JÜLICH
FORSCHUNGSZENTRUM

Parallelisation potential of image segmentation in hierarchical island structures on hardware-accelerated platforms in real-time applications

by Sergey Suslov

Motivation

Aim

- automated real-time segmentation of large image data sets (~4096^2)
- high quality segmentation for wide application area

Status of Technology

- qualitative change in computer performance (late 90s):
  - less frequency growth but more computing parallelism, HW specialisation
- increase in IC integration level enables HPC in workstations
- many calculation intensive applications have intrinsic parallelism
  - benefit from parallel platforms
- three main approaches:
  - symmetric multiprocessor (SMP)
  - configurable logic – Field-Programmable Gate Array (FPGA)
  - stream processing architecture – Graphics Processing Unit (GPU)

Goals

- adaptation of image segmentation method (Gray-value Structure Code (GSC))
  - FPGA
  - GPU
- evaluation and comparison of
  - performance
  - implementation efforts
  - implementation flexibility
- generalisation of results for complex data processing methods

Outline

- Image Segmentation / GSC Method
- Implementation Platforms
- Implemented Models
- Parallelisation Analysis and Implementation
- Results
- Conclusions

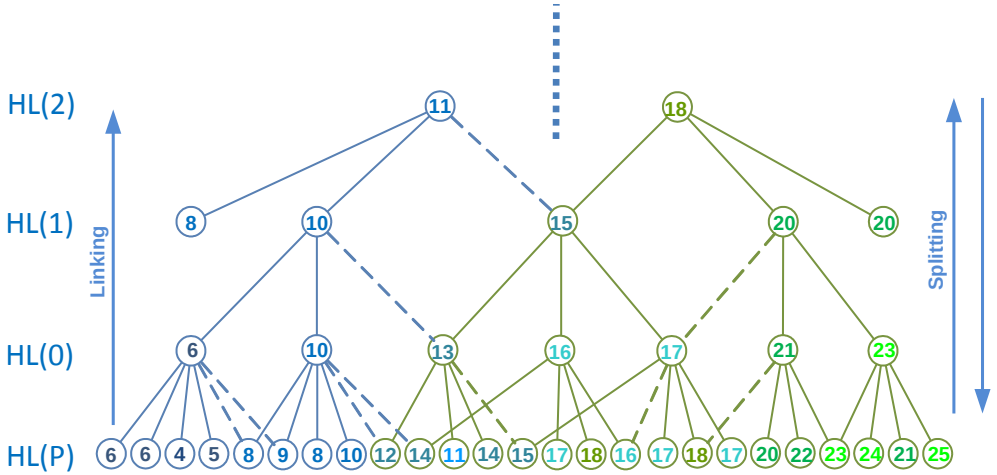
Image Segmentation

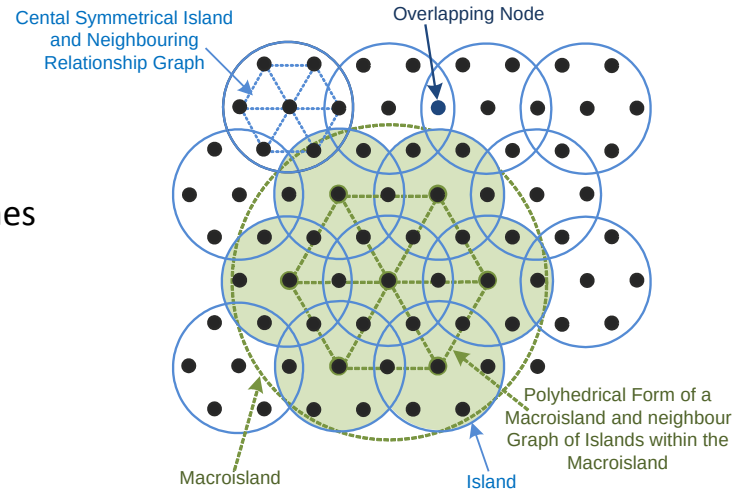
- spatial distribution of elements ⇒ image processing fits to parallel platforms
- typical image processing pipeline:

  pre-processing ⇒ segmentation ⇒ automated cognitive analysis
- image segmentation characterised:
  - applicability: universal ⇔ specific
  - control: automatic ⇔ human assisted
  - principle: discontinuity ⇔ homogeneity
  - main classes:
    - threshold-based or histogram-based techniques
    - edge-based methods
    - region-based methods
    - deformable models
    - classification-based methods
- real-time image segmentation: highly demanded task in image processing

Fundamentals of the GSC method

- hierarchical (multi-resolution) region-growing approach
- topology: hexagonal lattice
- region (R) forming criteria:
  - area
  - homogeneity of a feature (F)
  - spatial neighbourhood
- phases:
  - coding: ($island_{HL(P)}$; F(P); Neighbouring Graph) → $R_{HL(0)}$
  - linking: ($island_{HL(n)}$; $F(R_{HL(n)})$; Overlapping) → $R_{HL(n+1)}$, branches
  - splitting: generate disjoint segments (S) in region forest
  - result generation: F(S) → F(P)

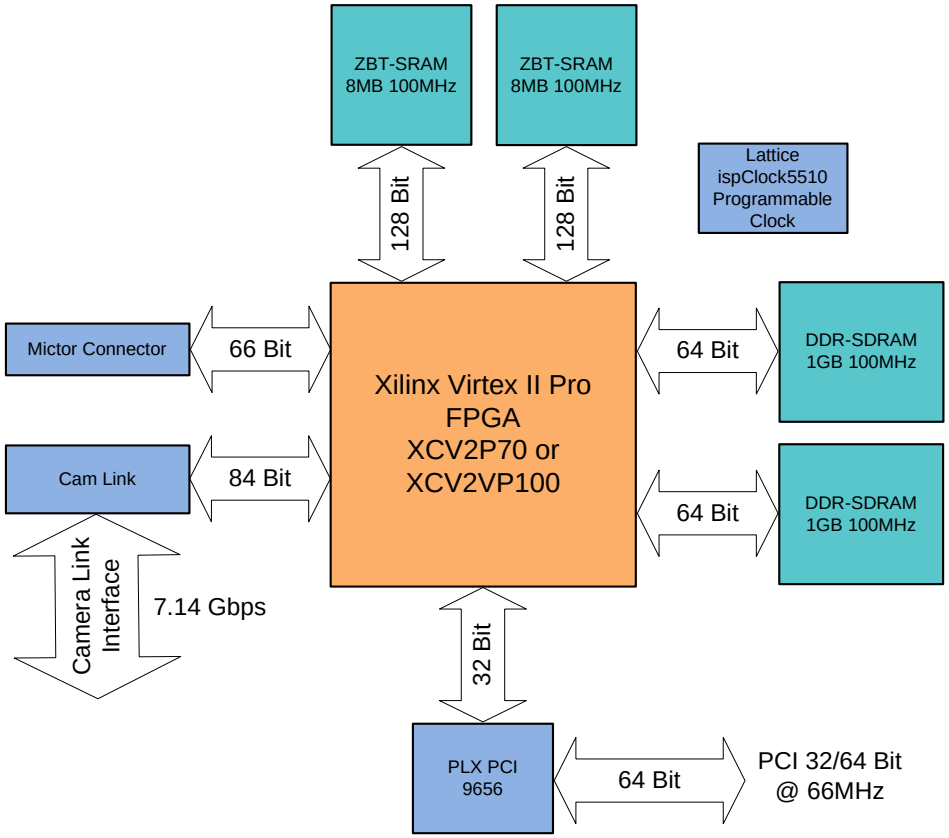




Implementation Platform FPGA

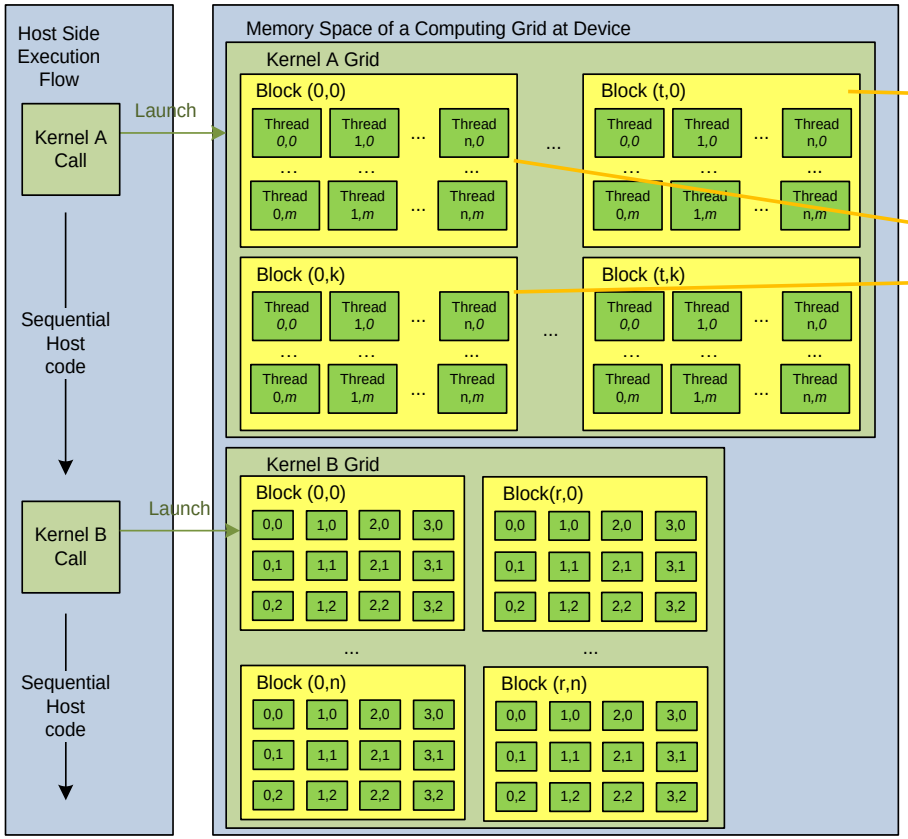
FPGA Expansion Board (Virtex II Pro)

- configurable logic (99,216 LCs @100Mhz)
- two DDR SDRAM banks (1GB@100MHz each)
- two ZBT SRAM banks (8MB@100MHz each)

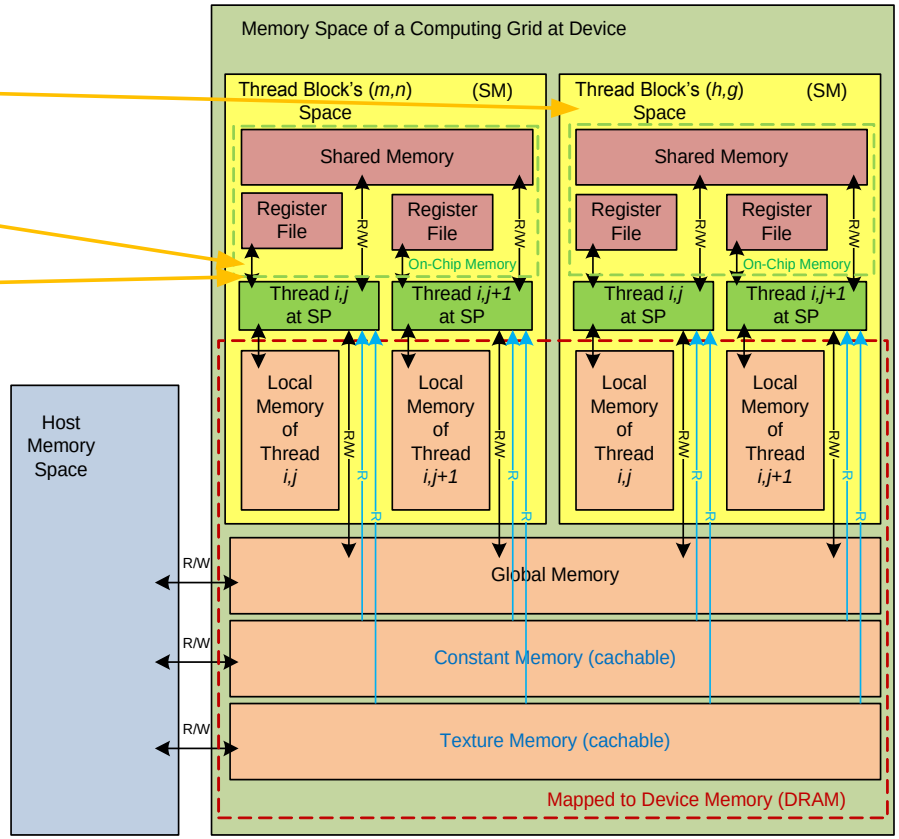


Implementation Platform GPU

Programming Model



Hardware Architecture

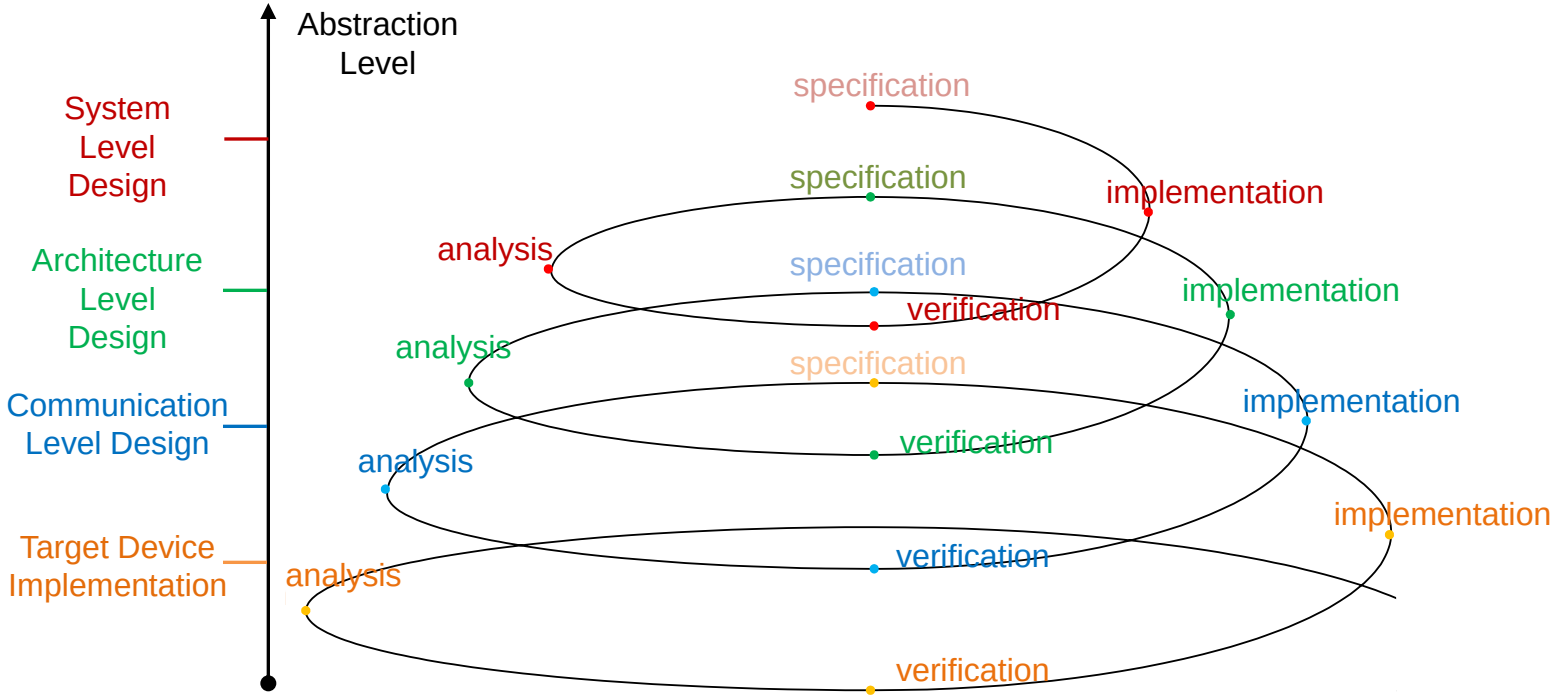


Tesla C1060: 240SP@602MHz; 4GB GDDR3@1.6GHz

Implemented Models

Methodology:

- top-down design approach
- gradual refinement
- cyclic: specification → implementation → verification → analysis



Implemented Models

Methodology:

- top-down design approach
- gradual refinement
- cyclic: specification → implementation → verification → analysis

Main GSC models:

(C, SystemC, SystemVerilog)

Common

- functional executable model
- static data-flow model

FPGA specific

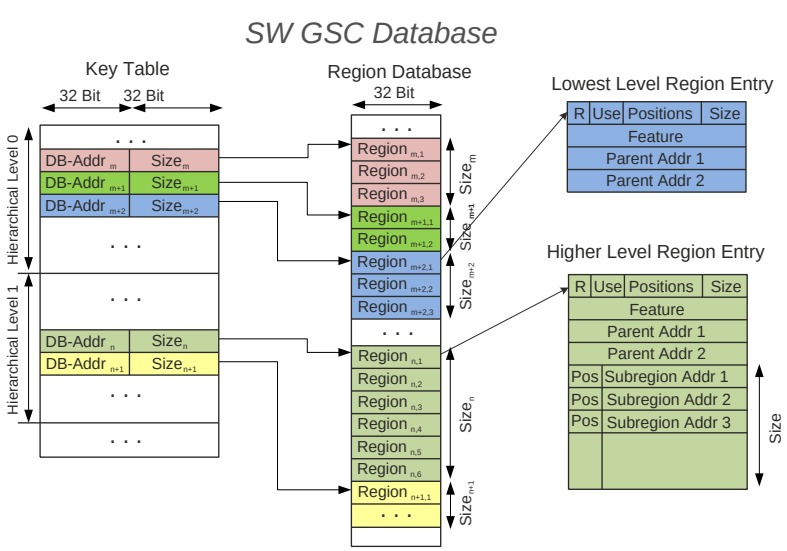
- static data-traffic model
- architectural executable model
- pre-implementation resource model
- interface and communicational model
- RTL synthesizable model

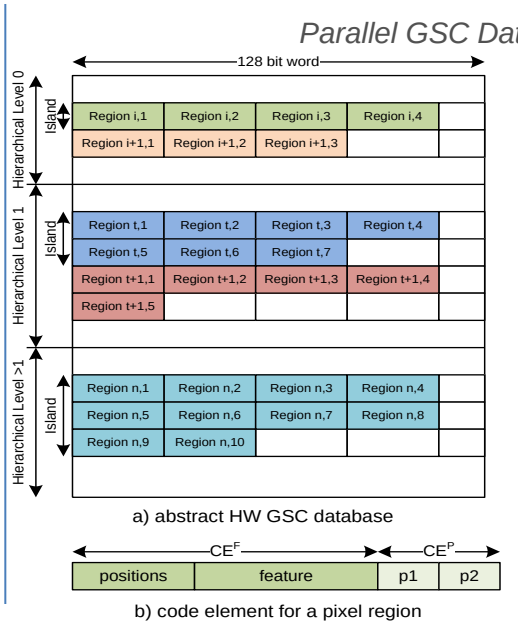
GPU specific

- SIMD functional model
- CUDA C program

GSC Hierarchy in Data-Structures

	Reference SW GSC	Parallel GSC	Parallel Implementation Applicability
Data allocation	dynamic, island key-table	regular 2D array for each HL(i)	schedulable seamless data streaming, no data access conflicts for concurrent processing
Island representation	arbitrary number of regions in islands(IS), region size is not fixed	IS size fixed, number of regions depends on HL(i), statistics based	fixed size ISs, predictable time for island processing
Hierarchy representation	bidirectional: upward and downward hierarchical pointers	upward pointers for parental regions	memory economy
Island addressing scheme	redirection via key-table	relative, defined by island coordinates	enables synch. prefetching of islands from adjacent levels
Memory layout	compact data structures	arrays of structure subfields	avoids load-modify-store operations, burst-oriented accesses



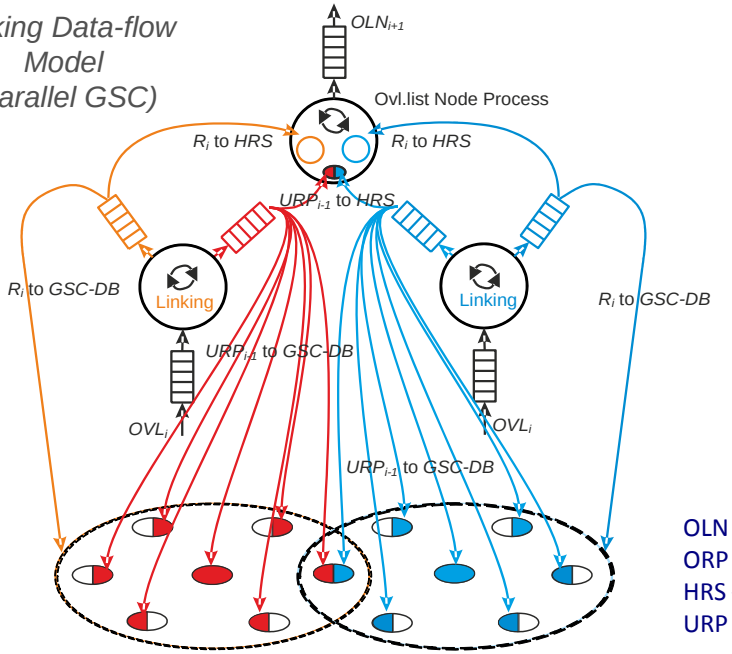


CE – code element, region info descriptor
ISi – island structure HL(i); [n] – index within an IS
feature – 8-bit int pixels, 8.4-bit fixed point internally

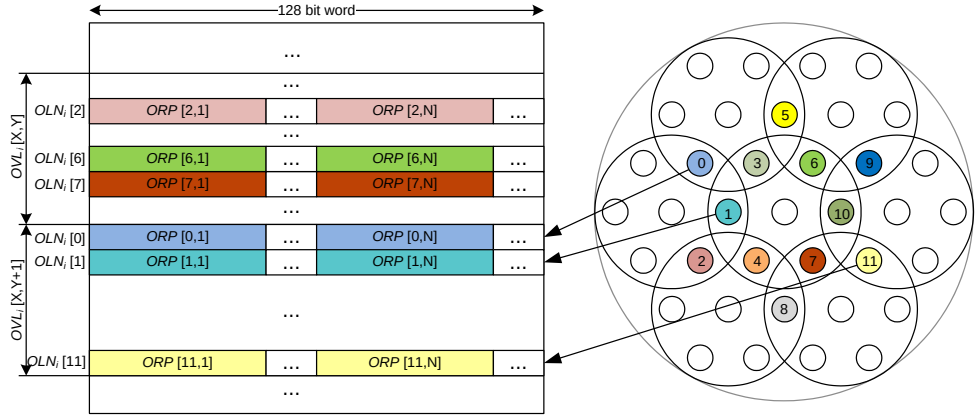
Features of GSC Implementations

	Reference CPU GSC	FPGA Parallel GSC	GPU Parallel GSC
Coding	recursive pixel-island graph traversing	structural, pipelined	SIMD, pipelined
Overlap detection	during linking $R_{HL(i)}$, second parent detection by database indirections: $R_{HL(i-1)} \rightarrow R_{HL(i-2)} \rightarrow R_{HL(i-1)}$	dedicated overlap-list structures $OVL_{HL(i)}$, during linking $R_{HL(i-1)}$, stored in separate OVL HL(i) array	similar to HW GSC, but OVLs are generated as the second pass over HL(i-1) after linking
Linking	sequential database search, recursive	topology aware OVL traversing, stack for rolling back to branch point	represents OVLs to overlap connectivity vectors(CV), bit-wise operation on CV table
Splitting / Result generation	recursive downward separation of hierarchical relationship tree, complete tree in a single pass	level-wise downpropagation, decides for most similar parent	similar to HW GSC

Linking Data-flow Model (Parallel GSC)



Overlap-list Structure



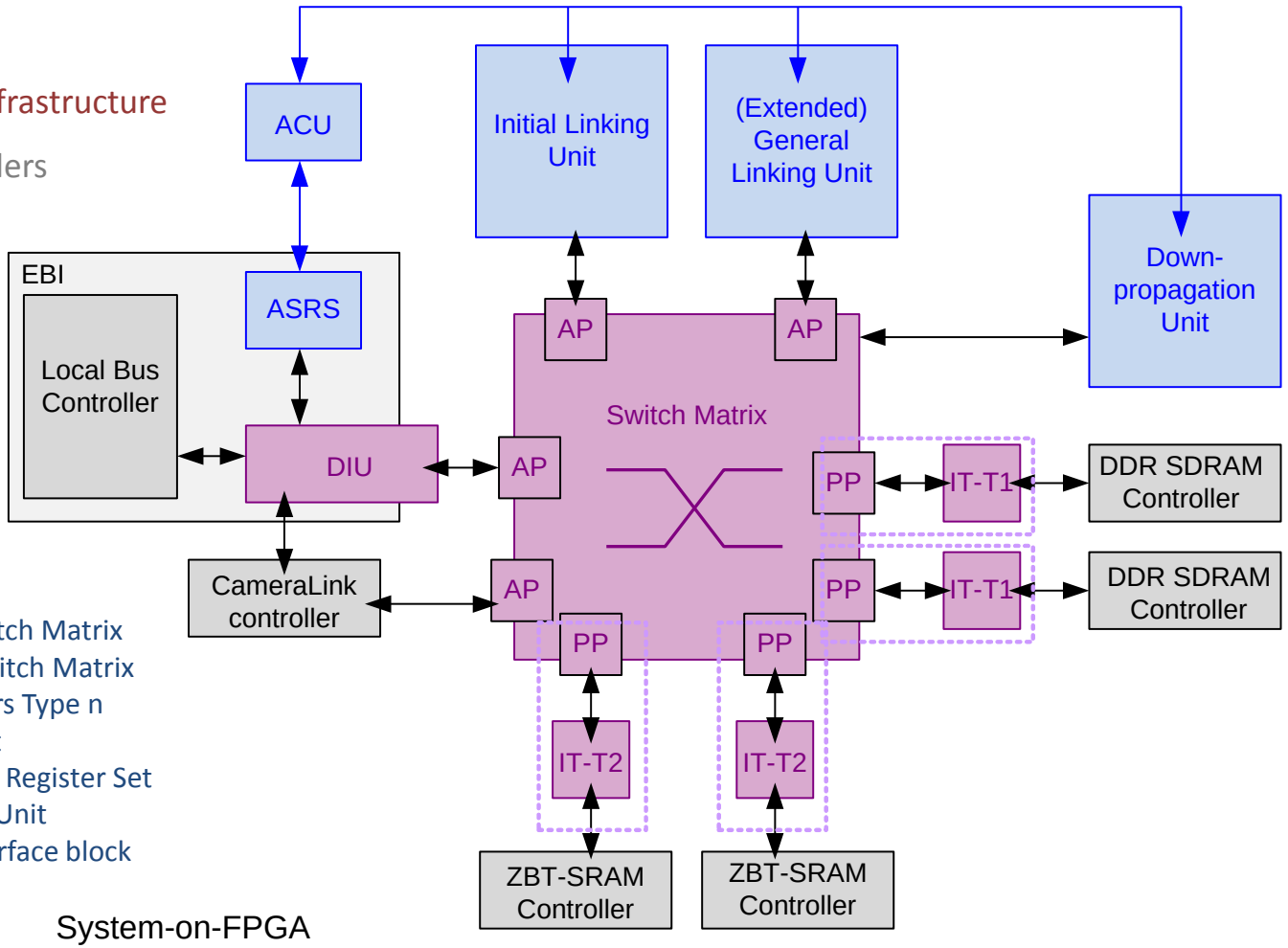
OLN – overlap node structure ∈ OVL
ORP – overlap region pair ∈ OLN
HRS – hierarchical relationship structure
URP – upward relationship pointers

R – region
GCS-DB – GSC database
i – subscript index denoting HL(i)
[m,n] – element position in an array

System on the FPGA Chip

System Blocks:

- GSC specific
- communication infrastructure
- peripheral controllers

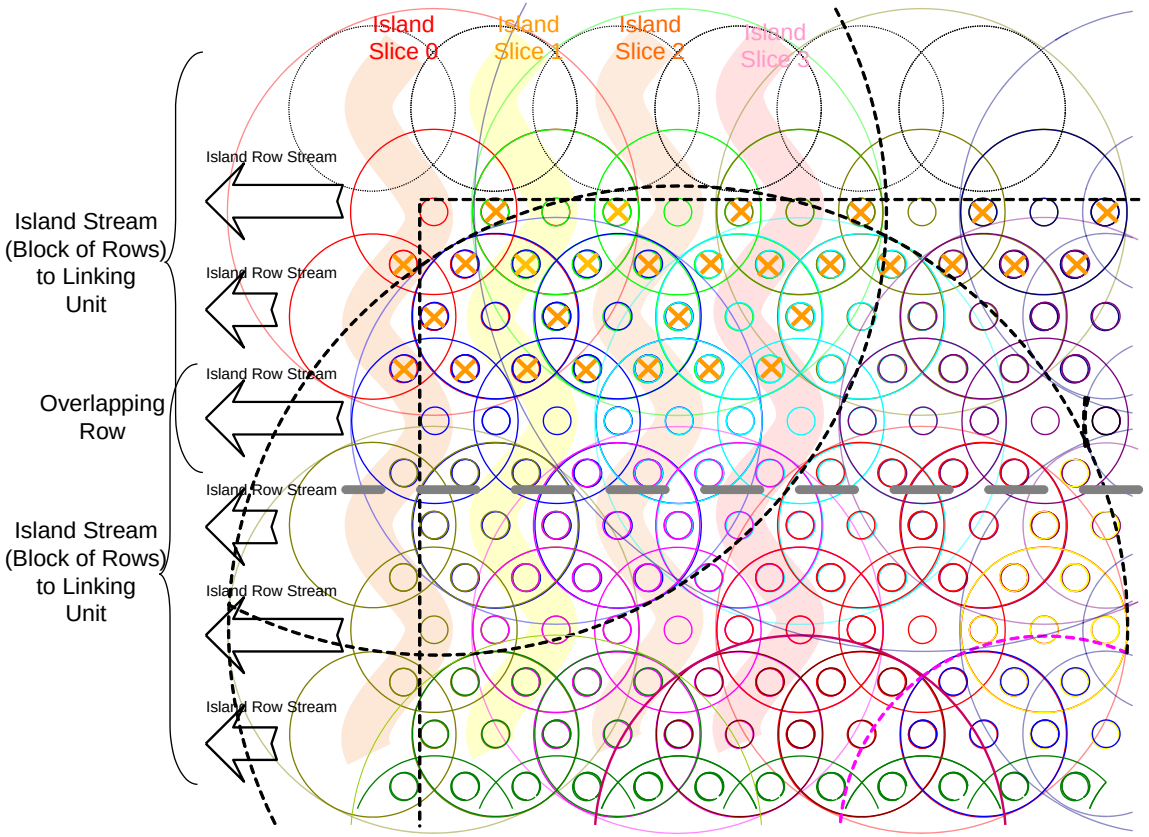


AP – Active Port of the Switch Matrix
PP – Passive Port of the Switch Matrix
IT-Tn – Interface Transactors Type n
DUI – Device Interface Unit
ASRS – Application Specific Register Set
ACU – Application Control Unit
EBI – Expansion Board Interface block

System-on-FPGA

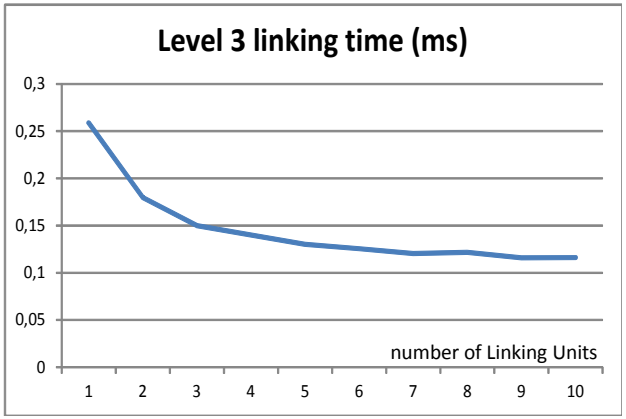
Data Streaming

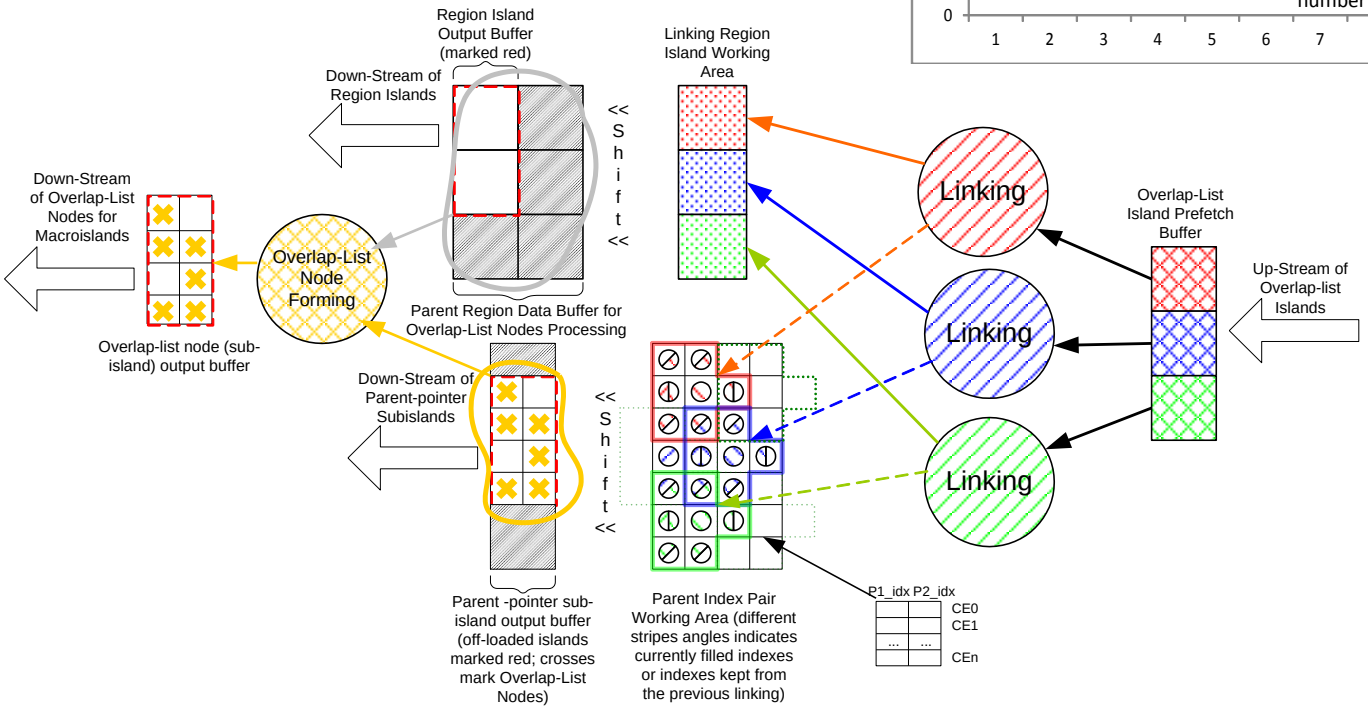
- island structures are buffered in regular scheme => continuous up- and downward streams of islands
- islands are processed column-synchronous => data shared within a column and among adjacent columns
- adjacent blocks of rows must overlap in one row of islands => no need for read-modify-write operations for updating parent pointers in subCEs and overlapping pairs in OLNs, inevitable for downpropagation



Balancing Pipelines (FPGA Linking Unit Example)

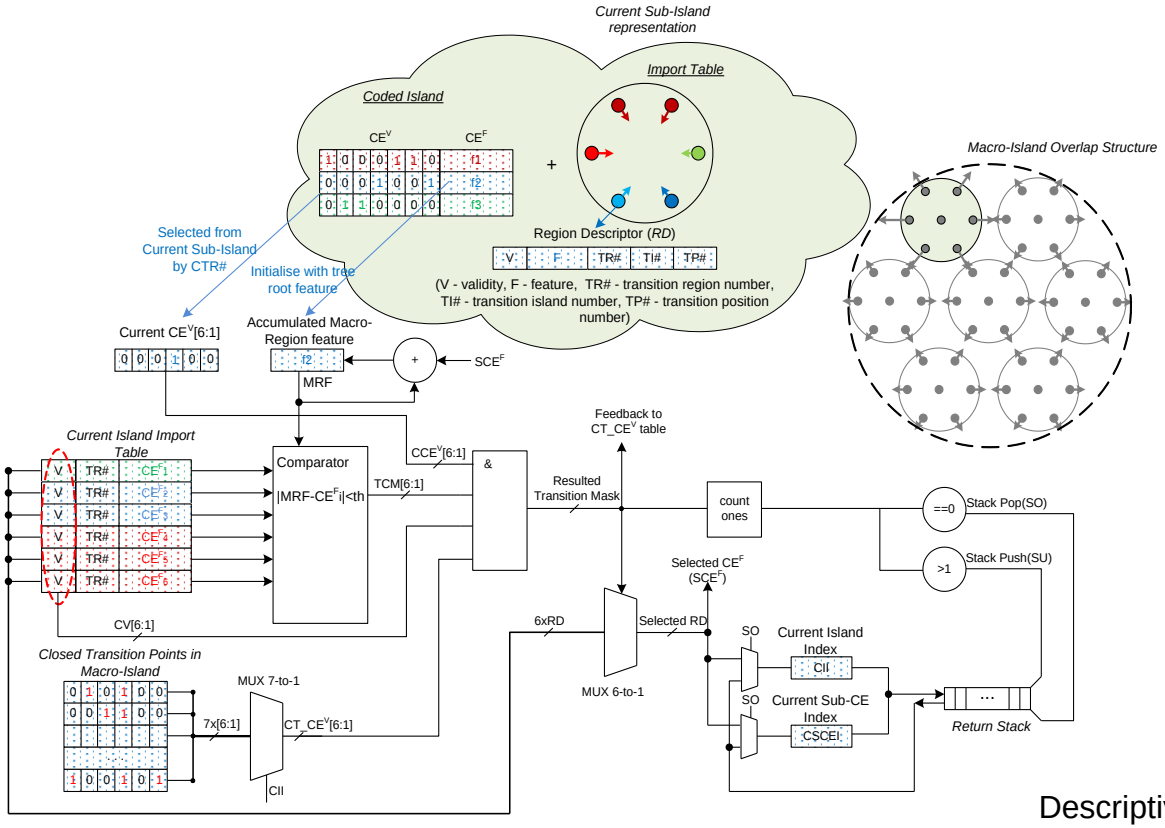
- pipeline throughput limited by the slowest stage
- processing power limited by memory throughput
- processing latency > data latency
  ⇒ increase processing parallelism
- channel bandwidth saturated
  ⇒ no need for excessive processing power
- traffic multiplexing: balance external channels

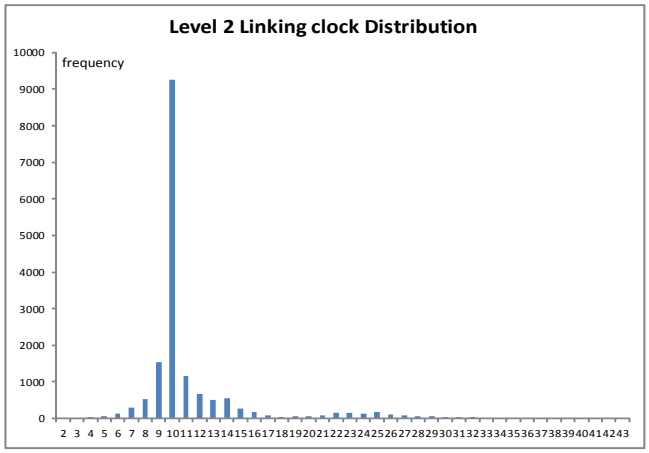




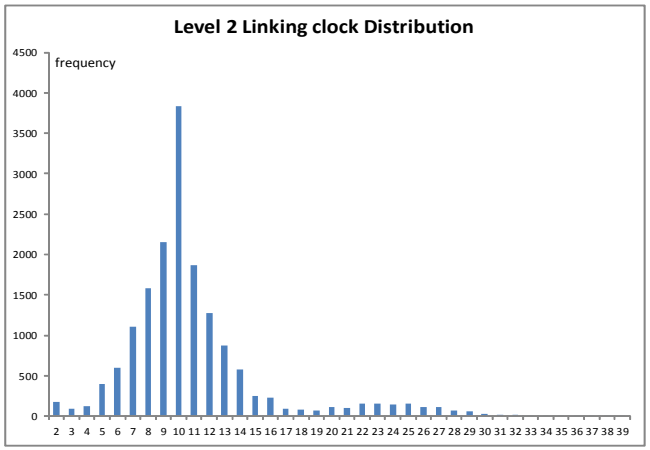
Topology-driven Linking

- traversing OVLs is guided by topological relationship of islands
- stack is used for storing branching points within linking trees
- transition gates are closed after passing through them to neighbours
- rolling back to branching points, if no gates left open in the current IS





Portrait



Aerial space photo of a town

Descriptive Statistics:

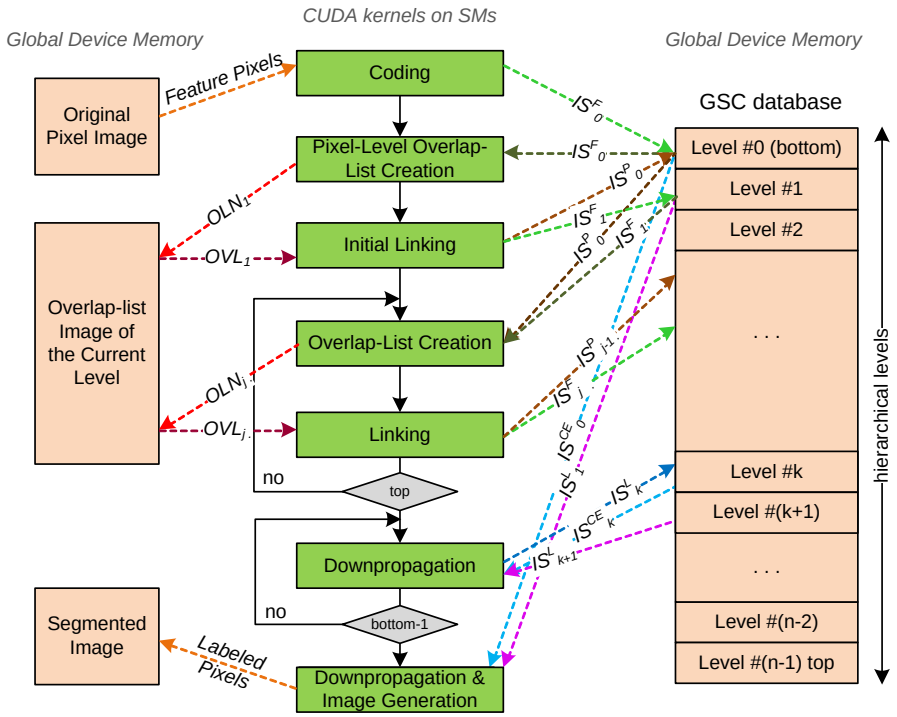
$$T_k, k: \sum_{1}^{k} \bar{p}_i \leqslant 0{,}95; \quad T_n, n: \forall \bar{p}_i \leqslant 0{,}01, i \in \lfloor n:m \rfloor;$$

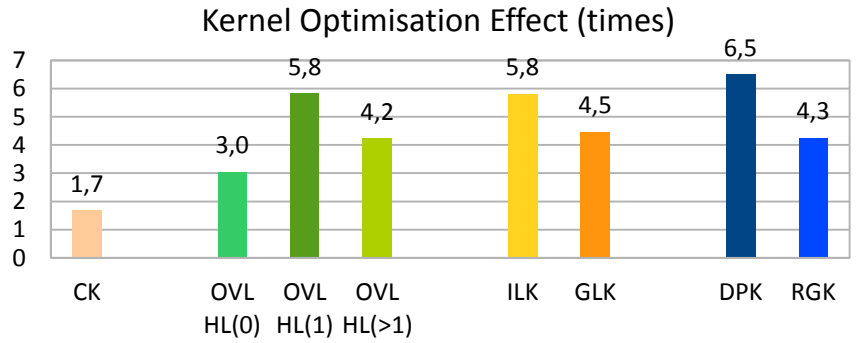
$$T_t = \left(\sum_{1}^{m} p_i * i\right) / \sum_{1}^{m} p_i$$

GPU implementation

Optimisation strategy

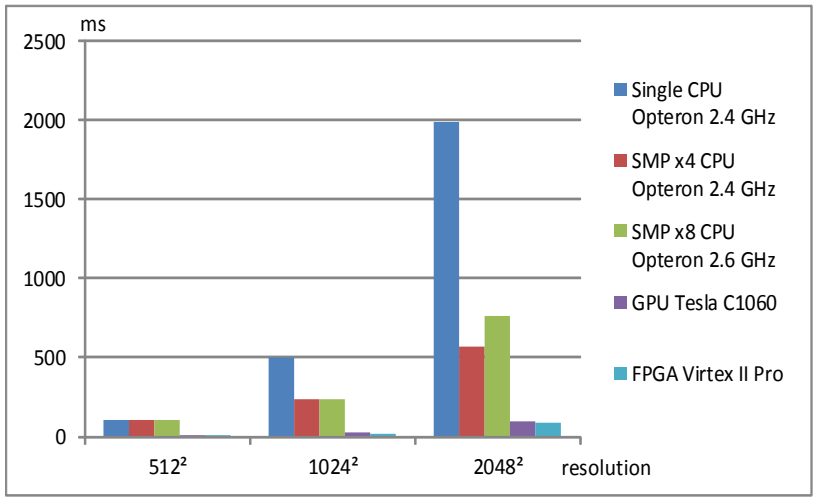
- Global memory optimisation
  - addressing scheme
  - data structure layout
- Memory architecture peculiarity exploitation
  - texture memory
  - constant memory
- Shared memory layout
  - addressing scheme
  - data volume optimization
  - memory reuse
- Kernel control-flow optimisation
  - execution path optimisation
  - branching minimisation
- Operation level optimisation
  - intrinsic GPU instructions
  - specialised math
- Block size parameters selection



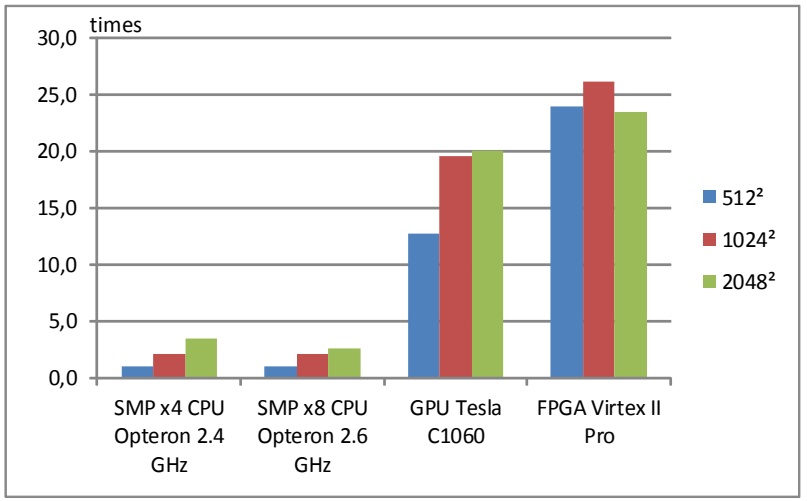


CK – coding
OVL HL(n) – OVL creation
ILK – initial linking
GLK – general linking
DPK – downpropagation
RGK – results generation

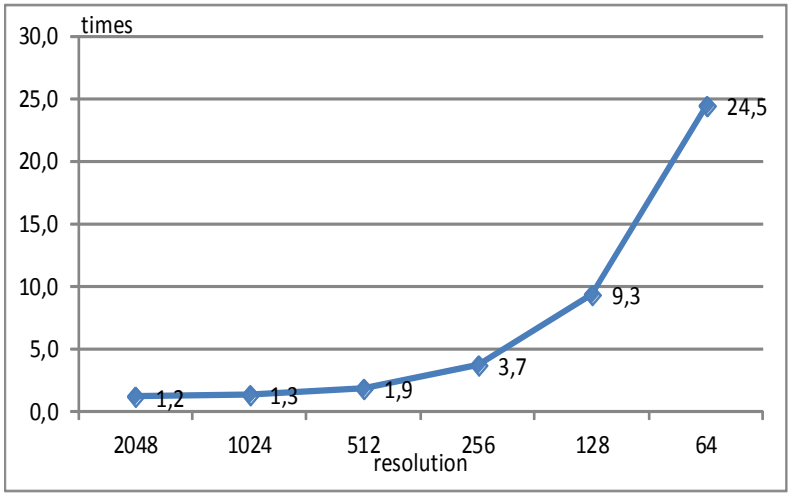
Results: Performance



Elapse Time

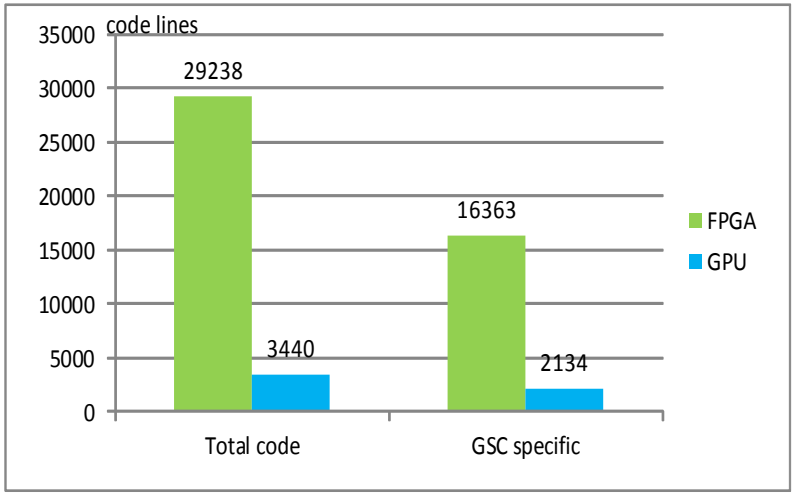


Performance Acceleration Ratios

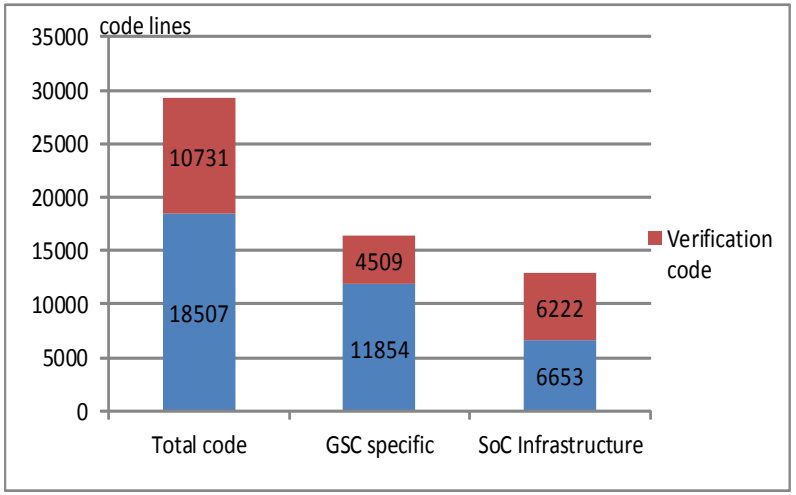


GPU to FPGA processing time ratios

Results: Efforts



Efforts for FPGA and GPU solutions



FPGA code structure

How to measure?

- Market price
  - most objective, but equivalents not always found in the market
- Manpower
  - straightforward, but additional administrative overheads, equivalent skills, and no education curve assumption
- Number of code lines
  - looks most naïve, but best if the designer is the same for different implementations: similar coding style, no education curve assumption, only matured code compared

Conclusions (1)

Performance

- 2048^2: 85 ms (FPGA), 99 ms (GPU) vs. 1985 ms (CPU)
- achieved acceleration endorses the use of the parallel GSC in real-time and interactive applications
- even older generation FPGA beat newer GPU processors

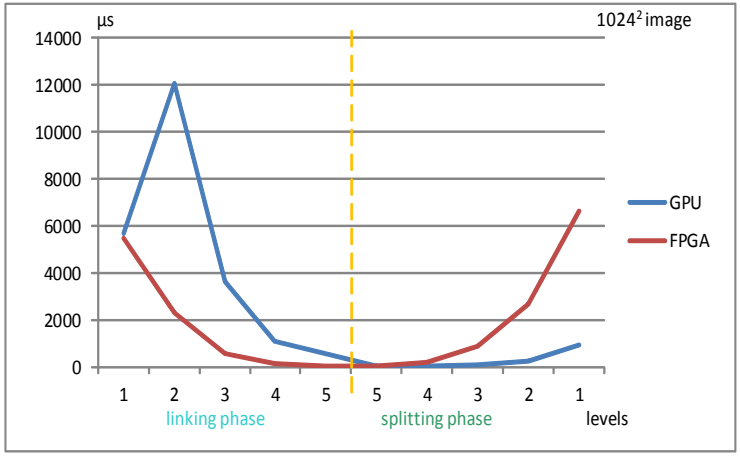
Efforts

- analysis of the method
  - extreme important
  - discover the parallelization potential of the algorithm
- development cycle is significantly longer for FPGA

Conclusions (2)

Flexibility

- FPGA
  - freedom for the organization of the computation
  - application specific buffering and data scheduling schemes
    - more effective bandwidth utilization
    - more elastic to data volumes
  - more flexible in data organization due to fine grained parallelism
  - lack of complex hardwired functional blocks

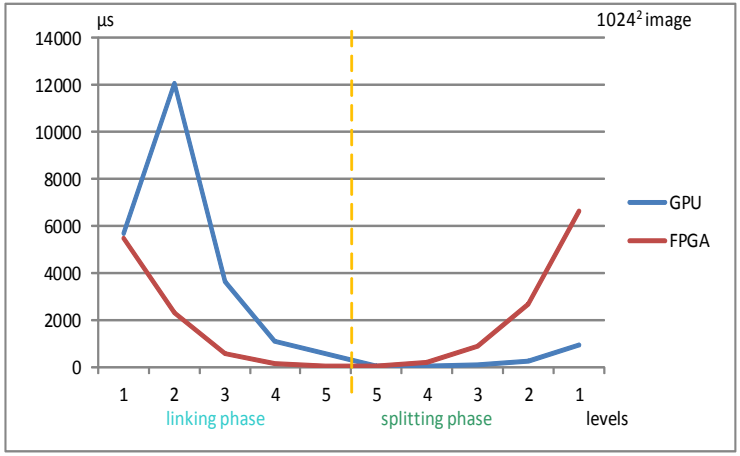


FPGA and GPU combined profile

Conclusions (3)

Flexibility

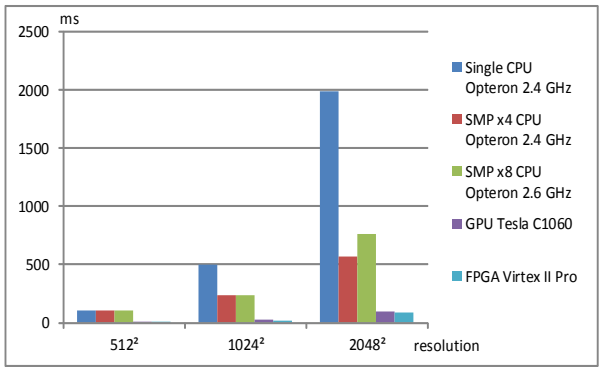
- GPU
  - adaptation of the algorithms to the target architecture
  - sensitive for complex code execution flow
    ⇒ branching minimization
  - effective for huge amount of data
  - ISA imposes restrictions on data compaction
  - sensitive to temporary data in on-chip memory
    ⇒ memory reuse techniques
  - sensitive to external data layout



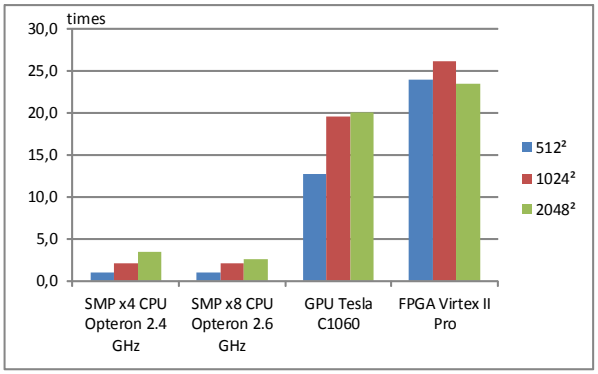
FPGA and GPU combined profile

Results Summary

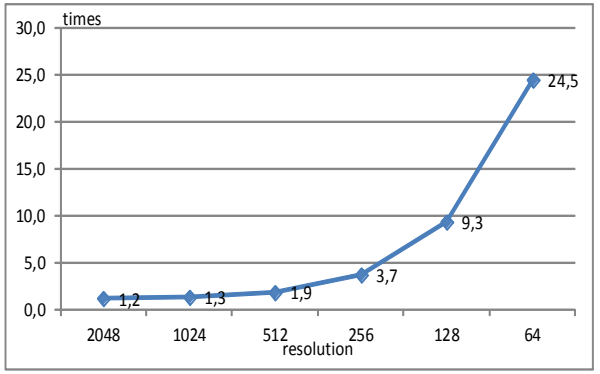
Performance



Elapse Time in ms

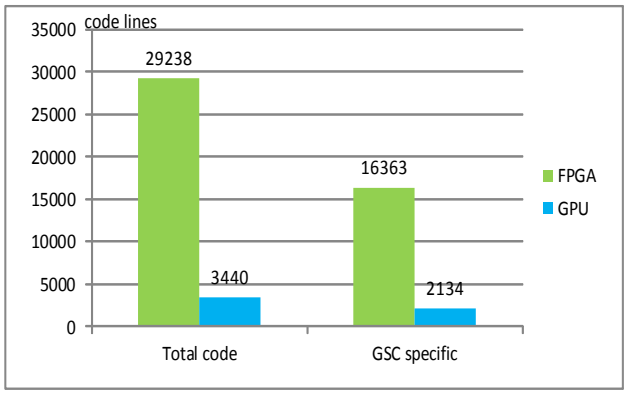


Performance Acceleration Ratios

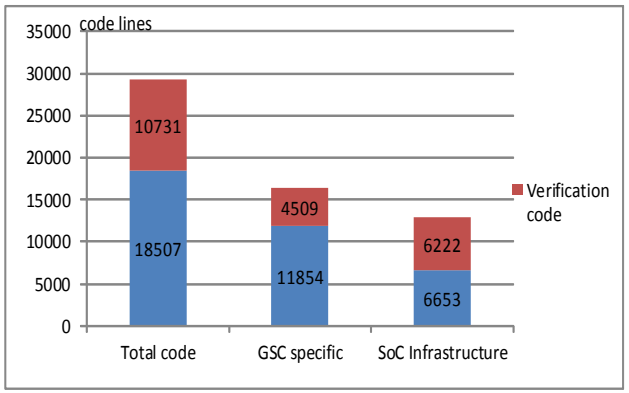


GPU to FPGA processing time ratios

Efforts



Efforts for FPGA and GPU solutions



FPGA code structure