Specification and architecture exploration of DSP system platforms

Authors : J.Quévremont, M.Sarlotte, B. Candaele

Contributions : F.Beaujean, C.Dietzi, D.Ragot

Speac Workshop

20th February 2004

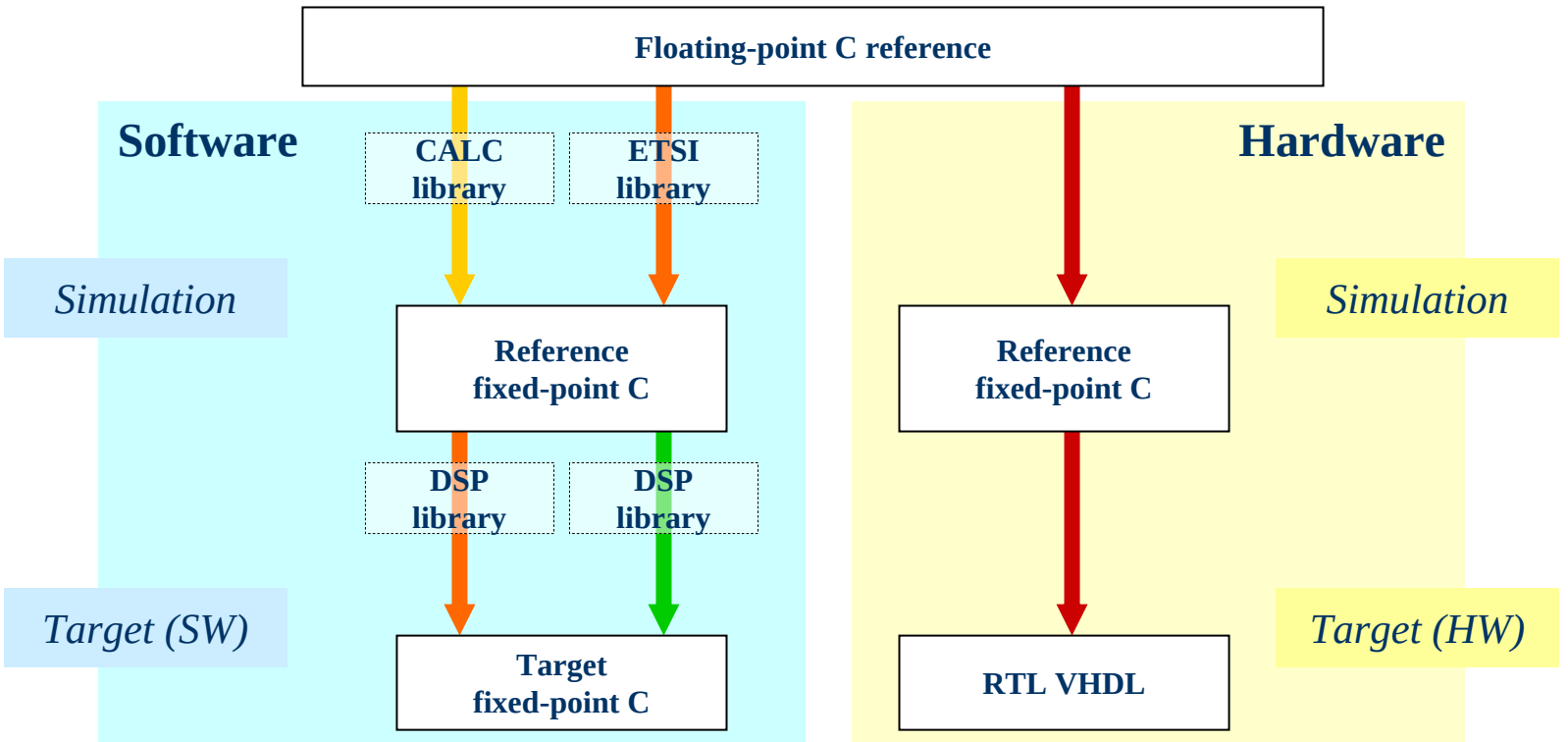
Contact: michel.sarlotte@fr.thalesgroup.com

Overview

- Current design flow for DSP applications
- New approach
- Main results
- Conclusions

Existing SLD flow for DSP (1)

Wide gap on HW part between the system engineering definition and the HW implementation tasks



Current development process for DSP (2)

Algorithm in Floating point

Algorithm in Fixed point

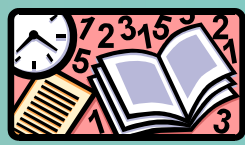
Engineering on the application definition and related performances assessment :

Algorithm
Dynamic
Latency
Data-rate
…

Signal Processing Engineering World

Frequency
Size
Memory
Latency
Ressource sharing
…

What is the best Architecture/ trade_off



Architecture 3

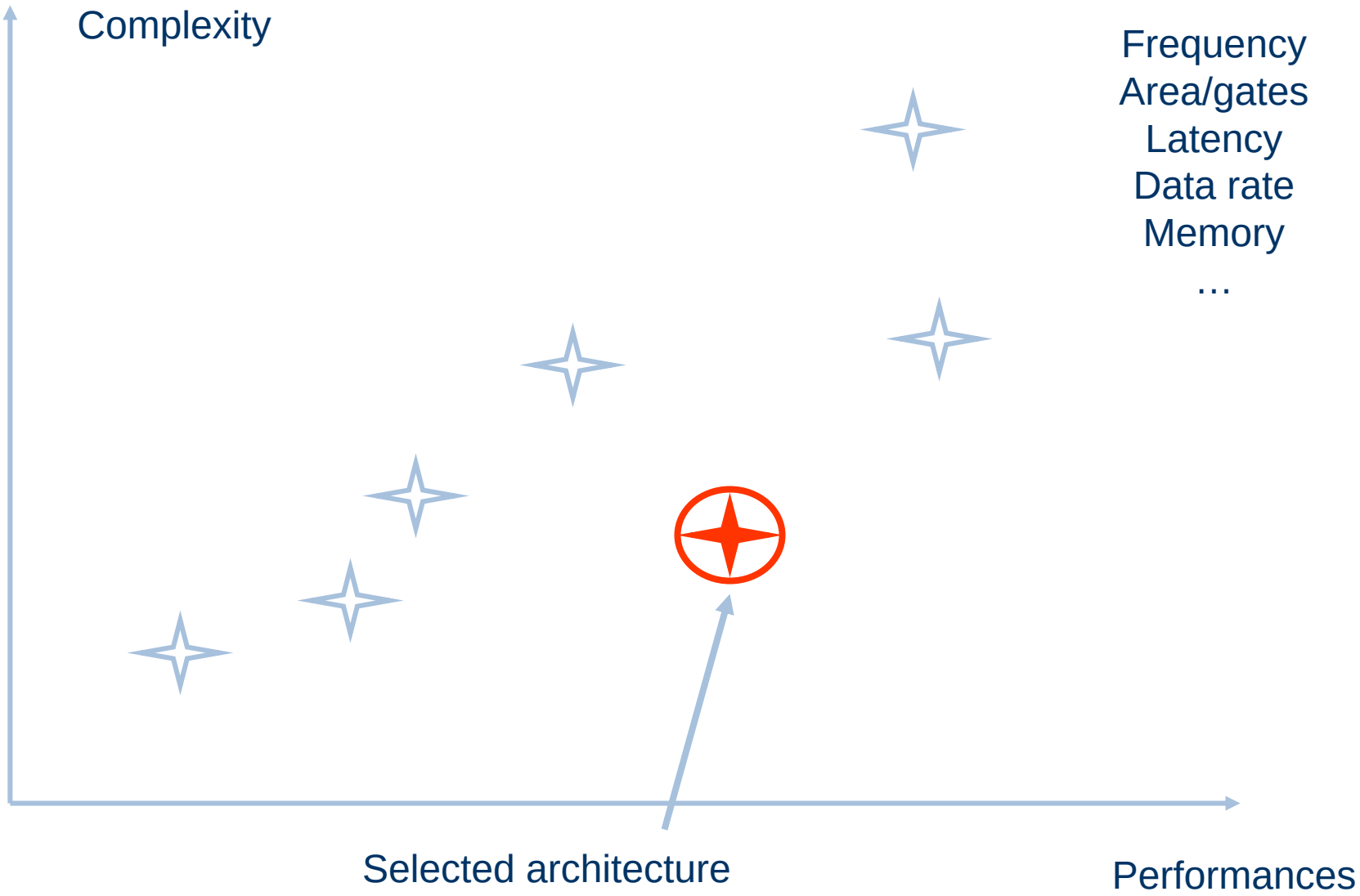
Architecture 1

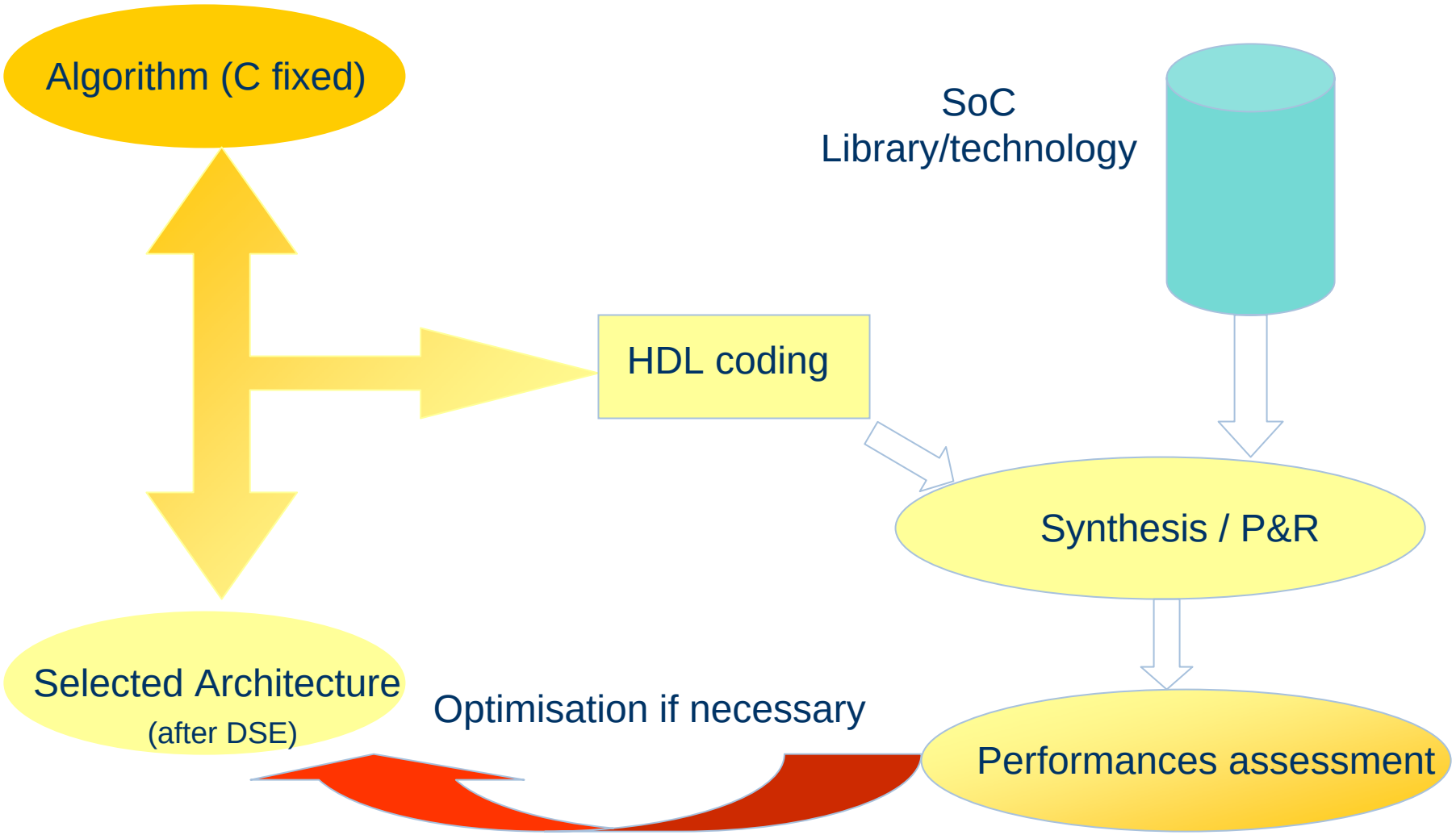
Architecture 2

SoC/ASIC/FPGA World

THALES

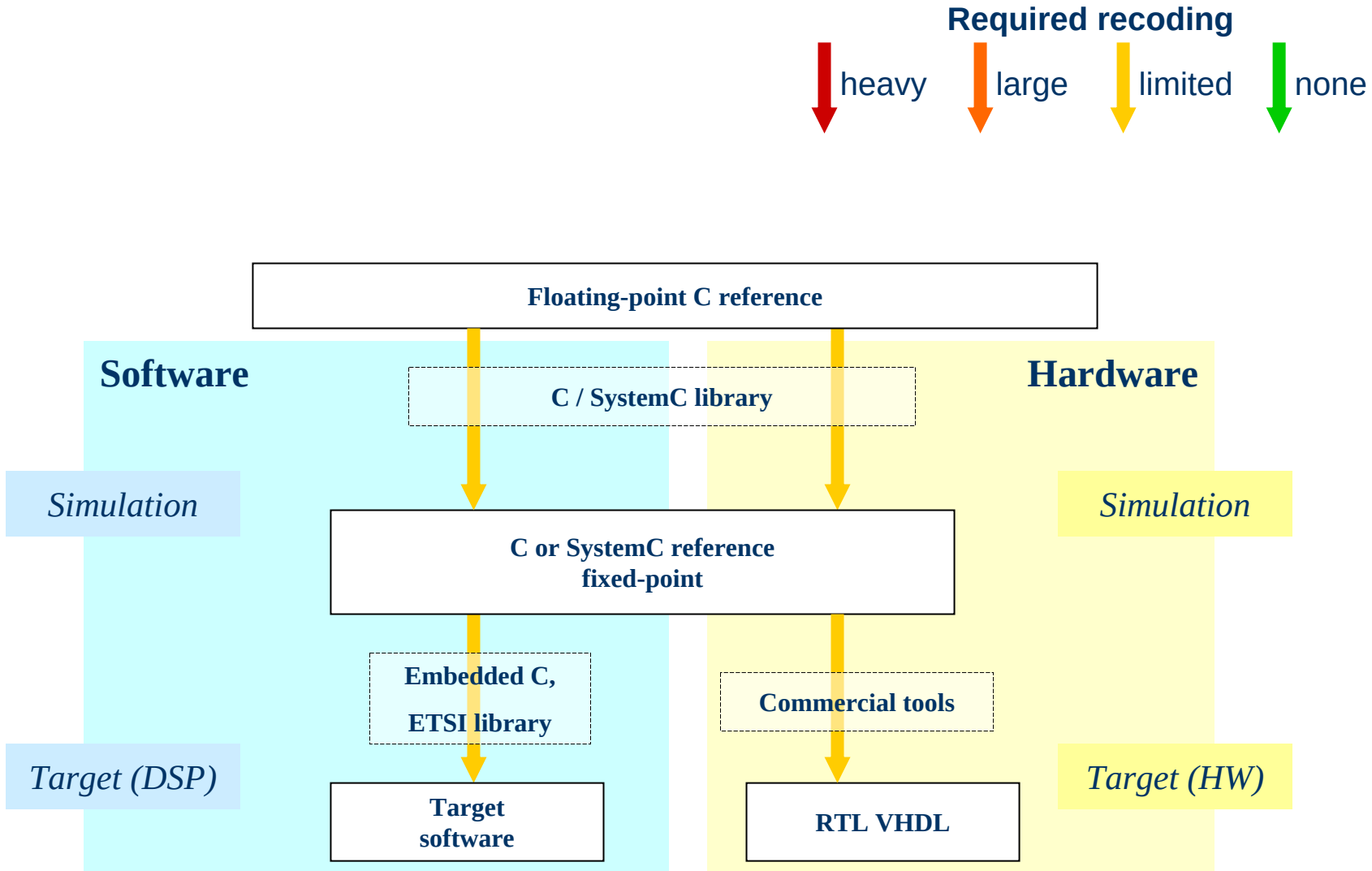
Design Space Exploration Graph



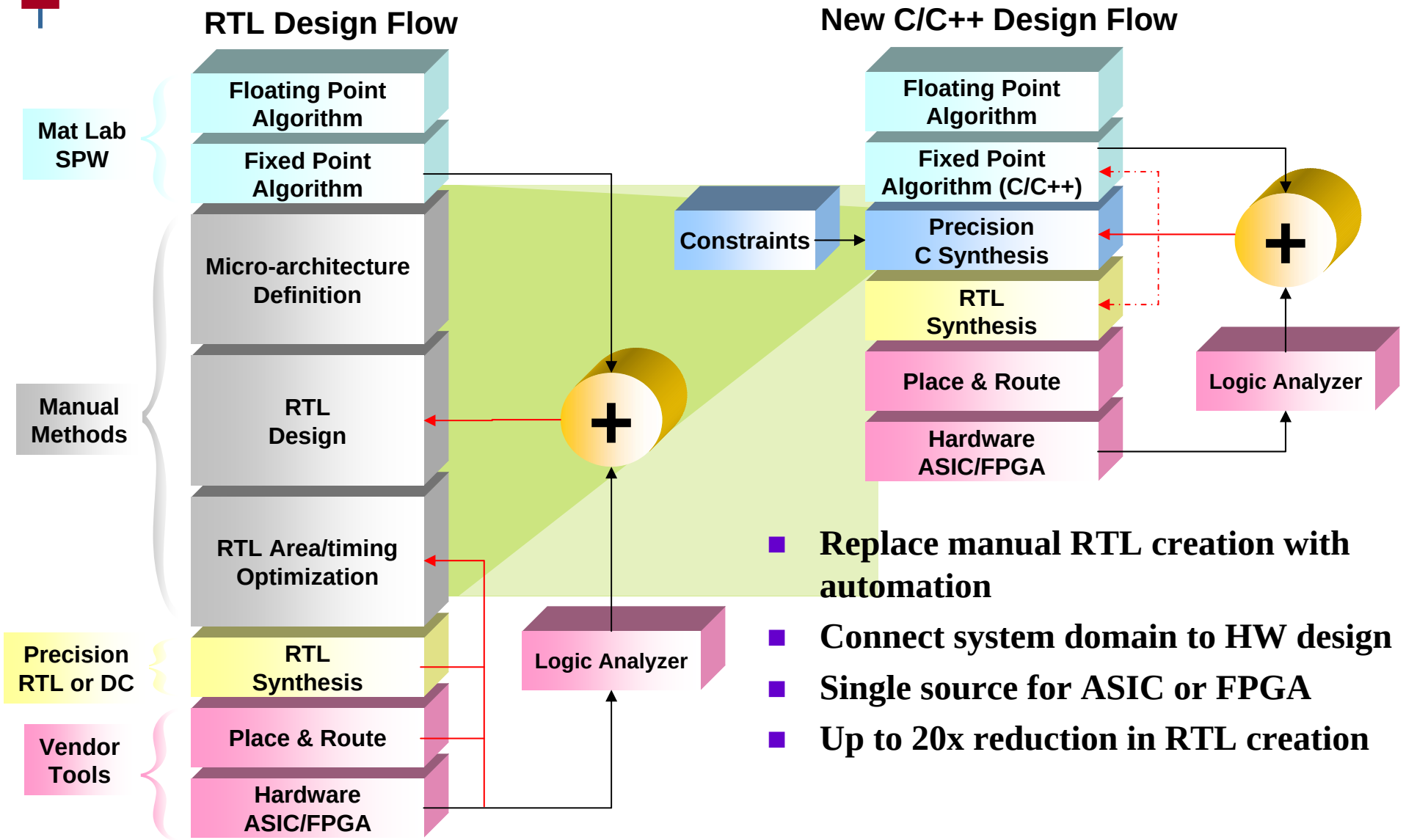


THALES

Target SLD flow for DSP



C/C++ hardware design flow for DSPs



- Replace manual RTL creation with automation
- Connect system domain to HW design
- Single source for ASIC or FPGA
- Up to 20x reduction in RTL creation

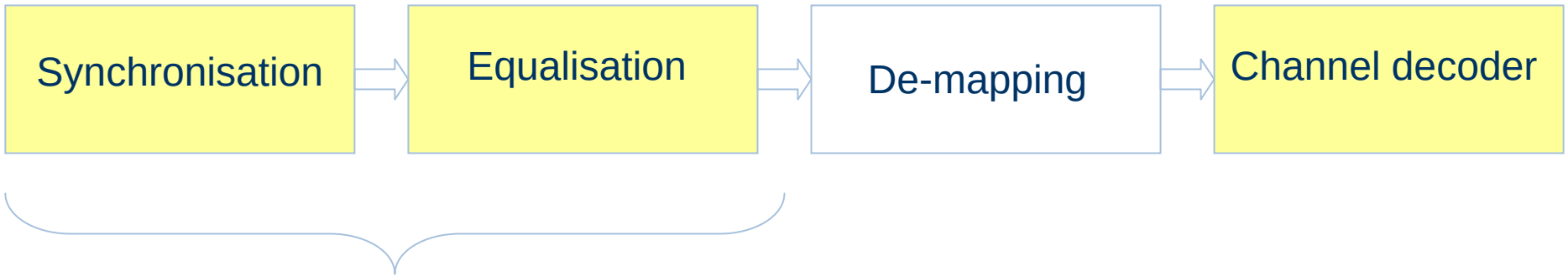
THALES

New development approach

- Focus on Digital Signal Processing Algorithms
- Modelisation in C fixed point
- Optimisation of the C code
- Design Space Exploration
  - Loop analysis
  - Ressource sharing
  - Pipelining
  - Memory mapping
- VHDL generation
- Synthesis and P&R
- Comparison with previous hand-made designs

Addressed modules during the study

Ref : Simplified modem overview



Pure digital signal processing
Based on filtering

Bit manipulation linked to DSP application

- Successful mapping of the algorithm in few days/ weeks
  - FFT (less than 1 day)
  - Viterbi (less than 2 weeks)
  - Synchronisation (less than 3 weeks)
  - Equalisation (less than 5 weeks)
- Achieved performances
  - FFT 120Mhz (80Mhz targeted)
  - Synchronisation 65Mhz (60Mhz targeted)
  - Egalisation 90/106Mhz (100Mhz targeted)
  - Viterbi 30Mhz (60Mhz targeted)

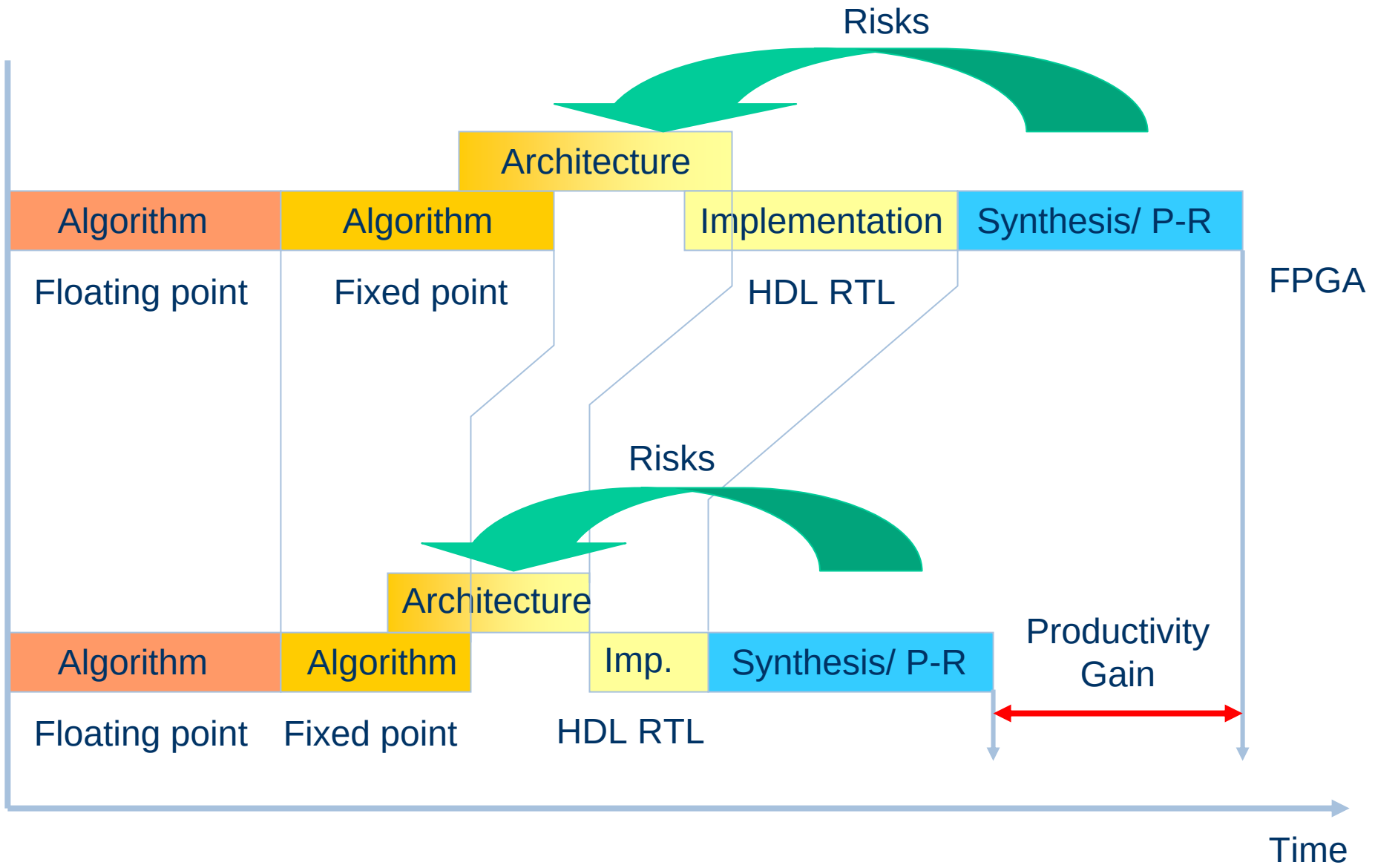
- Performances are closely linked to the original C algorithm code
  - C semantic to define and coding rules to implement
  - Recommendation : start from an "algorithm style" C reference and not from a "close to the architecture" C model
    - Viterbi module cycle cycle accurate C model returned bad results

Comparative results

	Results with C synthesis			Results from RTL approach			Results with SystemC		
Algorithm	**Frequency (MHz)**	**Surface**	**Time spent**	**Frequency (MHz)**	**Surface**	**Type**	**Frequency (MHz)**	**Surface**	
Viterbi	25	5240 CLB	3 weeks	60	2448 CLB	in-house	48	2600 CLB	2 months
FFT	111	1423 LC	3 days	158	~1400 LC	IP			
Synchronisatio	65	1295 CLB	3 weeks						
Equalization	106	7613 LC	5 weeks						

	C synthesis	System C	Hand-coded RTL
design productivity	good	moderate to good	moderate
architecture exploration and retargeting	easy	compliant	difficult
design optimisation	moderate to good	good	very good
reuse capability	very good for reference model	very good for reference model	good in HW IP dataflow

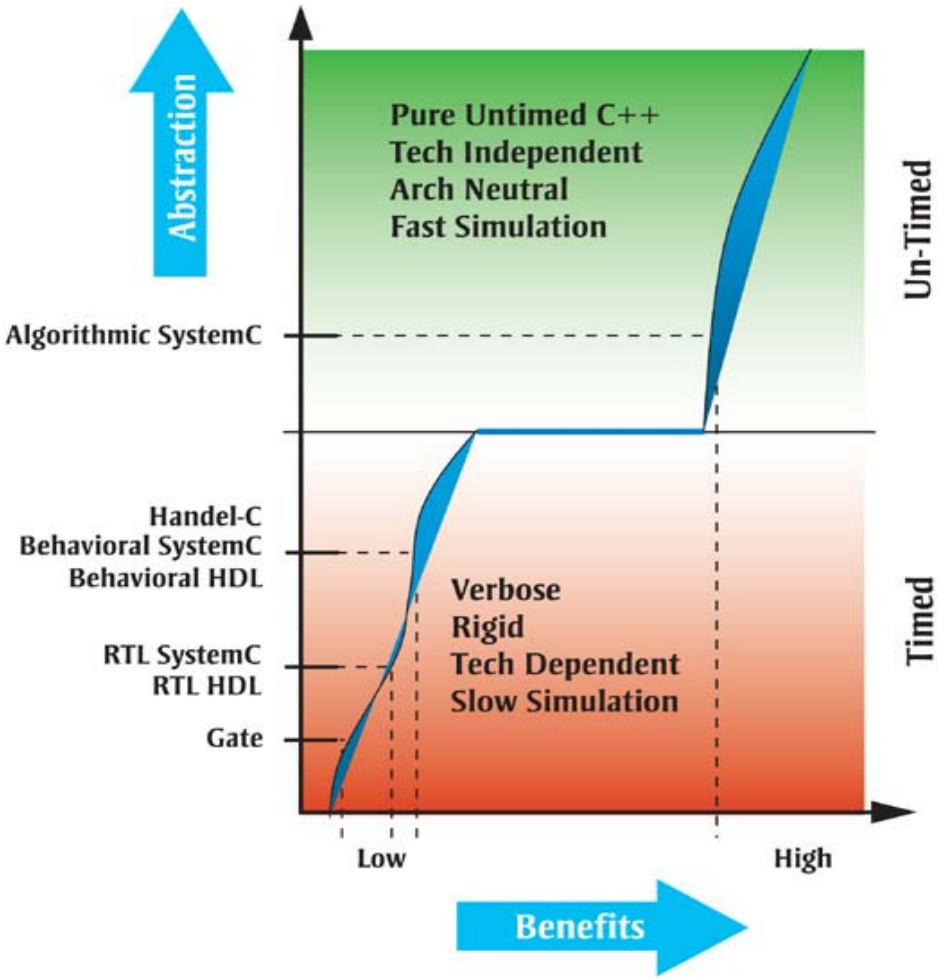
Benefit of such approach



THALES

- Advantages
  - Shorter design space exploration
  - Risk mitigation
  - Architecture trade-off
  - Single reference
- Drawbacks
  - New culture to set up
  - HDL is not the reference any more
  - Optimised implementation don't exist in all cases
  - Some limitations still exist
- Nevertheless, the approach is a real breakthrough to implement DSP algorithms in dedicated HW.

Advantage of the abstraction



Untimed C yields all the value

- 100 lines vs 1000 lines
- Simulation up to 10,000x faster
- Small to fast designs from the same source
- ASIC or FPGA from the same source
- RTL creation up to 20x faster

Conclusions

- C or SystemC based flow target better productivity:
  - A common reference
  - Quasi-automatic transition
  - Compliant TLM for architecture assessment

- Results :
  - SW path : no tools today for processor optimised code generation.
    Requires also a complete methodology move to C++ OO approach for DSP development (SystemC).
  - HW path : some promising solutions
    Applicable to FPGA and ASIC/SoC fast prototyping
    For algorithm/architecture fast exploration
    For DSP module developments