telecommunications & signal processing laboratory

McGill UNIVERSITY

GPU Acceleration for Fixed Complexity Sphere Decoder in Large MIMO Uplink Systems

Presented by Tianpei Chen Tianpei.chen@mail.mcgill.ca

Supervisor : Harry Leib Harry.Leib@mcgill.ca

Outline

Introduction

Introduce large MIMO uplink system, General Purpose Graphic Processing Unit Computing (GPGPU) , Compute Unified Device Architecture (CUDA) and microarchitecture of GPU.

Channel Model and Fixed Complexity Sphere Decoder (FCSD)

Provides background knowledge of discrete time MIMO channel model and details of Fixed Complexity Sphere Decoder.

GPU Based Acceleration of FCSD

Present implementation details of CUDA-FCSD

Performance and Analysis

Present computer simulation results, compare GPU and CPU implementation under different circumstances, analyze the factors that impact Acceleration performance

Conclusions

Give conclusions and discussions

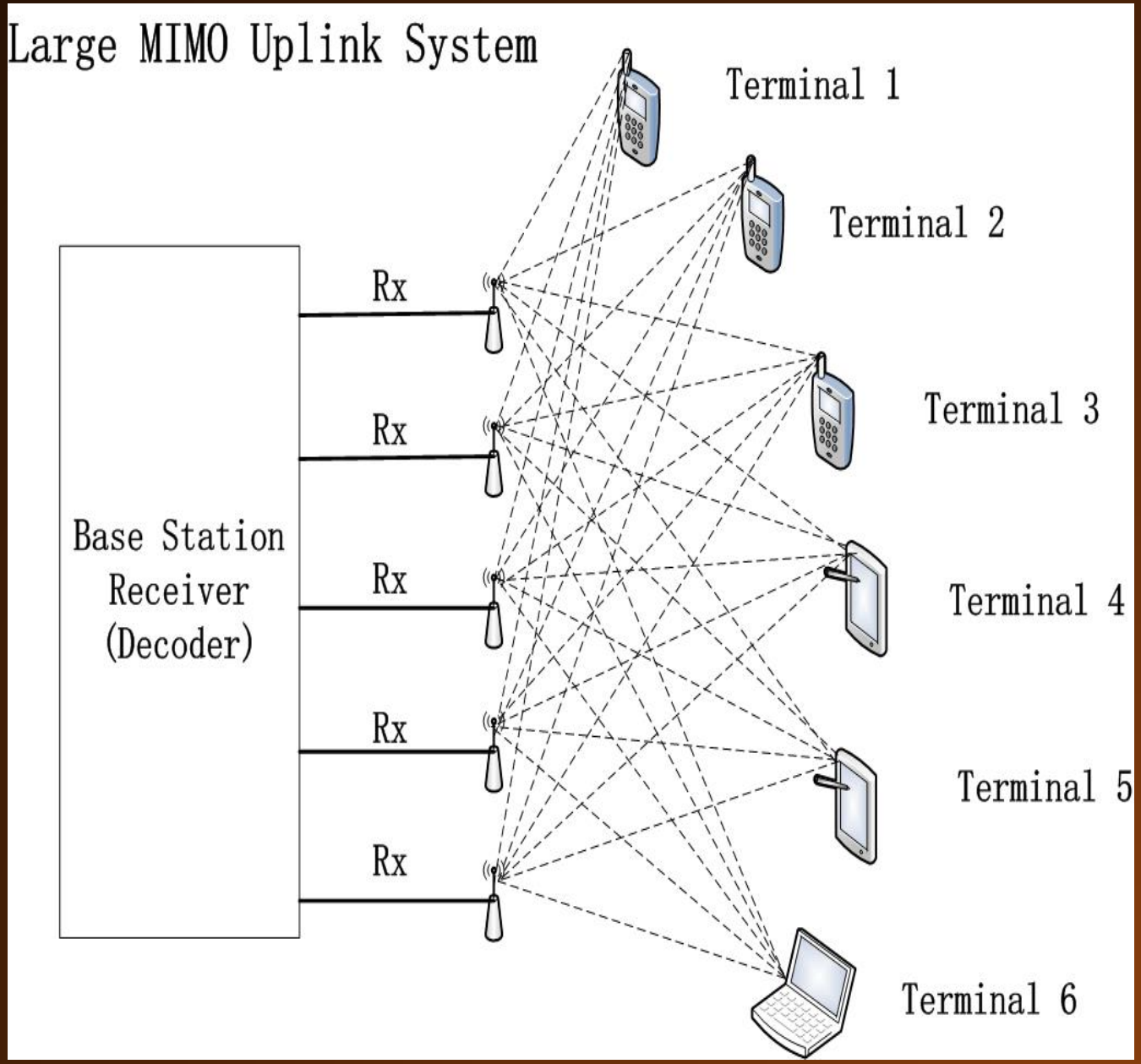
Introduction

- Large MIMO system
- General Purpose Graphics Processing Units Computing (GPGPU)
- Compute Unified Device Architecture (CUDA)
- Microarchitecture of GPU

Introduction

Large MIMO system

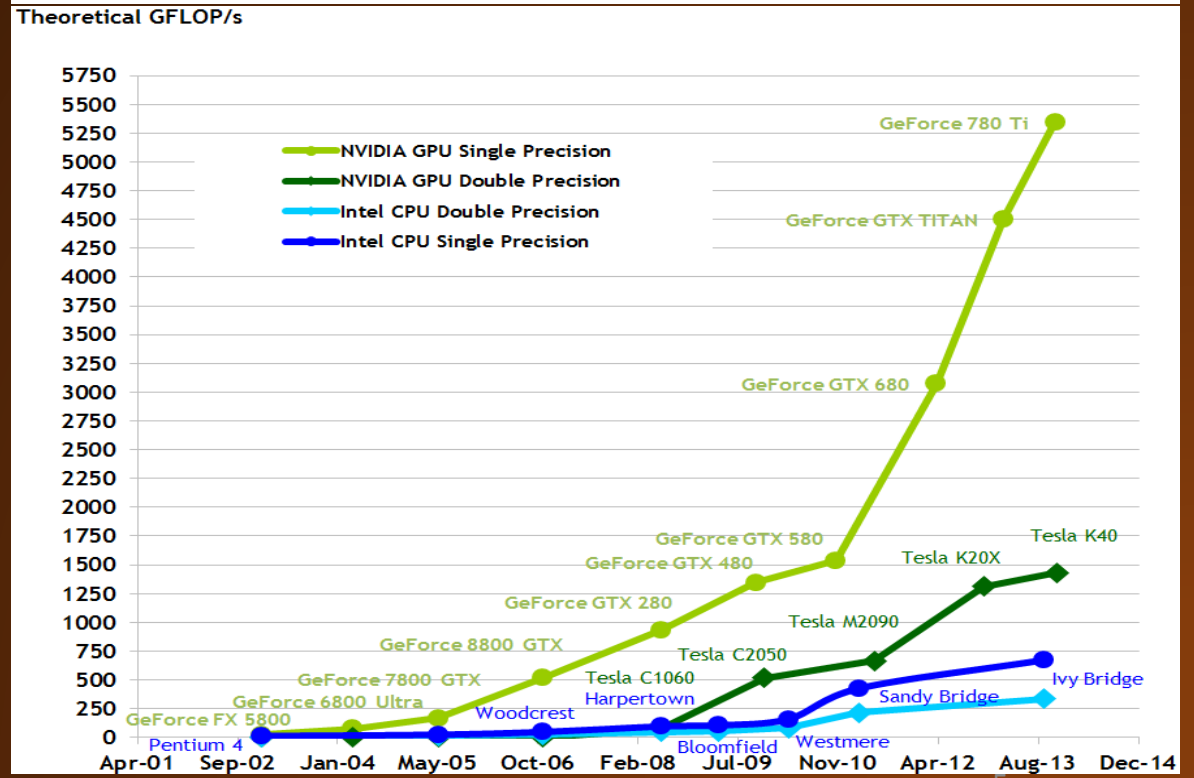
- Multiple Input Multiple Output (MIMO) technology has attracted immense research interests since it can improve spectrum efficiency as well as Quality of Service (QoS). Large Scale MIMO systems employ tens to hundreds of low-power low-price antennas at base station, serving several tens of terminals at the same time. Large MIMO system can achieve full potential of conventional MIMO systems while provide additional reduction of latency and improvement of power efficiency.



Introduction

General Purpose Graphics Processing Units Computing (GPGPU)

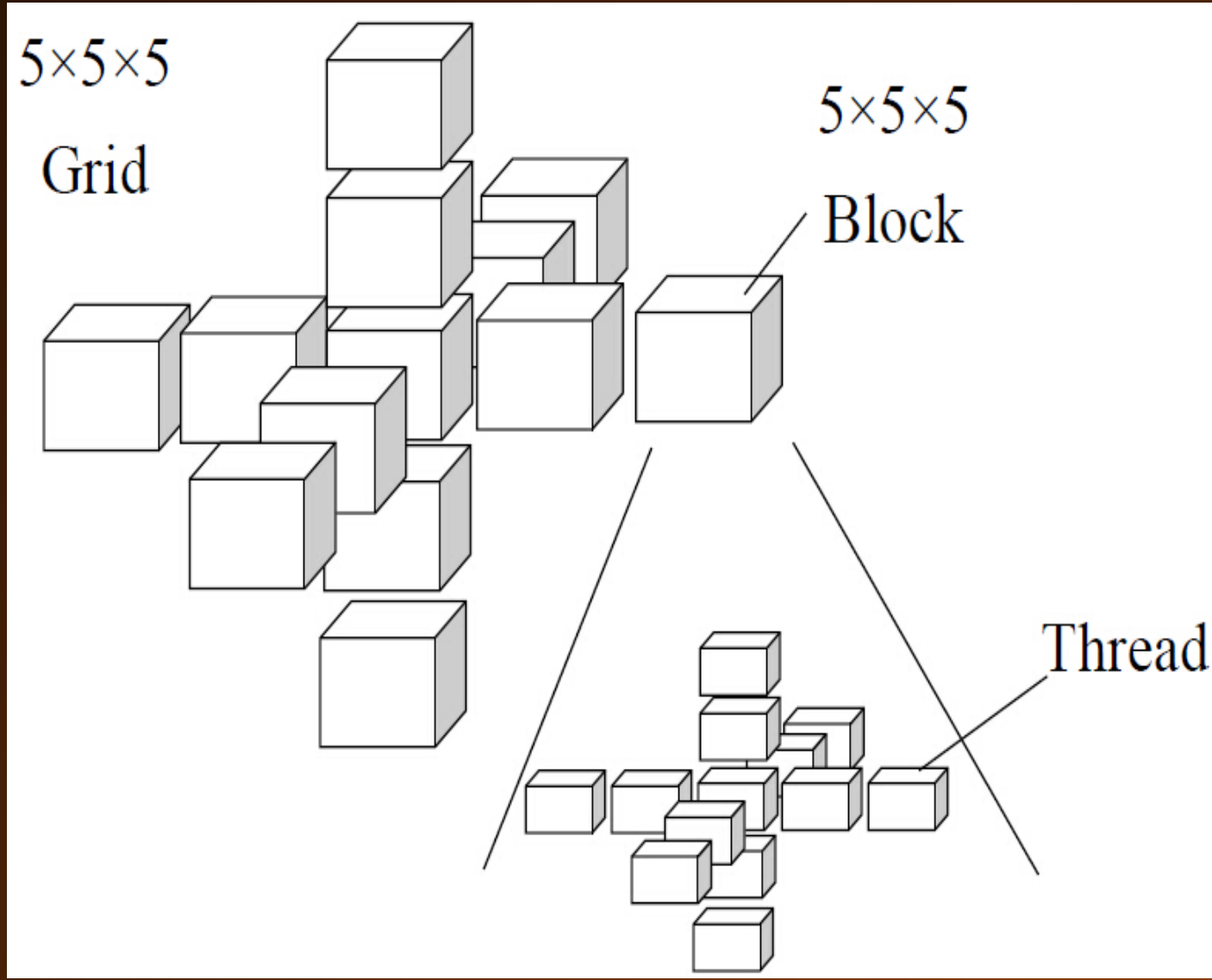
General Purpose Graphics Processing Units (GPGPU) computing is a common trend in industry and research area that using Graphics Processing Units (GPU) to handle massive computation applications which are traditionally implemented by Central Processing Units (CPU). Compared to a CPU, a GPU can naturally cope with computational intensive tasks, since most of the hardware resources (transistors) are allocated for data processing and parallel computation, rather than caching and flow control. GPU computing has been applied to different areas such as computer vision, signal processing, finance and medical electronics.



Introduction

Compute Unified Device Architecture (CUDA)

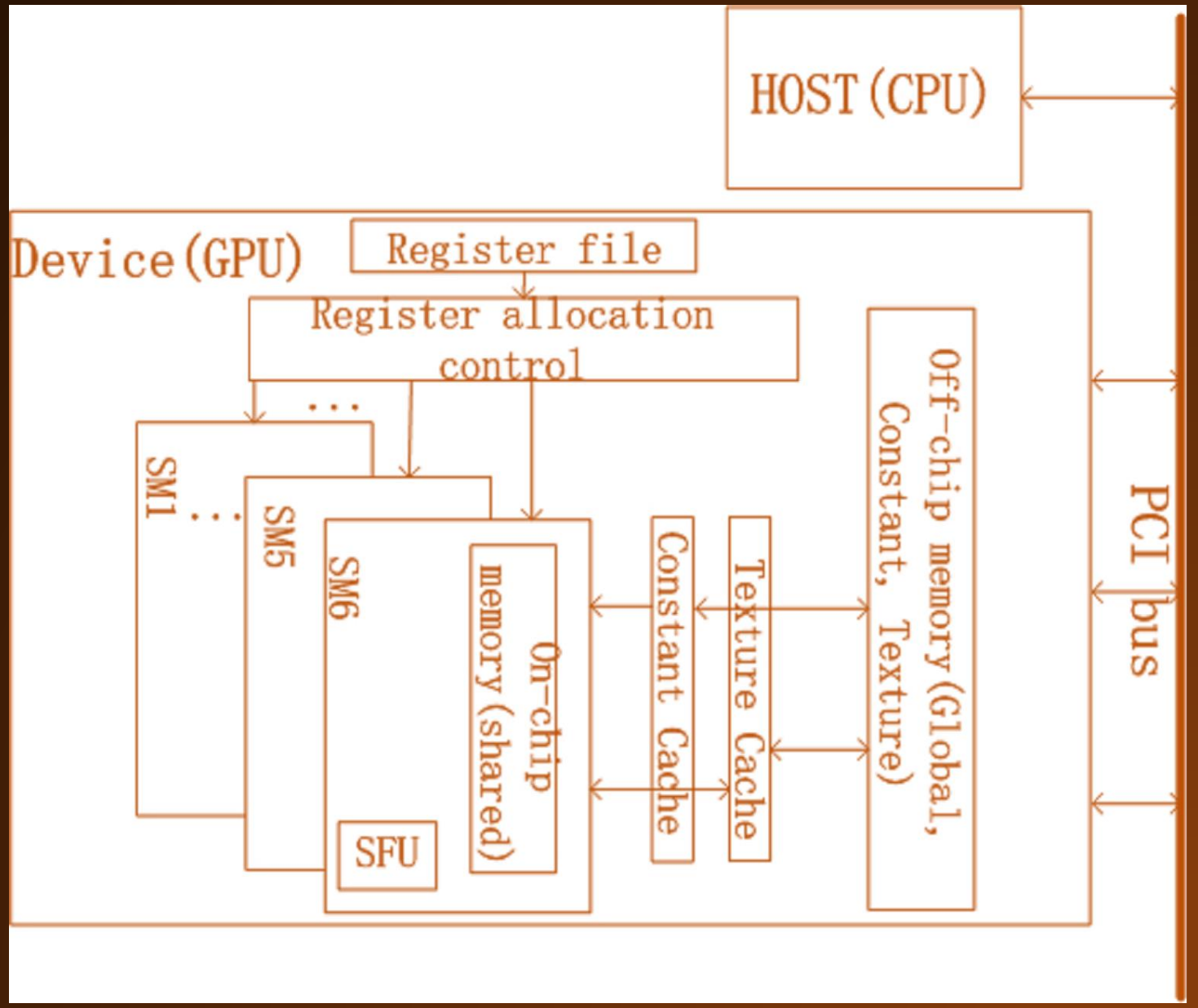
Compute Unified Device Architecture (CUDA) is a programming model that enables developers to use popular programming languages such as C, C++, Java and Python for GPU programming. It was introduced by NVIDIA in 2006. The serial code executes on host side (CPU) and parallel data processing code executes on device side (GPU). This methodology is called heterogeneous programming model which has been successfully applied in new generation supercomputers. CUDA has become a dominant framework for GPGPU because it do not require developers to learn a new programming language or know too much about hardware details. Another reason is CUDA have received continuous support from NVIDIA and CUDA developer community.



Introduction

Microarchitecture of GPU

GPU works as a coprocessor system. The major hardware resources are located at Stream Multiprocessors (SMs), which performs basic arithmetic operations, each SM has a special functional unites (SFU), which performs more complex arithmetic operations. A GPU contains five types of memory. registers, shared memory, global memory,
constant memory and texture memory.



Channel Model and Fixed Complexity Sphere Decoder (FCSD)

- MIMO channel Model

- Fixed Complexity Sphere Decoder

Channel Model and Fixed Complexity Sphere Decoder (FCSD)

MIMO channel Model

We consider a complex uncoded spatial multiplexing MIMO system with N_r receive and N_t transmit antennas, The discrete time MIMO channel model is given by

$$y = Hx + n \qquad (1)$$

$y \in C^{N_r \times 1}$ denotes the received symbol vector, $s \in C^{N_t \times 1}$ denotes the transmitted symbol vector, with components that are mutually independent and taken from signal constellation alphabet **O (e.g. 4QAM, 16QAM, 64QAM).** $\mathbf{n} \in \boldsymbol{C}^{N_r \times 1}$ denotes additive white Gaussian noise (AWGN) vector with zero mean components and variance N_o . $\boldsymbol{H} \in \boldsymbol{C}^{N_r \times N_t}$, denotes Rayleigh flat fading channel matrix, with independent identically distributed (i.i.d) circularly symmetric complex Gaussian components of unit variance. Let E_s denotes the average energy of transmitted symbols. Hence, $\frac{E_s}{N_o}$ is signal to noise ratio.

Assume the receiver has perfect channel state information (CSI), meaning that H is known, as well as the SNR. The task of the MIMO decoder is to recover s based on y and H.

Channel Model and Fixed Complexity Sphere Decoder (FCSD)

Fixed Complexity Sphere Decoder

From (1), the maximum likelihood detector (MLD) can be specified by

$$s_{ML} = arg \max_{s \in O^{N_t}} p(y|s, H) \quad (2)$$

Which can be rewritten as

$$s_{ML} = arg \min_{s \in O^{N_t}} |||y - Hx||^2 \quad (3)$$

Perform the QR factorization to H, we have

$$H = QR \quad (4)$$

Where $Q \in C^{N_t \times N_t}$ denotes unitary matrix and $\mathrm{R} \in C^{N_t \times N_t}$ denotes upper triangular matrix. For sake of brevity we gave the object function directly

$$s_{ML} = arg \min_{s \in O^{N_t}} |||R(\hat{S} - S)||^2 \quad (5)$$

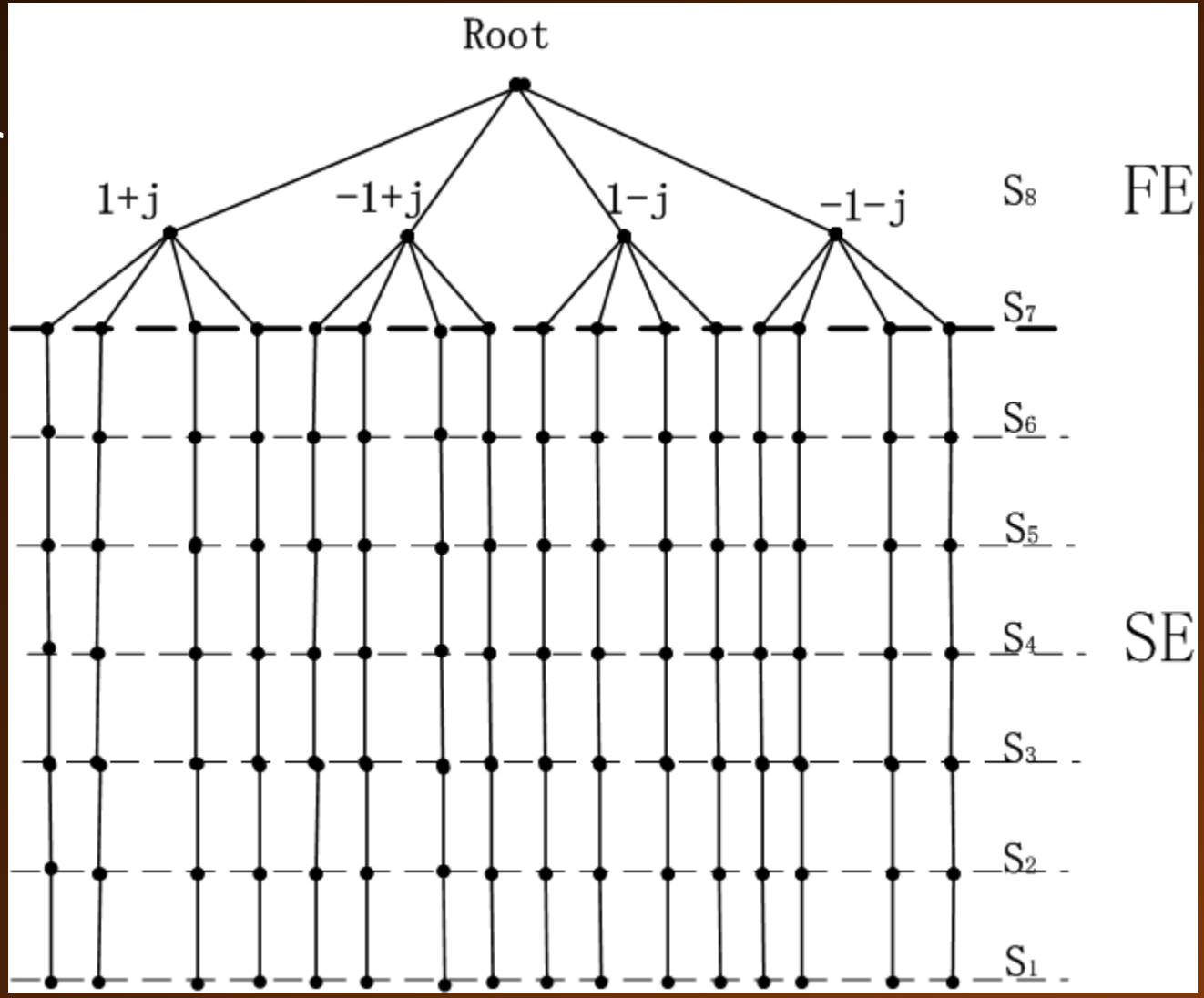
Where $\hat{s} = (H^H H)^{-1} H^H y$ denotes unconstrained estimation of s.

Channel Model and Fixed Complexity Sphere Decoder (FCSD)

Fixed Complexity Sphere Decoder

FCSD is a tree searching algorithm performs depth first searching.

At the first ρ node levels FCSD searches all the possible signal symbols exhaustively in constellation alphabet O, a process called Full Expansion (FE). at the remaining $N_t - \rho$ levels of nodes, the FCSD employs decision feedback, therefore there are only one branch expansion for each symbol node at this stage. This stage is called Single Expansion (SE). Thus the total number of the branches is fixed, only determined by FE. FCSD works as a constant number of independent multiple path tree searching algorithm.



Channel Model and Fixed Complexity Sphere Decoder (FCSD)

Fixed Complexity Sphere Decoder

- In conclusion FCSD tree searching process can be expressed as

FE stage:

$$s_i^F \in O^{N_t} \; if \; i = N_t, N_t\text{-}1, \ldots N_t - \rho + 1 \quad (6)$$

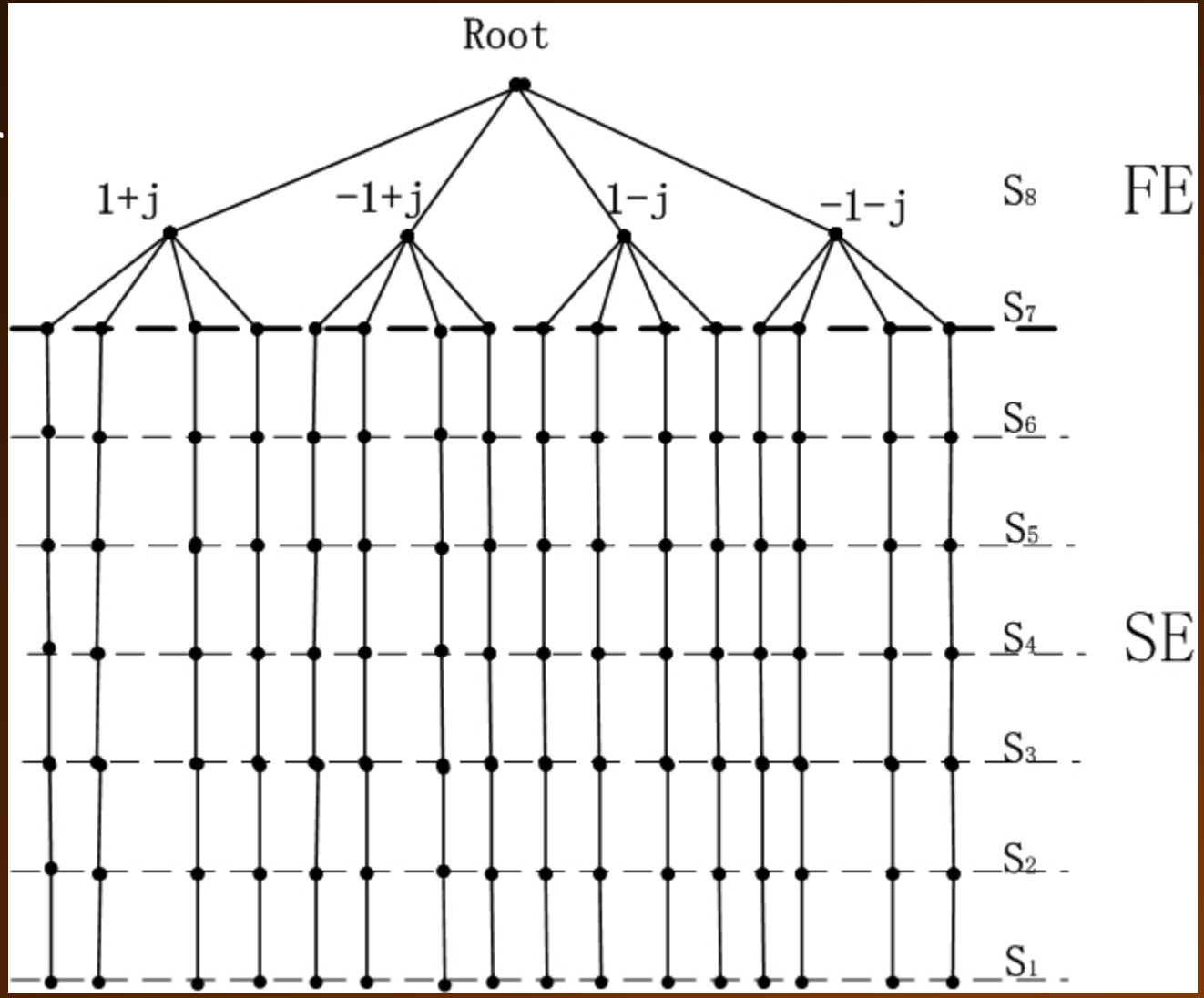
SE stage:

$$s_i^F = \mathbb{Q}[\widehat{s_i} + \sum_{j=i+1}^{N_t} \frac{r_{ij}}{r_{ii}} (\widehat{s_j} - s_j^F)] \quad (7)$$

Where s^F denotes one of solution candidate. $\mathbb{Q}$ denotes constellation quantization operation.

- At the post processing step, FCSD compares Euclidean distance and choose the optimal vector candidate.

$$E_u = ||R(\hat{s} - s)||^2 \quad (8)$$



GPU Based Acceleration of FCSD

- Preprocessing
- Integrated Parallel Acceleration of Paths Searching
- Memory Coalescing

GPU Based Acceleration of FCSD

Preprocessing

Preprocessing process of FCSD includes calculation of unconstrained estimation s hat in (), QR factorization as well as iterative channel ordering.

A FCSD channel ordering strategy is used in order to avoid error propagation in the serial path searching process. This channel ordering strategy is based on post processing SNR [reference]

$$\varphi_m = \frac{E_s}{N_o (H^H H)_m^{-1}}$$

Where φ_m denotes the post processing SNR of m th data stream.

The FCSD channel ordering works iteratively based on the following rule

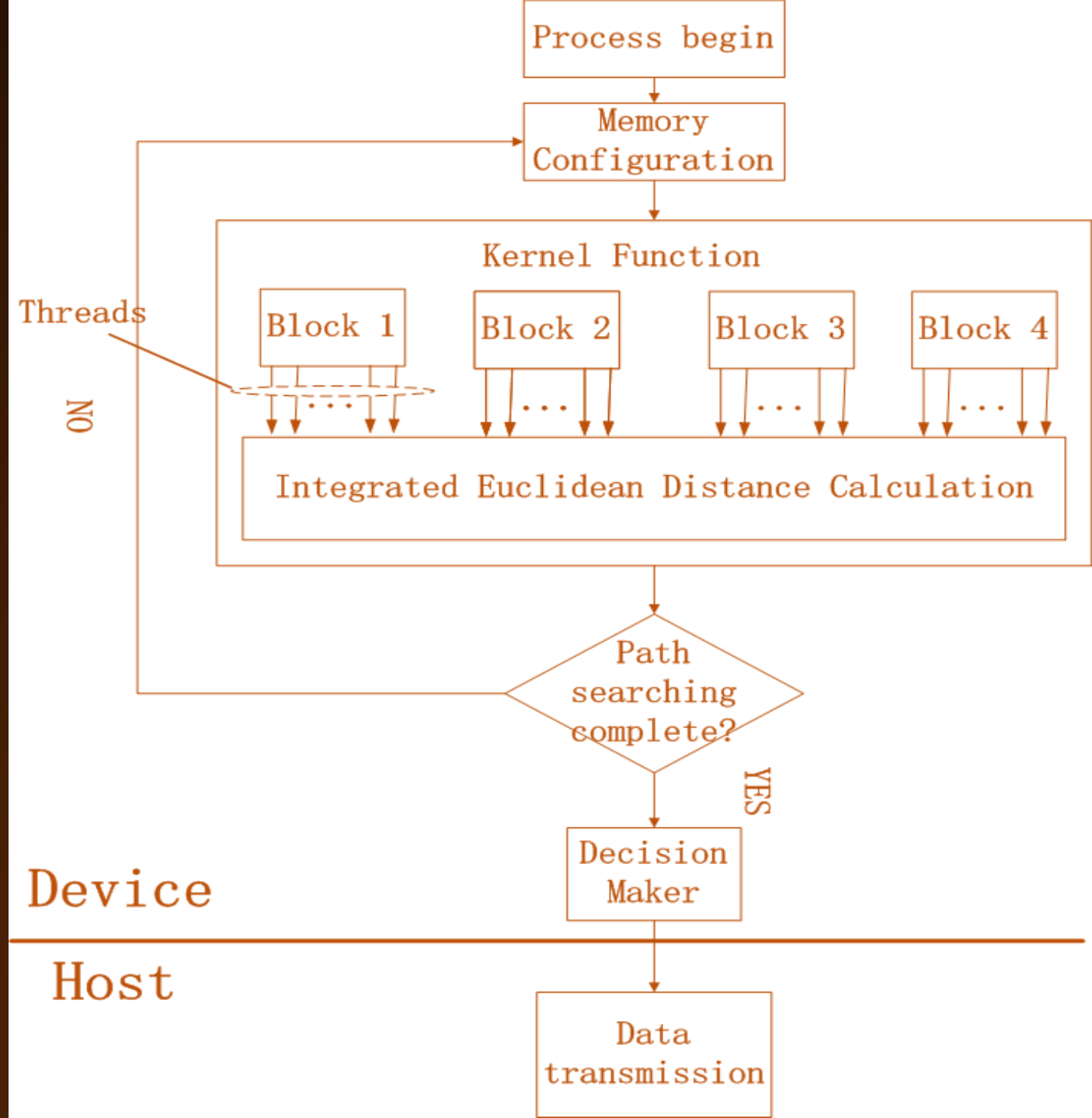
$$p = \begin{cases} arg\min_k (H_j^H H_j)_k^{-1} & FE\ stage \\ arg\max_k (H_j^H H_j)_k^{-1} & SE\ stage \end{cases}$$

As we can see, the preprocessing process contains a large amount of matrix operations (e.g. matrix-matrix multiplication, matrix-vector multiplication, matrix inverse and QR factorization). Which can be naturally coped by GPU parallel programming. We make use of the high performance CUDA basic linear algebra subroutines (cuBLAS) to accelerate preprocessing process.

GPU Based Acceleration of FCSD

Integrated Parallel Acceleration of Paths Searching

- Since the decision feedback searching paths has a serial nature, we match one path to one thread. In order to have the largest number of threads that can be parallelized, we use one dimensional blocks for widest expansion, and organize all the paths into several parallel blocks.



GPU Based Acceleration of FCSD

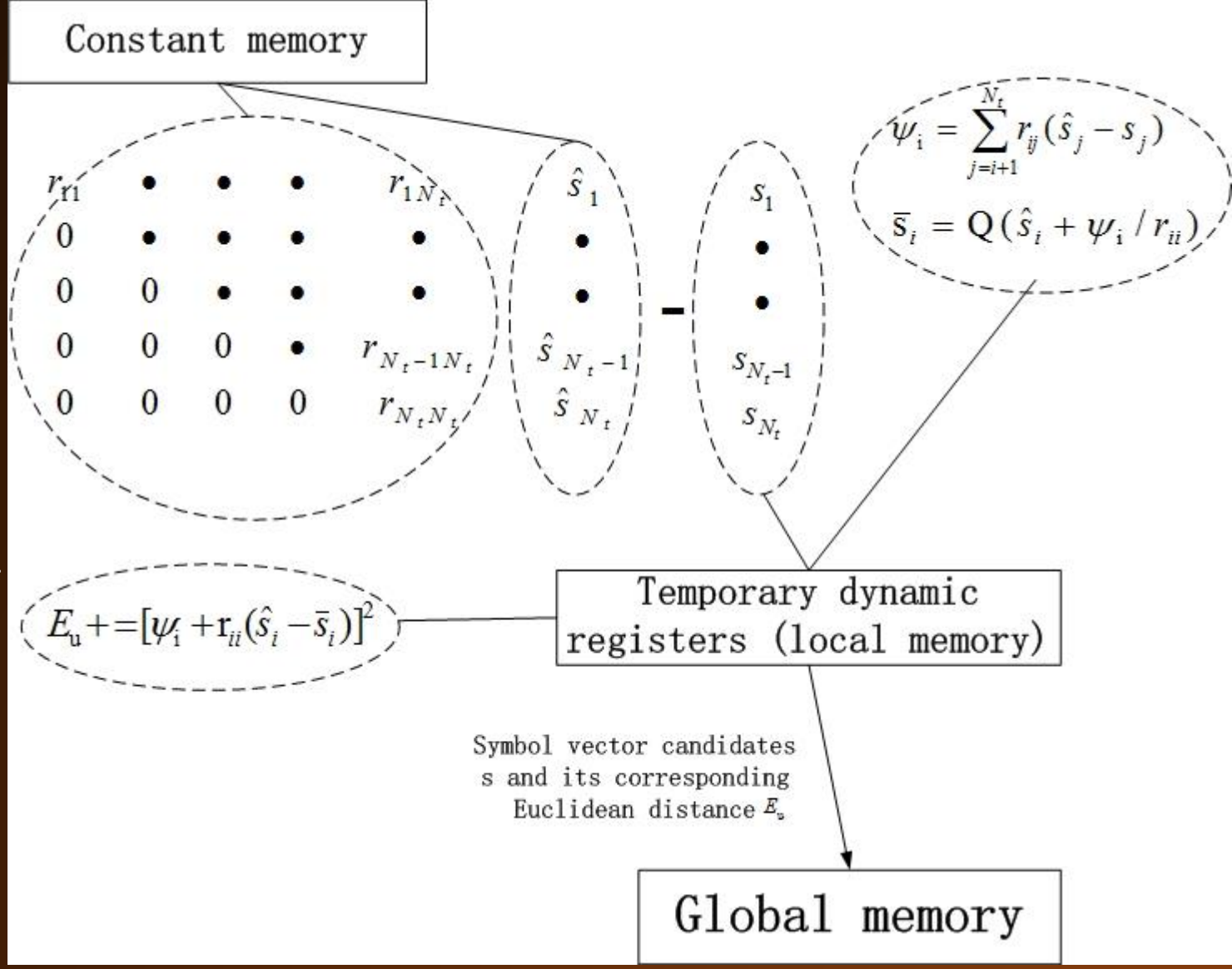
Integrated Parallel Acceleration of Paths Searching

- Recall the structural difference between CPU and GPU, modern CPUs bridge the sharp gulf between memory access speed and clock rate by caching, GPUs are computational intensive, therefore utilization of memory bandwidth is the bottleneck of overall performance.
- Based on (), at post processing stage, Euclidean distance of all the possible candidates are calculated and compared.

$E_u = \sum_{i=1}^{N_t} |\sum_{j=i}^{N_t} r_{ij}(\widehat{s_j} - s_j)|^2$

if we look at each searching path, we can find the value is partly calculated, at each node we have

$$\psi_i = \sum_{j=i+1}^{N_t} r_{ij}(\widehat{s_j} - s_j)$$



- We store this metric into local memory, at each node of searching path, the following operation is made $E_u \mathrel{+}= |\psi_i + (\widehat{s_i} - s_i)|^2$

GPU Based Acceleration of FCSD

Memory Coalescing

- Recall the structural difference between CPU and GPU, modern CPUs bridge the sharp gulf between memory access speed and clock rate by caching, GPUs are computational intensive, therefore utilization of memory bandwidth is the bottleneck of overall performance.
- Based on (), at post processing stage, Euclidean distance of all the possible candidates are calculated and compared.

$E_u = \sum_{i=1}^{N_t} |\sum_{j=i}^{N_t} r_{ij}(\widehat{s_j} - s_j)|^2$

if we look at each searching path, we can find the value is partly calculated, at each node we have

$$\psi_i = \sum_{j=i+1}^{N_t} r_{ij}(\widehat{s_j} - s_j)$$

Thank you!


telecommunications &
signal processing
laboratory
McGill
UNIVERSITY