GPU TECHNOLOGY CONFERENCE

Implementing Radar Algorithms on CUDA Hardware

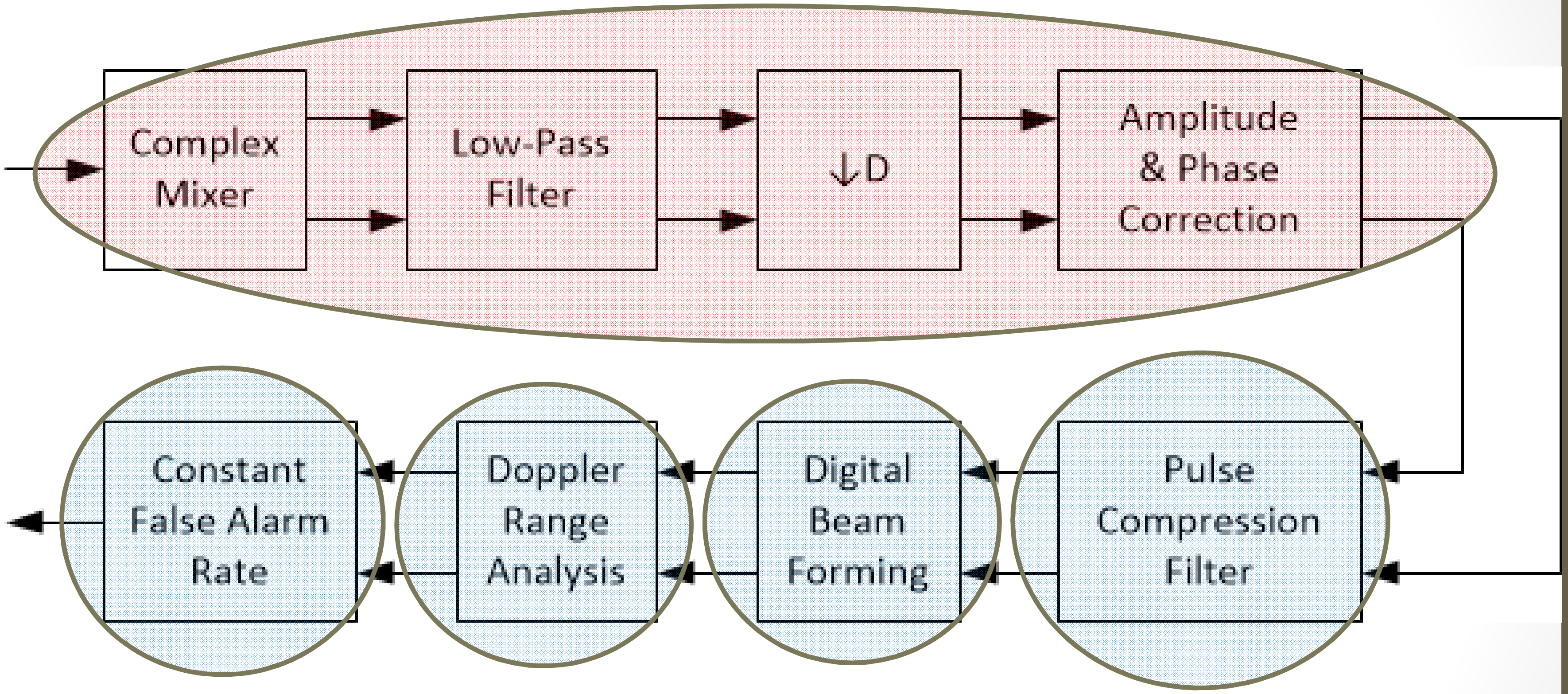
Pietro Monsurró, Alessandro Trifiletti, Francesco Lannutti

Dipartimento di Ingegneria dell'Informazione, Elettronica e delle Telecomunicazioni (DIET)

Università di Roma "Sapienza", Roma, Italia

SAPIENZA
UNIVERSITÀ DI ROMA

Radar signal processing



Beam-forming Doppler-Range Search Radar

SAPIENZA
UNIVERSITÀ DI ROMA

GPU TECHNOLOGY CONFERENCE

Advantages of GPUs

- Massive computing power
  - TeraFLOPS
  - Moore’s Law
- Large device memory bandwidth
  - Hundreds of GB/s
- C programming
  - Fast compilation
  - Fast prototyping

Disadvantages of GPUs

- Latency
- Require parallel algorithms
  - Identical operations for each block
  - Coalesced memory access
- PCIe bandwidth
  - PCIe bus is a bottleneck (8GB/s)

SAPIENZA
UNIVERSITÀ DI ROMA

GPU TECHNOLOGY CONFERENCE

Kernel #1 – Mixing & APC (1/3)

- IF input
  - 16 short int 80MS/s vectors of 256K elements
- BF output:
  - 16 complex float at 10MS/s (32K elements)
- Performs casting to float, mixing, low-pass filtering and down-sampling, amplitude & phase correction in <u>one</u> kernel

Kernel #1 – Mixing & APC (2/3)

$$\begin{cases} y_{4i} = x_{4i} \\ y_{4i+1} = jx_{4i+1} \\ y_{4i+2} = -x_{4i+2} \\ y_{4i+3} = -jx_{4i+3} \end{cases}$$

$$z_{8n} = \sum_{i=0}^{N-1} y_{8n-i} h_i$$

$$\left(h_i^k \equiv h_{4i+k}\right)$$

$$z_{8n} = \sum_{i=0}^{\frac{N}{4}-1} y_{8n-4i} h_i^0 + \sum_{i=0}^{\frac{N}{4}-1} y_{8n-4i-1} h_i^1$$
$$+ \sum_{i=0}^{\frac{N}{4}-1} y_{8n-4i-2} h_i^2 + \sum_{i=0}^{\frac{N}{4}-1} y_{8n-4i-3} h_i^3 =$$

$$= \left(\sum_{i=0}^{\frac{N}{4}-1} x_{8n-4i} h_i^0 - \sum_{i=0}^{\frac{N}{4}-1} x_{8n-4i-2} h_i^2 \right)$$
$$+j \left(\sum_{i=0}^{\frac{N}{4}-1} x_{8n-4i-1} h_i^1 - \sum_{i=0}^{\frac{N}{4}-1} x_{8n-4i-3} h_i^3 \right)$$

SAPIENZA
UNIVERSITÀ DI ROMA

GPU TECHNOLOGY CONFERENCE

Kernel #1 – Mixing & APC (3/3)

- 8:1 multirate filter
  - Polyphase architecture
- Input IF is $f_S/4$
  - Mixing by multiplying by $\{1,0,-1\}$ embedded in the polyphase filter
- Iterations on nearby inputs
  - Shared memory for inputs
- Filter and correction terms are constant
  - Constant memory for coefficients

Kernel #2 – Compression (1/4)

- Performs complex matched filtering with the complex coefficients of the BF transmitted pulse
  - The pulse is 100 samples long
    - 10MHz chirp waveform
  - Length increases process gain and processing time
- Filter coefficients in Constant memory
- Shared memory used to store inputs

Kernel #2 – Compression (2/4)

- Simplified benchmark for TD-FIR optimization
  - 2M x 10 points are filter through a 19-tap filter
  - All processing is real
- Results:

Simple implementation:	16.5ms
Using local TMP variable:	15.2ms
Using Constant memory:	7.1ms
Using Shared memory:	2.4ms

- L1/Shared Memory bandwidth is limiting factor

Kernel #2 – Compression (3/4)

- Time-domain processing faster than frequency-domain processing
  - Frequency-domain methods should be faster for long filters

- TD: $(N - L)L$ complex products (90K)
- FD: $2N \log_2 N + N$ complex products (21K)

SAPIENZA
UNIVERSITÀ DI ROMA

GPU TECHNOLOGY CONFERENCE

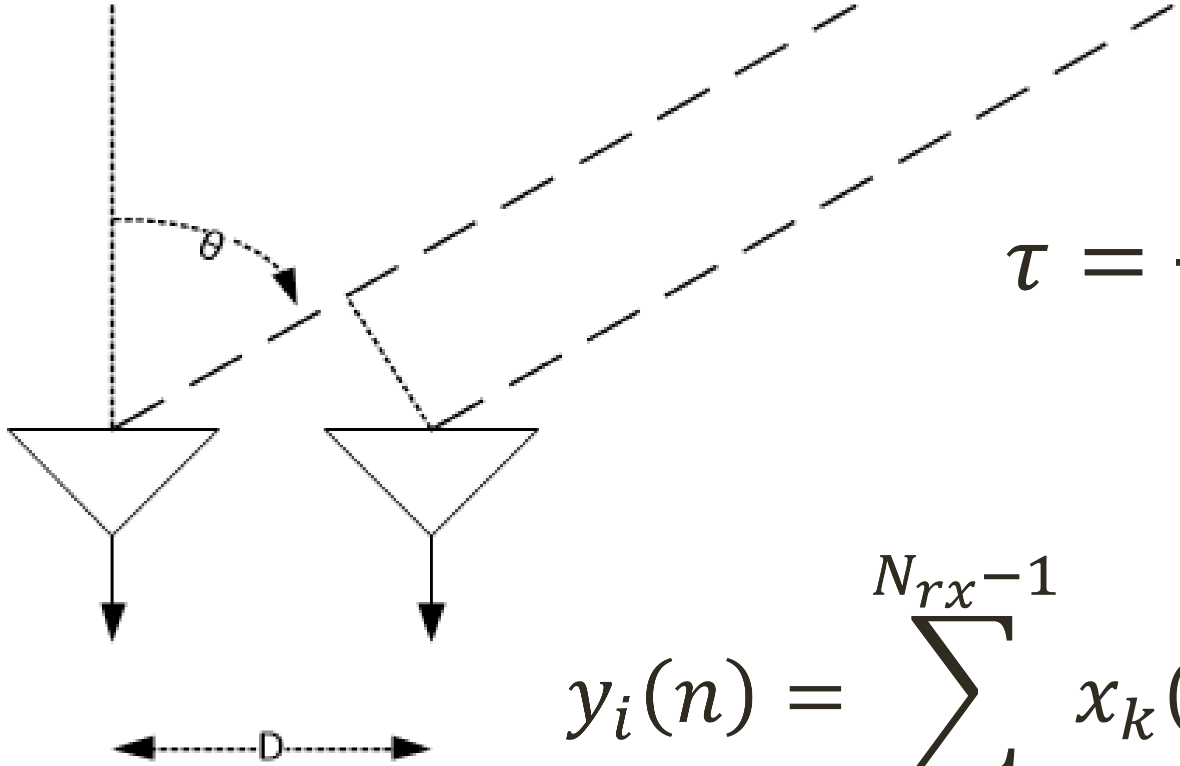
Kernel #2 – Compression (4/4)

- What's wrong with the FD approach?
  - 16 channels, 2048 points, 100 taps
  - Total time: 44us vs 53us
- Memory transfers take 1.5us @190GB/s
  - 2 vs 6 transfers
- Higher utilization
  - Compute: 55% vs 15% (25%)
  - Memory: 85% vs 25% (35%)

SAPIENZA
UNIVERSITÀ DI ROMA

GPU TECHNOLOGY CONFERENCE

Kernel #3 – Beam Forming (1/2)



$$\tau = \frac{2D}{c} \sin\theta$$

$$y_i(n) = \sum_{k=0}^{N_{rx}-1} x_k(n) e^{-j\omega_{rf}\tau_i k}$$

SAPIENZA UNIVERSITÀ DI ROMA

GPU TECHNOLOGY CONFERENCE

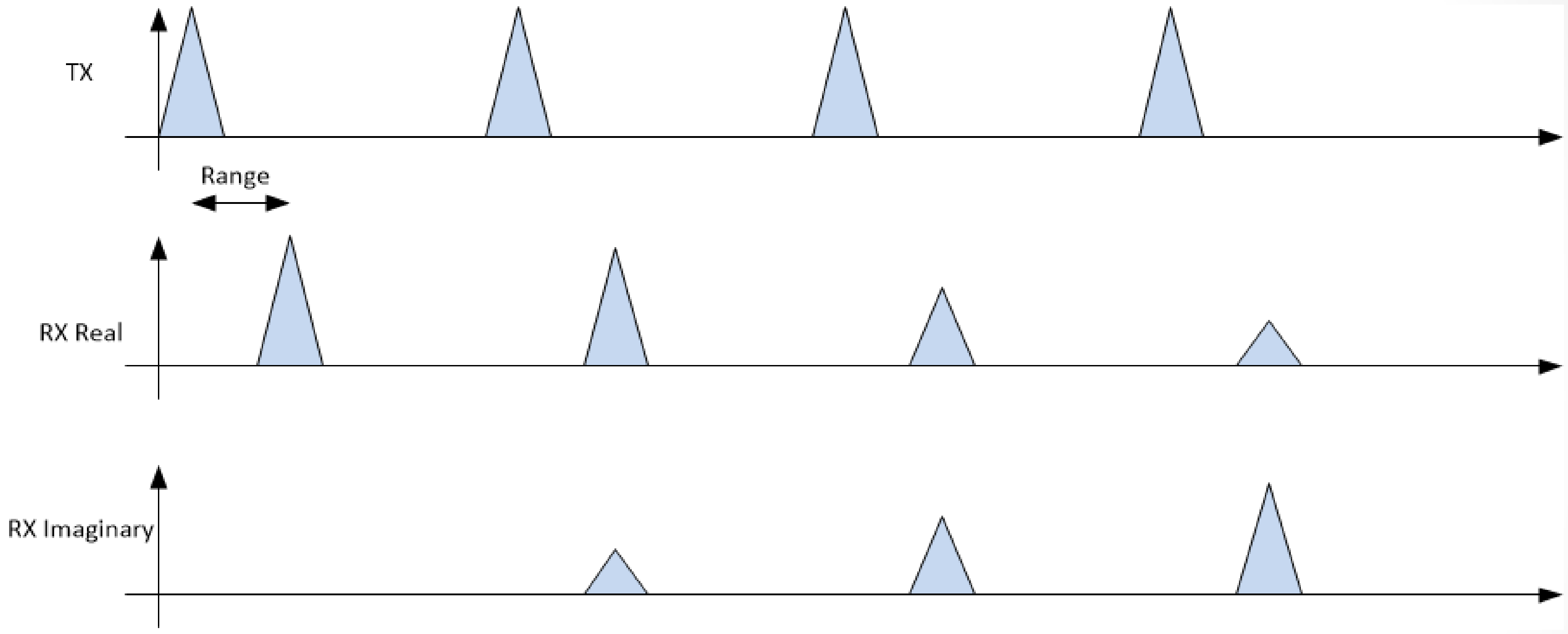
Kernel #3 – Beam Forming (2/2)

- Beam-forming rotates, scales and sums the receivers' outputs to form a directional beam
- There are 16 antennas and 16 beams which are processed in parallel
- The rotation matrix is stored in the Constant memory
  - No $\sin(\cdot)$ and $\cos(\cdot)$ are computed

SAPIENZA UNIVERSITÀ DI ROMA

GPU TECHNOLOGY CONFERENCE

Kernel #4 – Doppler/Range (1/3)



$$\Delta t = \frac{2R}{c} \qquad \Delta f = f_{RF}\frac{2V}{c}$$

Kernel #4 – Doppler/Range (2/3)

- Each target has its Doppler shift and delay, which are related to its speed and distance
- The Doppler/Range analysis is a series of time-shifted FFTs
- Each FFT has 32 points
- There are 1,000 FFTs for each of the 16 beams.
- The NVIDIA cuFFT library is used

Kernel #4 – Doppler/Range (3/3)

- The input of each 32-points FFT in one beam has a stride of 1,000 over a vector of 32,000 points
  - cufftPlanMany() is used
  - The Plan is iteratively launched 16 times, performing 1,000 32-points FFT each
- From CUDA 5.0 to CUDA 6.5, this kernel has slowed down from 150µs to 170µs

Kernel #5 – CFAR processing

- CFAR processing estimates the noise around the target by range and/or Doppler averaging of nearby cells
  - It is used to distinguish a true target from its surrounding noise
  - It has not been implemented efficiently

Asynchronous operations

- Memory operations on the PCIe bus can be performed in parallel with GPU processing
- Asynchronous streams
  - Require synchronization barriers
  - Can enhance throughput up to 100%
  - Cannot enhance latency

Constant and Shared memories

- Accesses to main memory has large bandwidth (200GB/s) but large latency (500 clock cycles)
- Constant coefficients can be stored in the fast Constant memory to reduce accesses to memory
- Data which is used often in one kernel can be stored in the Shared memory

SAPIENZA
UNIVERSITÀ DI ROMA

GPU TECHNOLOGY CONFERENCE

Algorithmic optimization

- Most functions can be performed with different algorithms:
  - Time-domain or frequency-domain
  - Poly-phase
  - More functions per kernel
- BLOCK / THREAD organization
  - “Empirical” optimization

Parametric coding

- The code is heavily parametric
  - Number of channels, beams, pulses and bins
  - Pulses' length and shape
  - IF-to-BF down-sampling
  - CFAR: number of range/bin averages
- Some parameters may be updated between frames by writing the Constant memory
  - To Be Done
  - KBs

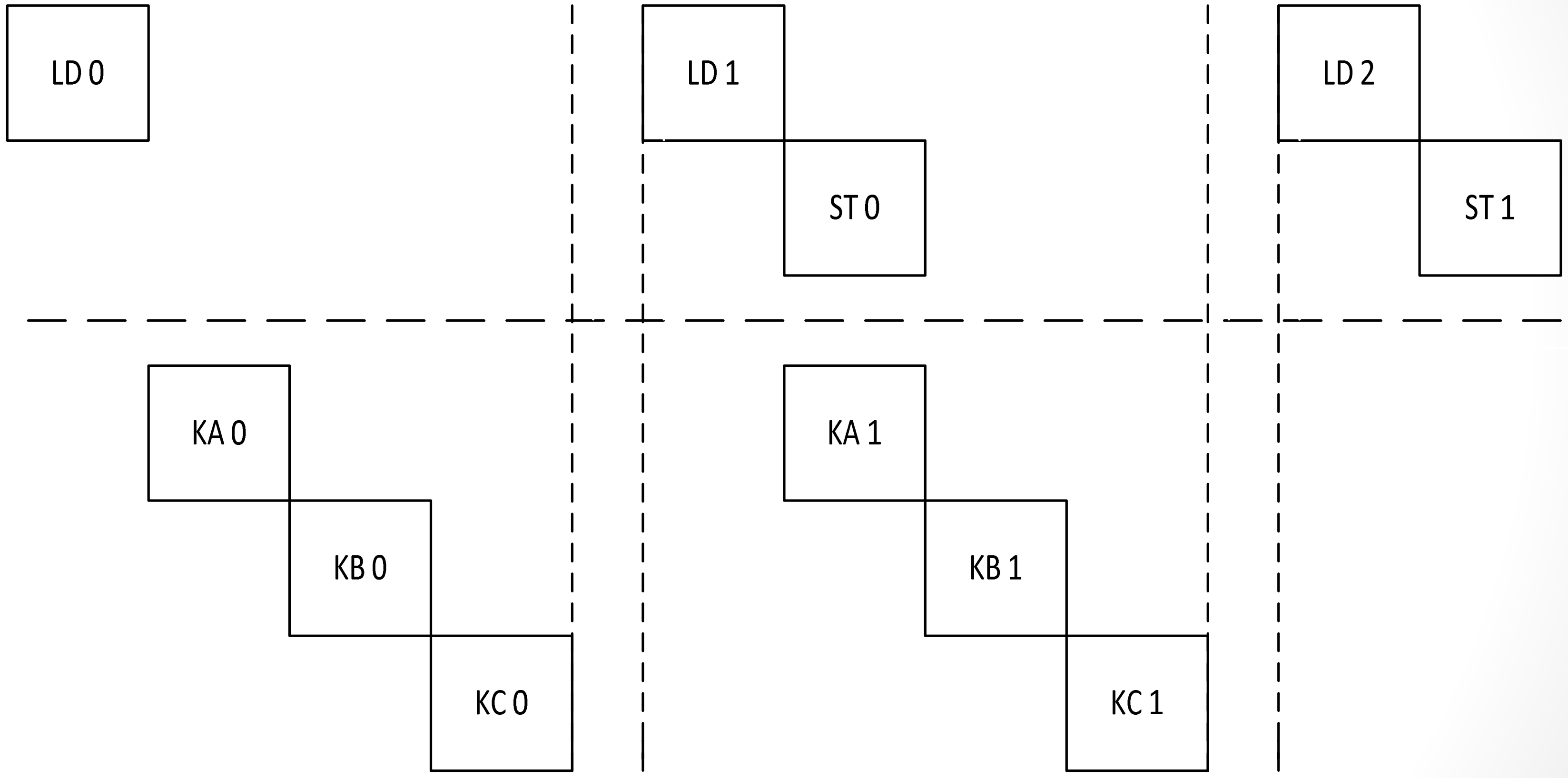
Performance (on GTX680)

Operation	*Time (µs) (5.0)*	*Time (µs) (6.5)*
DDC	220	200
PCF	620	430
DBF	300	280
FFT	150	170
CFAR	460	460
TOTAL (proc)	**1,770**	**1,520**
LOAD	1,300	1,300
STORE	330	330
TOTAL (mem)	**1,630**	**1,630**

SAPIENZA
UNIVERSITÀ DI ROMA

GPU TECHNOLOGY CONFERENCE

External synchronization



Conclusion (1/2)

- Most of the algorithms in a beam-forming pulse/Doppler radar can be parallelized
  - The PCIe bus is the bottleneck
- A tracking radar may be less efficiently implemented
  - (Low dimensional) adaptive filters may be harder to parallelize

Conclusion (2/2)

- The GPU need to work on a PC and together with a Data Acquisition Board (DAQ)
  - 16 x 80MSps ADCs, short int data: 2.56GB/s
- Ruggedization is required in a real system
  - Thermal & mechanical shocks
  - Electrical & Electromagnetic shocks
  - Environmental shocks
  - Ruggedized GPU SBCs are commercially available