MPEG-2 Decoding in a Stream Programming Language

Matthew Drake, Hank Hoffmann, Rodric Rabbah and Saman Amarasinghe

Massachusetts Institute of Technology

IPDPS
Rhodes, April 2006

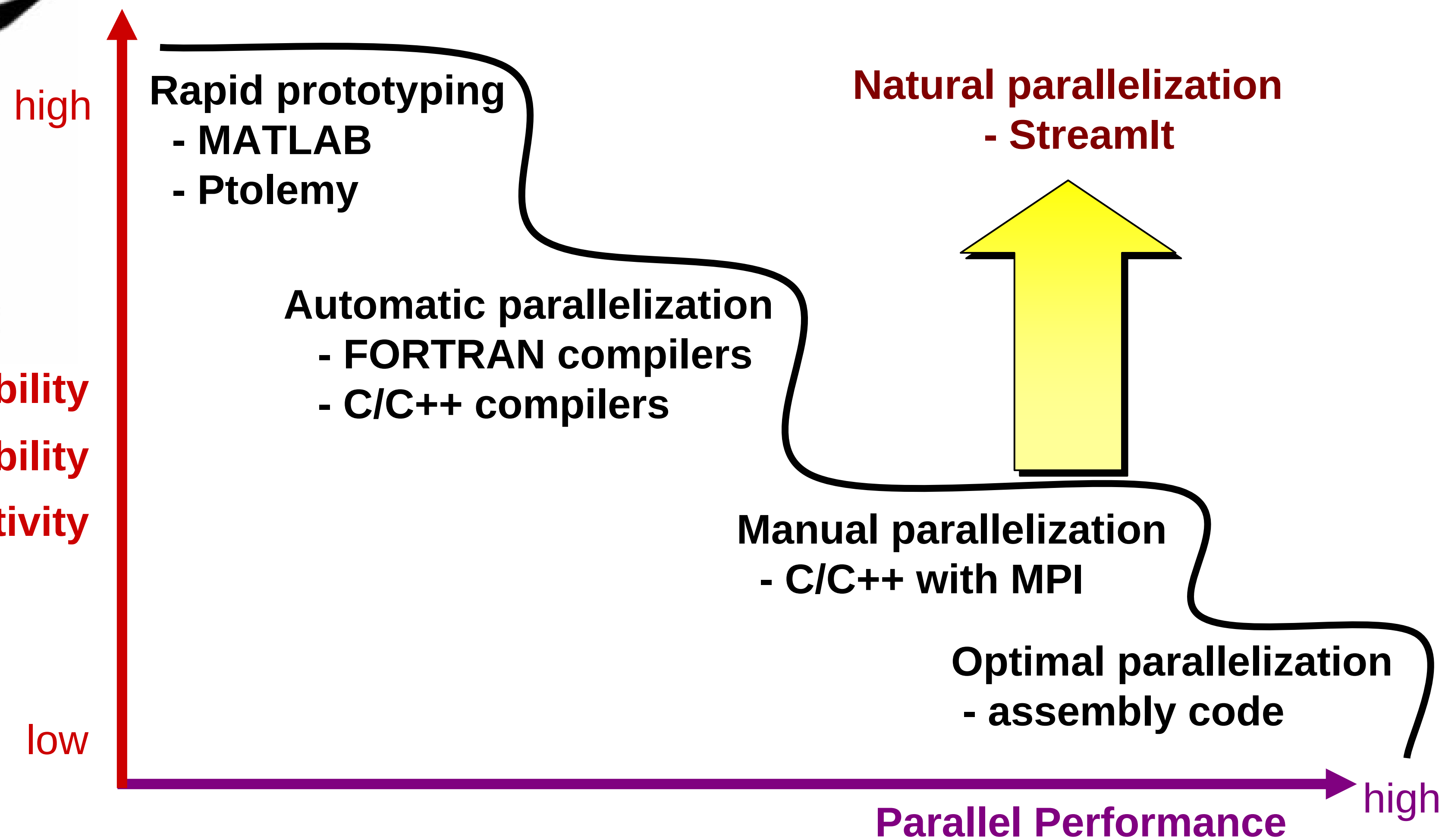
http://cag.csail.mit.edu/streamit

Stream Application Domain

- Graphics
- Cryptography
- Databases
- Object recognition
- Network processing and security
- Scientific codes
- …

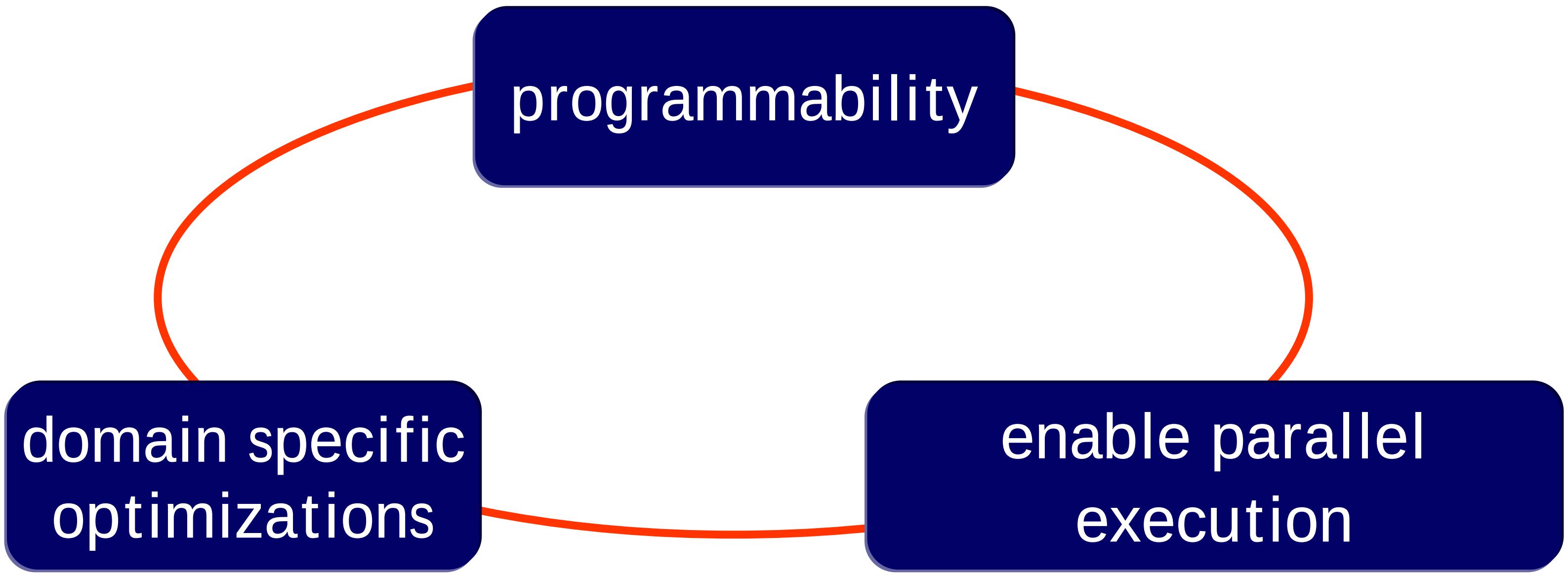
Parallel Programmer’s Dilemma

$$F(u,v)=\frac{2}{N}C(u)C(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N}$$



Compiler-Aware Language Design

boost productivity, enable faster development and rapid prototyping

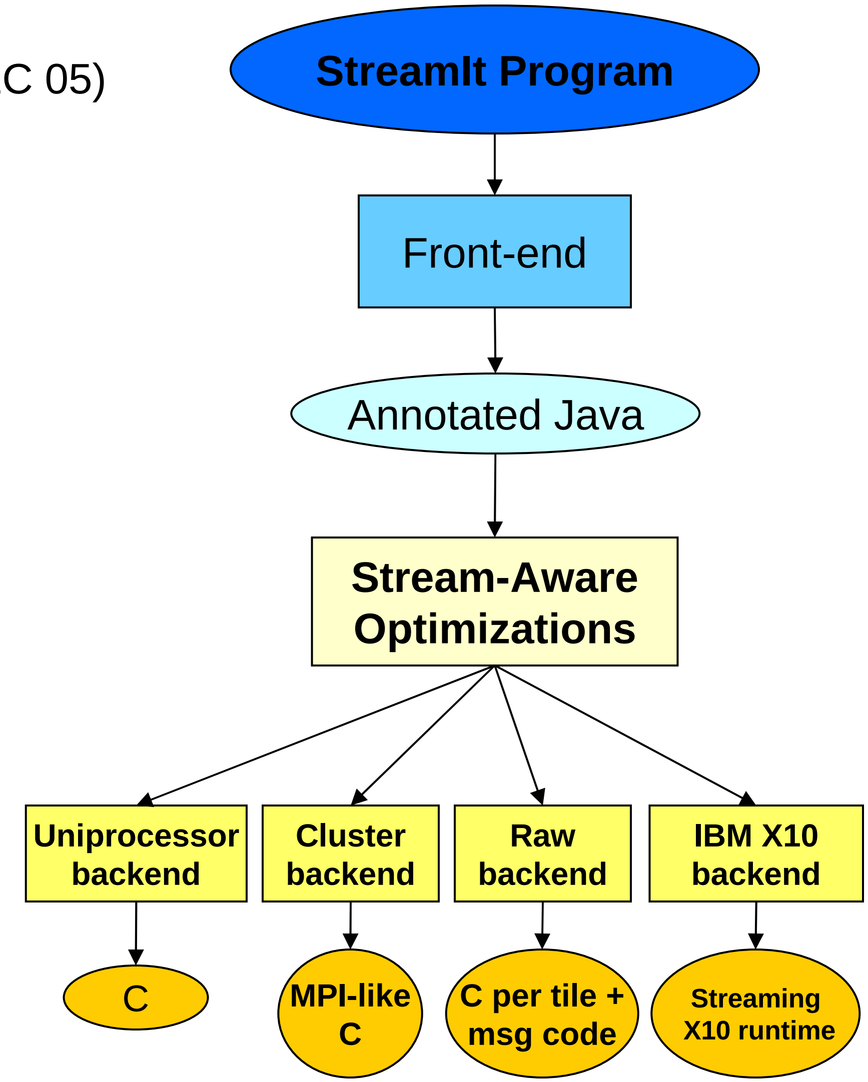


simple and effective optimizations for domain specific abstractions

target tiled architectures, clusters, DSPs, multicores, graphics processors, …

StreamIt Project

- **Language Semantics / Programmability**
  - StreamIt Language (CC 02)
  - Programming Environment in Eclipse (P-PHEC 05)
- **Optimizations / Code Generation**
  - Phased Scheduling (LCTES 03)
  - Cache Aware Optimization (LCTES 05)
- **Domain Specific Optimizations**
  - Linear Analysis and Optimization (PLDI 03)
  - Optimizations for bit streaming (PLDI 05)
  - Linear State Space Analysis (CASES 05)
- **Parallelism**
  - Teleport Messaging (PPOPP 05)
  - Compiling for Communication-Exposed Architectures (ASPLOS 02)
  - Load-Balanced Rendering (Graphics Hardware 05)
- **Applications**
  - SAR, DSP benchmarks, JPEG,
  - MPEG [IPDPS 06], DES and Serpent [PLDI 05], …



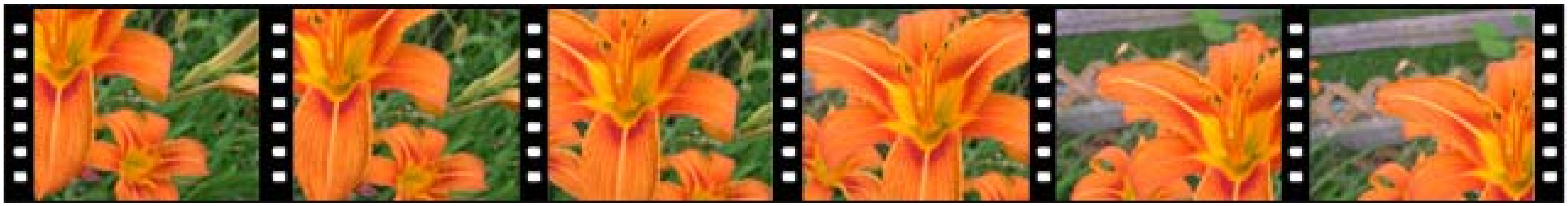
In This Talk

- MPEG-2 Overview
- StreamIt Application Development : MPEG-2 Decoding
- Natural expression of
  - Program structure
  - Parallelism
  - Data distribution
- Emphasis on programmability
  - Comparison/Contrast with C

Video Compression Algorithms

- Commonly used
- Order of magnitude reduction in data needed for representation
- Decreases storage requirements
- Internet and wireless transmission feasible

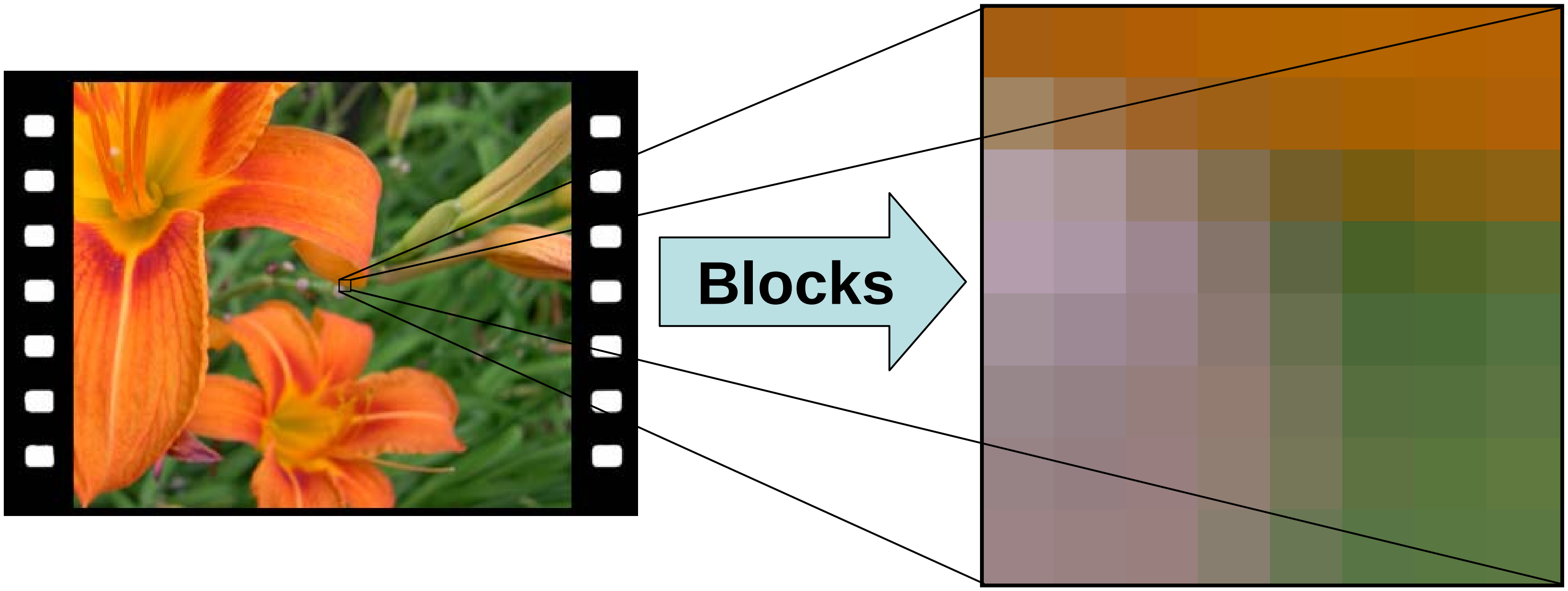
MPEG-2 Overview



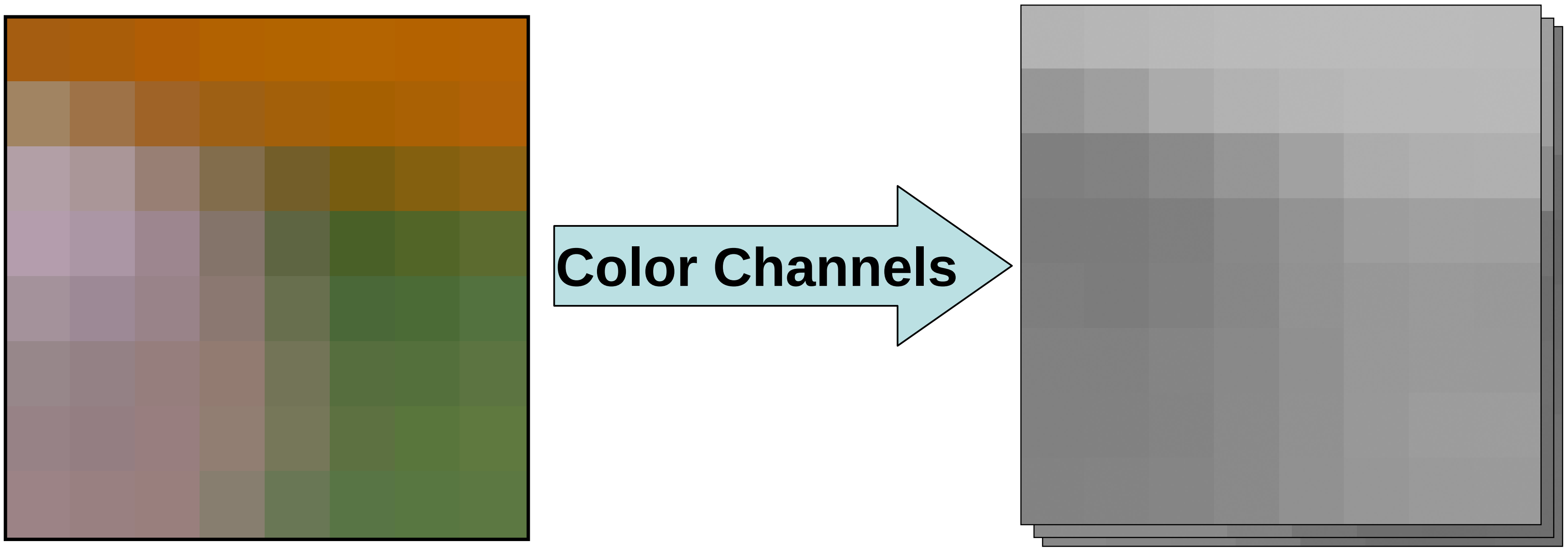
- Temporal compression eliminates redundancies between pictures
- Spatial compression eliminates data within a picture based on a human perception model

MPEG-2 Temporal Compression

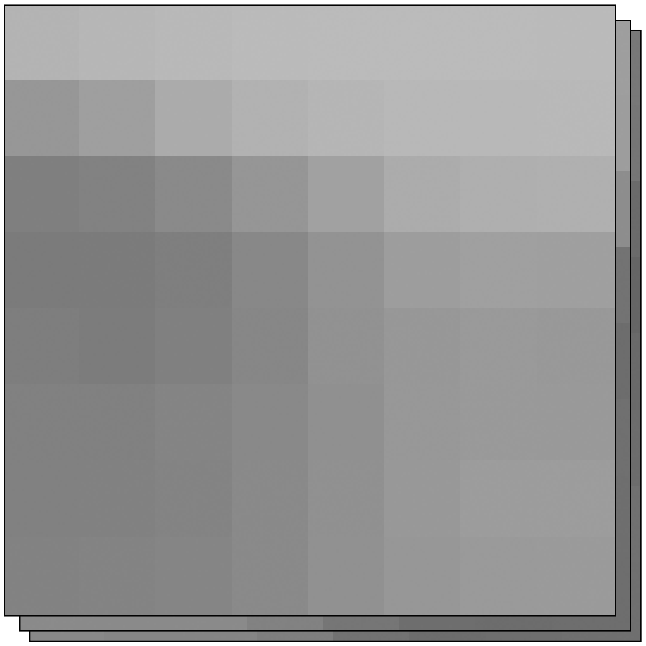
MPEG-2 Spatial Compression

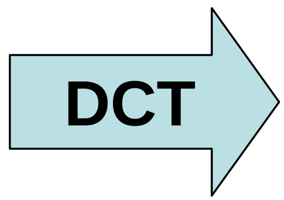


MPEG-2 Spatial Compression



MPEG-2 Spatial Compression





horizontal frequency

vertical frequency

50	-	-3	3	0	1	0	0
7 41	34 2	-3	-2	0	0	0	0
27	9	0	-2	0	-1	-1	-1
8	9	1	-1	0	0	0	0
-1	5	2	2	0	-1	0	1
-2	0	2	1	0	0	0	-1
-1	0	1	1	0	0	0	0
-1	-2	0	1	0	0	-1	0

MPEG-2 Spatial Compression

horizontal frequency →, vertical frequency ↓

50	-	-3	3	0	1	0	0
7 41	34 2	-3	-2	0	0	0	0
27	9	0	-2	0	-1	-1	-1
8	9	1	-1	0	0	0	0
-1	5	2	2	0	-1	0	1
-2	0	2	1	0	0	0	-1
-1	0	1	1	0	0	0	0
-1	-2	0	1	0	0	-1	0

Quantization →

horizontal frequency →, vertical frequency ↓

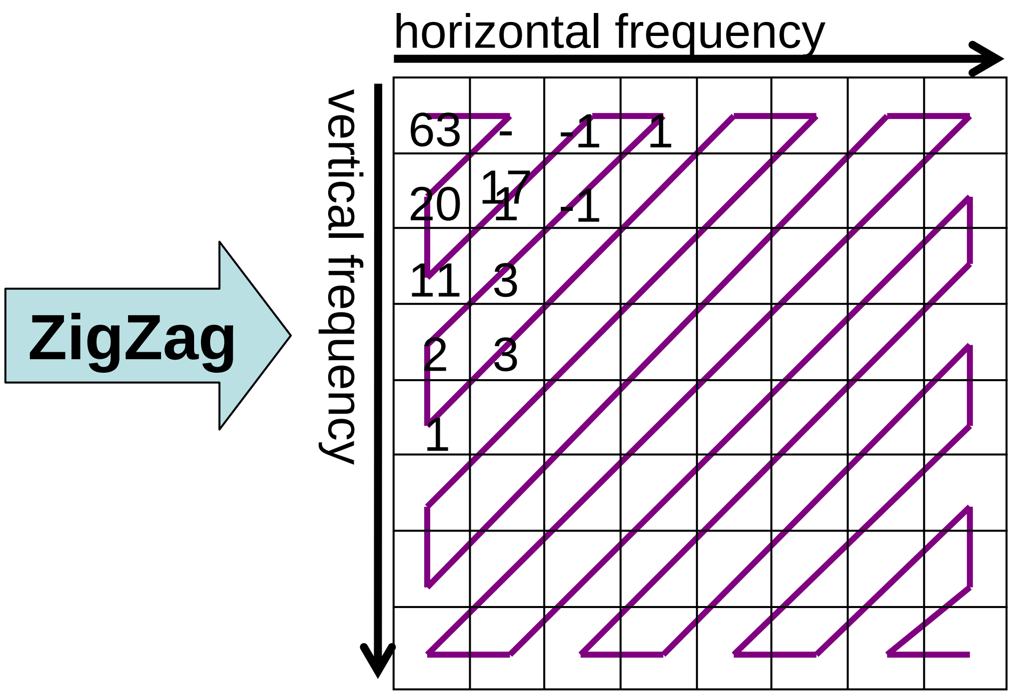
63	-	-1	1	0	0	0	0
20	17 1	-1	0	0	0	0	0
11	3	0	0	0	0	0	0
2	3	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

MPEG-2 Spatial Compression

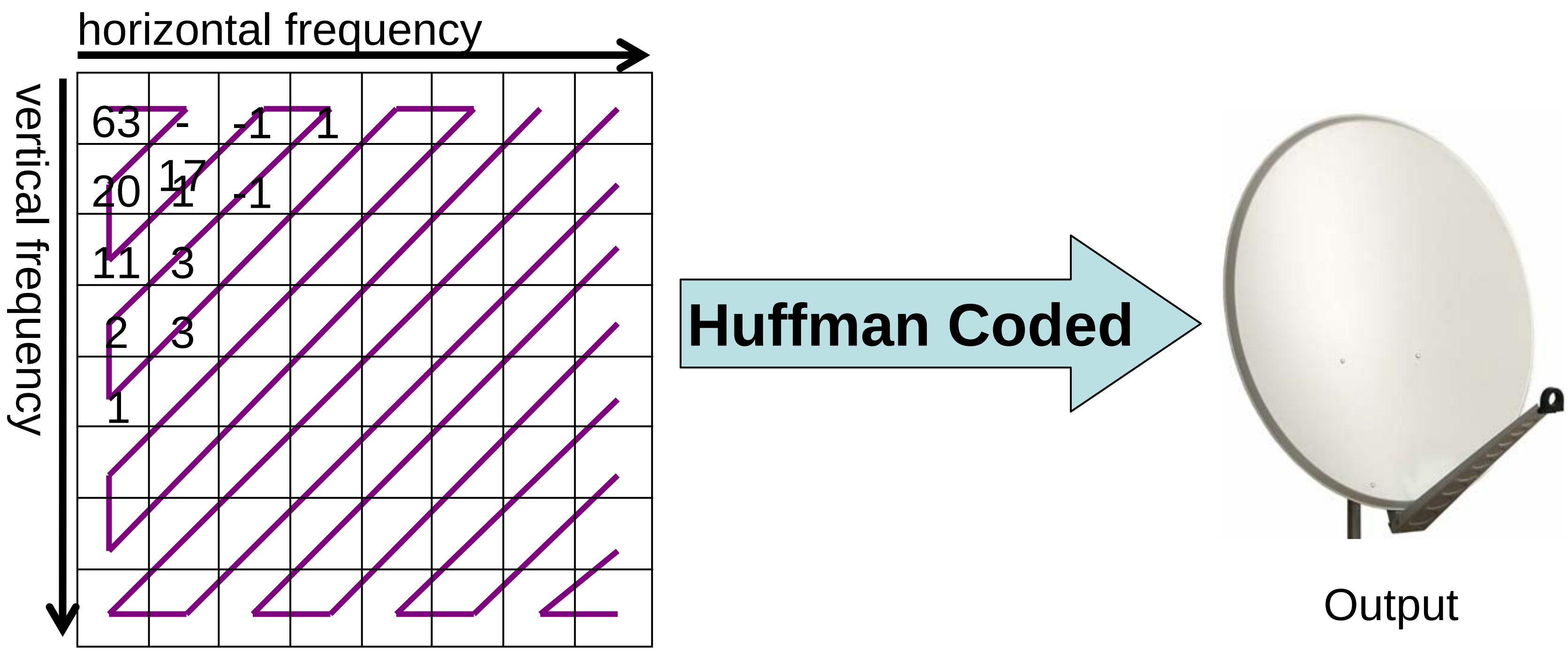
horizontal frequency

vertical frequency

63	-	-1	1	0	0	0	0
20	17 1	-1	0	0	0	0	0
11	3	0	0	0	0	0	0
2	3	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

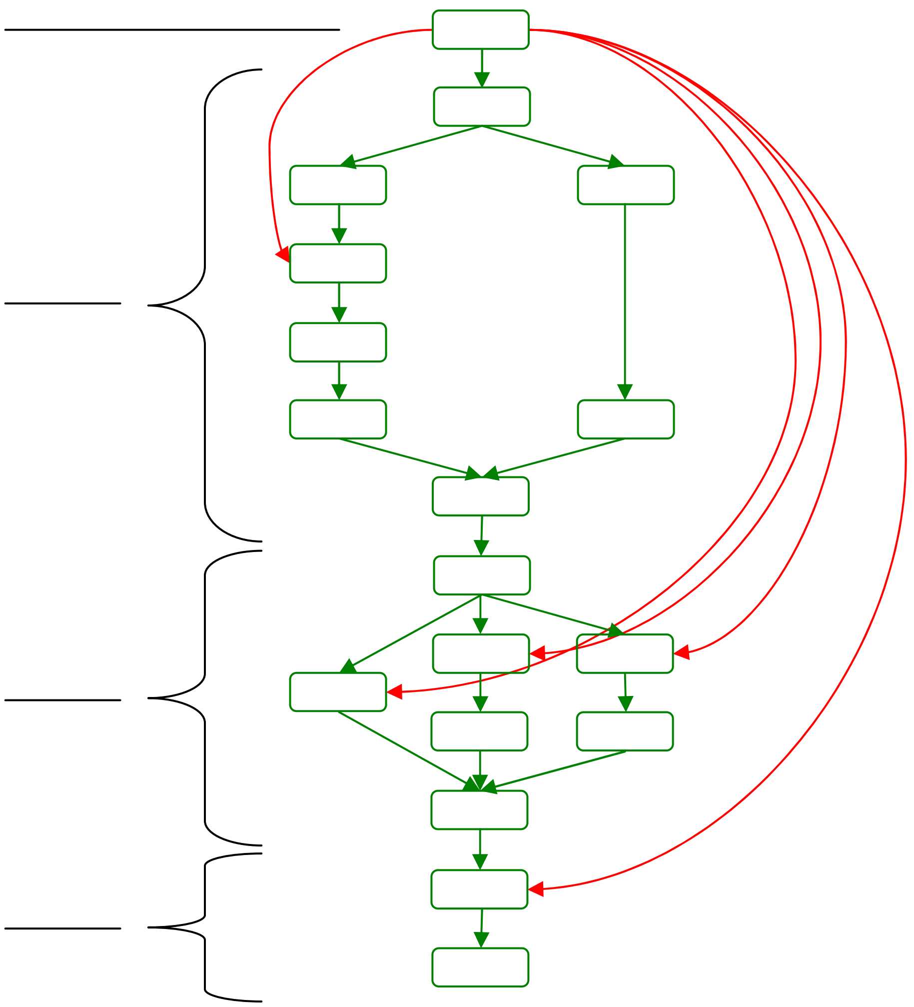


MPEG-2 Spatial Compression

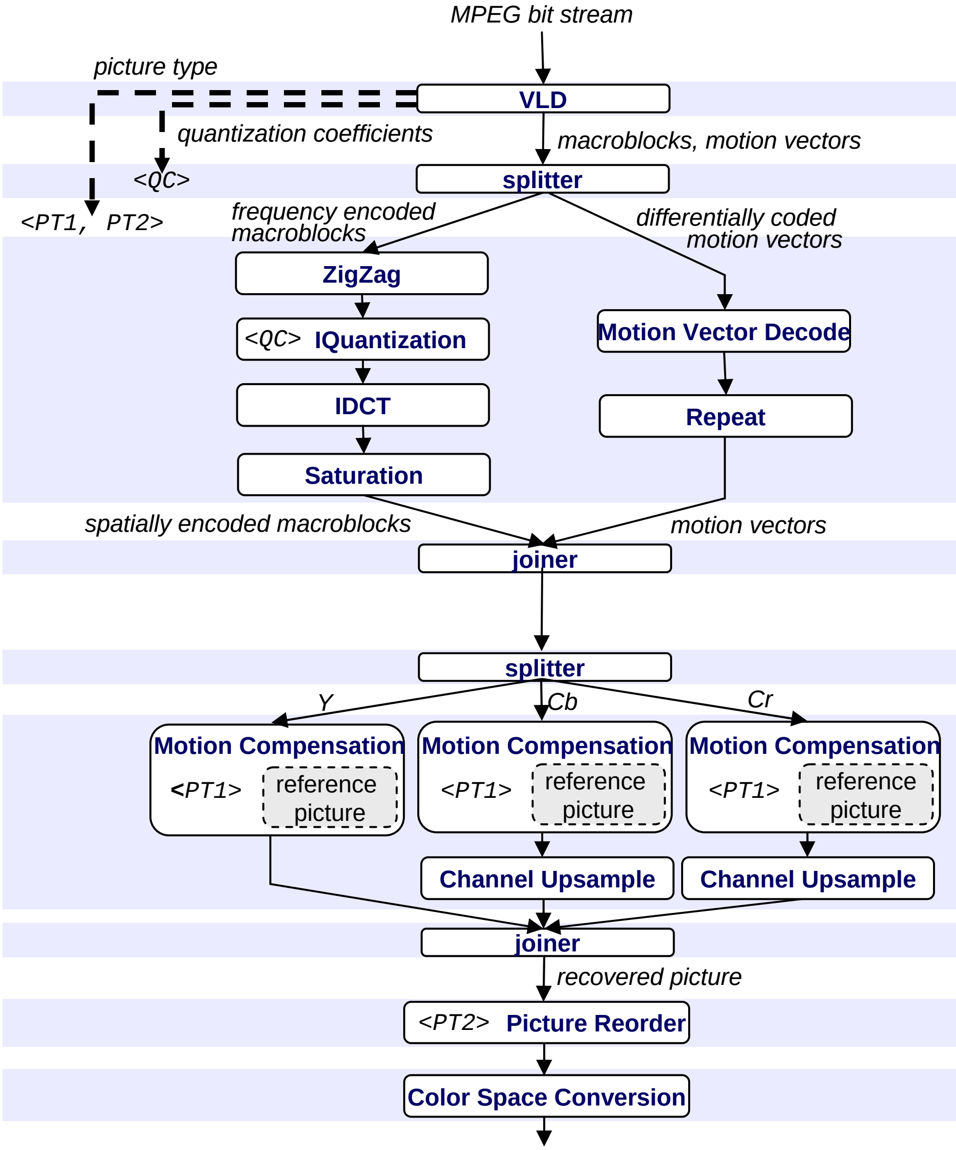


Stream Composition of MPEG-2 Decoder

- Variable length decoding
- Spatial decoding
  - block decoding in parallel with motion vector decoding
- Temporal decoding
  - all color channels motion compensated in parallel
- Color space conversion and data ordering

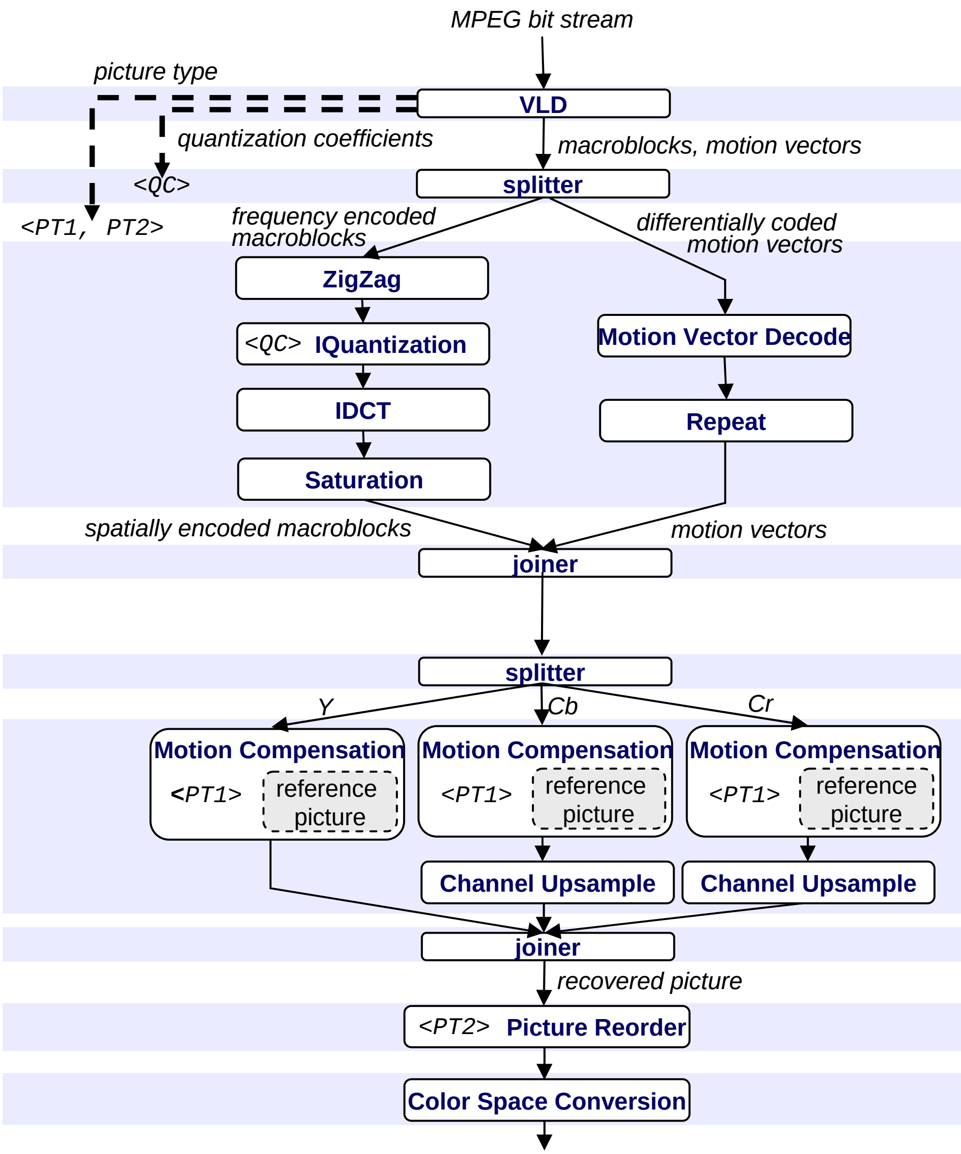


Application Design



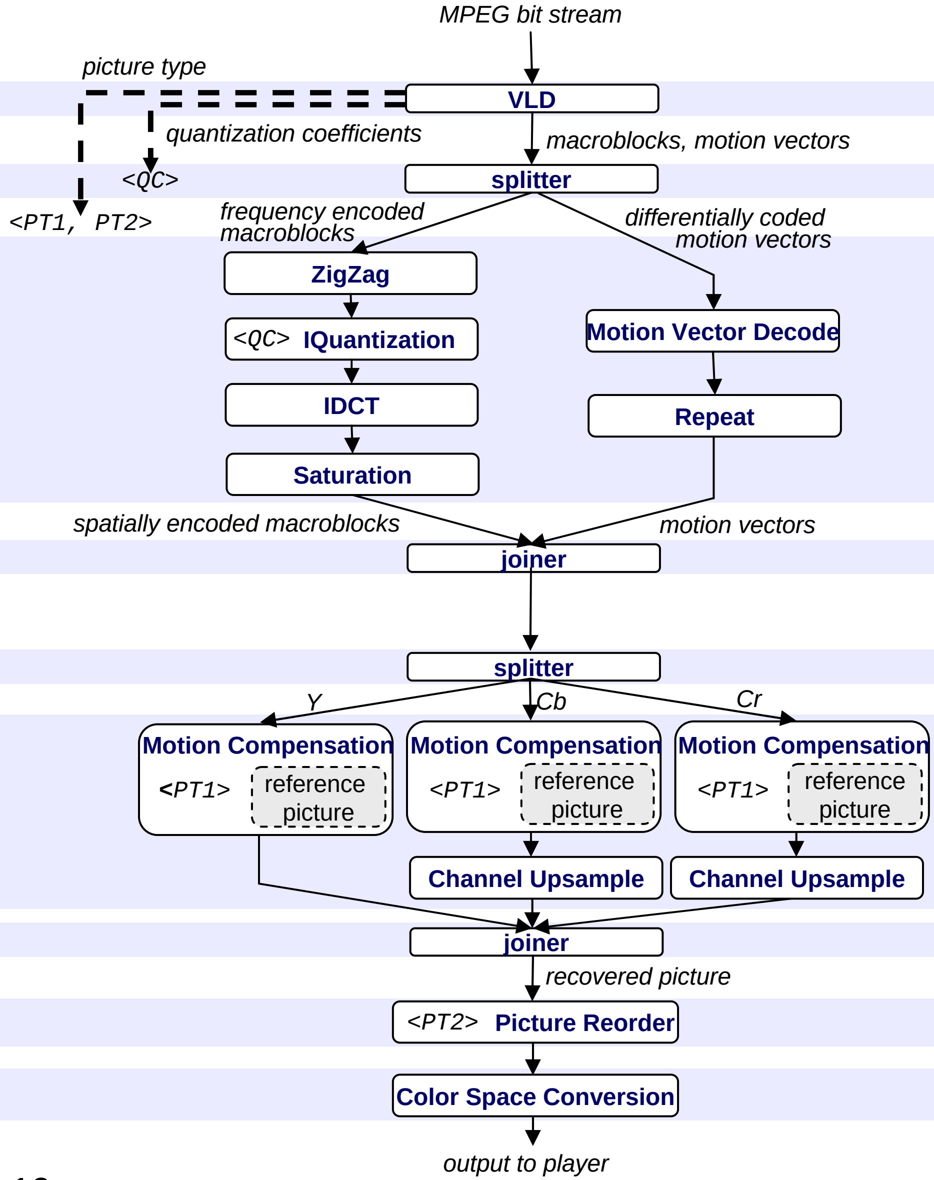
- Structured block level diagram describes computation and flow of data
- Conceptually easy to understand
  - Clean abstraction of functionality

StreamIt Philosophy



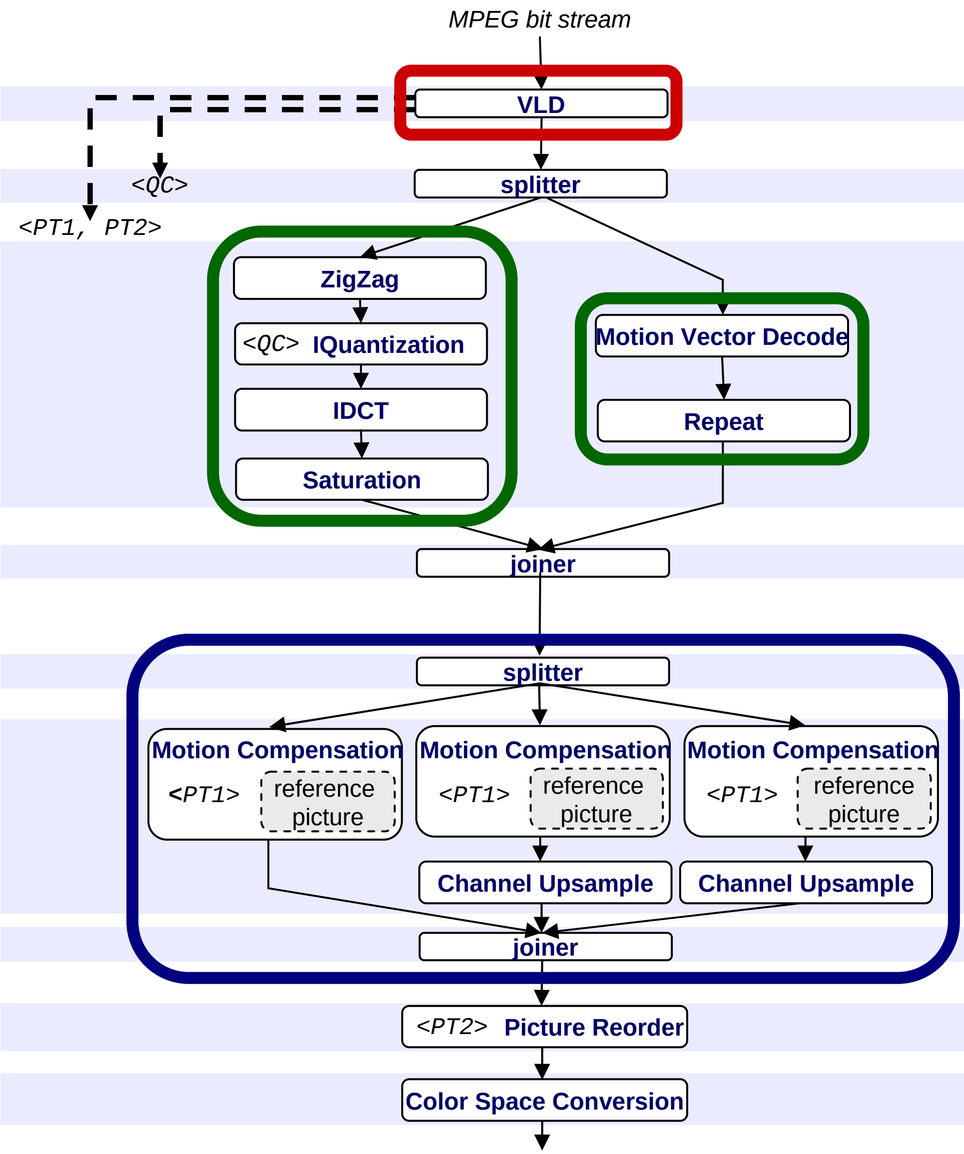
- Preserve program structure
  - Natural for application developers to express
- Leverage program structure to discover parallelism and deliver high performance
- Programs remain clean
  - Portable and malleable

StreamIt Philosophy




```
add VLD(QC, PT1, PT2);
add splitjoin {
    split roundrobin(N*B, V);

    add pipeline {
        add ZigZag(B);
        add IQuantization(B) to QC;
        add IDCT(B);
        add Saturation(B);
    }
    add pipeline {
        add MotionVectorDecode();
        add Repeat(V, N);
    }

    join roundrobin(B, V);
}
add splitjoin {
    split roundrobin(4*(B+V), B+V, B+V);

    add MotionCompensation(4*(B+V)) to PT1;
    for (int i = 0; i < 2; i++) {
        add pipeline {
            add MotionCompensation(B+V) to PT1;
            add ChannelUpsample(B);
        }
    }

    join roundrobin(1, 1, 1);
}
add PictureReorder(3*W*H) to PT2;

add ColorSpaceConversion(3*W*H);
```

Stream Abstractions in StreamIt




```
add VLD(QC, PT1, PT2);
add splitjoin {
    split roundrobin(N*B, V);

    add pipeline {
        add ZigZag(B);
        add IQuantization(B) to QC;
        add IDCT(B);
        add Saturation(B);
    }
    add pipeline {
        add MotionVectorDecode();
        add Repeat(V, N);
    }
    join roundrobin(B, V);
}
add splitjoin {
    split roundrobin(4*(B+V), B+V, B+V);

    add MotionCompensation(4*(B+V)) to PT1;
    for (int i = 0; i < 2; i++) {
        add pipeline {
            add MotionCompensation(B+V) to PT1;
            add ChannelUpsample(B);
        }
    }

    join roundrobin(1, 1, 1);
}
add PictureReorder(3*W*H) to PT2;

add ColorSpaceConversion(3*W*H);
```

filters

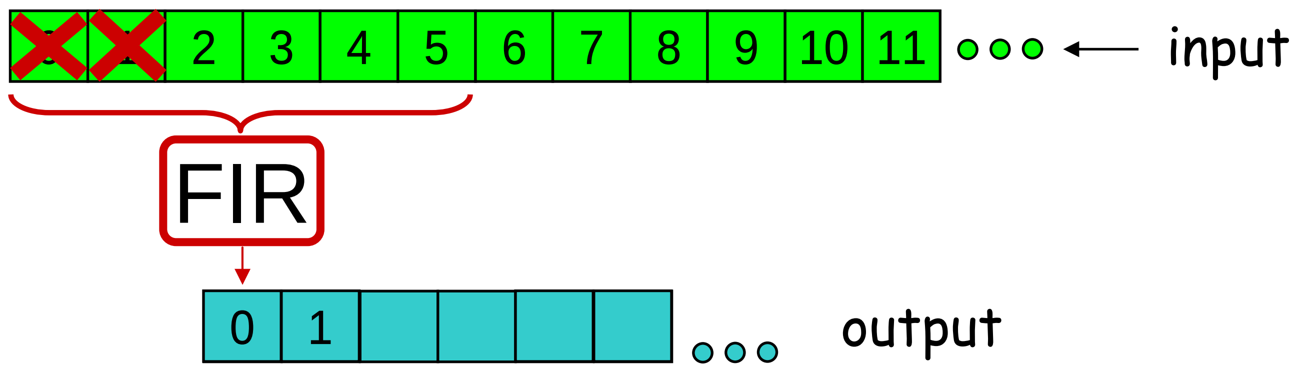
pipelines

splitjoins

StreamIt Language Highlights

- Filters
- Pipelines
- Splitjoins
- Teleport messaging

Example StreamIt Filter

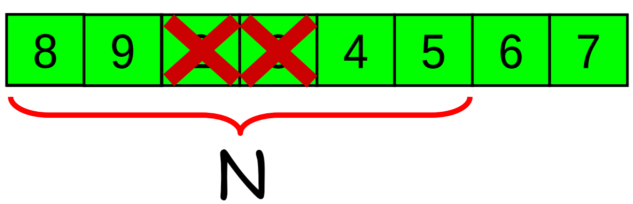



```
float→float filter FIR (int N) {

    work push 1 pop 1 peek N {
        float result = 0;
        for (int i = 0; i < N; i++) {
            result += weights[i] * peek(i);
        }
        push(result);
        pop();
    }
}
```

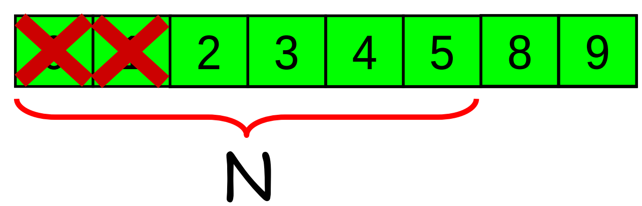
Compiler Managed Buffers

- Sliding window computation is very common in multimedia and scientific codes
- There are various implementation strategies for managing peek buffers

Circular Buffer:



Copy-Shift:



- Compiler recognizes peek buffers and chooses best implementation strategy for an architecture

FIR Filter in C

```
void FIR(
  int* src,
  int* dest,
  int* srcIndex,
  int* destIndex,
  int srcBufferSize,
  int destBufferSize,
  int N) {

  float result = 0.0;
  for (int i = 0; i < N; i++)
  result += weights[i] * src[(*srcIndex + i) % srcBufferSize];
  dest[*destIndex] = result;
  *srcIndex = (*srcIndex + 1) % srcBufferSize;
  *destIndex = (*destIndex + 1) % destBufferSize;
}
```

- FIR functionality obscured by buffer management details
- Programmer must commit to a particular buffer implementation strategy

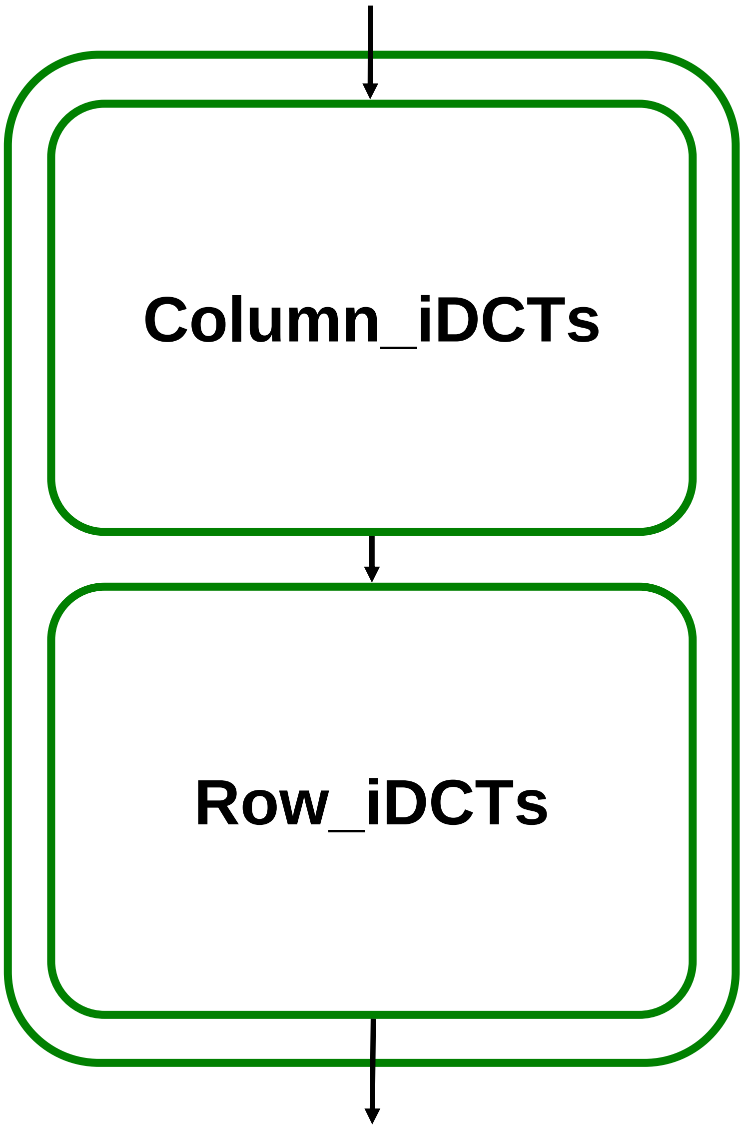
StreamIt Language Highlights

- Filters
- Pipelines
- Splitjoins
- Teleport messaging

Example StreamIt Pipeline

- Pipeline
  - Connect components in sequence
  - Expose pipeline parallelism

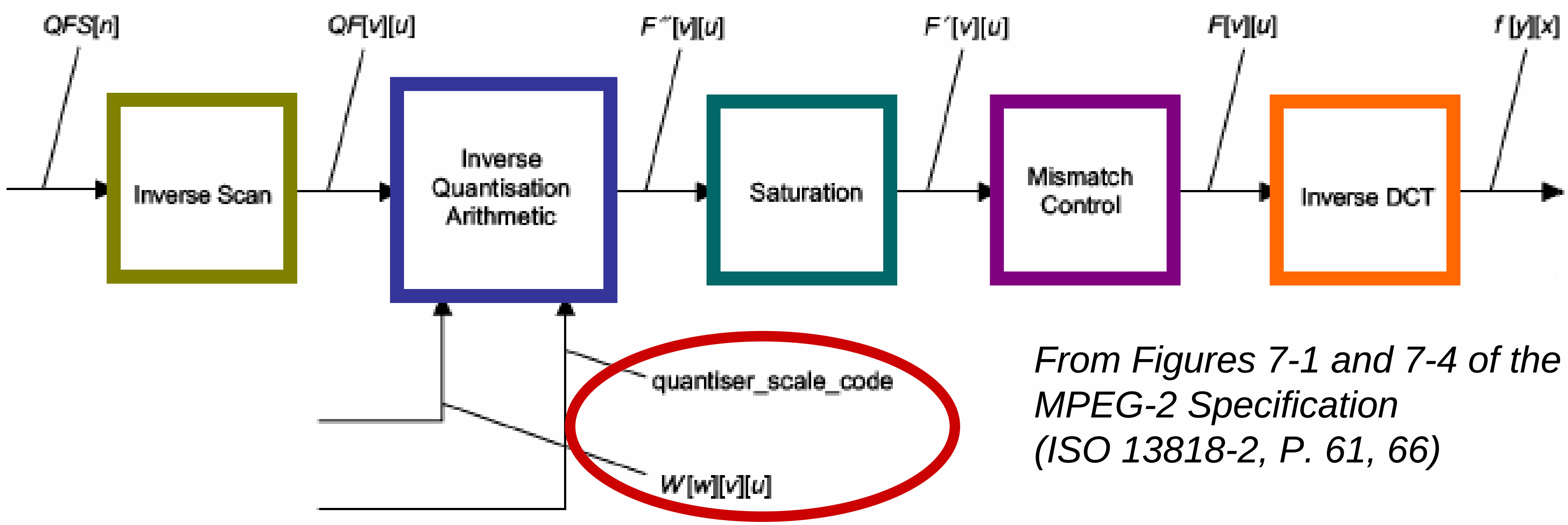
```
float→float pipeline 2D_iDCT (int N)
{

        add Column_iDCTs(N);
        add Row_iDCTs(N);

}
```



Preserving Program Structure

```
int->int pipeline BlockDecode(
         portal<InverseQuantisation> quantiserData,
         portal<MacroblockType> macroblockType) {
    add ZigZagUnordering();
    add InverseQuantization() to quantiserData, macroblockType;
    add Saturation(-2048, 2047);
    add MismatchControl();
    add 2D_iDCT(8);
    add Saturation(-256, 255);
}
```

Can be reused for JPEG decoding



From Figures 7-1 and 7-4 of the MPEG-2 Specification (ISO 13818-2, P. 61, 66)

exchange of control-relevant information

In Contrast: C Code Excerpt

```
EXTERN unsigned char *backward_reference_frame[3];
EXTERN unsigned char *forward_reference_frame[3];
EXTERN unsigned char *current_frame[3];
...etc...
```

```
Decode_Picture {
  for (;;) {
    parser()
    for (;;) {
      decode_macroblock();
      motion_compensation();
      if (condition)
        then break;
    }
  }
  frame_reorder();
}
```

```
decode_macroblock() {
  parser();
  motion_vectors();
  for (comp=0;comp<block_count;comp++) {
    parser();
    Decode_MPEG2_Block();
  }
}
```

```
motion_vectors() {
  parser();
  decode_motion_vector
  parser();
}
```

```
motion_compensation() {
  for (channel=0;channel<3;channel++)
    form_component_prediction();
  for (comp=0;comp<block_count;comp++) {
    Saturate();
    IDCT();
    Add_Block();
  }
}
```

```
Decode_MPEG2_Block() {
  for (int i = 0;; i++) {
    parsing();
    ZigZagUnordering();
    inverseQuantization();
    if (condition) then
      break;
  }
}
```

- Explicit for-loops iterate through picture frames
- Frames passed through global arrays, handled with pointers
- Mixing of parser, motion compensation, and spatial decoding

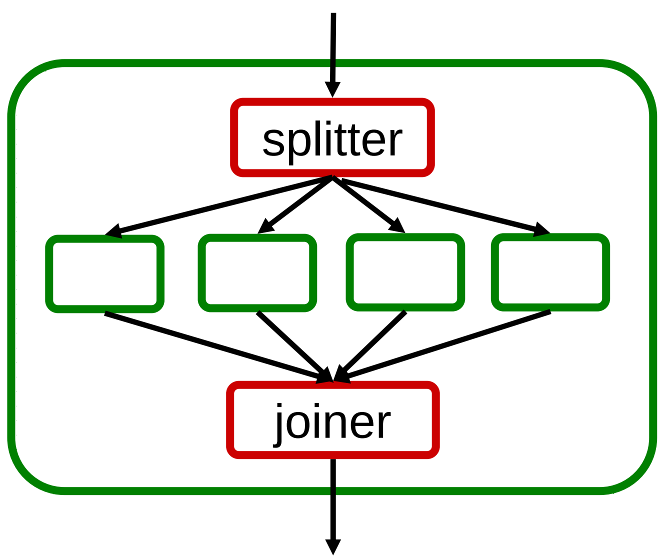
StreamIt Language Highlights

- Filters
- Pipelines
- Splitjoins
- Teleport messaging

Example StreamIt Splitjoin

- Splitjoin
  - Connect components in parallel
  - Expose data parallelism and data distribution

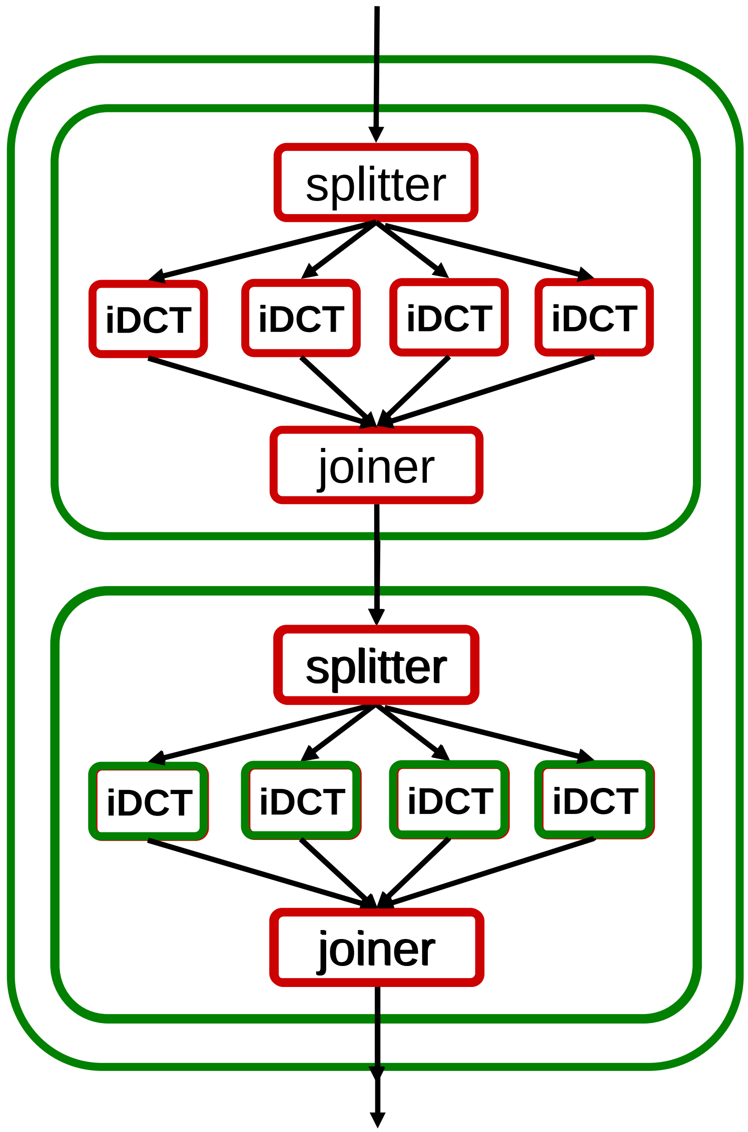
```
float→float splitjoin Row_iDCT (int N)
{
    split roundrobin(N);
    for (int i = 0; i < N; i++) {
        add 1D_iDCT(N);
    }
    join roundrobin(N);
}
```



Example StreamIt Splitjoin

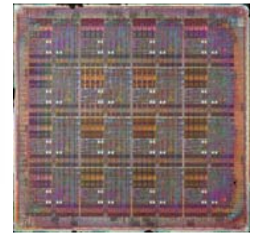
- Splitjoin
  - Connect components in parallel
  - Expose data parallelism and data distribution

```
float→float splitjoin Row_iDCT (int N)
{
    split roundrobin(N);
        for (int i = 0; i < N; i++) {
            add 1D_iDCT(N);
        }
        join roundrobin(N);
}
```

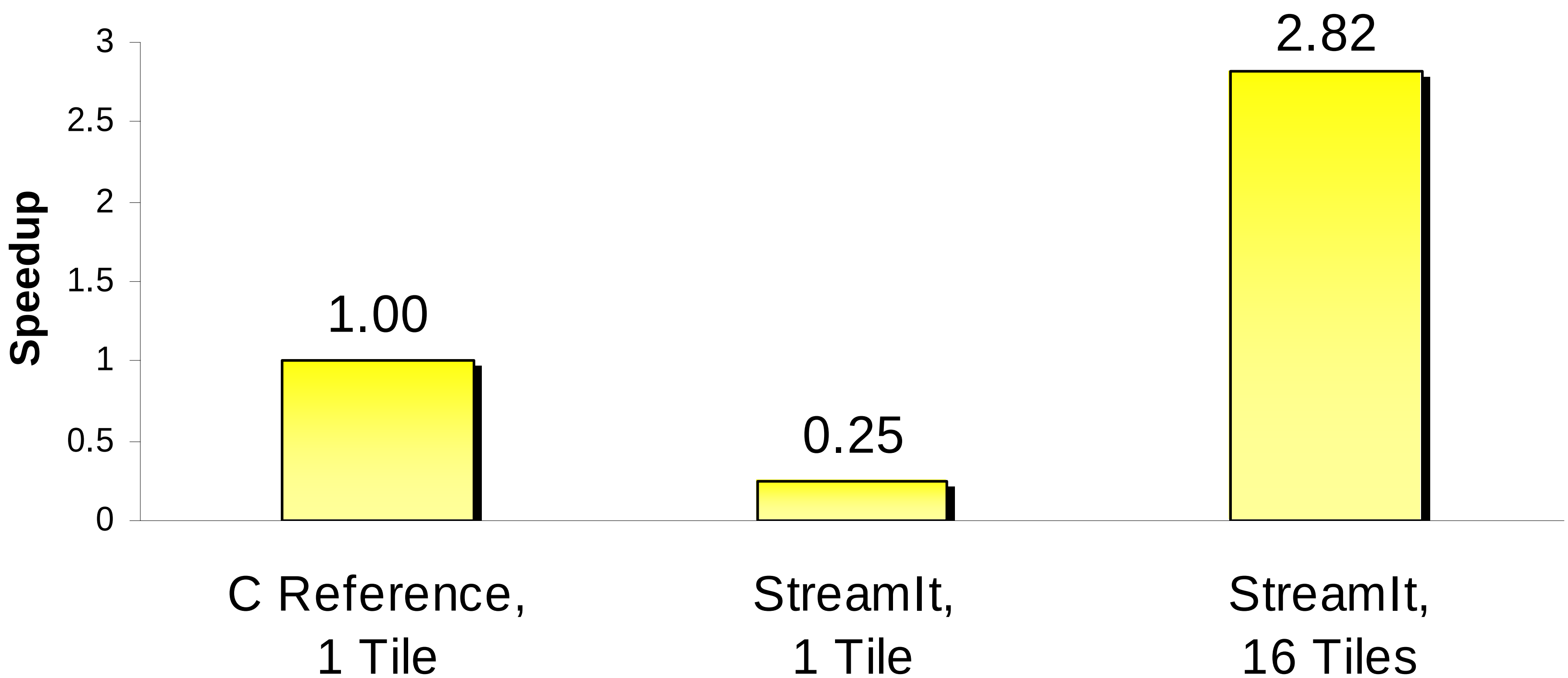


StreamIt Parallel Performance

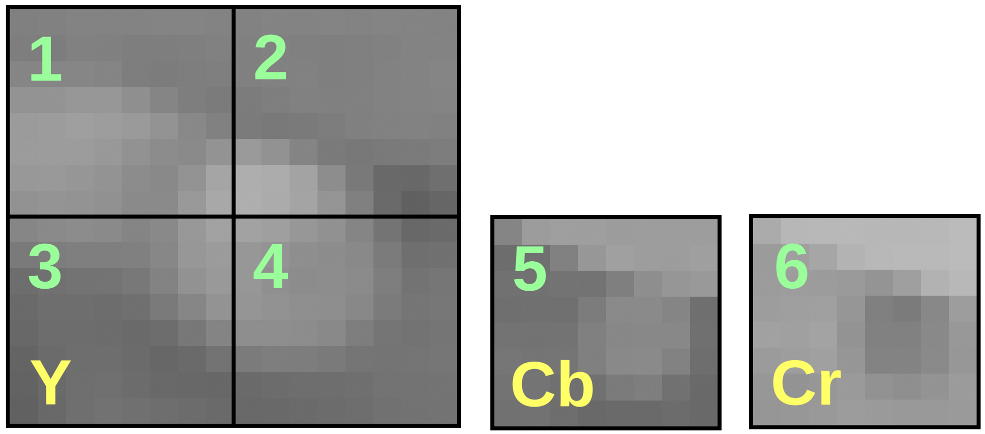
$$F(u,v) = \frac{2}{N} C(u) C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\frac{(2x+1)u\pi}{2N} \cos\frac{(2y+1)v\pi}{2N}$$



2D Discrete Cosine Transform on MIT Raw Architecture



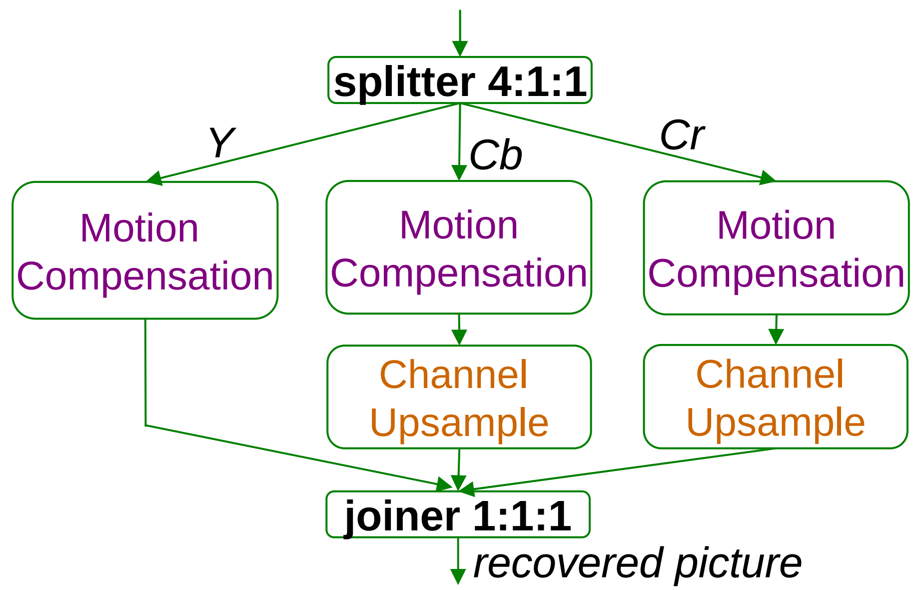
Naturally Expose Data Distribution



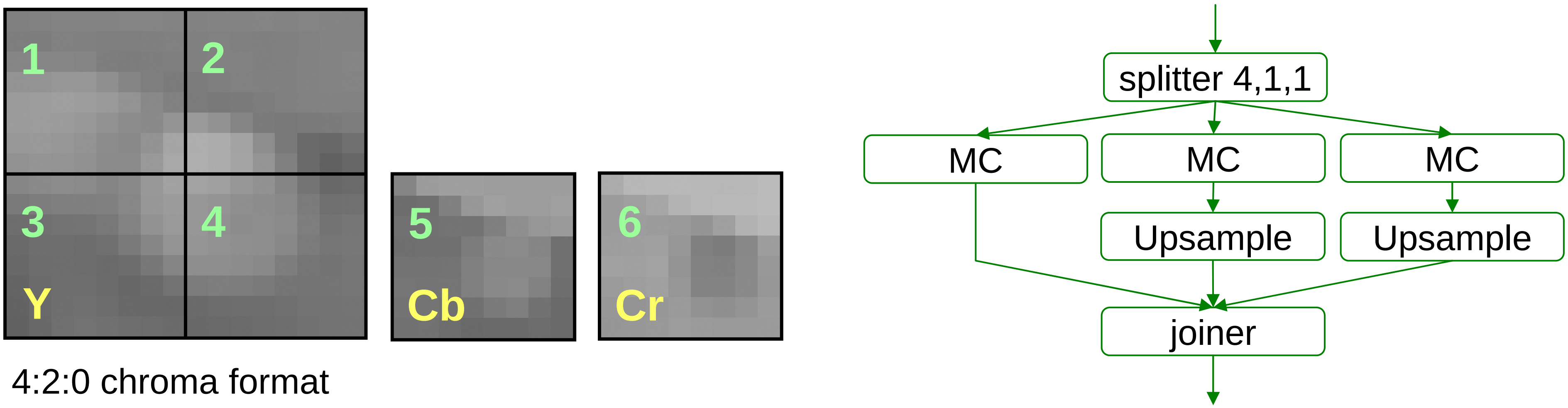
scatter macroblocks according to chroma format

```
add splitjoin {
   split roundrobin(4*(B+V), B+V, B+V);

   add MotionCompensation();
   for (int i = 0; i < 2; i++) {
       add pipeline {
           add MotionCompensation();
           add ChannelUpsample(B);
       }
   }

   join roundrobin(1, 1, 1);
}
```

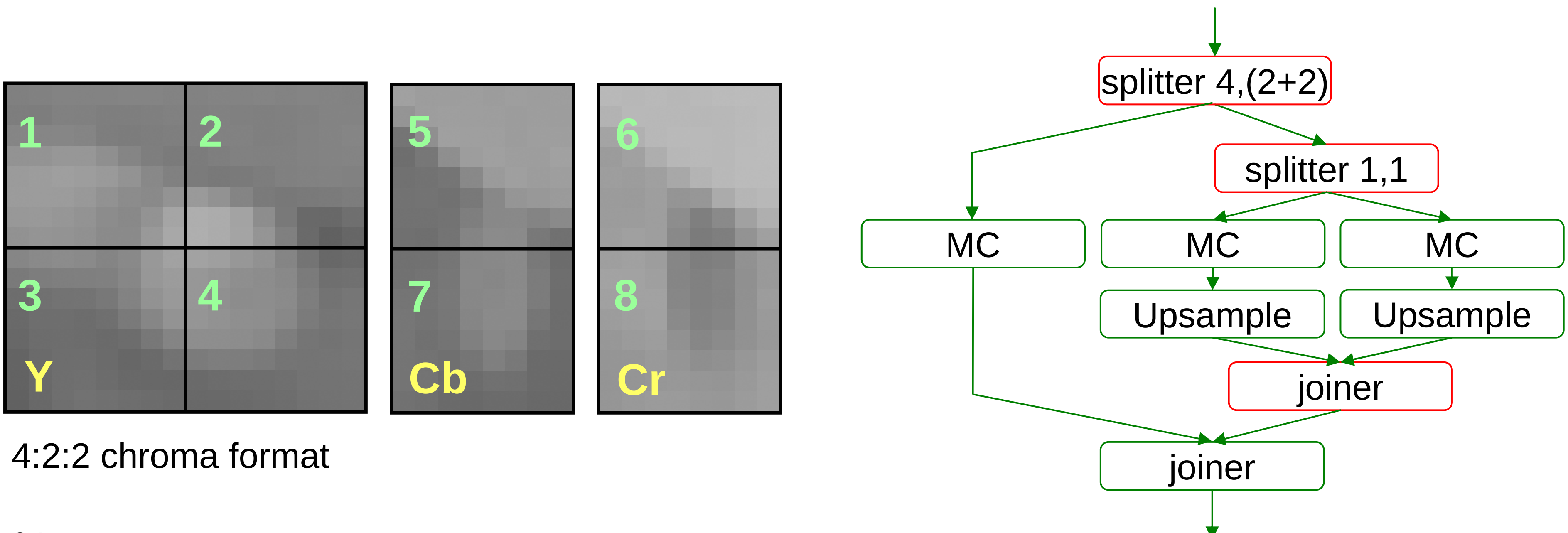
gather one pixel at a time



Stream Graph Malleability



4:2:0 chroma format

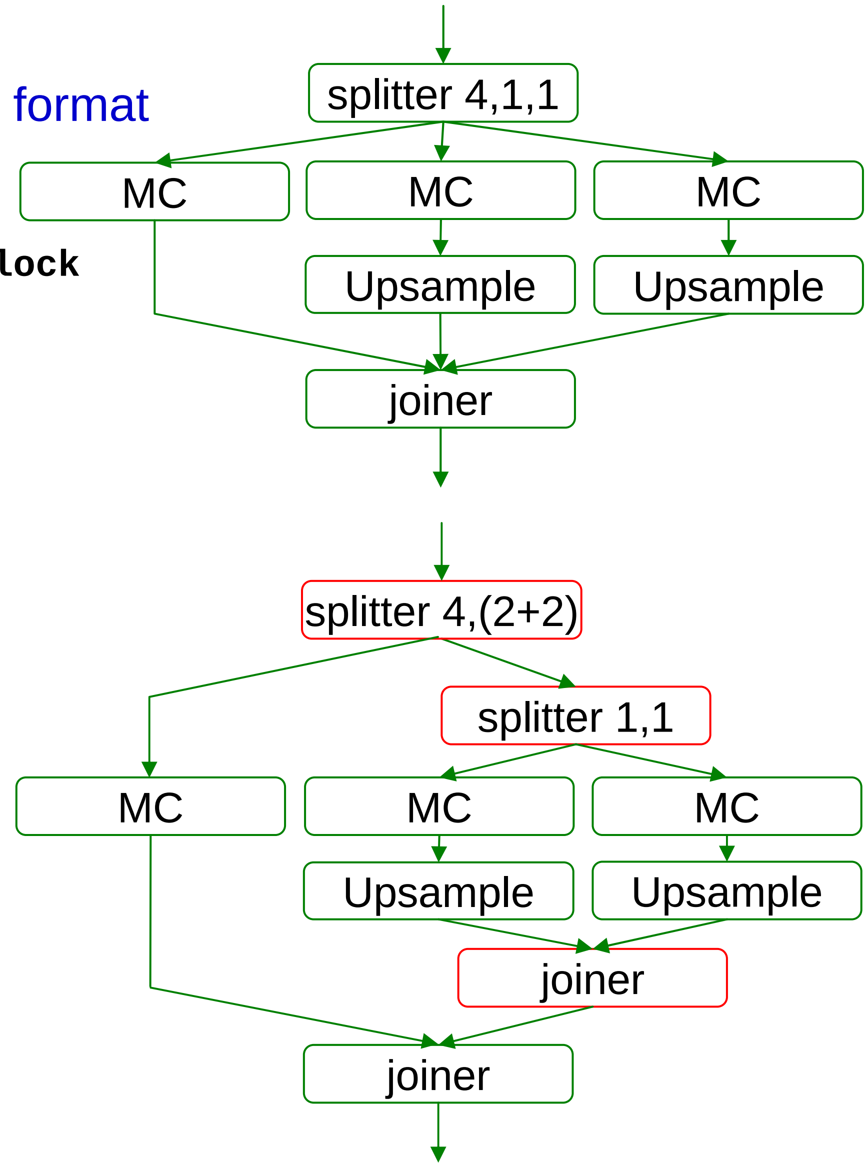


4:2:2 chroma format

StreamIt Code Sample

blue = code added or modified to support 4:2:2 format

```
// C = blocks per chroma channel per macroblock
// C = 1 for 4:2:0, C = 2 for 4:2:2
add splitjoin {
   split roundrobin(4*(B+V), 2*C*(B+V));

   add MotionCompensation();
   add splitjoin {
      split roundrobin(B+V, B+V);

      for (int i = 0; i < 2; i++) {
         add pipeline {
             add MotionCompensation()
             add ChannelUpsample(C,B);
         }
      }

      join roundrobin(1, 1);
   }

   join roundrobin(1, 1, 1);
}
```



In Contrast: C Code Excerpt

blue = pointers used for address calculations

```
/* Y */
form_component_prediction(src[0]+(sfield?lx2>>1:0),dst[0]+(dfield?lx2>>1:0),
                          lx,lx2,w,h,x,y,dx,dy,average_flag);

if (chroma_format!=CHROMA444)  {
   lx>>=1; lx2>>=1; w>>=1; x>>=1; dx/=2;
}
if (chroma_format==CHROMA420)  {
   h>>=1; y>>=1; dy/=2;
}

/* Cb */
form_component_prediction(src[1]+(sfield?lx2>>1:0),dst[1]+(dfield?lx2>>1:0),
                          lx,lx2,w,h,x,y,dx,dy,average_flag);

/* Cr */
form_component_prediction(src[2]+(sfield?lx2>>1:0),dst[2]+(dfield?lx2>>1:0),
                          lx,lx2,w,h,x,y,dx,dy,average_flag);
```

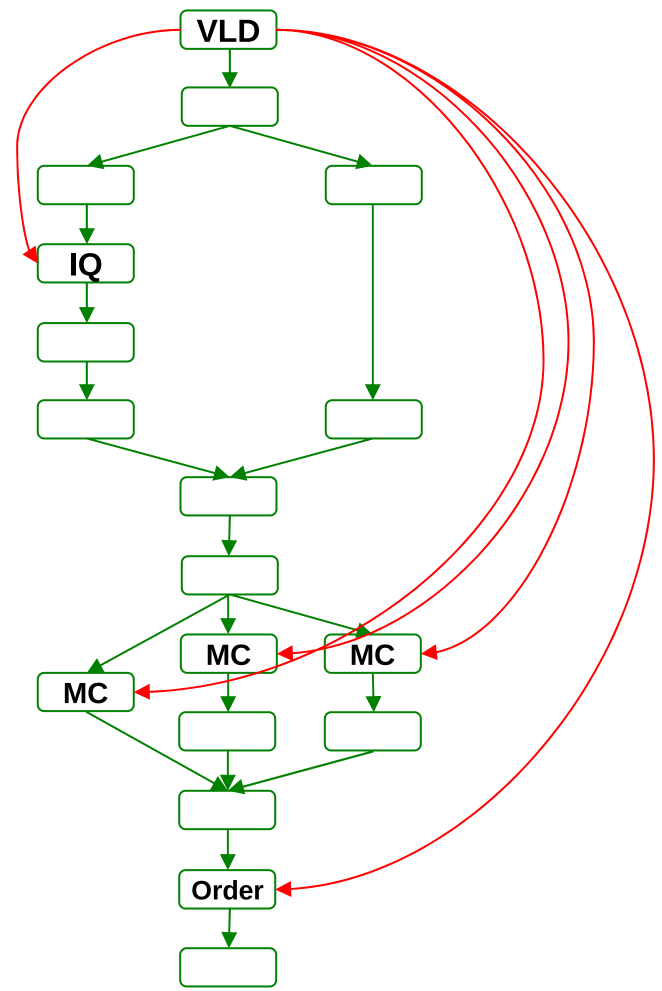
Adjust values used for address calculations depending on the chroma format used.

StreamIt Language Highlights

- Filters
- Pipelines
- Splitjoins
- Teleport messaging

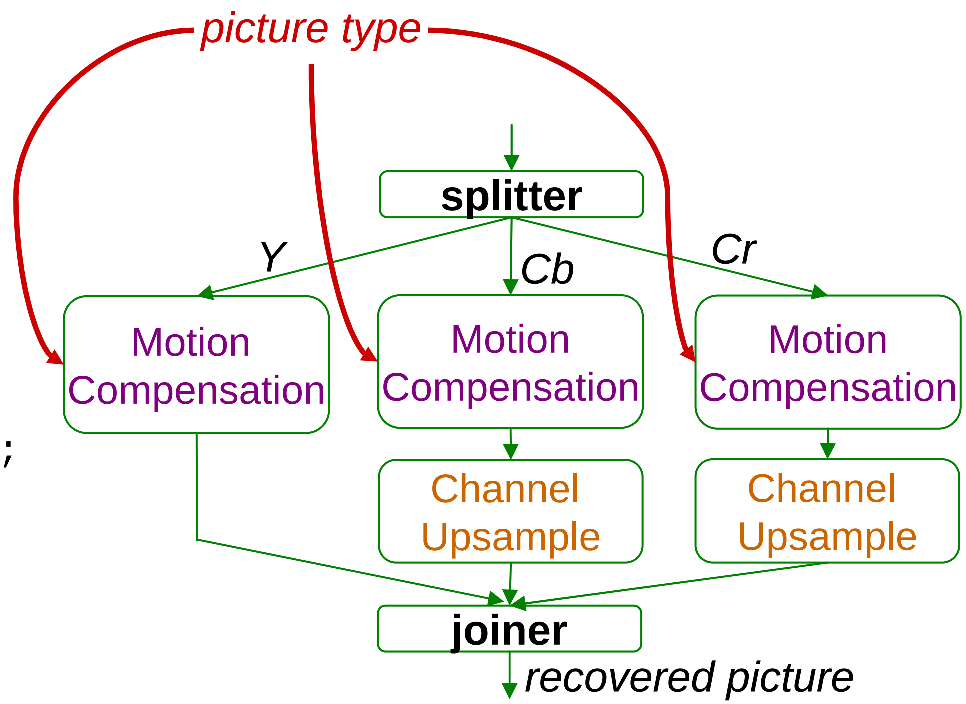
Teleport Messaging

- Avoids muddling data streams with control relevant information
- Localized interactions in large applications
  - A scalable alternative to global variables or excessive parameter passing



Motion Prediction and Messaging

```
portal<MotionCompensation> PT;

add splitjoin {
   split roundrobin(4*(B+V), B+V, B+V);

   add MotionCompensation() to PT;
   for (int i = 0; i < 2; i++) {
       add pipeline {
           add MotionCompensation() to PT;
           add ChannelUpsample(B);
       }
   }

   join roundrobin(1, 1, 1);
}
```

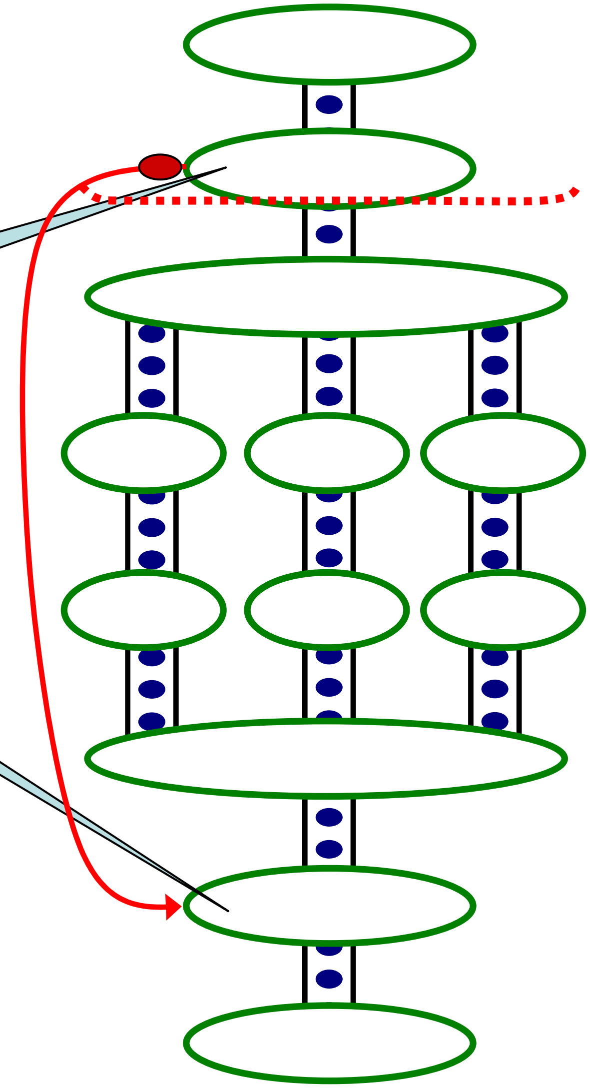


Teleport Messaging Overview

- Looks like method call, but timed relative to data in the stream

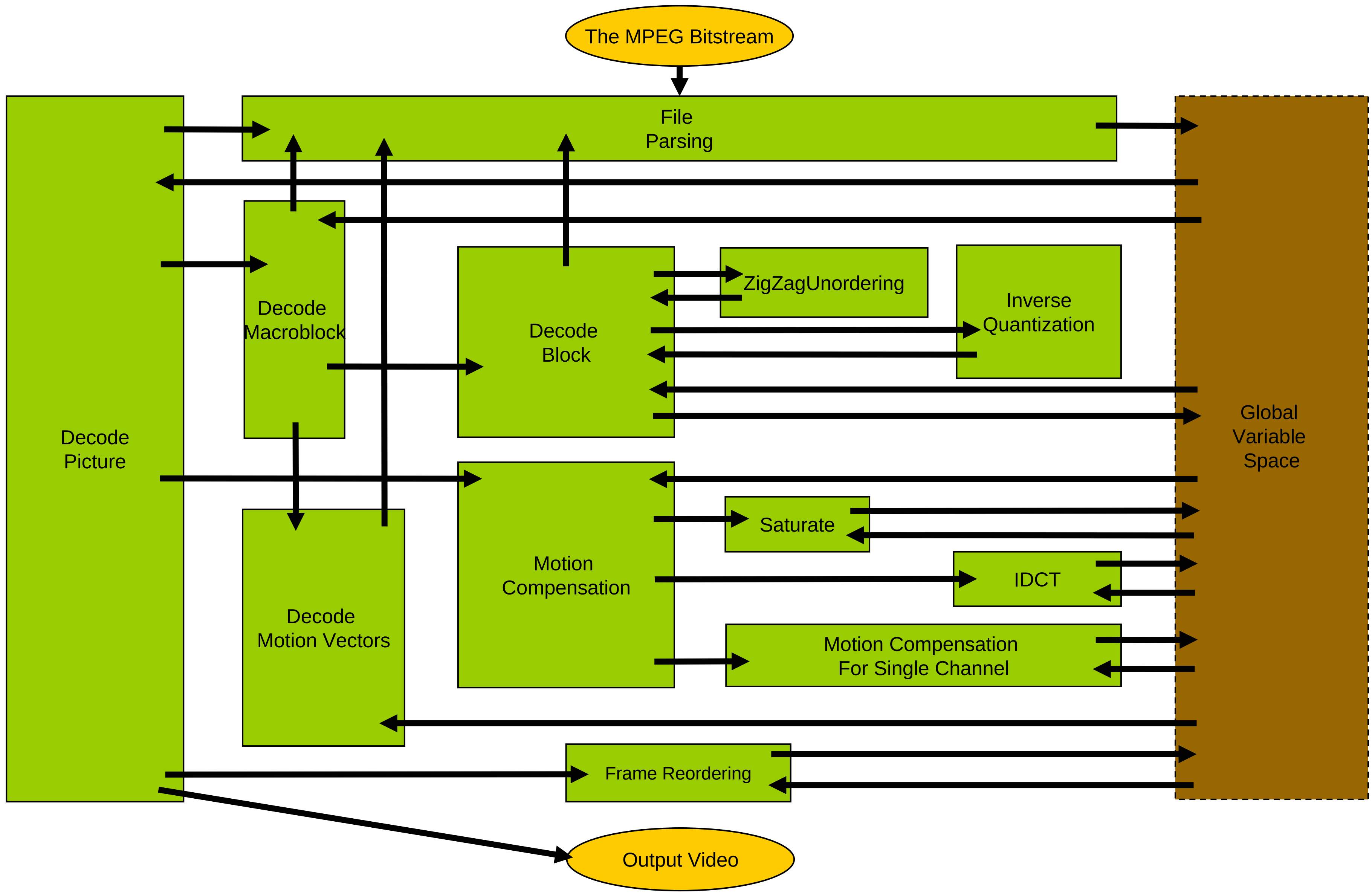
```
TargetFilter x;
if newPictureType(p) {
    x.setPictureType(p) @ 0;
}
```

```
void setPicturetype(int p) {
    reconfigure(p);
}
```



- Simple and precise for user
  - Exposes dependences to compiler
  - Adjustable latency
  - Can send upstream or downstream

Messaging Equivalent in C



Language Comparison: Programmer's Perspective

	C	StreamIt
Correctness and Performance	Mixed together	Separation of concerns
Buffer management	Programmer managed	Compiler managed
Scheduling	Programmer managed	Compiler managed

Language Comparison: Compiler's Perspective

	C	StreamIt
Memory Model	Global address space	Distributed (private) address spaces
Parallelism	Implicit	Explicit
Communication	Obscured	Exposed
Transformations	Limited	Global

Implementation

- Functional MPEG-2 decoder
  - Encoder recently completed
- Developed by 1 programmer in 8 weeks
- 2257 lines of code
  - Vs. 3477 lines of C code in MPEG-2 reference
- 48 static streams, 643 instantiated filters

Related Work

- Synchronous Dataflow and Extensions
  - *Synchronous Piggybacked Dataflow*
    - C. Park, J. Chung, S. Ha 1999
    - C. Park, J. Jung, S. Ha 2002
  - *Blocked Dataflow*
    - D.-I. Ko, S. S. Bhattacharyya 2005
  - *Hierarchical Dataflow*
    - S. Neuendorffer, E. Lee 2004
- Implementations
  - *MPEG2 Decoding and Encoding*
    - E. Iwata, K. Olukotun 1998
  - *Parallel MPEG4 Encoding*
    - I. Assayad, P. Gerner, S. Yovine, V. Bertin 2005
- Stream Oriented Languages
  - Esterel, Lustre, Signal, Lucid, Cg, Brook, Spindle, StreamC, Occam, Parallel Haskell, Sisal

Ongoing and Future Work

- MPEG-2 performance evaluation
- Inter-language interfaces
  - StreamIt to native C, and vice versa
- More applications

we want to hear from you!

Conclusions

- StreamIt language preserves program structure
  - Natural for programmers
- Parallelism and communication naturally exposed
  - Compiler managed buffers, and portable parallelization technology
- StreamIt increases programmer productivity, doesn't sacrifice performance

The End

StreamIt
http://cag.csail.mit.edu/streamit