HDTV in StreamIT:
A Case Study

Andrew A. Lamb
MIT
Laboratory for Computer Science
Computer Architecture Group
(8/2/2002)

Resources/Acknowledgements

- ATSC (Advanced Television Systems Committee) Standard A53, revision B (HDTV standard)
- Vanu PowerPoint presentation on their HDTV implementation

Outline

- **“Cultural” Overview**
- Reed-Solomon
- Convolutional Interleaving
- Trellis Coding/Viterbi Decoding

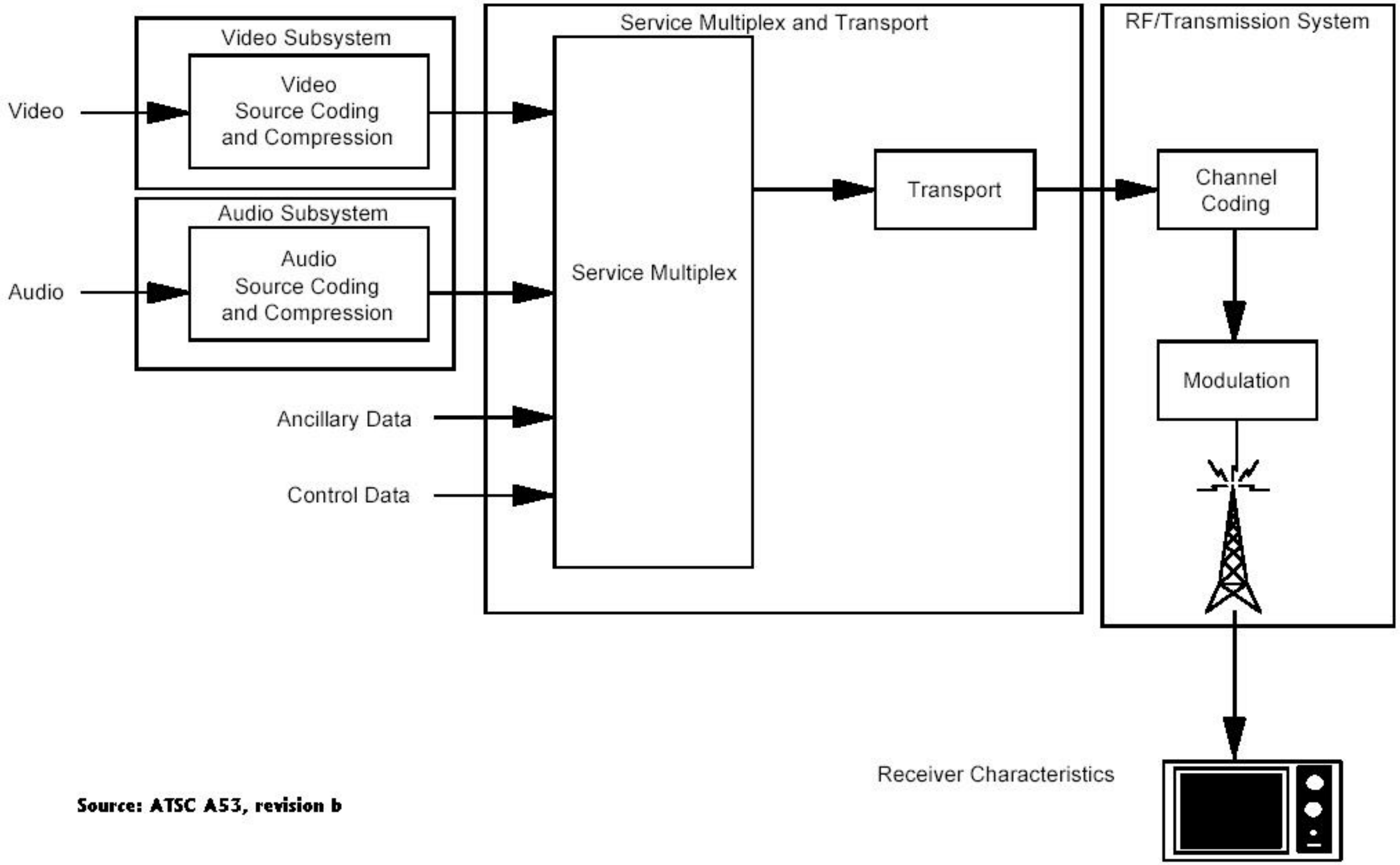
HDTV

- HDTV == ATSC A53 rev. B standard

- Spec was created by “Digital HDTV Grand Alliance”
  - AT&T (now Lucent Technologies), General Instrument, North American Philips, Massachusetts Institute of Technology, Thomson Consumer Electronics, the David Sarnoff Research Center (now Sarnoff Corporation) and Zenith Electronics Corporation

  - (http://www.atsc.org/history.html)

A53 Revision b



Source: ATSC A53, revision b

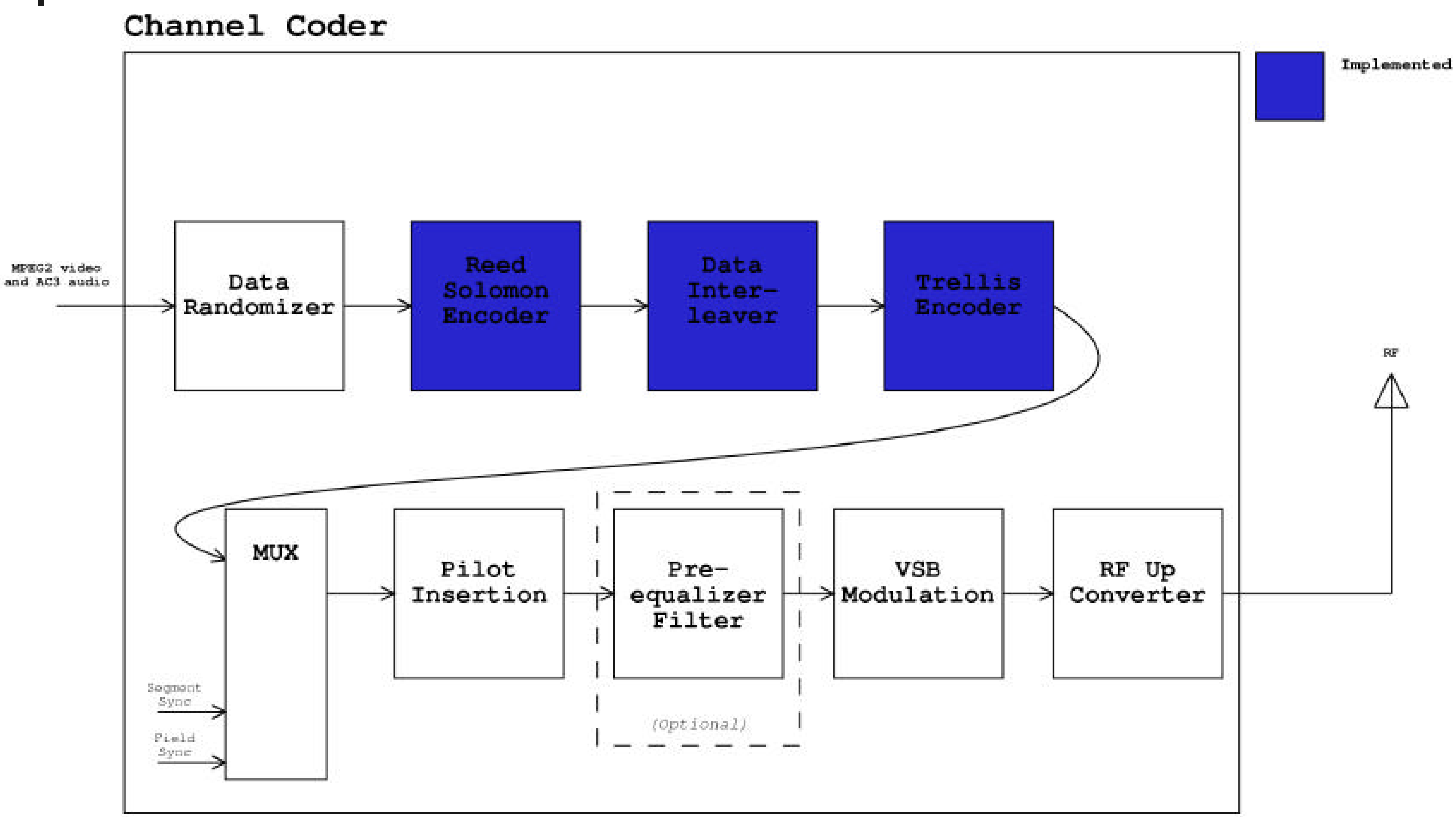
A53 Revision b (cont.)

- A53b describes service multiplexing and transport along with channel coding.
- ISO/IEC IS 13818-2, International Standard (1996) describes MPEG-2
- A52a describes AC3 encoding
- Doesn’t describe **decoding**.

A53 Revision b (cont.)

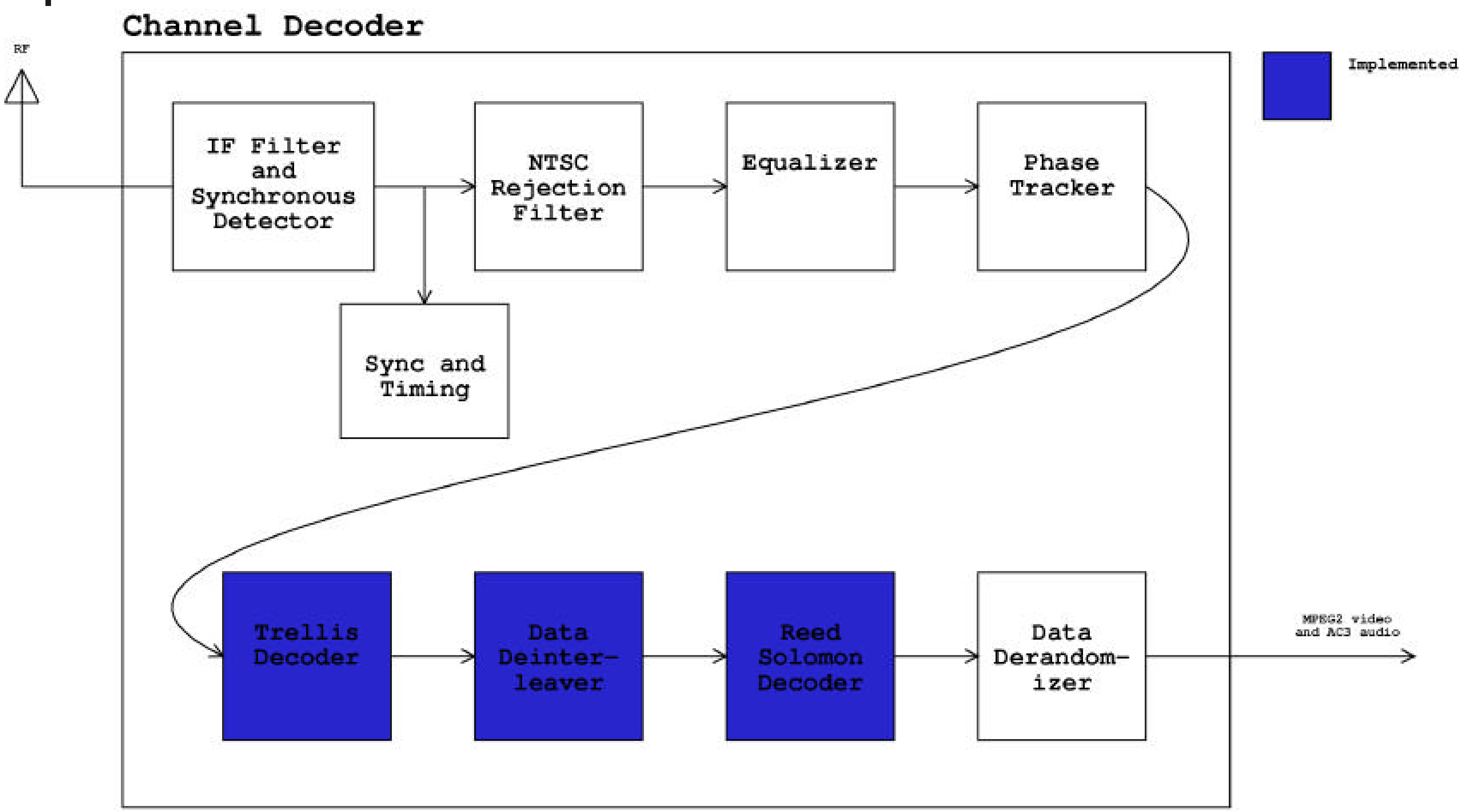
- Channel coding has static rates.
- Comes in 2 flavors
  - 8-VSB for terrestrial cable television
  - 16-VSB for satellite television
  - (the 8 means that a transmitted symbol can take on 1 of 8 values.)
- We focus on the channel encoding and decoding of 8-VSB.

8-VSB Channel Coder



Source: ASTC Standard A53 revision b

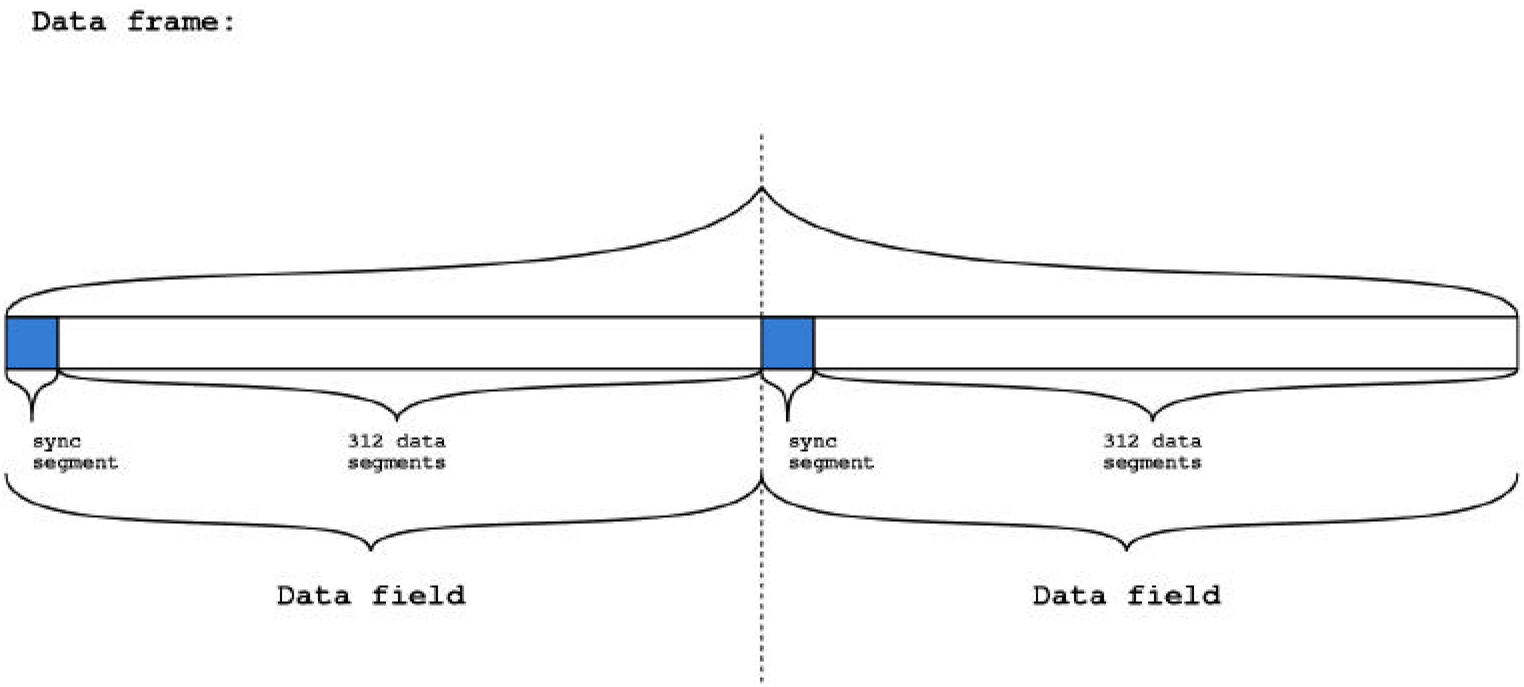
8-VSB Channel Decoder



Source: Vanu HDTV presentation

Data rates

- Data frame = 2 data fields, each with 313 segements (1 sync, 312 data)



Data rates (cont.)

- Each data segment encodes 188 bytes of MPEG-2 stream (1 sync, 187 data).
- Transmitted as 828 8 level symbols

---

187 data bytes	+20 parity bytes →	207 bytes
207 bytes	8 bits per byte →	1656 bits
1656 bits	2/3 trellis encoder →	2484 bits
2484 bits	3 bits/symbol →	828 symbols

Outline

- “Cultural” Overview
- **Reed-Solomon**
- Convolutional Interleaving
- Trellis Coding/Viterbi Decoding

Reed-Solomon Codes

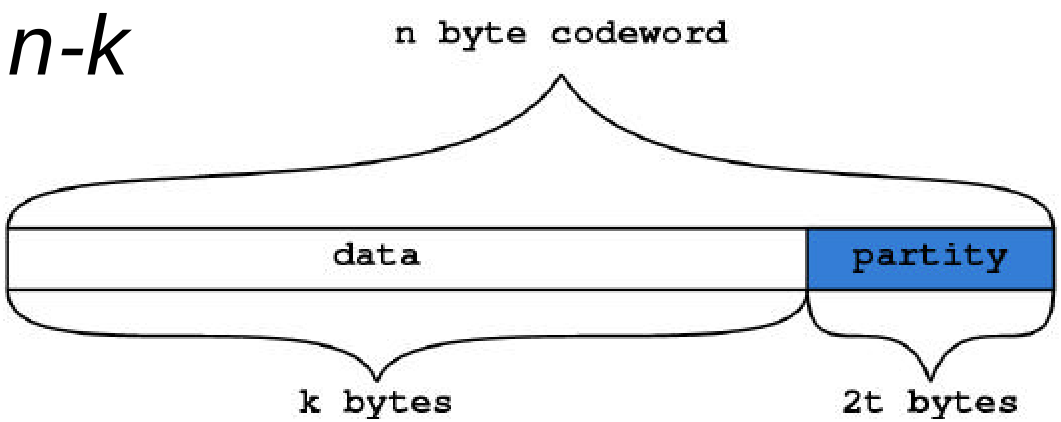
- Also in CDs, Cellphones, ADSL, DVD, etc.
- FEC (forward error correction)
- (eg it appends parity symbols to data symbols)

Irving S. Reed and Gustave Solomon (researchers at Lincoln Labs) published original paper in SIAM, 1960.

- (http://www.siam.org/siamnews/mtc/mtc193.htm)

Reed-Solomon Codes (cont.)

- Specified as RS(n,k)
- Takes k symbols of data and produces an n symbol codeword (e.g. it adds n-k parity symbols)
- Can correct up to t symbol errors, where $2t = n-k$



StreamIT Implementation

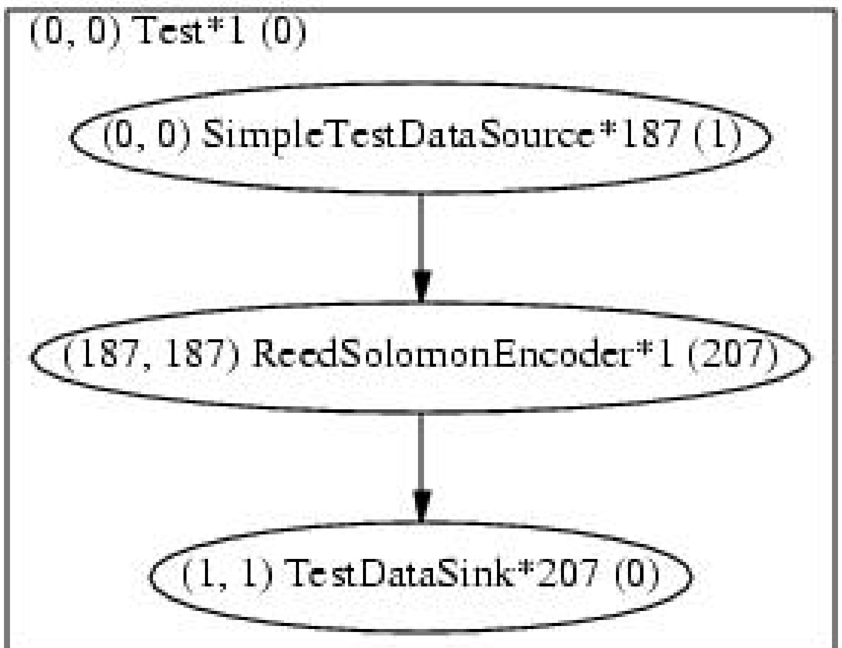
- Copied *rscode*, an open source RS implementation
  (written by Henry Minsky, formerly of the AI lab)

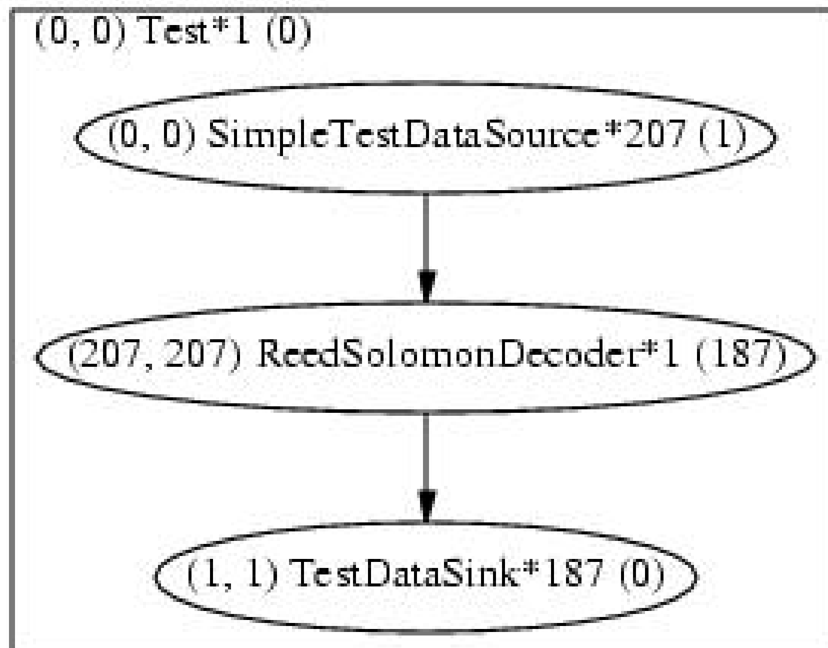
- Converted the rscode from **c** to **Java** and fooled with fields.
- Works, and corrects artificially introduced errors as promised.

http://rscode.sourceforge.net/

RS Stream Graphs

- Monolithic, coarse grained implementation.





StreamIt Implemention Notes

- Both the encoder and decoder have the same code b/c you can't share static code among streams.

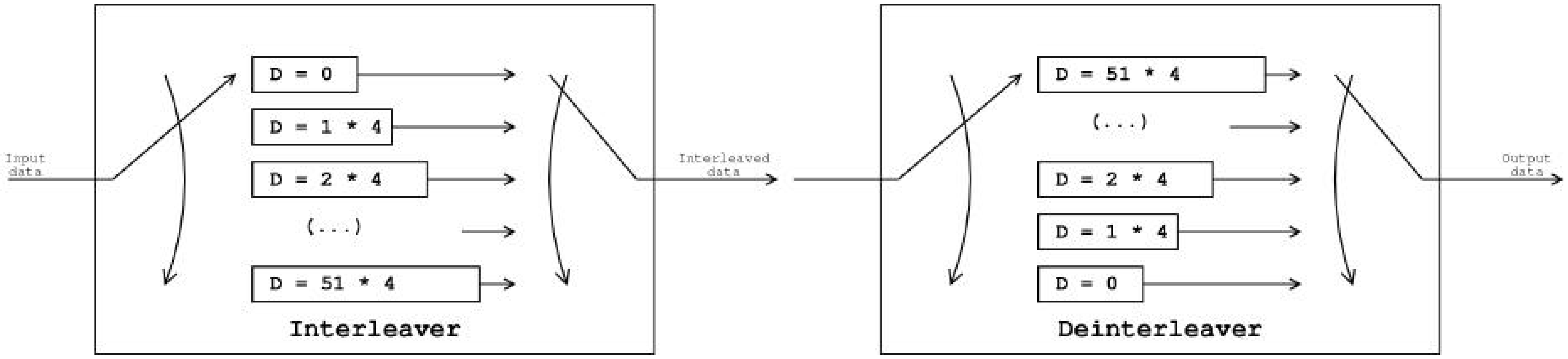
- It was fairly straightforward for the **c** to **Java** conversion – nice feature for migrating existing applications.

Outline

- “Cultural” Overview
- Reed-Solomon
- **Convolutional Interleaving**
- Trellis Coding/Viterbi Decoding

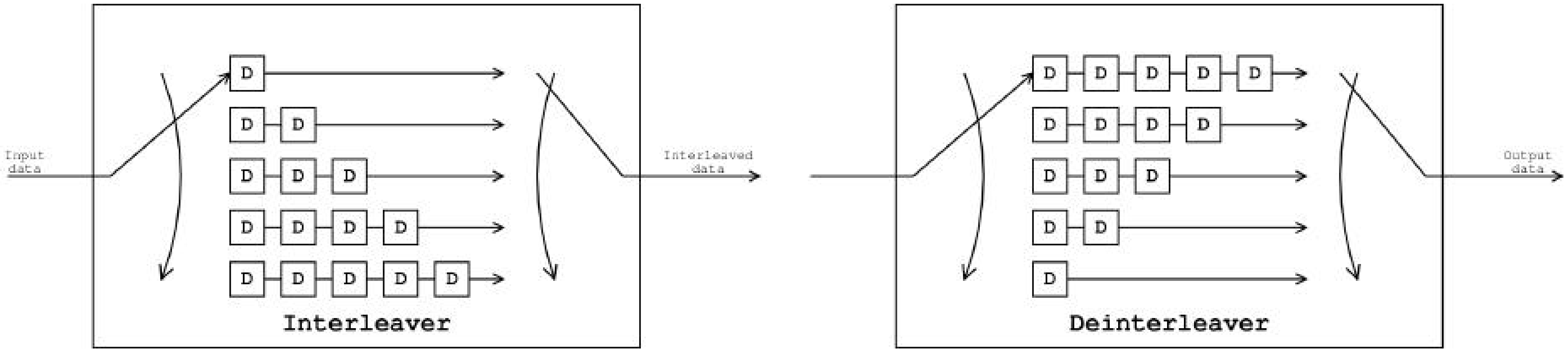
Convolutional Interleaver

- Spreads a burst of channel errors out in space over encoded data.



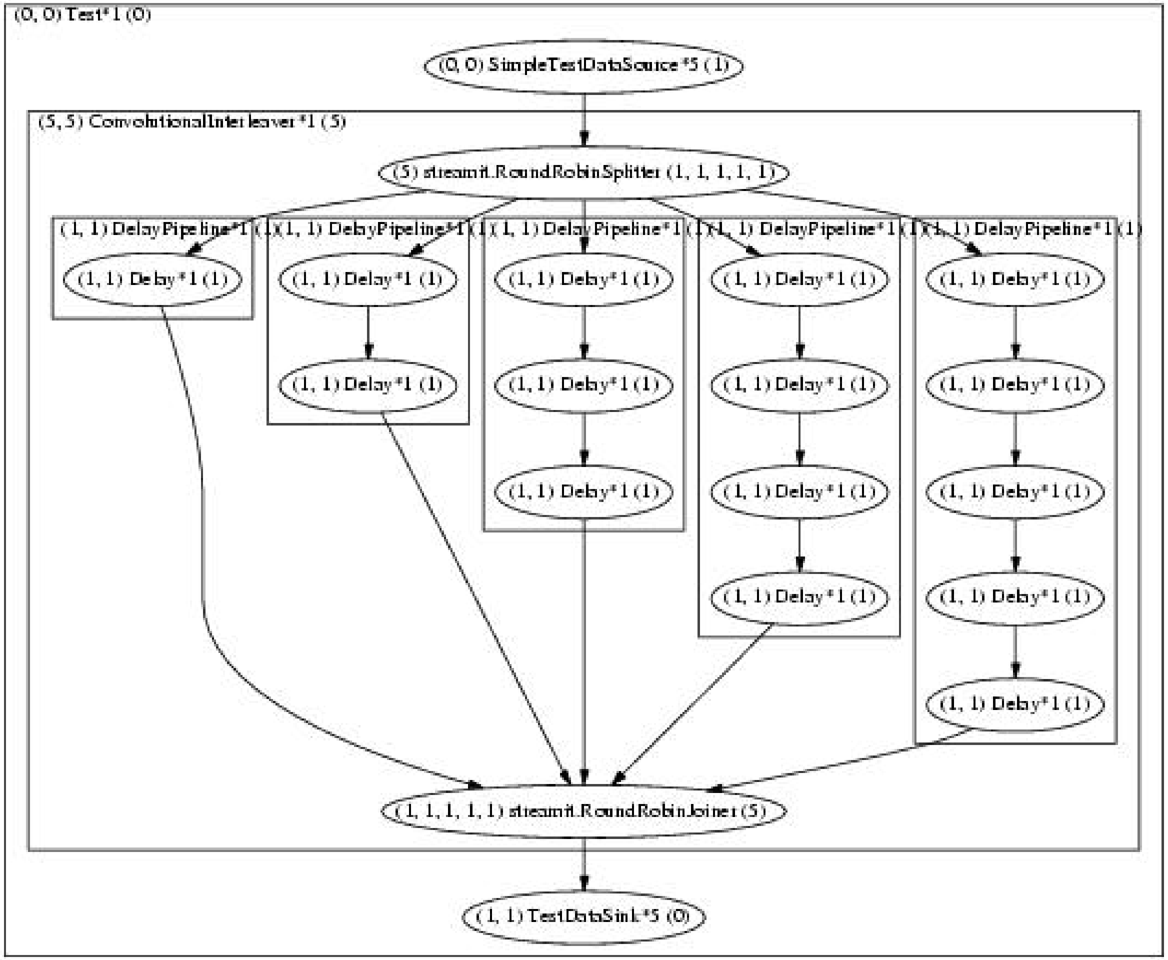
Delay

- I found it easier to understand the convolutional interleaver this way:

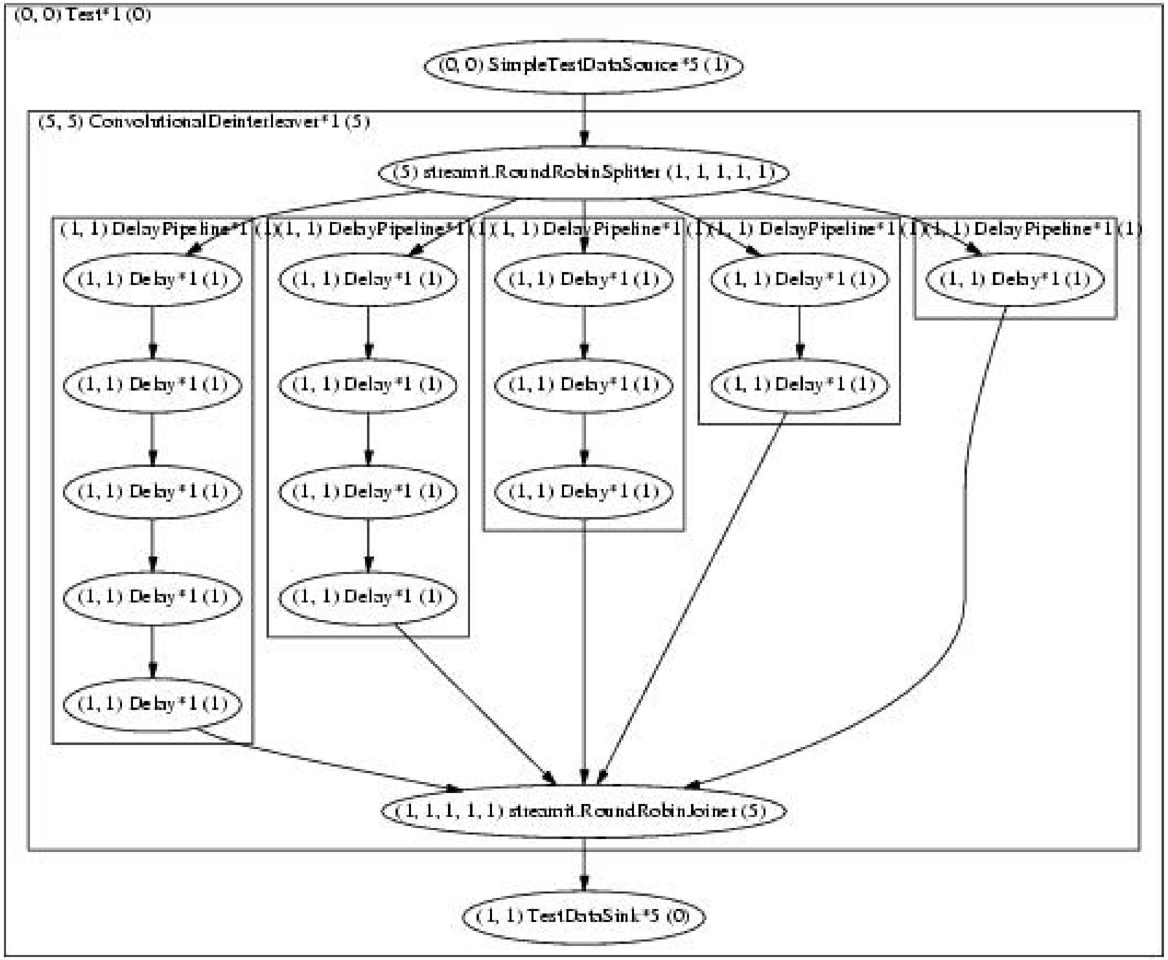


- Need to “prime” with 30 elements.

Interleaver Stream Graph



Deinterleaver Stream Graph



StreamIt Implemention Notes

- The stream graphs closely resembles the block diagram, which is good.
- It was very easy, once we figured out what a convolutional interleaver did, to implement in StreamIt.
- No way to ignore first 30 elements

Outline

- “Cultural” Overview
- Reed-Solomon
- Convolutional Interleaving
- **Trellis Coding/Viterbi Decoding**

Trellis Encoding

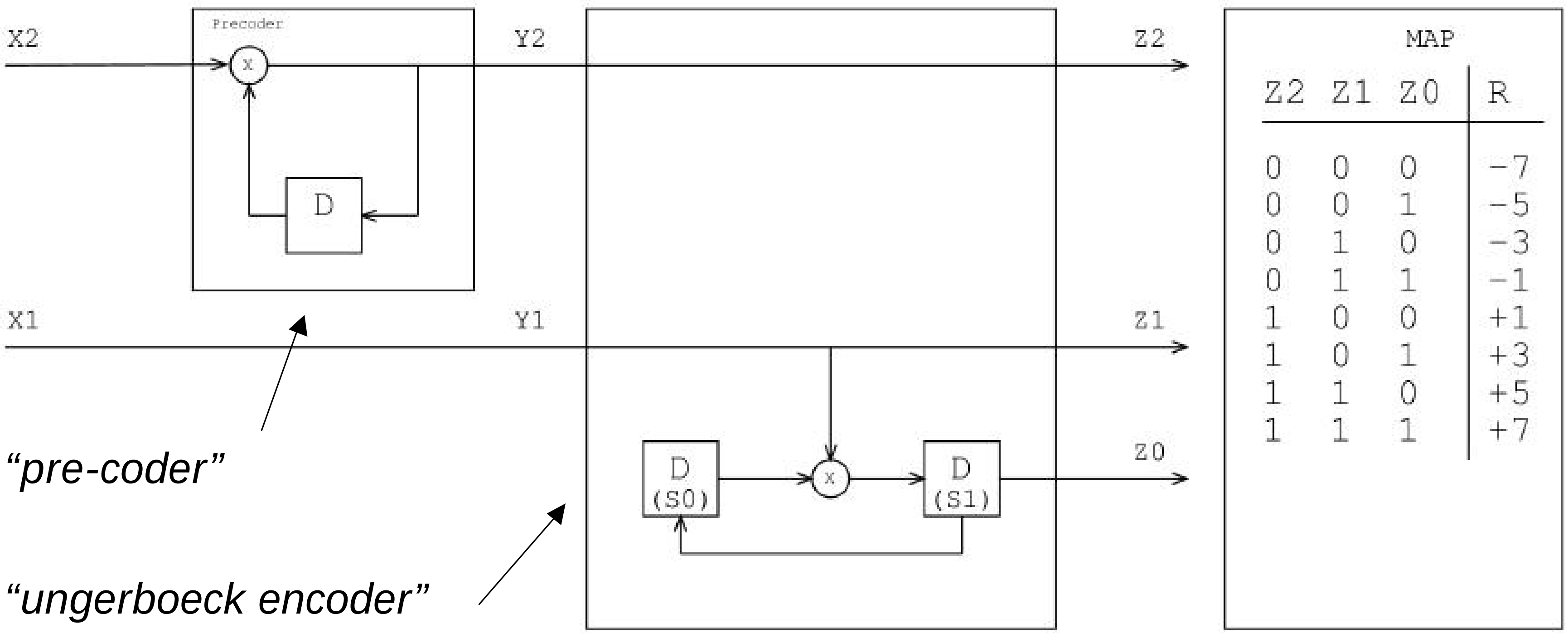
- In HDTV, the trellis encoder is a convolution encoder followed by a symbol mapper.
- This provides good noise immunity (somehow)
- HDTV encoder uses 12 trellis coders in parallel

Convolution Encoding

- Input is an m bit symbol
- Output is an n bit symbol
- m/n is the rate (HDTV uses 2/3)
- K is the reach: the number of output symbols that each input symbol affects (HDTV has K=3)

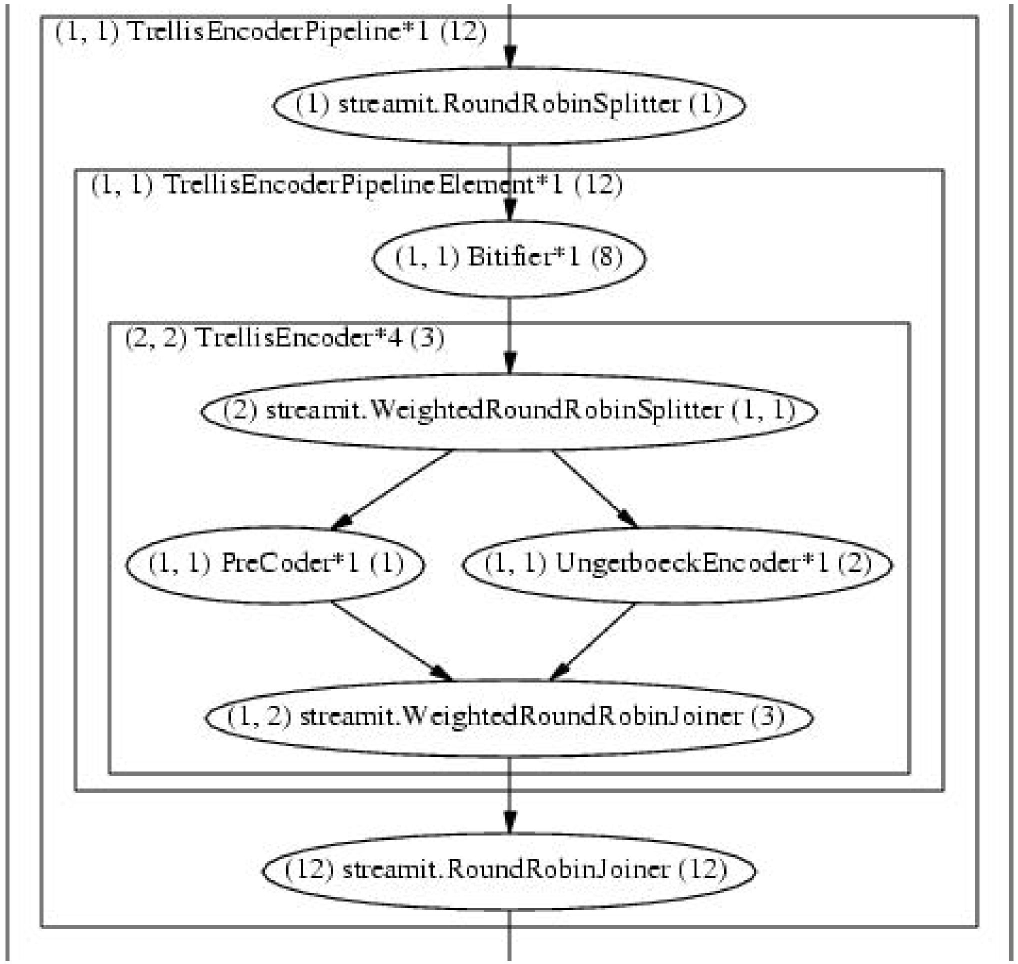
Trellis Encoding (cont.)

- Trellis Encoder used in HDTV



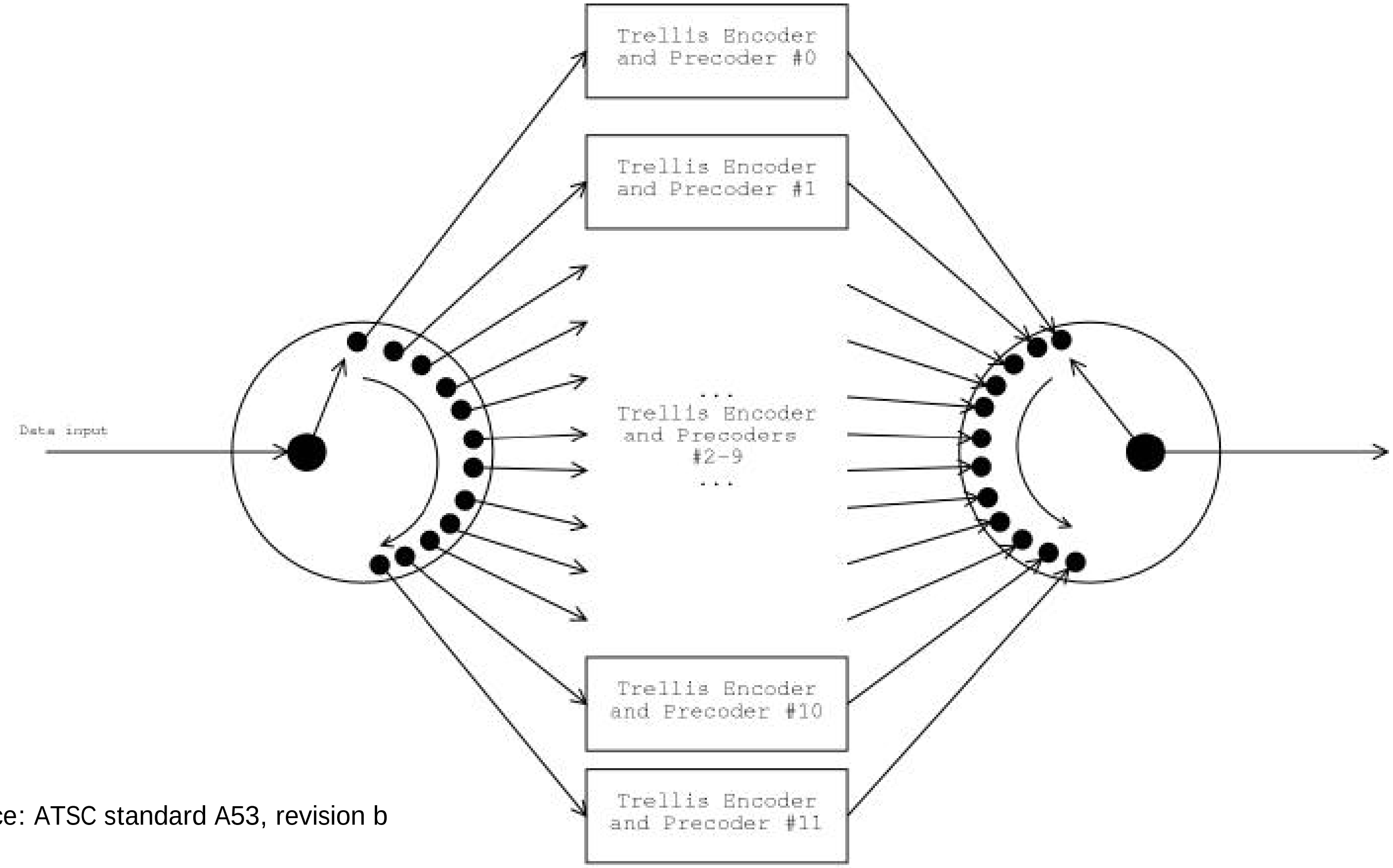
Source: ATSC standard A53, revision b

Trellis Encoder Stream Graph



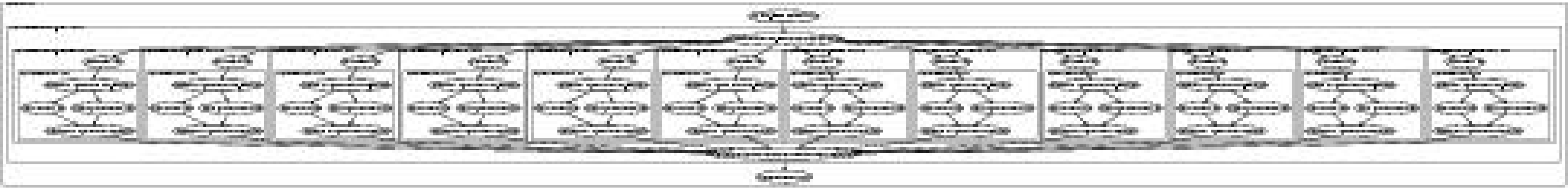
Trellis Encoding (cont.)

- 12 Trellis Encoders



Source: ATSC standard A53, revision b

Trellis Split-Join Stream Graph



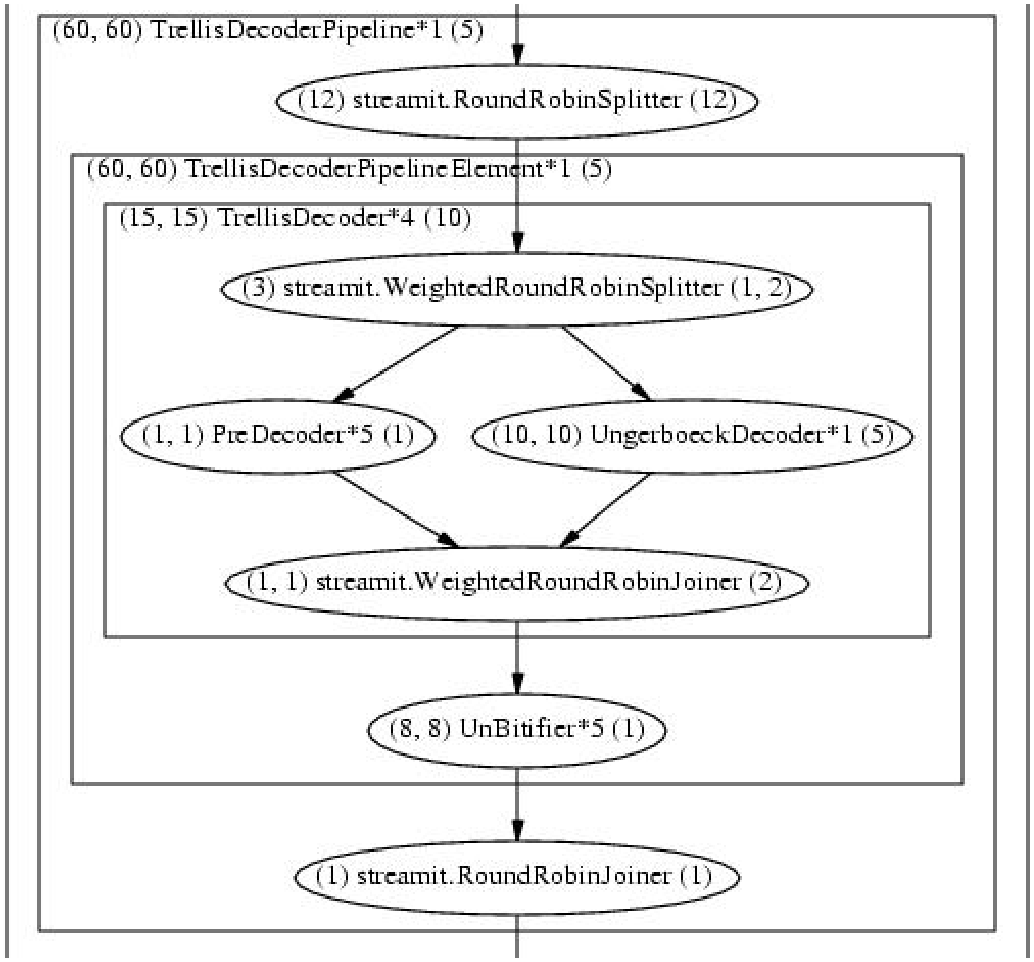
Trellis Decoding

- Not specified by A53b, I chose to use Viterbi decoding.

  (after Andrew Viterbi, founder of Qualcomm. B.S., M.S. MIT ’57, PhD USC ’62. Also VI-A at Raytheon)

- Uses a dynamic programming approach
- Reads in a lot of input before producing a lot of output.
- Not fine grained (currently?).

Trellis Decoder Stream Graph



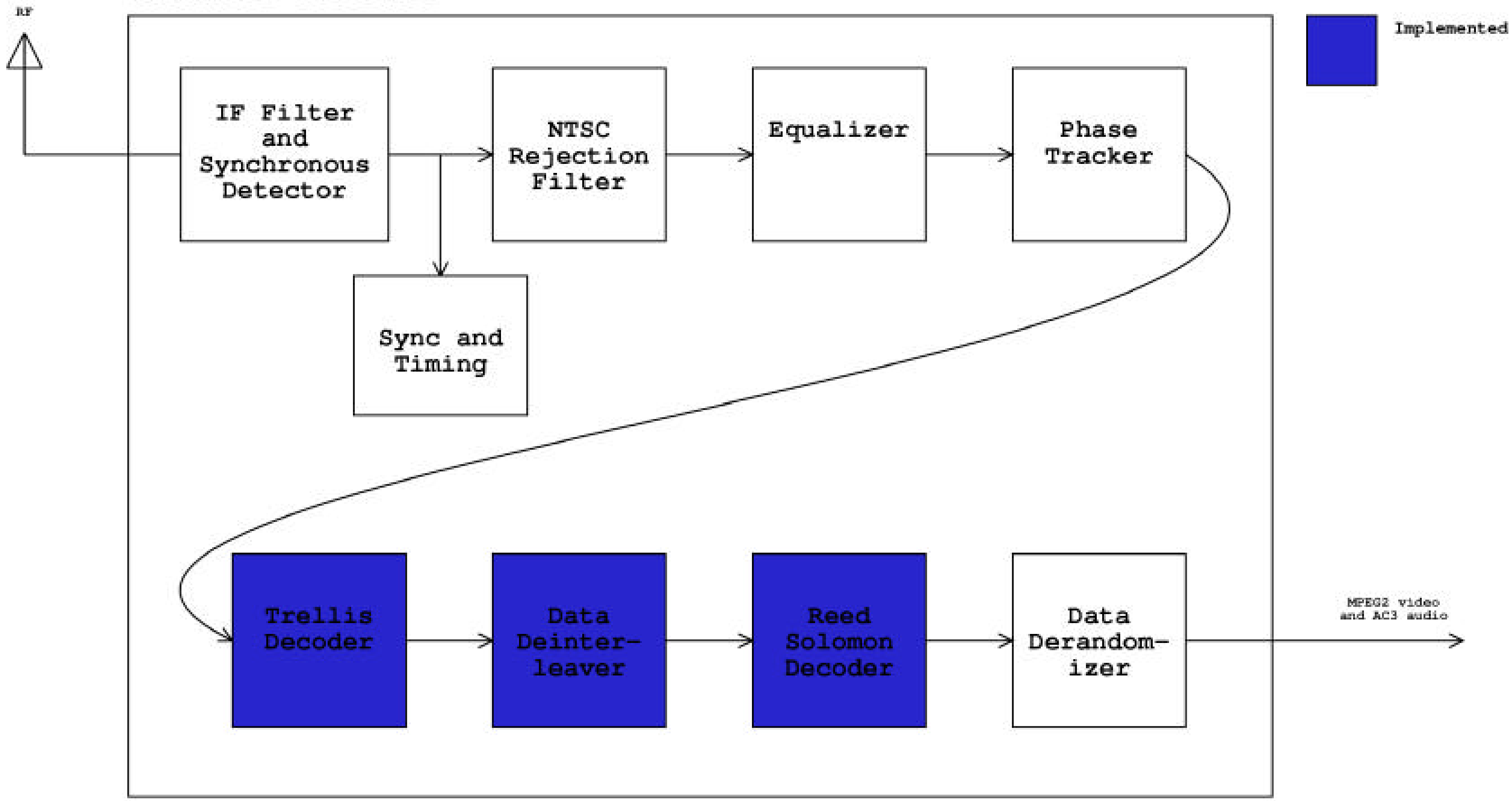
StreamIt Implemention Notes

- The stream graph is reasonably close to the block diagram. The Ungerboeck decoder (which implements the Viterbi algorithm) is monolithic.

- The encoders are naturally written to operate on bits (implementation uses integers) and so it is very inefficient.

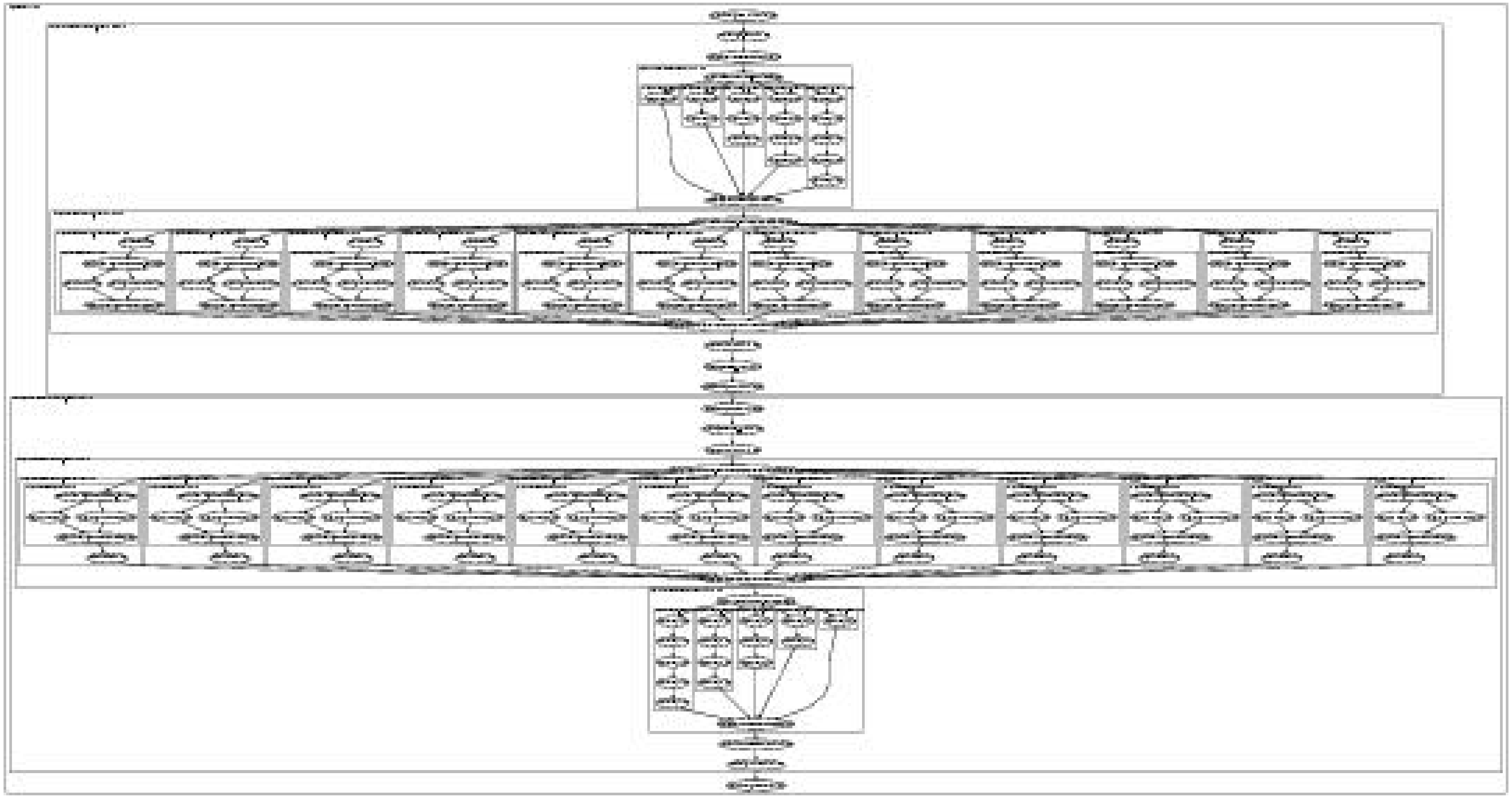
HDTV Decoder (again)

Channel Decoder



Source: Vanu HDTV presentation

Whole Stream Graph



Source Code Comparison

Block	StreamIT Code lines	Total lines	Vanu Lines(?)
RS Encoder	190	364	(?)
RS Decoder	195	367	~450
Interleaver	41	83	(?)
Deinterleaver	49	83	~250
Trellis Encoder	52	118	(?)
Trellis Decoder	207	417	~500
Randomizer(?)	—	—	~210