StreamIt

Tarun Pondicherry
COS 597C
November 18, 2010

Overview

- Motivation
- Stream Programming
- StreamIt Language
- StreamIt Parallelization
- Conclusions

Overview

- Motivation
- Stream Programming
- StreamIt Language
- StreamIt Parallelization
- Conclusions

Motivation

- As argued by “StreamIt Cookbook”:
- Why von Neumann languages (C/C++/Java) successful?
  - Abstract out differences of von Neumann machines
  - Efficient mapping to von Neumann machine
- “Today von Neumann languages are a curse!”
  - Efficient mapping to parallel architectures difficult
  - Force programmer to take target architecture into account
  - Force programmer to *explicitly* parallelize: deal with threads, communication and synchronization

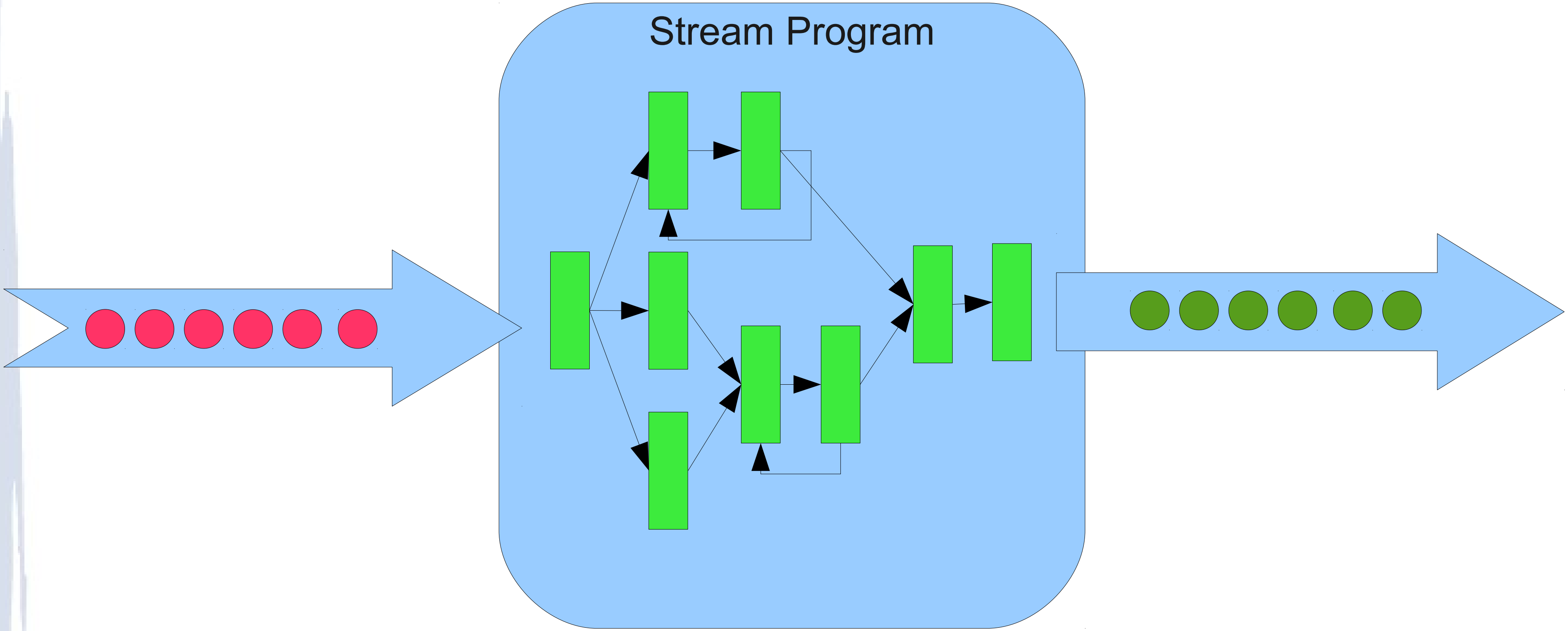
Motivation

- Want *implicitly* parallel abstraction that
  - Abstracts out differences in parallel architectures (number of cores, communication methods, synchronization methods, etc.)
  - Allows efficient mapping to parallel architecture
  - Directly exposes tasks that can run in parallel
- Stream Programming Abstraction
  - Trade off generality for performance and ease of programming
  - Many applications naturally fit paradigm
  - Implicitly parallel
  - Allows efficient mapping by compiler

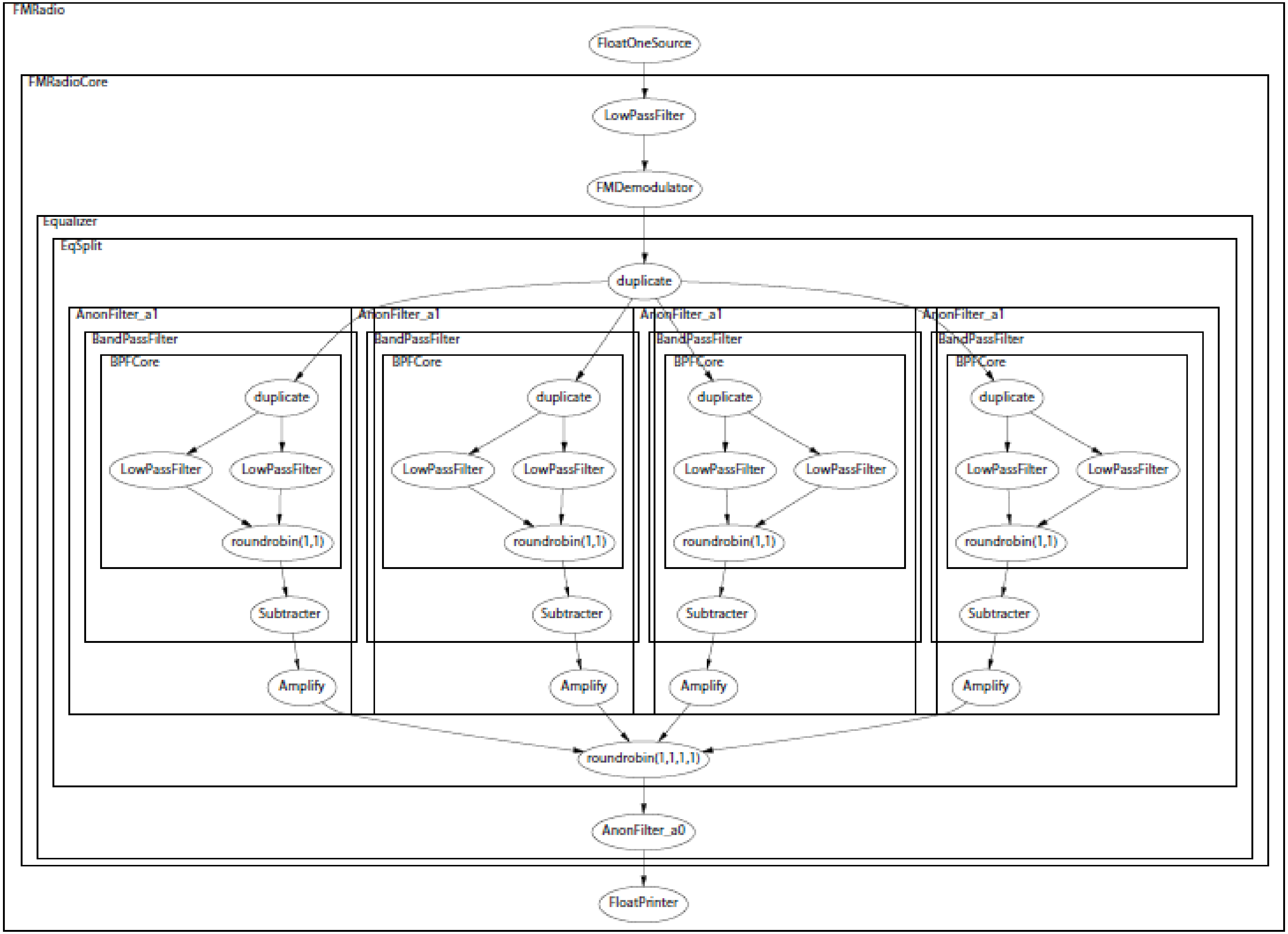
Overview

- Motivation
- Stream Programming
- StreamIt Language
- StreamIt Parallelization
- Conclusions

Stream Programming



Stream Programming: Example



Source: http://groups.csail.mit.edu/cag/streamit/papers/streamit-cookbook.pdf

Stream Programming: Applications

- Mobile
  - Compression (LZW)
  - Encryption (DES)
- Desktop
  - Streaming audio / video (MPEG-2)
  - Graphics (Depth of Field)
- Servers
  - Software routers
  - Modulation / Demodulation (Cell phone)

Stream Programming: Properties

- Large/Unbounded amount of data
  - Short lifetime per data item
  - Minimal processing per data item
- Regular, repeating computation
  - Static structured graph of filters
  - Independent actors
  - Explicit communication

Stream Programming: Properties

- Filter is autonomous unit of computation
- Each filter
  - Own PC
  - Own Address space
- Filters
  - Unaware of execution order
  - Communicate explicitly
- Stream program consists of a static structured graph of filters

Overview

- Motivation
- Stream Programming
- StreamIt Language
- StreamIt Parallelization
- Conclusions

StreamIt Language

- Program consists of filters connected by structured graph
- Filter
    - Autonomous unit of computation
    - Single input, single output
    - Stateful / Stateless
    - Composable
- Pipeline
    - Filters connected by FIFO queues
- Structured Stream
    - Pipeline
    - Splitjoin and Feedback Loop

StreamIt Language: Filter

```
int->int filter Averager(int n) {
  work pop 1 push 1 peek n {
    int sum = 0;
    for (int i = 0; i < n; i++)
      sum += peek(i);
    push(sum/n);
    pop();
  }
}
```

Code Source: http://groups.csail.mit.edu/cag/streamit/papers/streamit-cookbook.pdf

StreamIt Language: Filter

- Declaration
  - Defines input / output data type (int, float, bit, complex, struct)
  - Parameters for filter instantiation (parameters constant in init/work)
- Init Function
  - Called once to set up state
- Work Function
  - Called forever
  - Defines number of items popped from input stream, pushed to output stream, peeked at from input stream
  - Push/pop rates do not have to match

StreamIt Language: Pipeline

IntSource

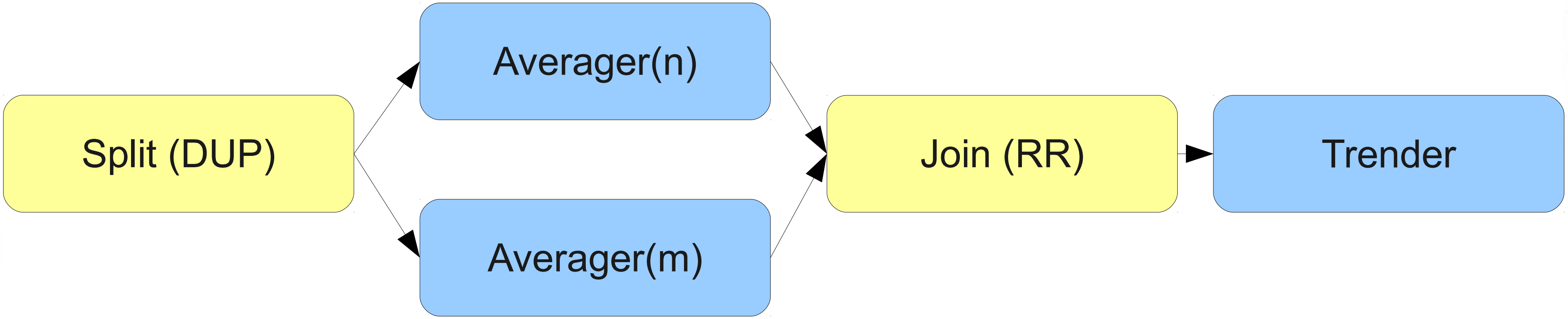
Averager(10)

IntPrinter

```
void->void pipeline MovingAverage {
  add IntSource();
  add Averager(10);
  add IntPrinter();
}
```

Code Source: http://groups.csail.mit.edu/cag/streamit/papers/streamit-cookbook.pdf

StreamIt Language: SplitJoin



```
float->float splitjoin
dualAverager(int n, int m) {
  split duplicate;
  add Averager(n);
  add Averager(m);
  join roundrobin;
}
```

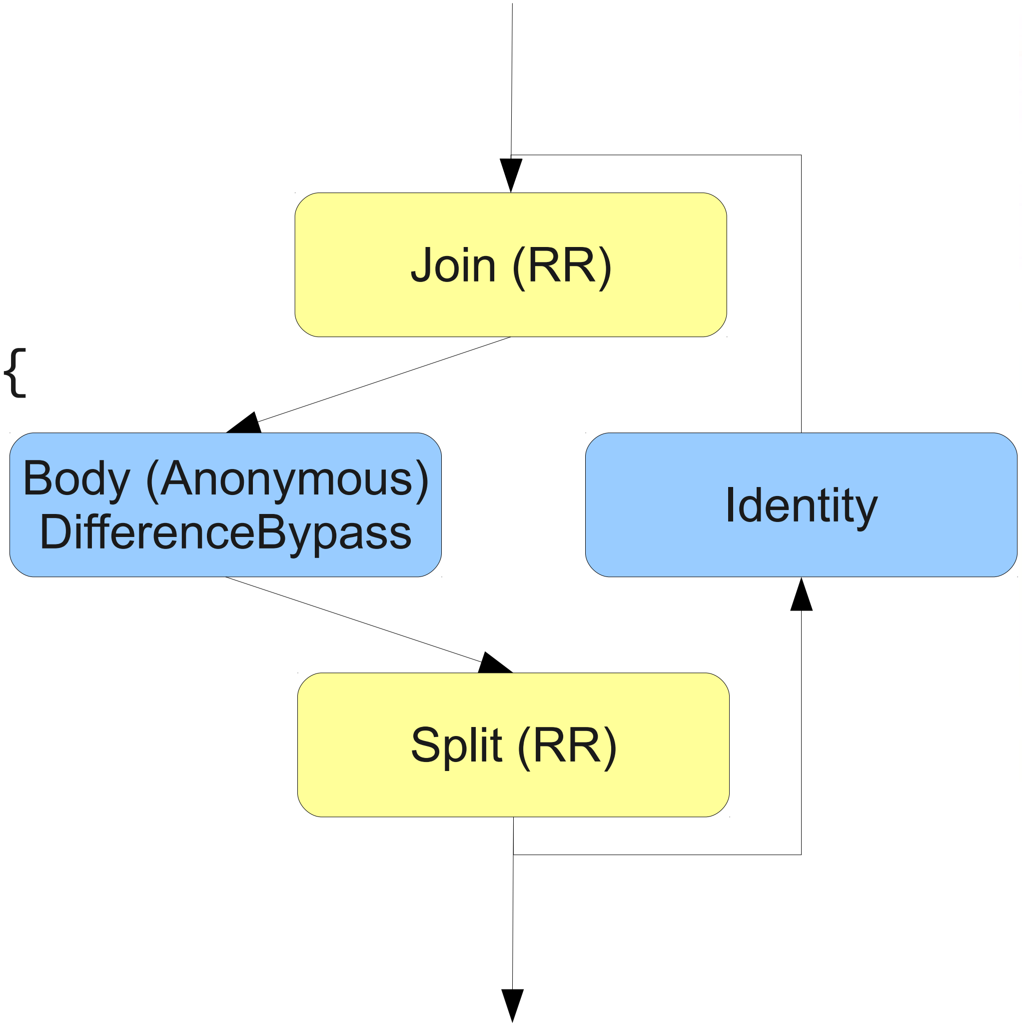
```
float->int filter Trender{
  work pop 2 push 1 {
    float a = pop();
    float b = pop();
    if (a > b) { push(1); }
    else { push(0); }
  }
}
```

StreamIt Language: SplitJoin

- Split: divide stream into multiple streams
    - Duplicate
    - Round robin
- Join: combine streams into single stream
    - Round robin
- Can specify flow rate from each input/output filter

StreamIt Language: FeedbackLoop

```
int->int feedbackloop
EdgeDetecter {
   join roundrobin(1, 1);
   body int->int filter {
      work pop 2 push 2 peek 2 {
         push(peek(0)-peek(1));
         push(peek(0));
         pop();
         pop();
      }
   }
   loop Identity<int>;
   split roundrobin(1, 1);
   enqueue(0);
}
```



StreamIt Language: FeedbackLoop

- Join: combine input and loop feedback
- Body: filter for forward operation
- Loop: filter for reverse operation
- Split: split forward operation output into output and loop feedback input
- Enqueue: initial values on joiner feedback path

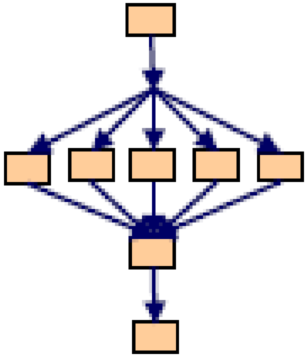
- Deadlock if loop filter output becomes empty
- Prequeue values with enqueue to ensure joiner always has input at startup
- Immediate data dependency with slow body can slow execution of feedback block

Overview

- Motivation
- Stream Programming
- StreamIt Language
- StreamIt Parallelization
- Conclusions

StreamIt Parallelization

- Structured Graph Exposes Parallelism
  - Task parallelism (example Threads)
  - Data parallelism (example Do All)
  - Pipeline parallelism (example ILP)
- Target architectures vary
  - Granularity
  - Topology
  - Communication
  - Memory



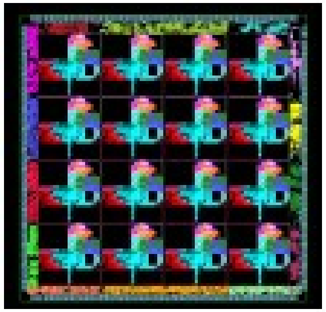


Image Source:http://groups.csail.mit.edu/cag/streamit/talks/pact03tutorial/index.html

StreamIt Parallelization: Compiler

- Partitioning/Placement
  - Coarsen Granularity
  - Data Parallelize
  - Software Pipeline
- Scheduling
- Code Generation (output C/Java code)

StreamIt Parallelization: Compiler

- Naïve Partitioning / Scheduling
- Fine-grained data parallelism

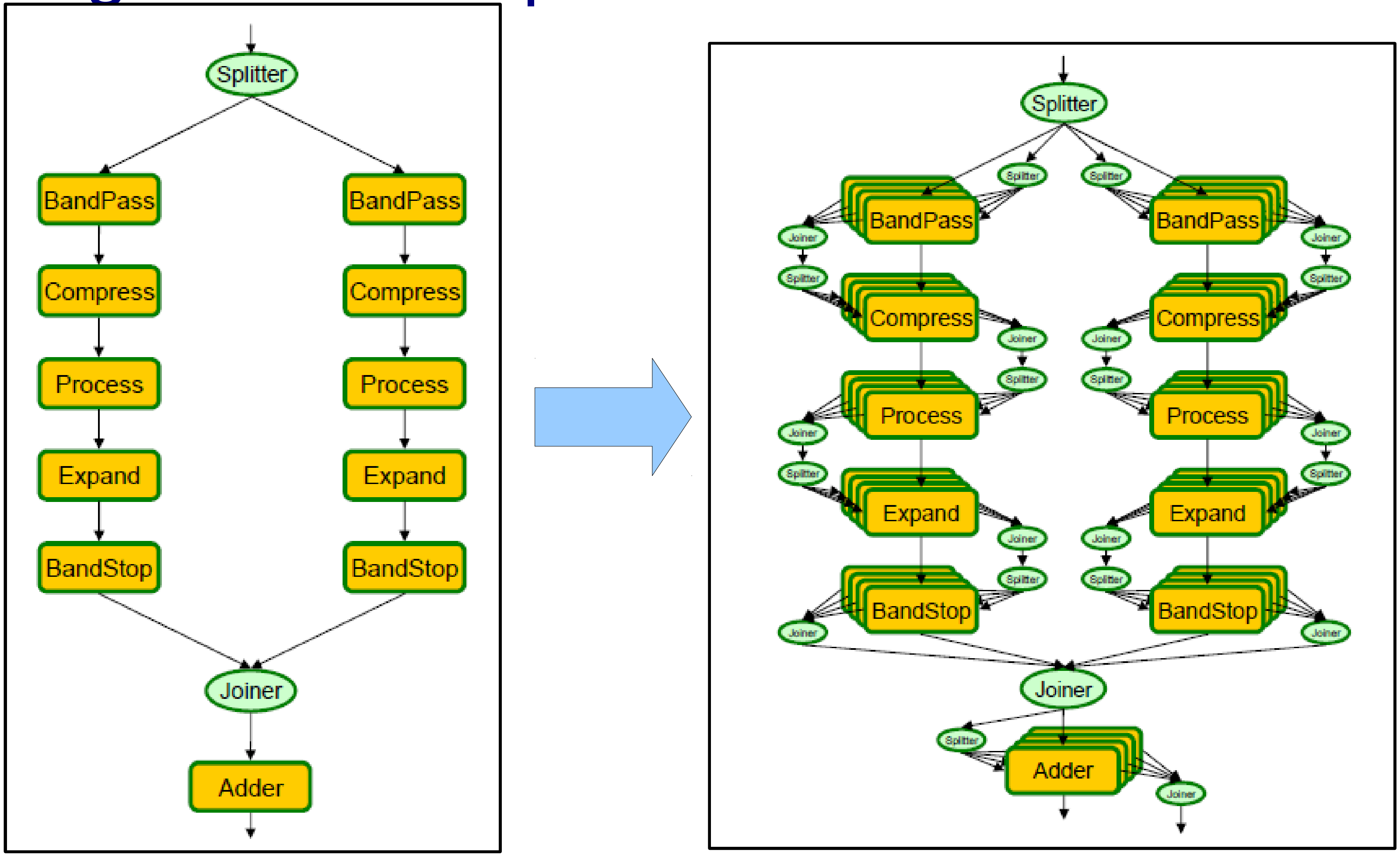
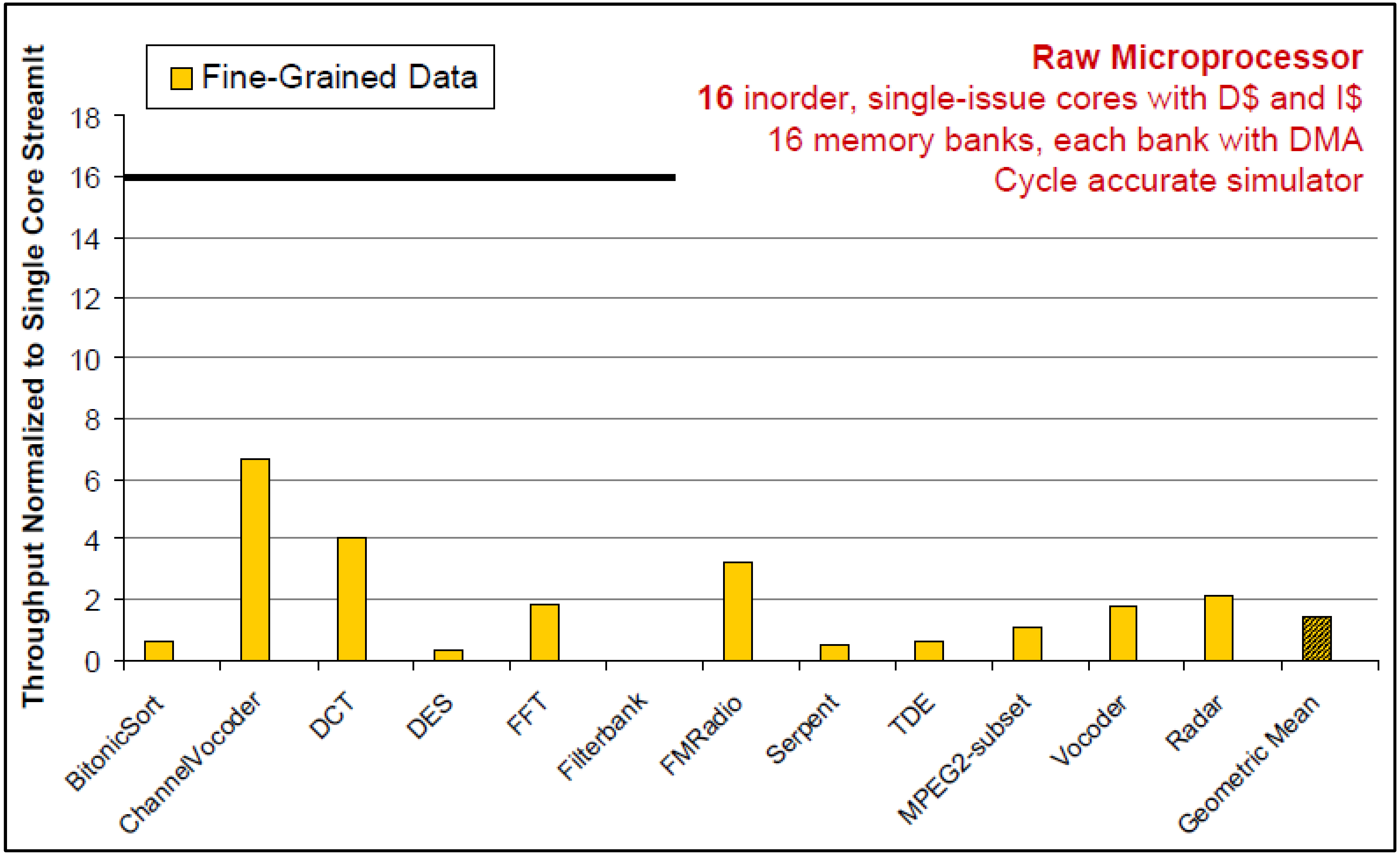


Image Source:http://groups.csail.mit.edu/cag/streamit/talks/pact03tutorial/index.html

StreamIt Parallelization: Compiler

- Large synchronization overhead

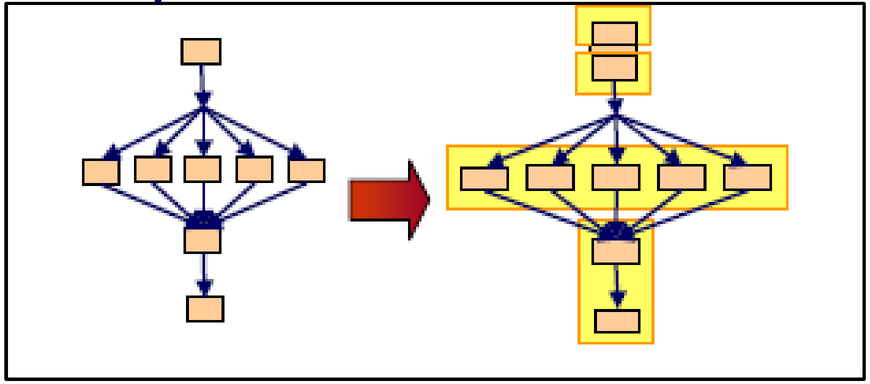


Raw Microprocessor
16 inorder, single-issue cores with D$ and I$
16 memory banks, each bank with DMA
Cycle accurate simulator

Image Source:http://groups.csail.mit.edu/cag/streamit/talks/pact03tutorial/index.html

StreamIt Parallelization: Compiler

- Ideal Partitioning
  - Each filter has dedicated tile
  - Each filter performs same amount of work



- Compiler Algorithm

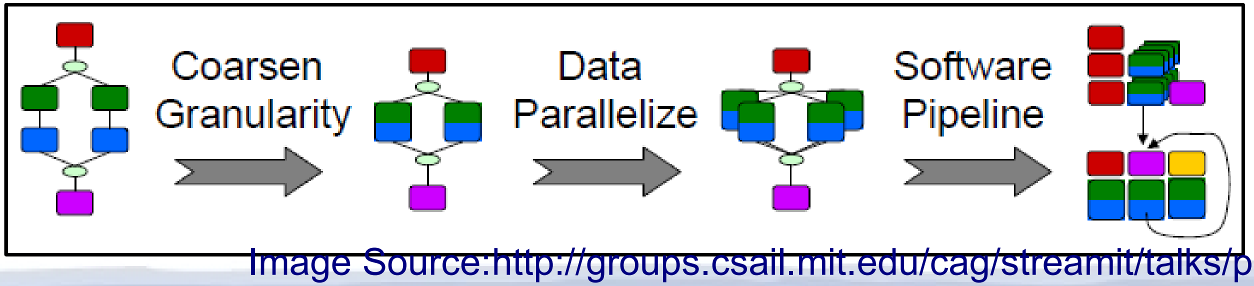


Image Source:http://groups.csail.mit.edu/cag/streamit/talks/pact03tutorial/index.html

StreamIt Parallelization: Compiler

- Coarsen Granularity, Data Parallelization

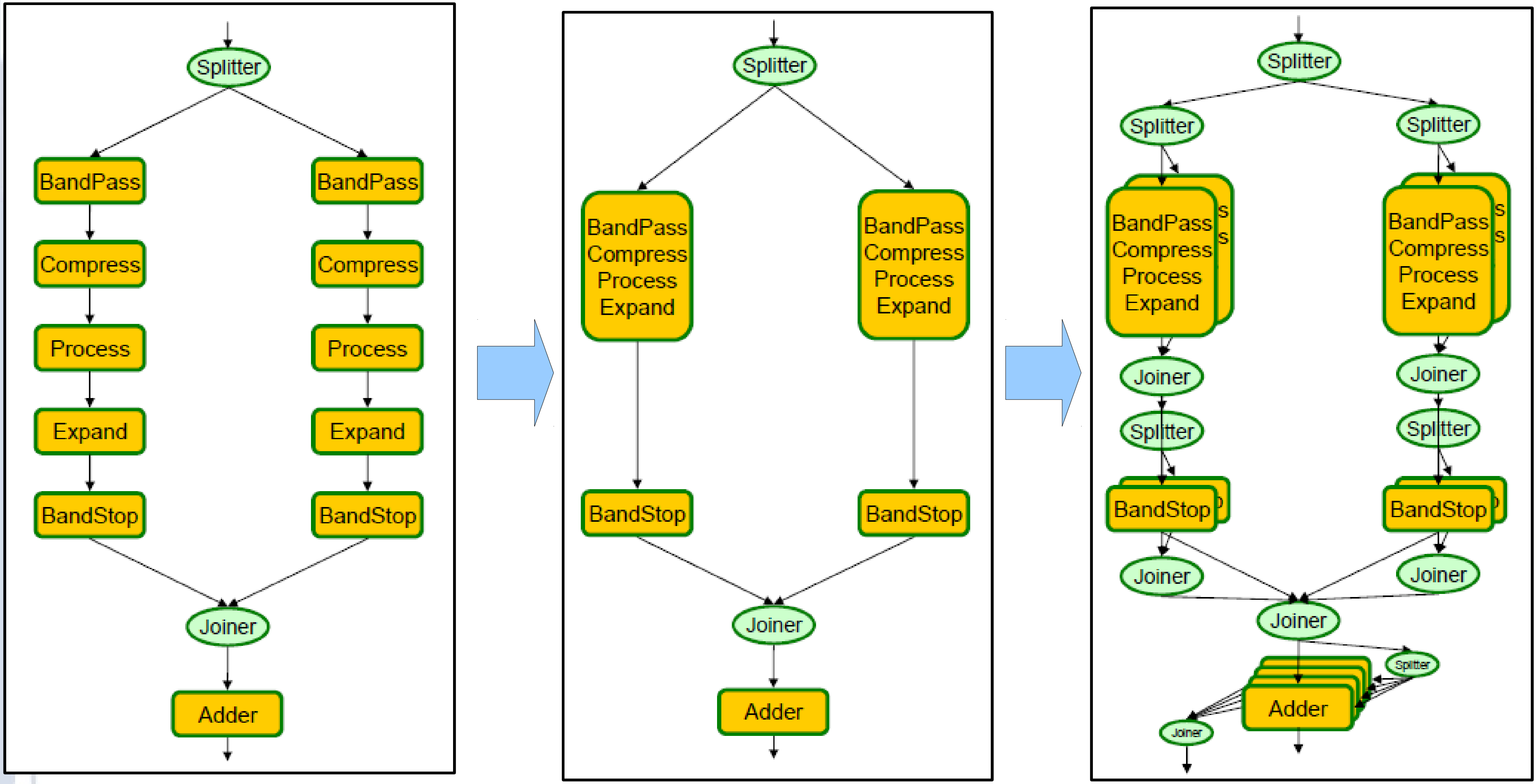


Image Source:http://groups.csail.mit.edu/cag/streamit/talks/pact03tutorial/index.html

StreamIt Parallelization: Compiler

- Task Parallelization

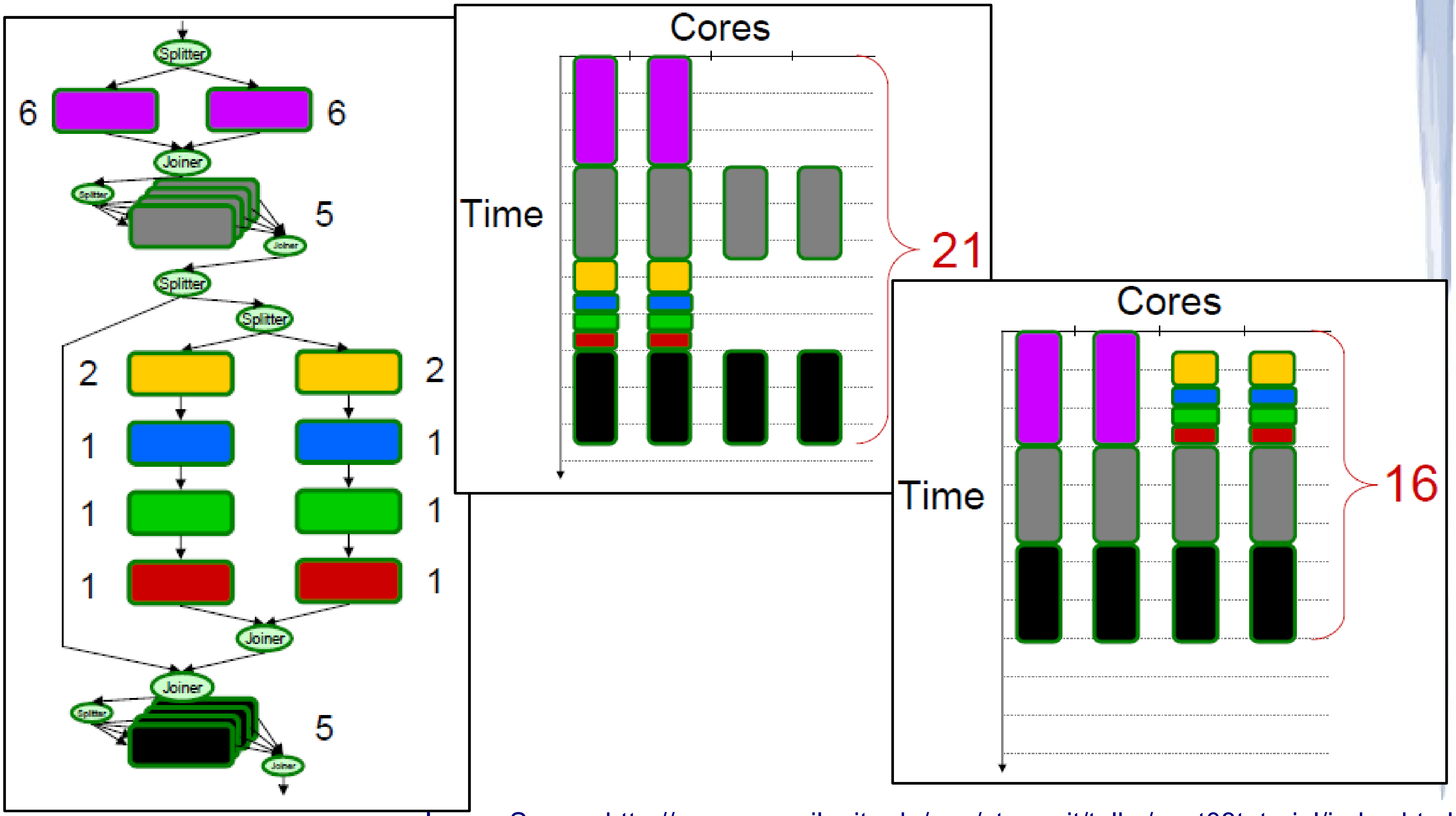


Image Source:http://groups.csail.mit.edu/cag/streamit/talks/pact03tutorial/index.html

StreamIt Parallelization: Compiler

- Lower synchronization overhead

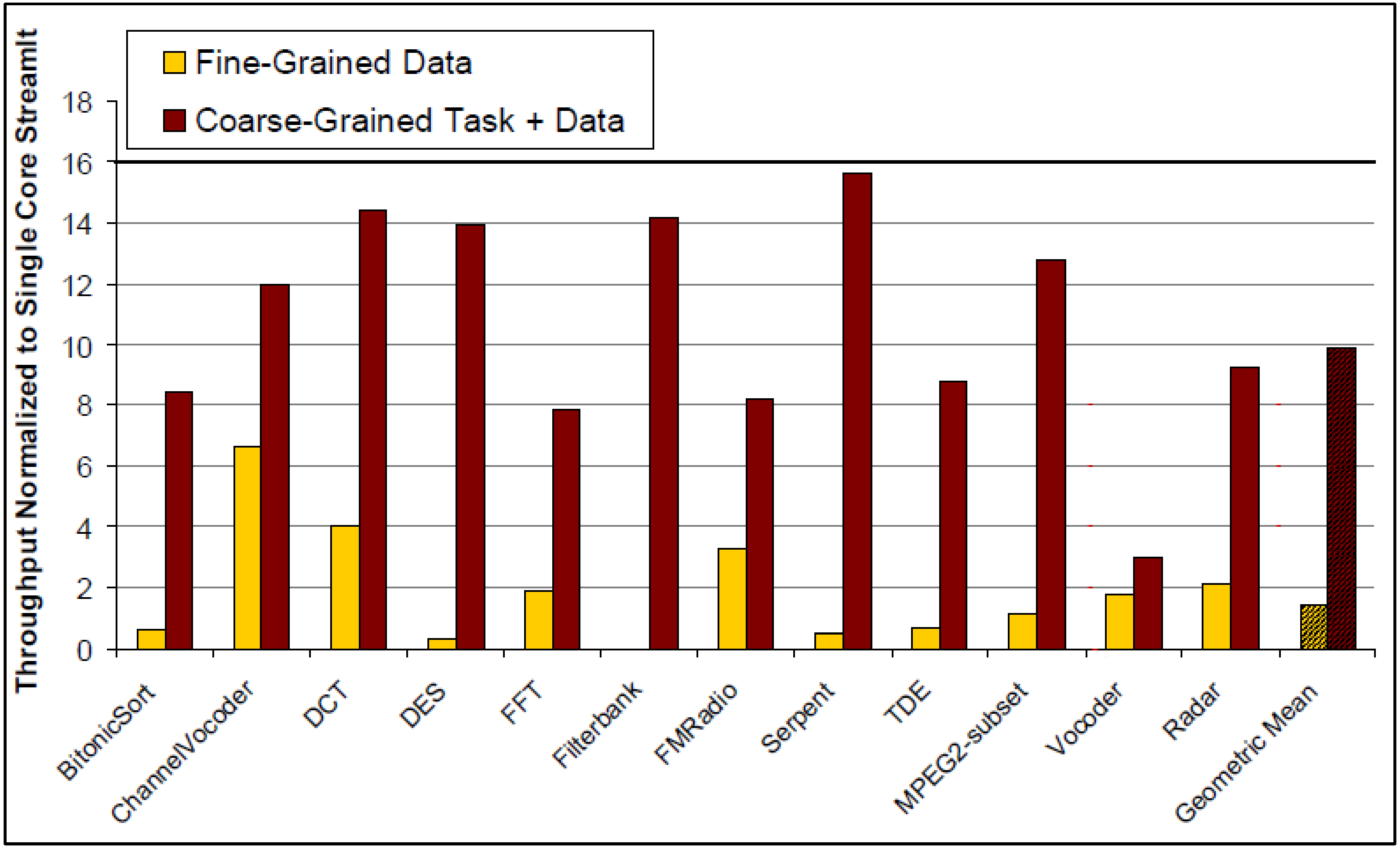


Image Source:http://groups.csail.mit.edu/cag/streamit/talks/pact03tutorial/index.html

StreamIt Parallelization: Compiler

- Software Pipeline

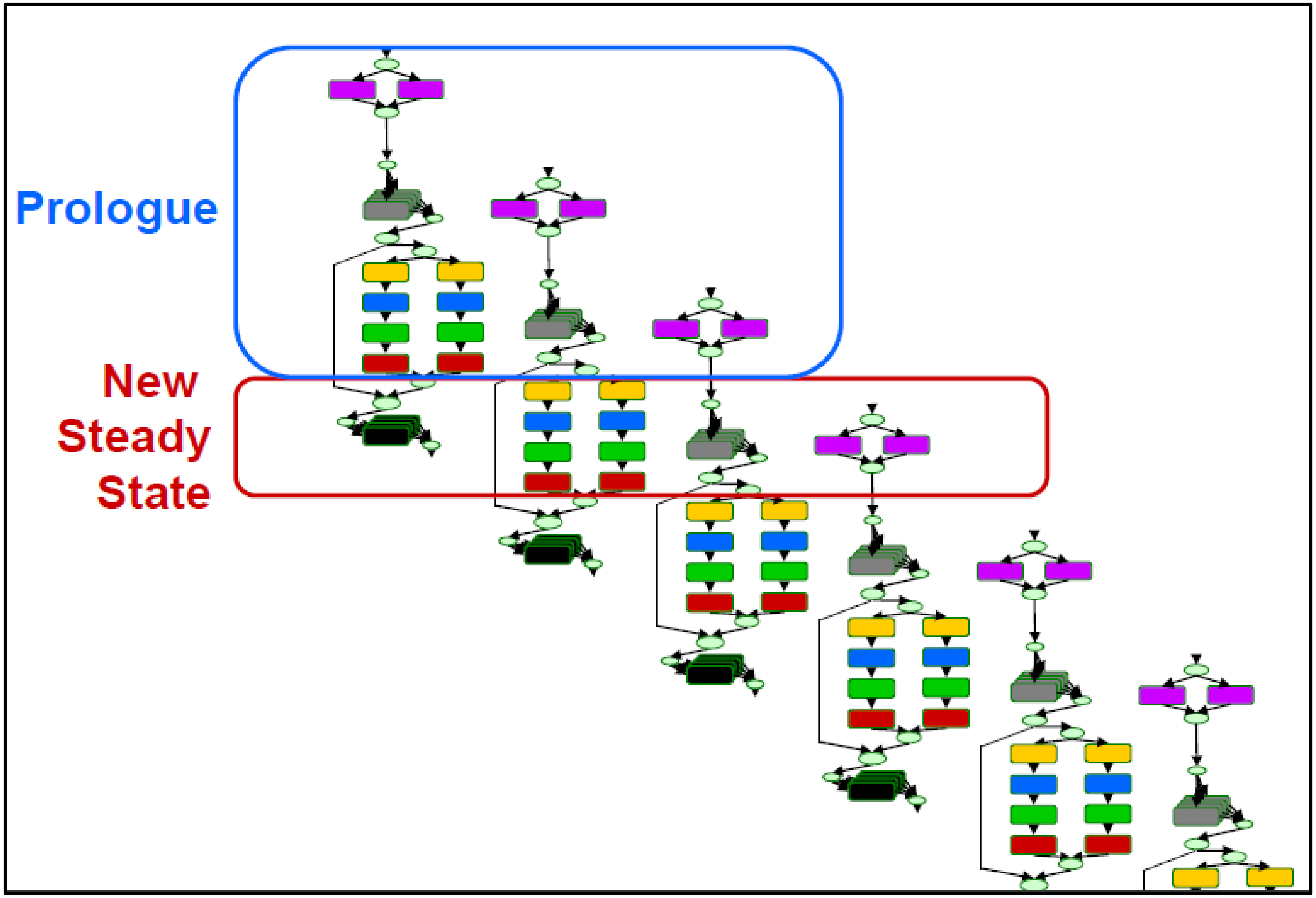


Image Source:http://groups.csail.mit.edu/cag/streamit/talks/pact03tutorial/index.html

StreamIt Parallelization: Compiler

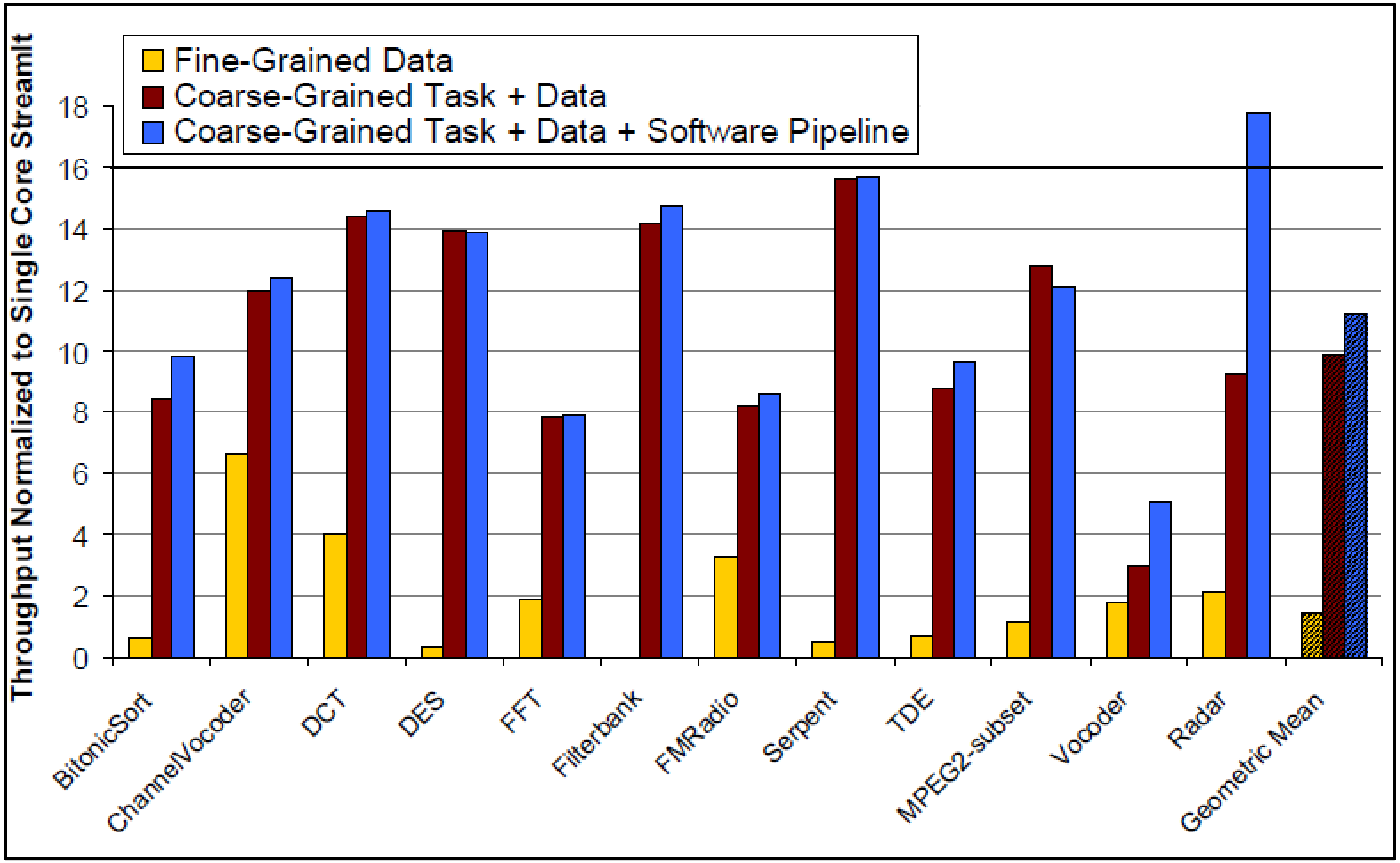


Image Source:http://groups.csail.mit.edu/cag/streamit/talks/pact03tutorial/index.html

Streamit Parallelization: Scheduler

- Steady state remains same for each cycle
  - Find steady state schedule at compile time
  - All data rates known at compile time
  - Solve system of linear equations to find steady state schedule
- Find prologue schedule
- Low scheduling overhead at runtime

Conclusions

- Stream Programming
  - Programmer does not need to focus on concurrency
  - Programmer does need to be aware of which techniques run more efficiently in parallel
  - Exposes task, data and pipeline parallelism
- Compiler can manage parallelism
  - Choose granularity
  - Perform load balancing
  - Take care of concurrency issues
  - Optimize for given architecture

References

- Language Reference
  - http://groups.csail.mit.edu/cag/streamit/papers/streamit-lang-spec.pdf
- StreamIt Cookbook
  - http://groups.csail.mit.edu/cag/streamit/papers/streamit-cookbook.pdf
- Presentations
  - http://people.csail.mit.edu/mgordon/mgordon-phd-defense.pdf
  - http://groups.csail.mit.edu/cag/streamit/talks/StreamIt-IBM-PL-Day-04.pdf
  - http://research.microsoft.com/en-us/um/people/thies/thesis-defense.pdf
  - http://groups.csail.mit.edu/cag/streamit/talks/pact03tutorial/index.html
  - http://groups.csail.mit.edu/cag/streamit/talks/StreamIt-NEPLS-8-02.ppt
- Images and line of thought on some slides taken directly from presentations