Using reconfigurable HW for high dimensional CAF computation

Antonín Heřmánek, Michal Kuneš

Institute of Information Theory and Automation
Academy of Sciences of the Czech Republic,
Prague, Czech Republic
email: {hermaneke, xkunes}@utia.cas.cz

March 27, 2006

Outline

Motivation
- Cross Ambiguity function

System Settings

Algorithms for FFT

System Architecture
- Architecture of the L0 stage
- Architecture of the L1 stage
- Architecture of the L2 stage
- Overall architecture
- Time Scheduling

Implementation and results

- Modern radar system use Passive Coherent Location (PCL)
- Systems with no emitation – only “listen”
- PCL exploit high-power commercial transmitters (FM, TV, etc.)
- evaluates a function related to direct and reflected signals from localized targets
- The function to be evaluate = Cross Ambiguity function (CAF)

Cross Ambiguity function

CAF represents the power spectral density distribution of the cross-correlation between direct and reflected signals:

$$A(\tau, k) = \sum_{n=0}^{N-1} s_1(n) s_2^*(n+\tau) e^{-j2\pi kn/N} \tag{1}$$

- s_1, s_2 discrete-time analytic (complex) signals
- N is the integration period
- τ is time delay
- k Doppler frequency offset

The magnitude of $A(\tau, k)$ peaks when τ and k/N are equal to the TDOA and/or FDOA between signals s_1 and s_2.

Cross Ambiguity function – cont.

The effective CAF computation use direct application of the FFT into signal product of the signals s_1 and s_2.

$$A(\tau, k) = \mathrm{FFT}(s_1(n)s_2^*(n+\tau)) \tag{2}$$

To calculate CAF using this formula for all values of τ and k, an individual FFT computation is required for each value of τ.

System Settings

The basic requirements on CAF calculation engine for PCL system signal processing are as follows:

- Sampling frequency: 100-200 kHz
- Effective bit resolution of input signals: 18-24 bits
- Integration interval: $N = 2^{17} = 131\,072$ samples
- Frequency resolution: < 1 Hz
- Accuracy of the CAF calculation: absolute error about 10^{-10} compared to the IEEE 64-bit floating-point arithmetic
- Maximum number of time delays: < 600
- Maximum frequency range: $\langle -300 : +300 \rangle$Hz (601 spectral coefficients)
- Total time of computation: < 1 sec

Algorithms for FFT

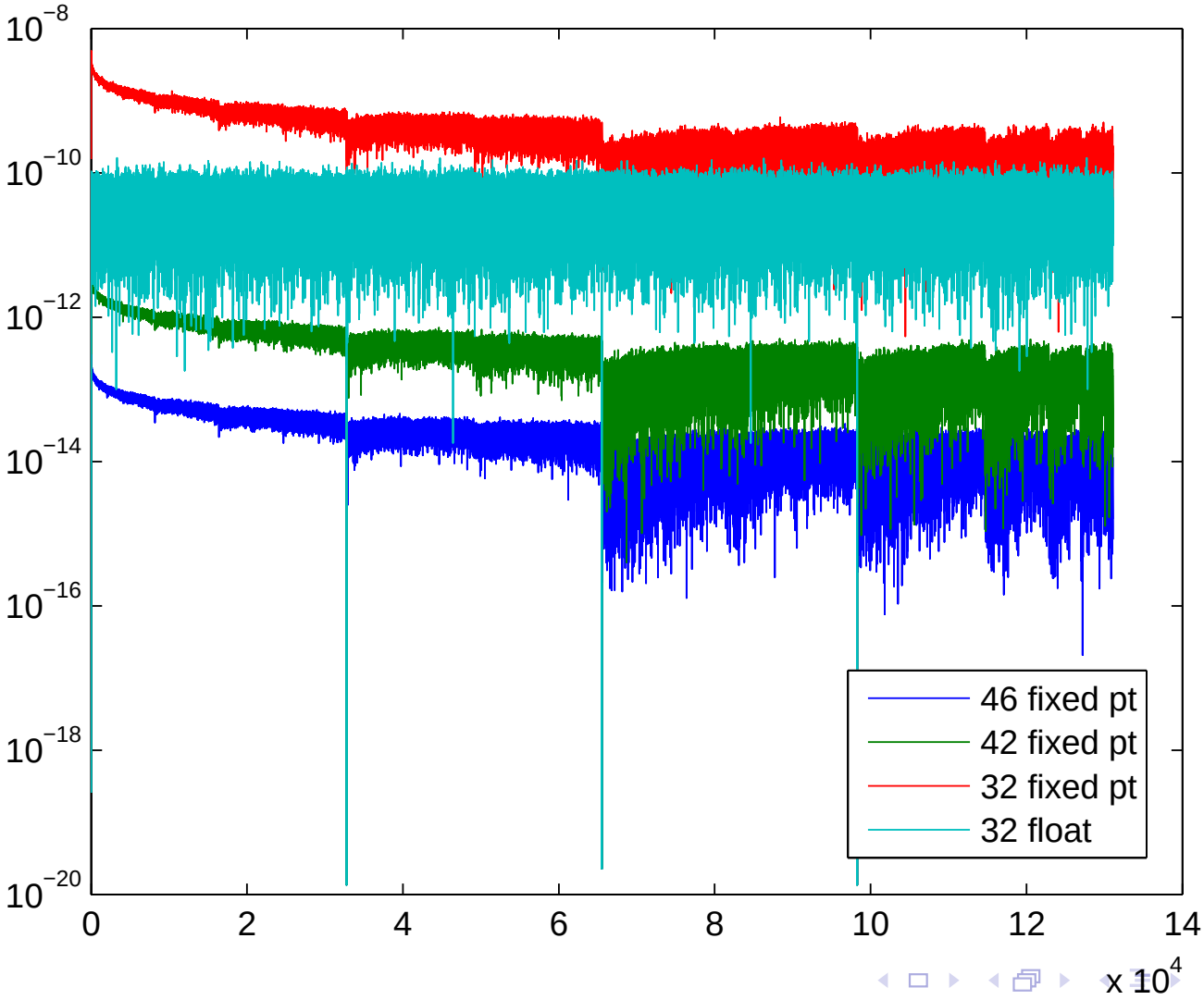
Known algorithms

- FFT (radix -2, radix-4 , split radix)
- Zoom-FFT
- Pre-weighted FFT
- Goertzel algorithm

Suitable algorithms for PCL

- FFT (any radix) only!
- other methods inserts a systematic error and/or are more complex for a given interval of freqency bins

Numerical reqirements



Numerical reqirements – cont.

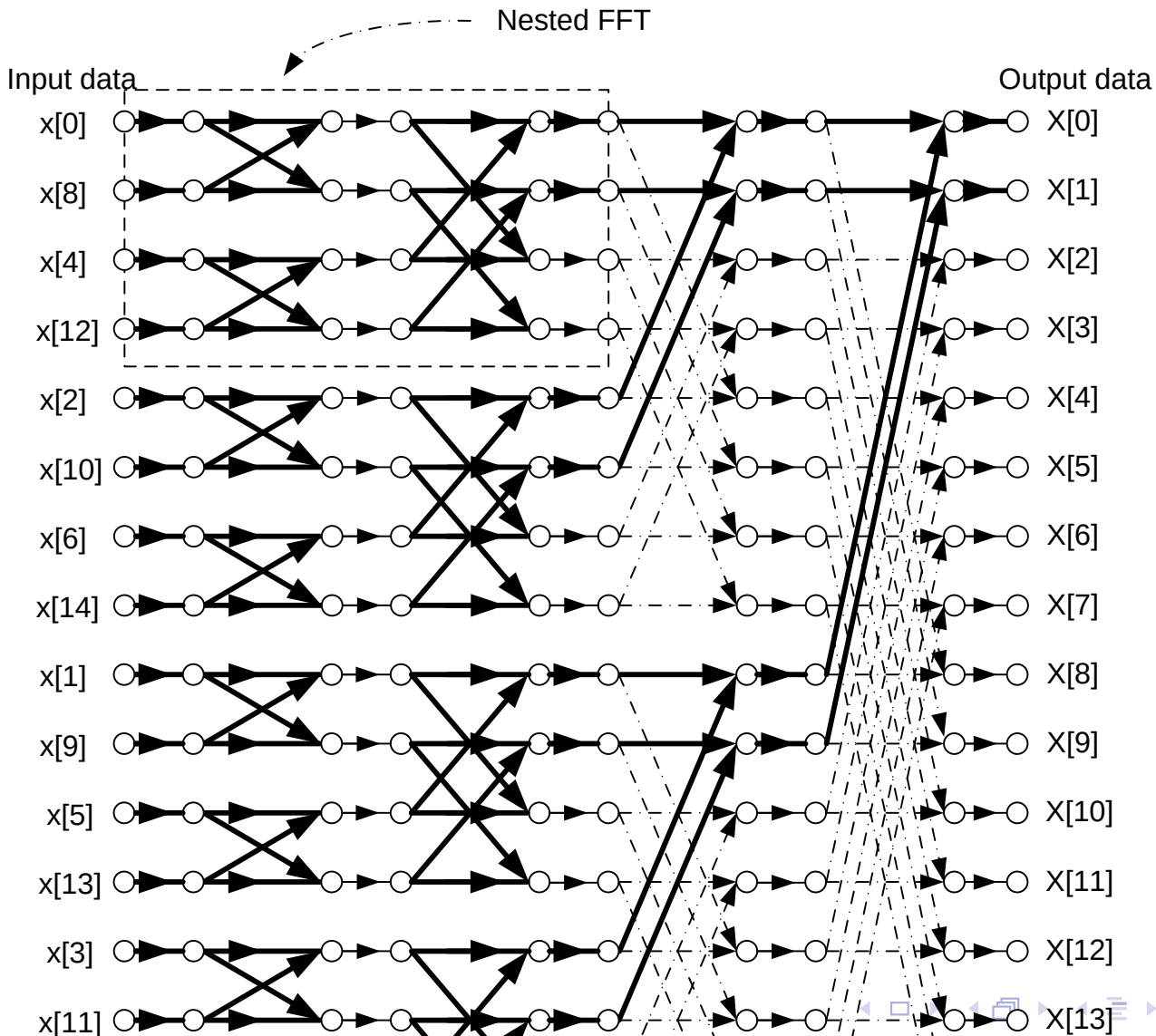
Conclusion

- we need at least 42 bits in fixed point
- 32-bit IEEE floating point is on the boundary

Available FFT on FPGA

- non of the commercial nor open source such long and such accurete FFT IP core
- Xilinx: FFT length $2^{16} = 65\,536$ at 24 bits (fixed pt. or block floating pt.)
- Altera: $2^{14}c = 16384$ at 24 bits fixed point

FFT with reduced number of operation



System Architecture

Structure of long FFT calc

Stage L0: signal multiplication $s_1(n)s_2^*(n+\tau)$

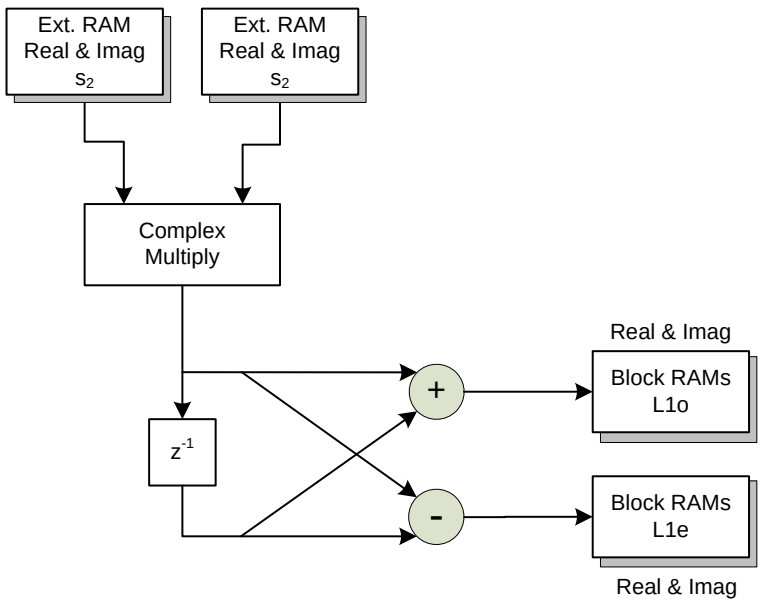
Stage L1: parallel calculation of nested FFTs of size N_f

Stage L2: calculation of the rest of butterflies

Timing

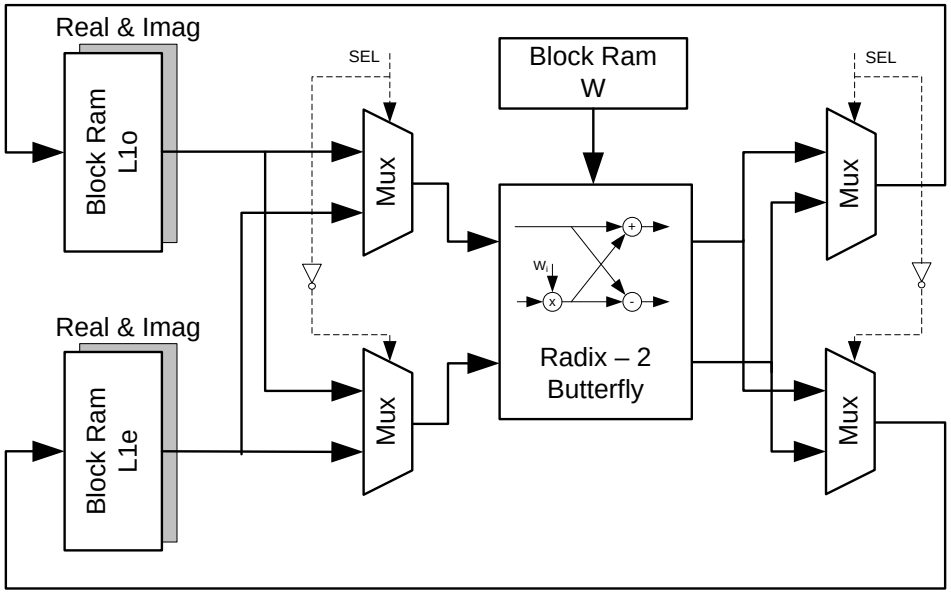
- stage L1 represents $N/N_f = 128$ nested FFTs – using mono processor approach, takes $589,824$ cycles.
- stage L2, takes approximately $76,928$ cycles.
- it follows that the most computationally intensive part is the computation of stage L1.

Block structure of L0 stage



- All data paths and operation are complexnumbers/paths.
- to shorten the calculation L1 stage, the first radix-2 butterfly (addition/subtraction only) is calculated in stage L0

Block structure of the L1 stage



- Since the calculation of the nested FFTs is the most computationally intensive operation, four L1 cores are used in parallel
- Using two dual ported memories, two input and two output values are read/stored at the same time – the Radix-2 core is fully utilized.

Architecture of the L2 stage

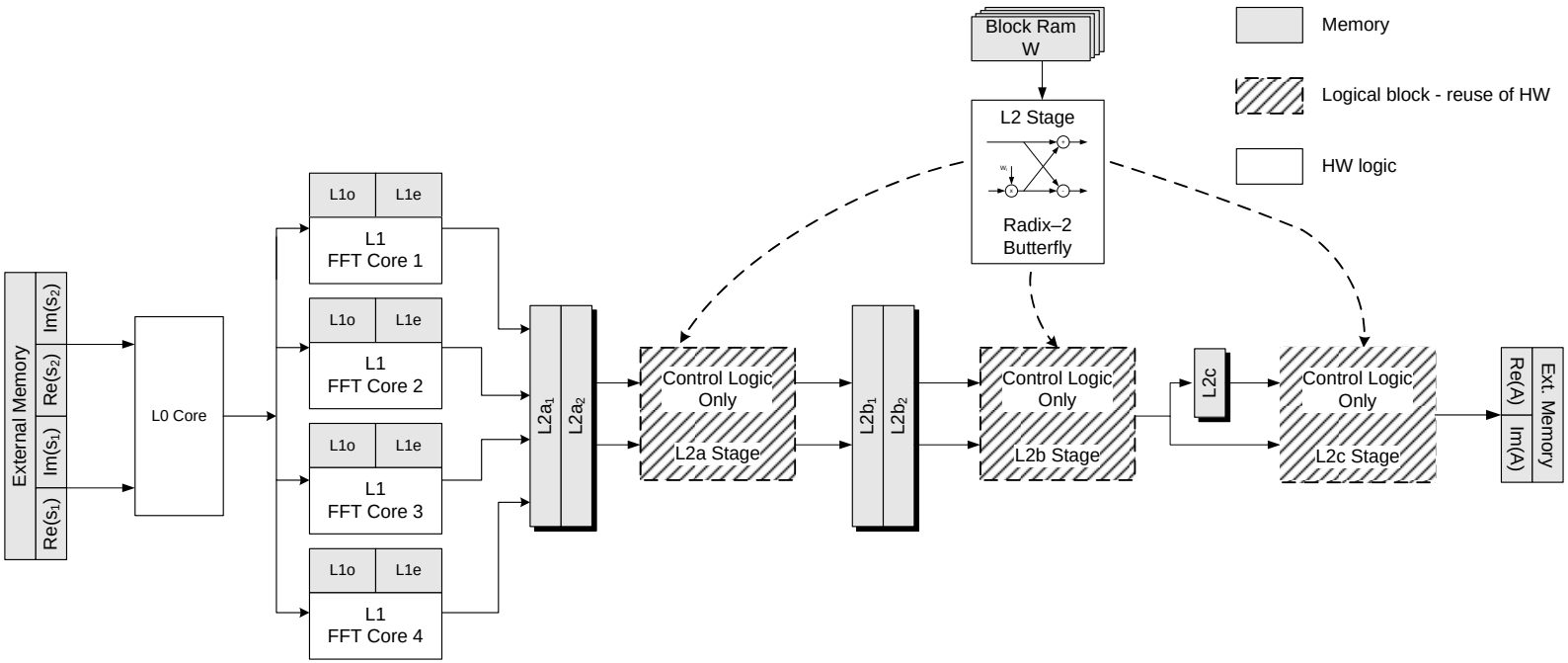
- L2 architecture is the same as L1
- it uses a set of tables W of size Δ each.
- To optimize the size of intermediate memory and time schedule, the L2 block is divided to tree logic parts:

  L2a, L2b implements three stages of FFT i.e. from $2^3\Delta$ inputs and calculates Δ intermediate values

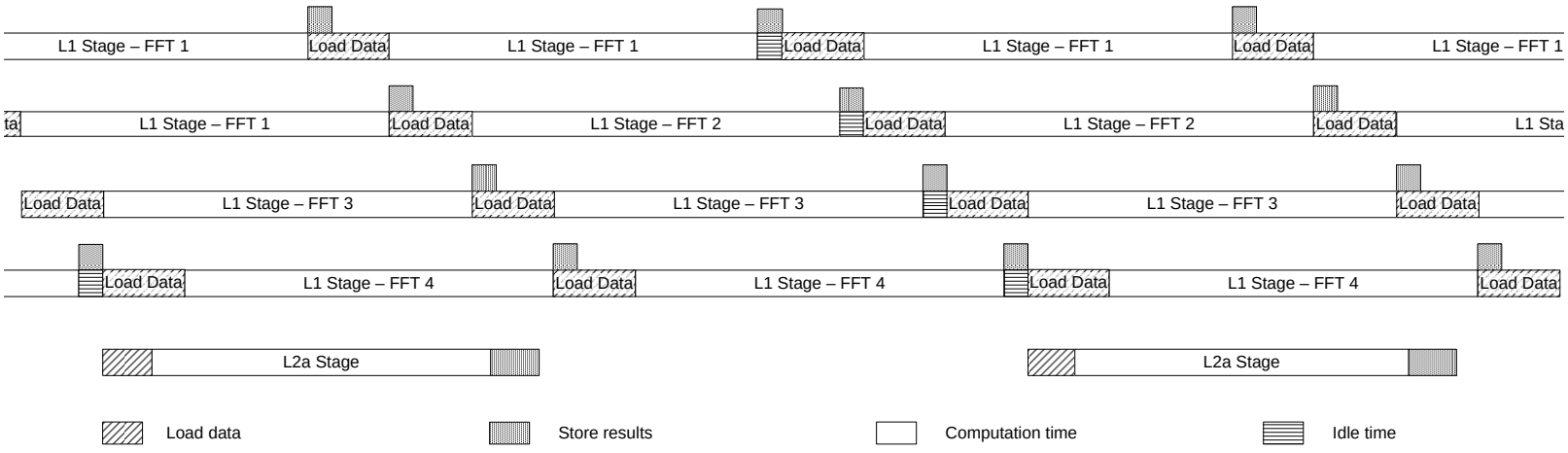
  L2c L2c implements the final one stage i.e. from 2Δ values calculates final Δ results. It use one table W

- all three parts L2a, L2b and L2c use the same hardware core – only one part can run at a time

Block architecuture of the system



Time Scheduling



Time Scheduling – cont.

Stage	Clock Cycles	Comp. 33MHz	Comp. 66MHz
data load L1	1034	$31.3\mu s$	$15,7\mu s$
data store L1	304	$9.21\mu s$	$4.61\mu s$
data load L2a,b	624	$18.9\mu s$	$9.4\mu s$
data store L2a,b	606	$18.3\mu s$	$9.1\mu s$
data load L2c	609	$18.4\mu s$	$9.2\mu s$
data store L2c	609	$18.4\mu s$	$9.2\mu s$
computation L1	4805	$146\mu s$	$72.8\mu s$
computation L2a, L2b	4263	$129\mu s$	$64.6\mu s$
computation L2c	608	$18.4\mu s$	$9.21\mu s$
computation 1xFFT	$219 \cdot 10^3$	$6,64ms$	$3,32ms$
computation CAF	$131 \cdot 10^6$	$3,98s$	$1,99s$

Implementation and results

Utilization of XC4VSX55		
DSP48s	216	42%
RAMB16s	170	53 %
LUTs	25394	51 %
Slices	16482	67 %
Flip Flops	19266	39 %

Utilization of EP2S180		
DSP9s	624	81 %
M512s	846	90 %
M4Ks	472	61 %
M-RAMs	4	44 %
ALUTs	48924	34 %
ALMs	27180	37 %
Registers	25729	17 %

END