High-level OpenCL implementation of FDAS in Pulsar Search on FPGAs

Oliver Sinnen Haomiao Wang
&
Prabu Thiagaraj (Manchester Uni)

Parallel and Reconfigurable Computing

Department of Electrical and Computer Engineering
University of Auckland

Computing for SKA, 2016

Outline

Outline

Where within SKA project

- Square Kilometre Array (SKA) project will be the world's largest radio telescope array, and is currently in Phase 1 (SKA1)
- Initially three telescopes for SKA1: LOW, **MID** and SURVEY
- SKA1-MID packages: **Central Signal Processor (CSP)** and Science Data Processor (SDP)
- CSP system is comprised of 3 major sub-elements

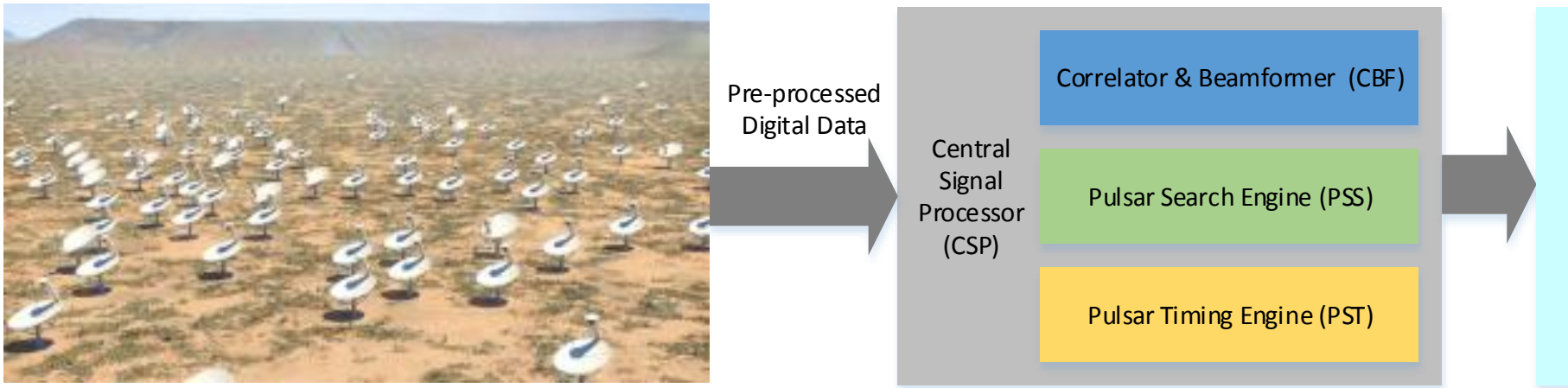


Figure: Structure of SKA1-MID CSP system

Pulsar Search

- Pulsar is highly magnetized and rotating neutron star, emitting electromagnetic radiation, which is a pulse when observed
- Period of signal of a (single) pulsar is very precise
  - Makes it research object for a range of areas
- Special interest: **binary pulsar**
  - Pulsar with a binary companion (other pulsar or black hole) is ideal object to test general relativity among other things
- Because binary pulsar has own orbit period (which is unknown), it is impossible finding binary pulsars using general pulsar search methods

Pulsar Search

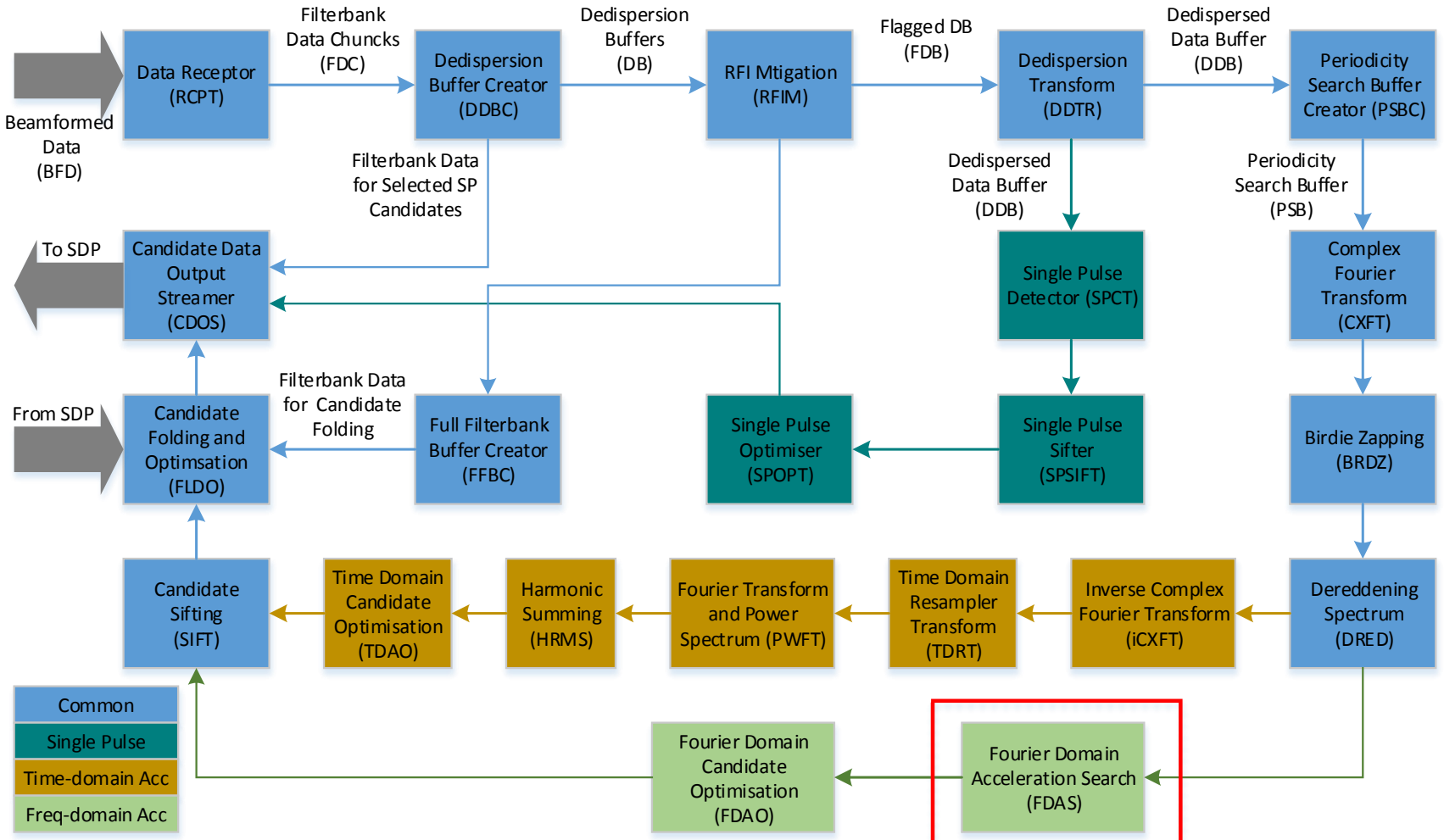
- When orbit period of binary pulsar is order of magnitude higher than observation time, **frequency derivatives** of pulsars are **constant**.
- Using **matched filtering** technique in Fourier domain to recover the signal into single bin.

$$A_{r_0} \backsimeq \sum_{k=[r_0]-m/2}^{[r_0]+m/2} A_k A^*_{r_0-k},$$

where frequency r_0 is unknown.

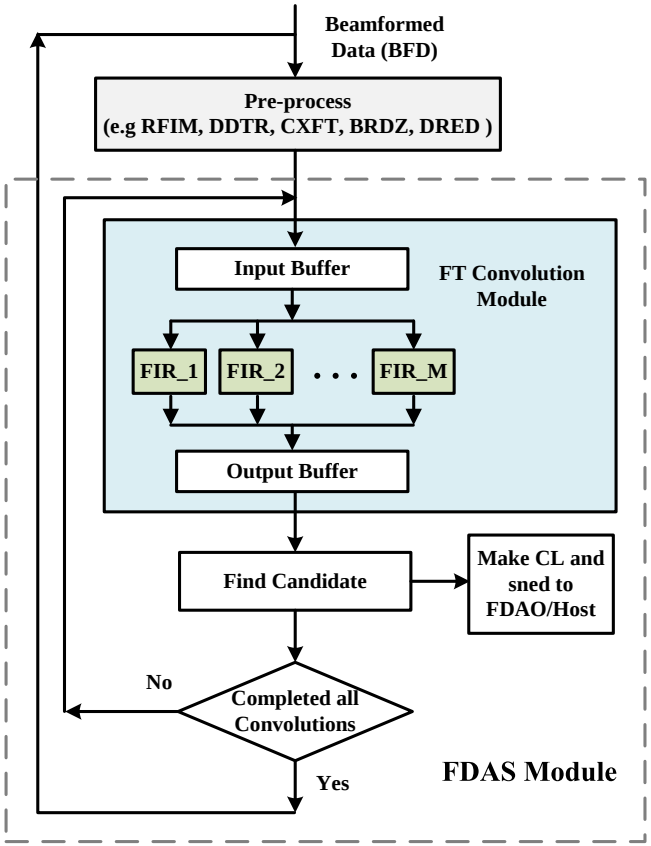
- Summation is computed at a range of frequencies r.

Block Overview of Pulsar Search Engine



Fourier-domain Acceleration Search (FDAS)

FDAS module is applied to search for (binary) pulsars with constant frequency derivatives in frequency-domain



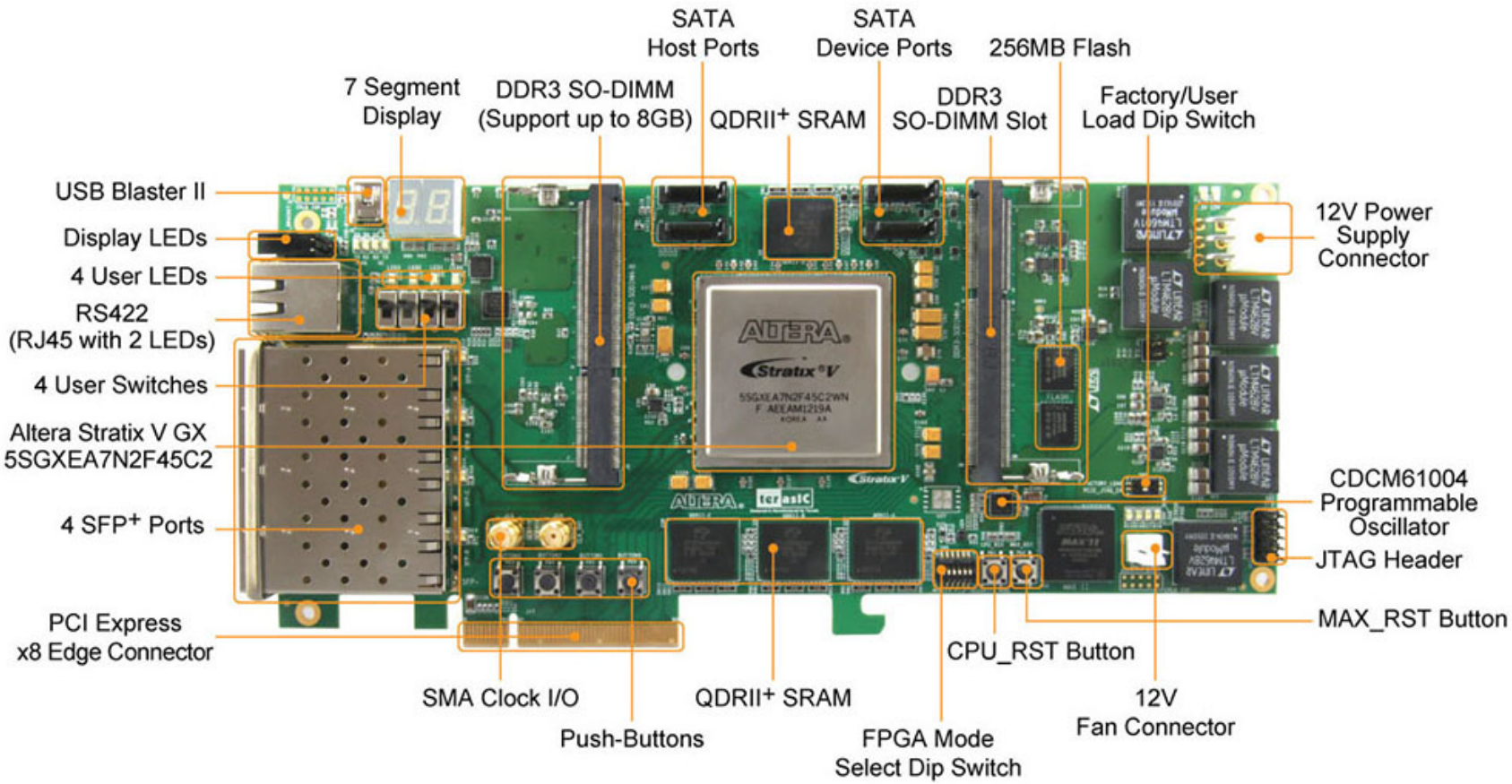
Specification of task

Main part of FDAS module is convolution with numerous filters

Parameter	Destriiption	Value
B	# of beams	$1000 \sim 2000$
N	# of complex samples per group	2^{22}
M	# of templates/filter	84
K	# of average template/filter length	222
W	Workload of one beam	6.26×10^{11}
t_{limit}	Time of executing one sample group	$88ms$
P	Performance needed per beam	$7.1 TFLOPS$

OpenCL for FPGA

Altera SDK for OpenCL (AOCL) 1.0



Outline

Method Overview

Objective:
Estimating required number of FPGAs in *time-domain*:

- Maximum Performance
- Parallelisation of Multiplications

***K*-tap FIR filter *m*:**
output y_m is convolution of filter coefficients (h_m) with input signal (x_m),

$$y_m[i] = \sum_{k=0}^{K-1} x_m[i-k]h_m[k], \text{ for } i = 0, 1, \ldots N-1.$$

If h and x are all complex single precision floating-point (SPF) data, 8 **operations (flops)** are needed to accumulate one product.

Maximum Performance

- Estimate based on maximum achievable operation performance of FPGA
- Overall workload: $W = 8NKM$ operations
- => number of needed FPGAs

$$F = \lceil \frac{P}{p} \rceil = \lceil \frac{W}{pt_{limit}} \rceil$$

- Implemented complex FIR filters with OpenCL

For DE5 board with Stratix V A7 FPGA, our best optimized implementation got $p \approx 110$ GFlops
=> need **65** FPGAs – per beam!

Parallelisation of Multiplications

- Focusing on critical operation: multiplication
- DSP blocks (27 x 27 bit multipliers) are used on FPGA to implement SPF multiplication
- DSP blocks are limited resource on FPGA
- Lower bound on required FPGAs
-

$$F = \lceil \frac{4 W_{CM}}{f \cdot t_{limit} \cdot D} \rceil .$$

With W_{CM} being number of complex multiplications, f operation frequency and D number of DSP blocks on FPGA

- Stratix V A7 : $\approx$ **60** FPGAs needed, Stratix V D5 (top-end current FPGA): $\approx$ **10** FPGAs need
- Stratix 10 top-end has roughly 2 x # DSPs and higher f

Parallelisation of Multiplications

- Focusing on critical operation: multiplication
- DSP blocks (27 x 27 bit multipliers) are used on FPGA to implement SPF multiplication
- DSP blocks are limited resource on FPGA
- Lower bound on required FPGAs
- $$F = \lceil \frac{4 W_{CM}}{f \cdot t_{limit} \cdot D} \rceil .$$

  With W_{CM} being number of complex multiplications, f operation frequency and D number of DSP blocks on FPGA
- Stratix V A7 : $\approx$ **60** FPGAs needed, Stratix V D5 (top-end current FPGA): $\approx$ **10** FPGAs need
- Stratix 10 top-end has roughly 2 x # DSPs and higher f

Relaxing Requirements

- Data type (from SPF to fixed-point)
- Input Length (N ranges from 2^{16} to 2^{22})
- Filter number and size (reduce $\sum K_i$)
- Time limitation (increase t_{limit})

Relaxing Requirements

By applying relaxation methods, number of FPGAs decreased. If all these methods are applied, over 90% of FPGAs can be saved.

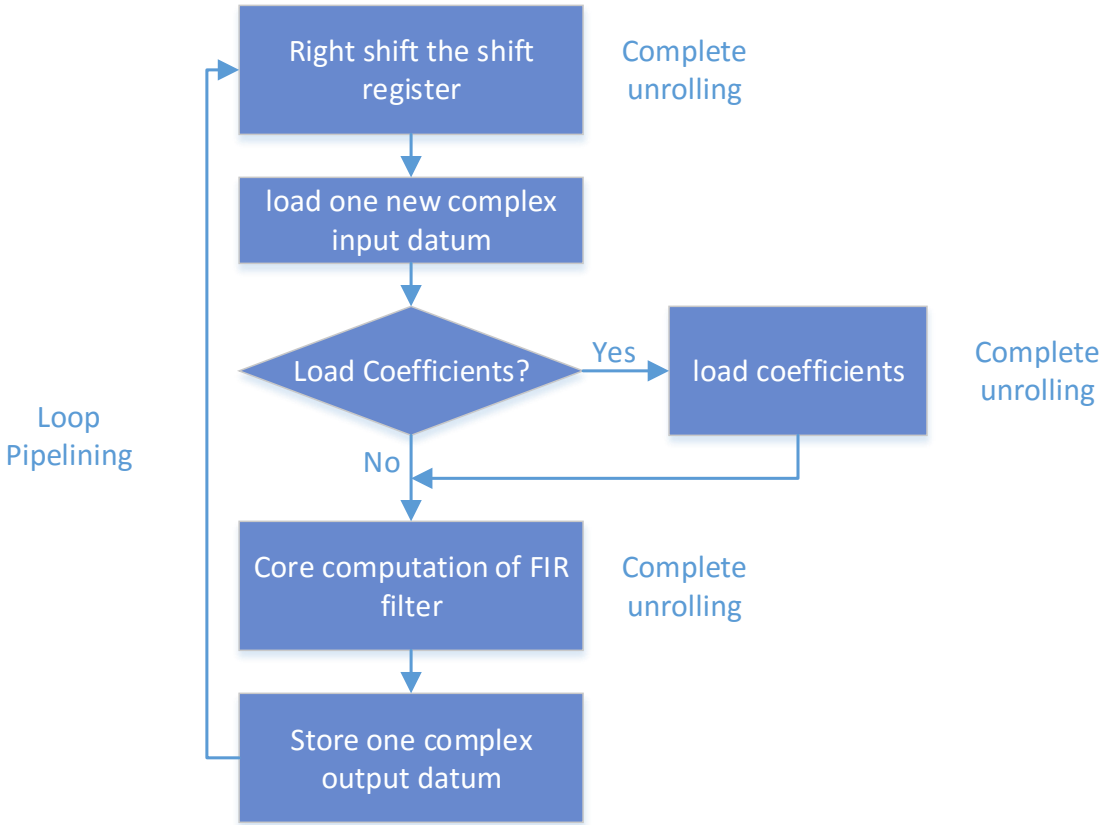
Table: Influence of Four Different Relaxation Methods

Factors	Relaxation methods	Reduced number
Precision	16-bit+16-bit	54.69%
Input size	$\frac{N}{2}$, $\times 8$	50.77%
Filter number and size	$\frac{1}{2}\sum_{i=1}^{M} K_i$	49.23%
Time limit	$2t_{limit}$	49.23%

Outline

General FIR Filter Kernel

- Complete unrolling of inner loops
- Loop pipelining of overall loop



Decomposition of FIR Filter

Large-tap filters cannot be completely unrolled!
Splitting large-tap filter into a group of sub-FIR filters.

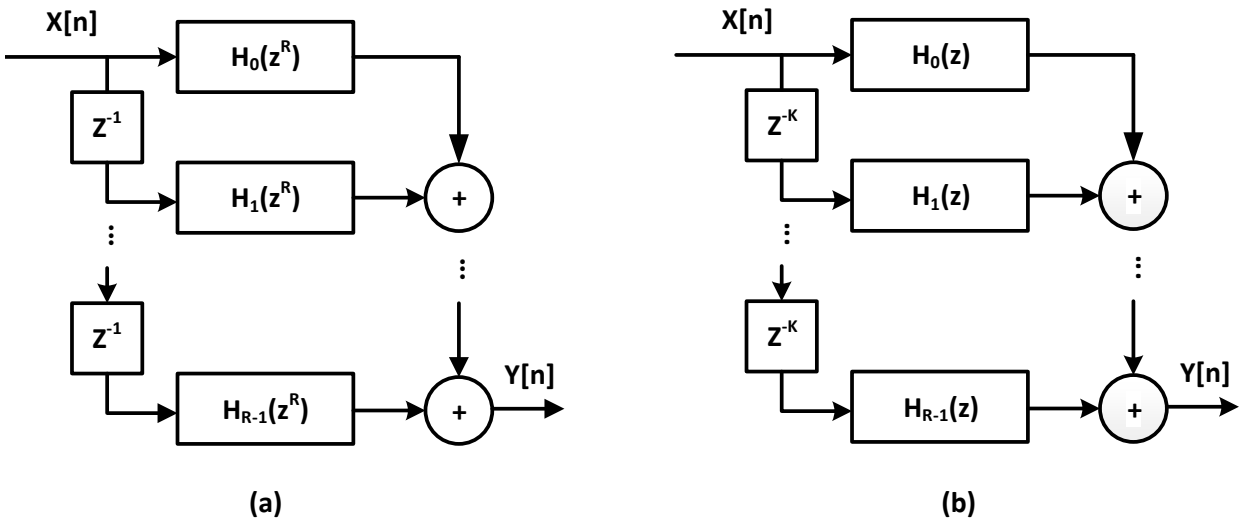
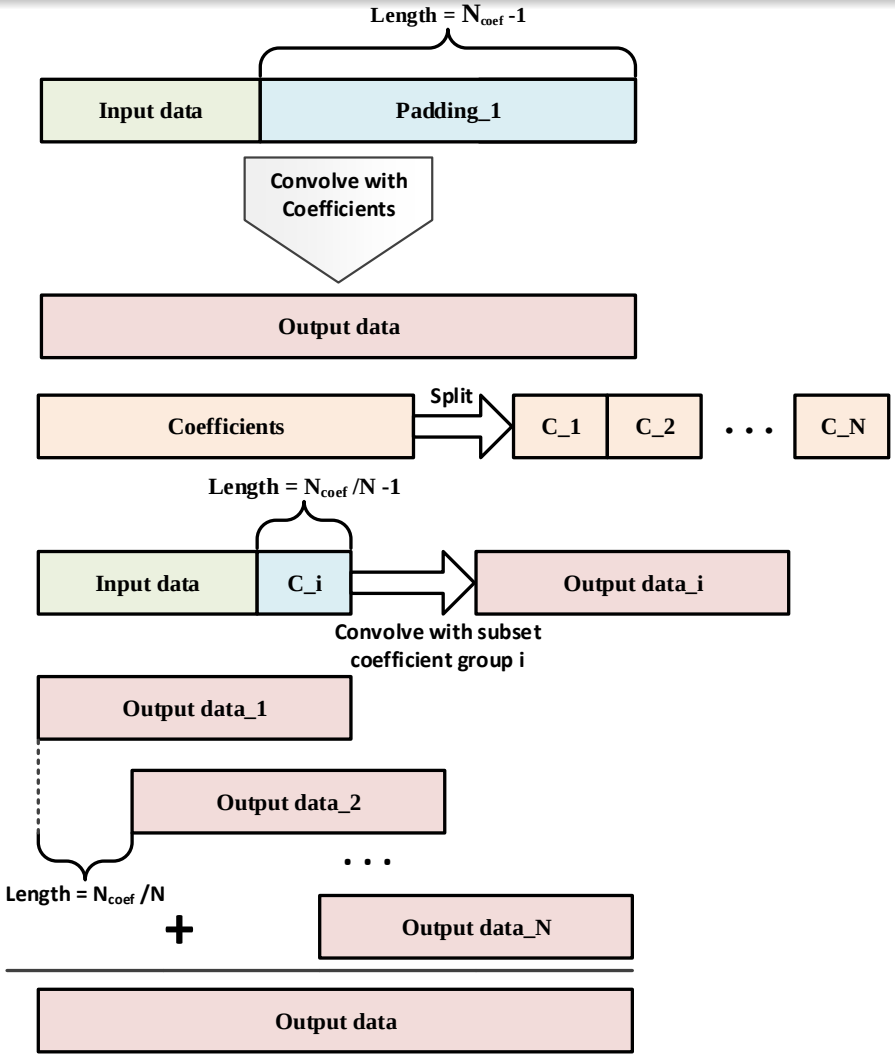


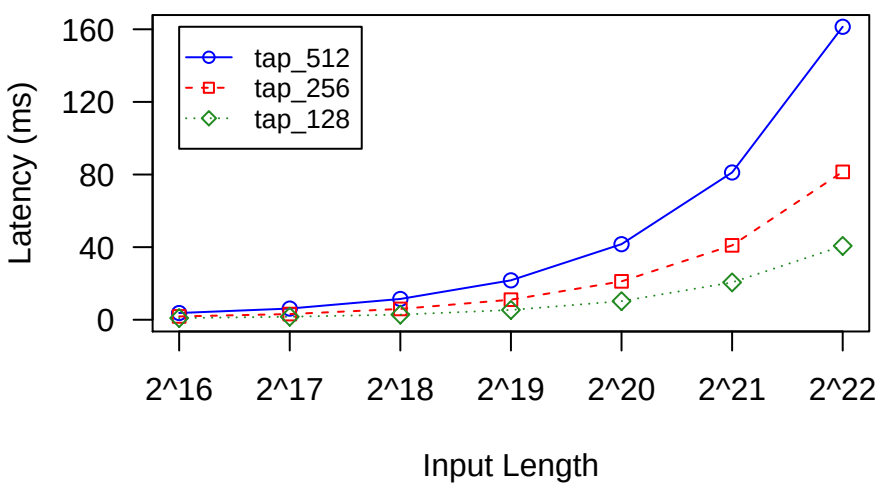
Figure: Structure of Overlap-add method (a)Split-in-frequency (b) Split-in-time

Split-in-time Overlap-add Method



Performance

Decomposed kernel using SIT-OLA method, kernel is tested by executing 128-tap, 256-tap and 512-tap FIR filters.



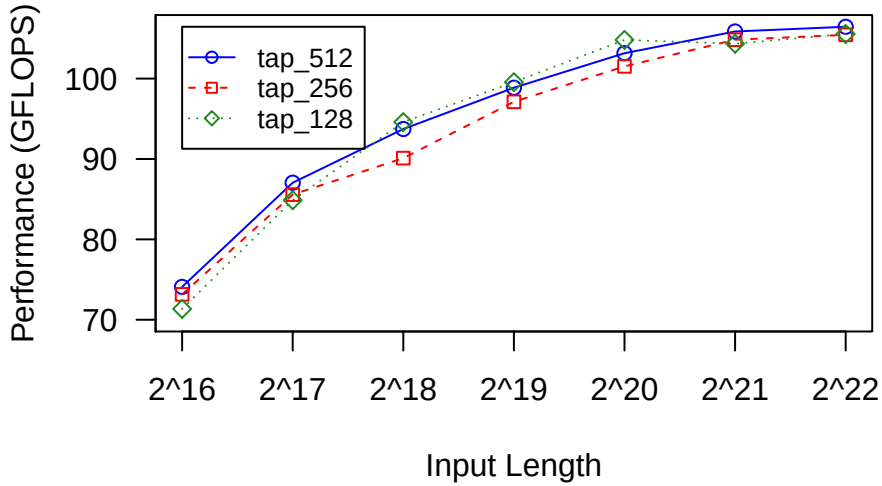


Figure: Peformance and latency of decomposed FIR filter

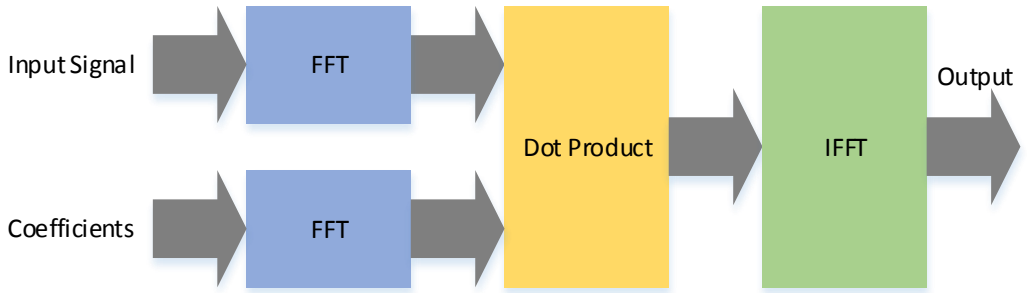
Outline

Flow of Frequency-domain

- Based on the convolution theorem

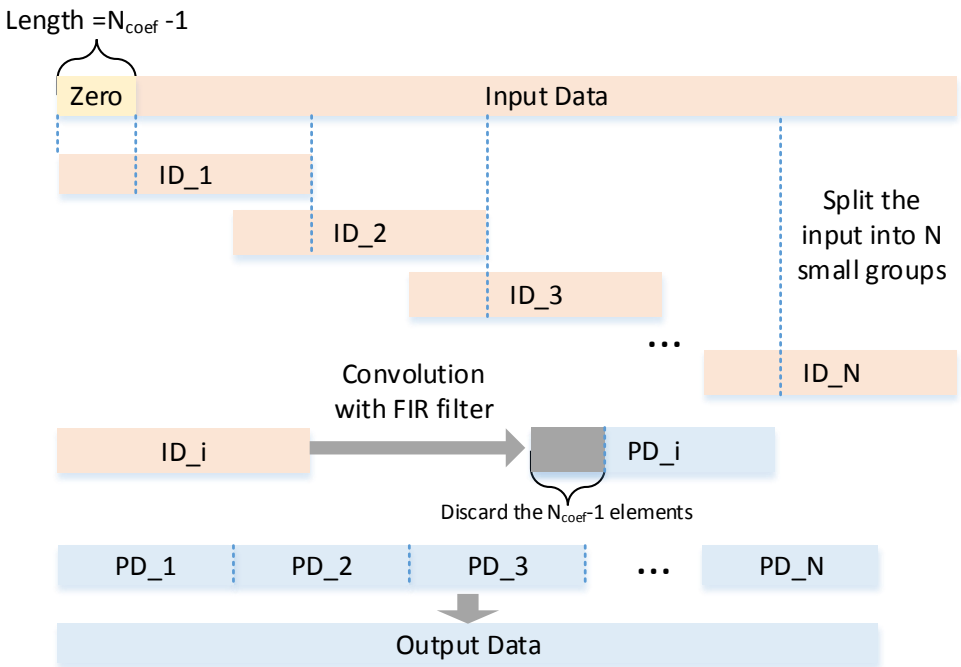
$$\mathscr{F}\{f * h\} = \mathscr{F}\{f\} \cdot \mathscr{F}\{h\}$$

- => convolution by Fourier transform, dot product and inverse Fourier transform
- computation complexity of one filter changed from $O(NK)$ to $O(NlogN)$.



Decomposition of Input

- Input length N too large, needs to be split
- Overlap-save method used
  - good: no adding needed



Two structures of OpenCL kernel

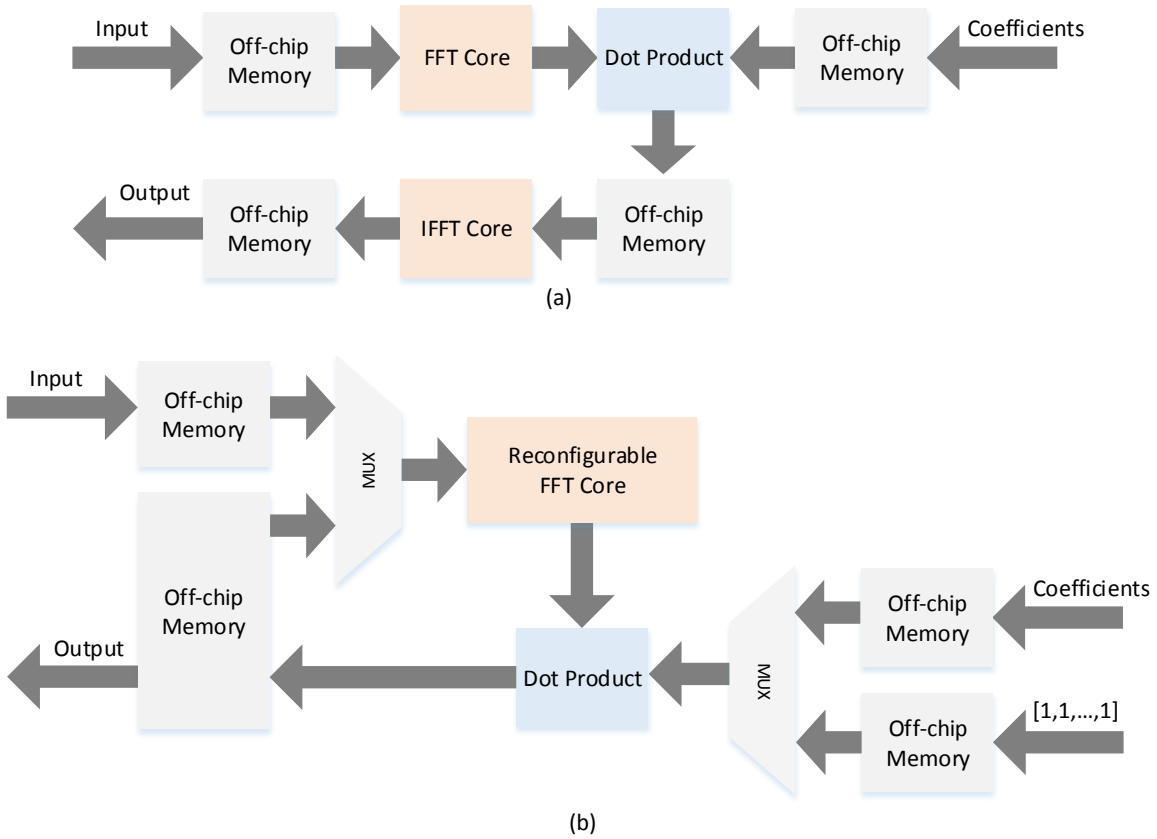
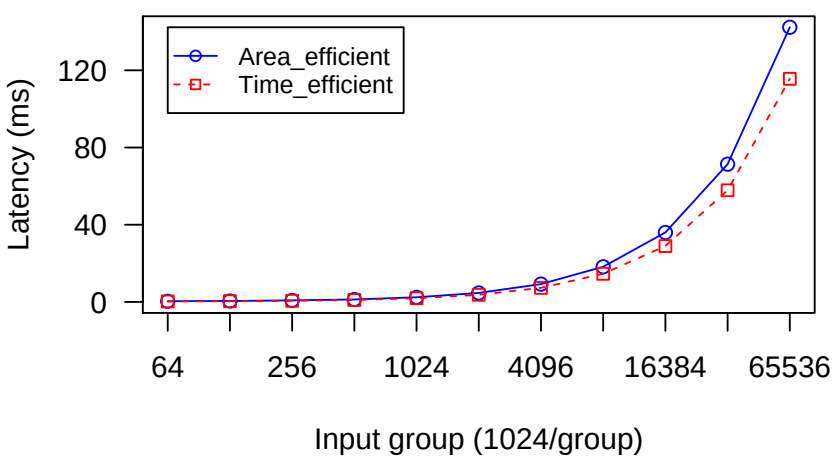


Figure: Structure of Frequency-domain FIR filter (a) time-efficient structure (b) area-efficient structure

Performance

Time-efficient and area-efficient kernels are tested by executing input groups of different length. Length of one group is 1024.



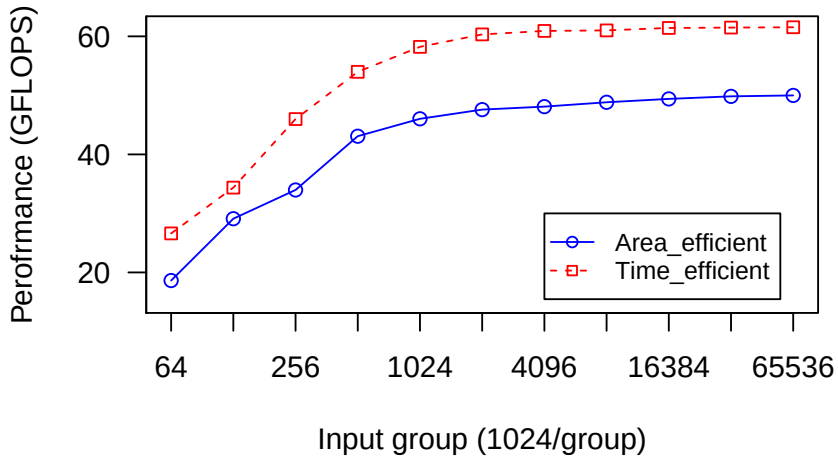


Figure: Performance and latency of frequency FIR filter kernels

Outline

Optimisation of Time-domain Implementation

For FIR filter in transposed structure, one Input[n] multiplies with K coefficients, which can be represented as a multiple constant multiplication (MCM) problem. Using Addition/Subtraction and shift instead of multiplication.

- Common sub-expression elimination (CSE) algorithms
- Adder graph (AG) based algorithms

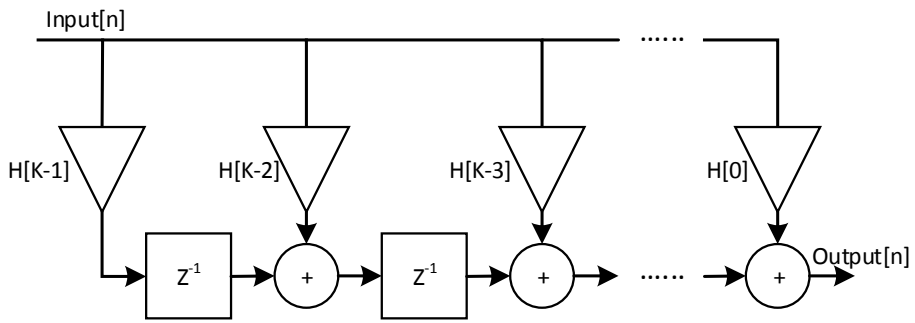


Figure: FIR Filter in Transposed Structure

Optimisation of Frequency-domain Implementation

- Optimisation of FPGA based implementation of FFT
- Decimation in time (DIT)
- Decimation in frequency (DIF)

High-level generation tool

- Developing an automatic compiler/tool that chooses best domain (time or frequency) for filter
- Based on input of coefficients compiler generates optimized code, e.g. using CSE algorithm

Summary

- Complex FIR filter are implemented in both time and frequency-domain using OpenCL
- Both time and frequency-domain FIR filter kernels can execute complex large-tap FIR filters
- Time-domain FIR filter is suitable for relative small-tap complex FIR filters
- Frequency-domain FIR filter has advantage over time-domain FIR filter in large-tap complex FIR filter