: 06*

Context size: 170 bytes

****

Restoring the application to zc706

Today's limitations

* Task migration should happen only during computation time: restricted model of behavior

```
Loop
    read_input (&data_in)
    compute
    write(data_out)
```

* The smallest FPGA (resources) limits the process
* 50 MHz (on Altera) for IP after migration

* Limited to (and between):
  * Xilinx Zynq: Zedboard and ZC706
  * Altera: ARRIA V and DE1 SoC

Conclusion and Perspectives

* Solution for task migration between heterogeneous FPGA
  * Context switch method and tools
  * Design flow
  * Validation environment

* Next steps
  * Management of I/O (communication consistency)

* Demo at FPL conference demo night (31st of Aug. Lausanne, Switzerland)

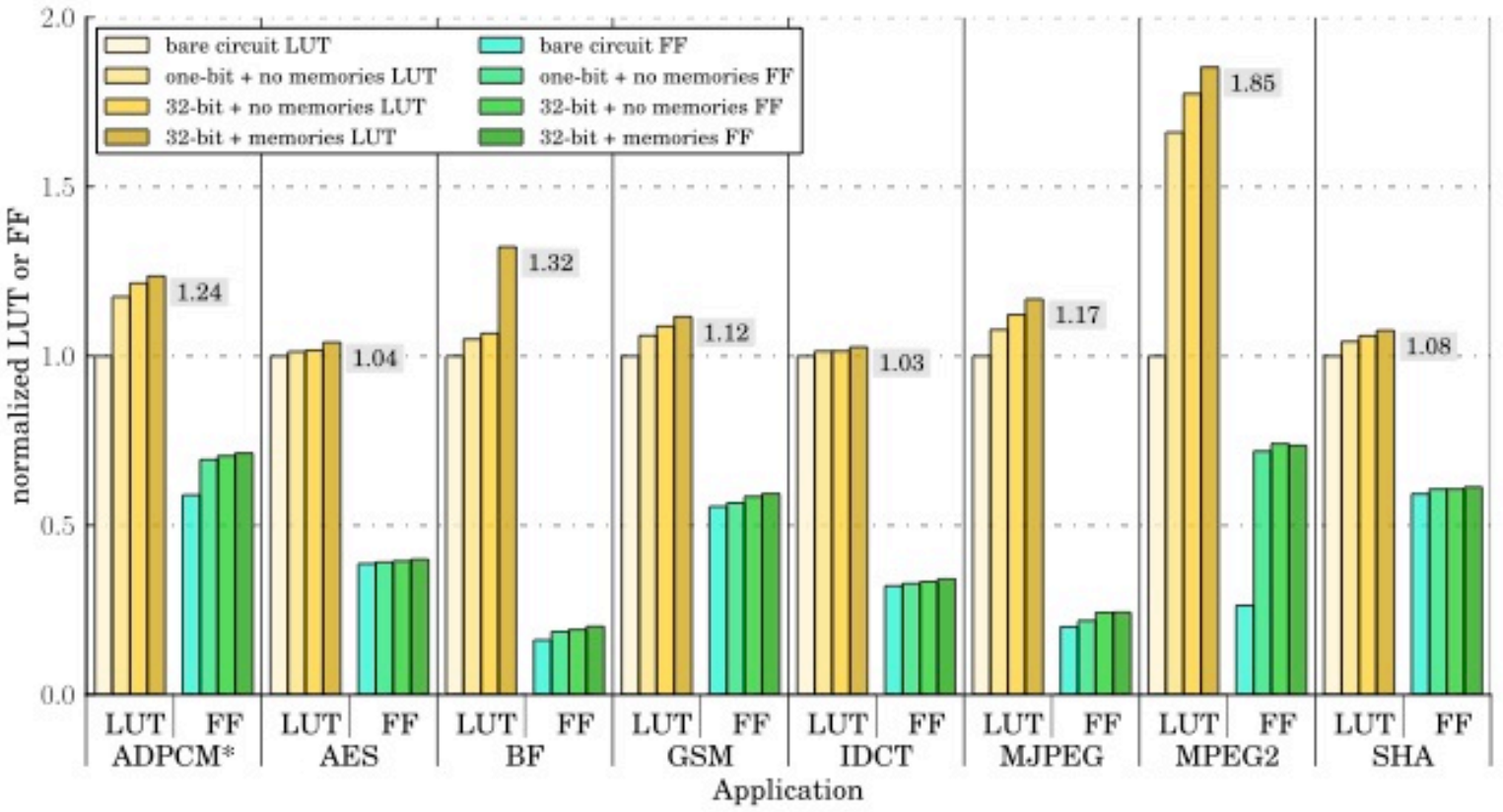
Annex 1: Extracted bit reduction

* CHStone benchmark suite app. + MJPEG dec. + IDCT
* T_{lat} = 15000 cycles

	# of PSC /Checkpoints /states	Checkpoint ratio	Full Scan-chain (bit)	Mean PSC (bit)	Gain (full / mean)
adpcm	6 / 6 / 131	4.6 %	6304	2778	2.3 x
aes	3 / 5 / 1053	0.5 %	1616	163	9.9 x
blowfish	5 / 6 / 112	5.4 %	568	124	4.6 x
gsm	12 / 14 / 1712	0.8 %	864	59	14.6 x
idct	2 / 2 / 233	0.9 %	1000	36	27.8 x
mjpeg	43 / 104 / 887	11.7 %	5762	444	13.0 x
mjpeg2	12 / 19 / 94	20.2 %	1384	774	1.8 x
sha	7 / 10 / 134	7.5 %	3168	176	18.0 x
Mean		6.4 %			**11.5 x**

Annex 2: Area overhead

* Post logic synthesis area results expressed in LUT and FF (ISE 14.7)



Annex 3: Low exec. time overhead

* Post mapping critical path and frequency results (ISE 14.7)
* Full: 32 bit-wide mechanism with memory extraction

	Critical path (ns)		Frequency (MHz)		
	bare	**full**	**bare**	**full**	**%**
adpcm	9.654	9.709	103.581	102.997	0.56 %
aes	5.173	5.178	193.298	193.141	0.08 %
blowfish	5.87	5.874	170.364	170.242	0.07 %
gsm	12.963	13.462	77.141	74.281	3.71 %
idct	12.628	12.628	79.188	79.188	0.00 %
mjpeg	41.102	44.006	24.33	22.724	6.60 %
mjpeg2	13.038	14.476	76.696	69.082	9.93 %
sha	8.969	8.973	111.492	111.44	0.05 %
Mean					**2.6 %**