Viterbi Decoding using Imagine

Njuguna Njoroge

Ayodele Thomas

John Davis

Andrew Lin

EE482C Project

May 30, 2002

Viterbi Project Outline

- Implementing Viterbi
  - TI hardware
  - Imagine Hardware
- DSP Hardware
- Imagine Hardware
- Results
- Feedback/Feedforward and Dependencies
- Conclusions

Viterbi a la DSP

- Macros:
  - Butterflies
- Traceback Loop
- Explicit Memory Management
- Bit manipulation a plus!

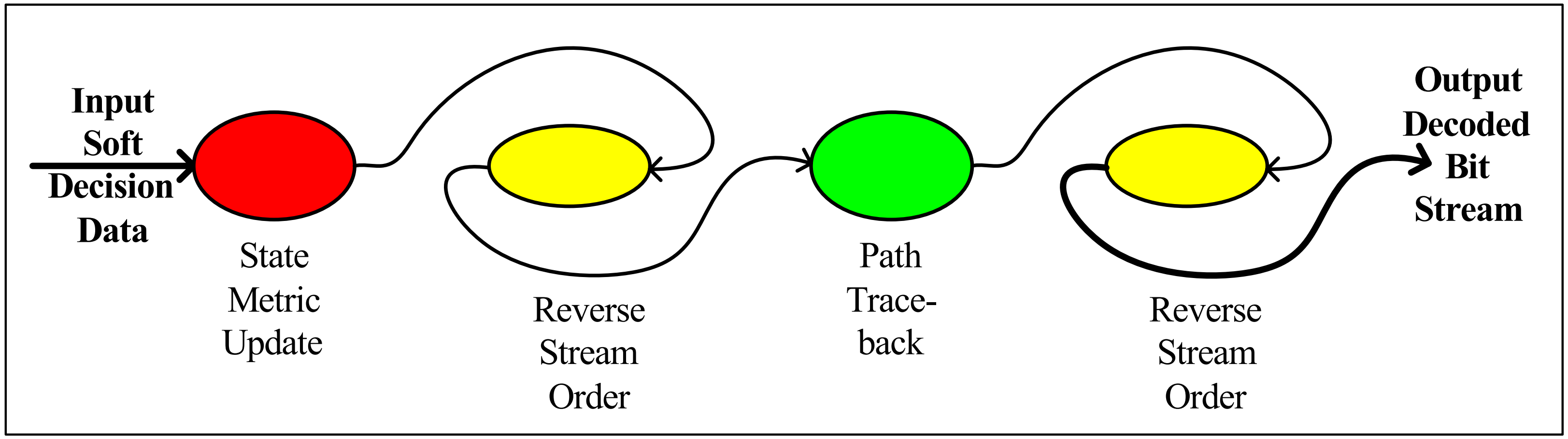
DSP Hardware

- The TI TMS320C54x DSP (or ‘C54x for short)
- It has:
  - 40-bit barrel shifter
  - 2 40-bit accumulators
  - 1 40-bit ALU (can be used as two 16-bit ALUs)
  - 1 40-bit Adder
  - 1 17x17-bit Multiplier (combine this with adder and you have a MAC)
- And, it also has some special Viterbi hardware
  - Compare-Select-Store Unit (CSSU) – this can compare two 16-bit values, and store the decision and the value in 1 cycle.
  - Special 16-bit Transition Register (TRN) to store decision bits.

DSP Results

- Input stream: 1K G0,G1 pairs
- Output stream: 1K bits
- Cycle times:
  - Initilization: 308 (.4%)
  - State Metric Update: 9604x8 (87.4%)
  - Traceback: 1344x8 (12.2%)

Viterbi! Can you Imagine?



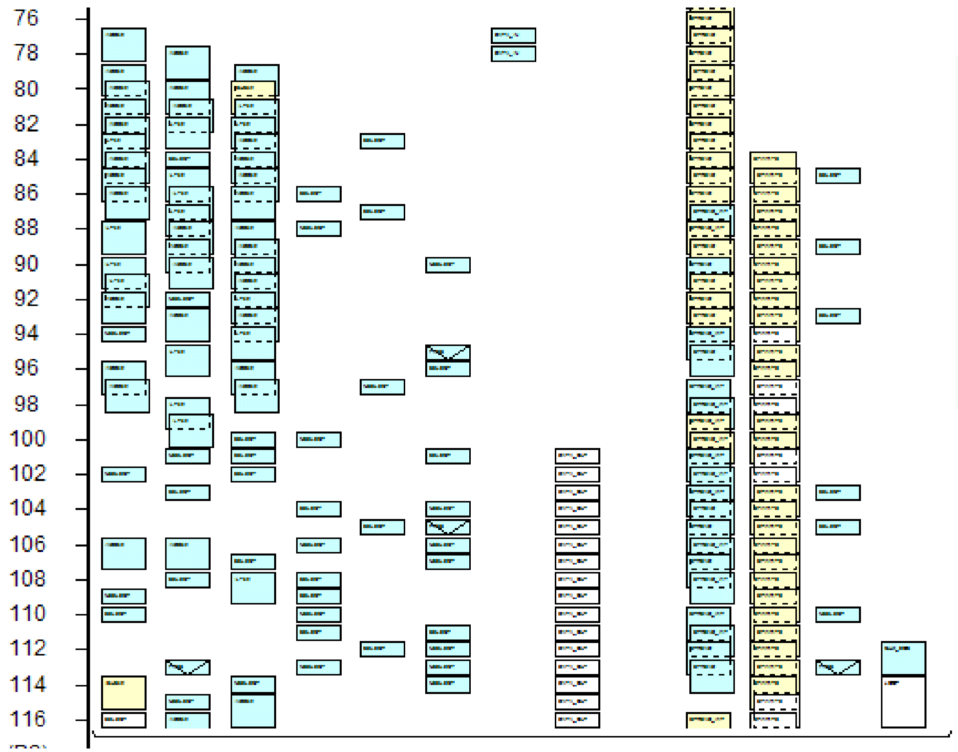
- Streams = lines
  - Input, intermediate, and output data
- Kernels: ellipses
  - State metric table update via local distance calculation, reverse stream order, traceback to reconstruct/decode data, and reverse output data

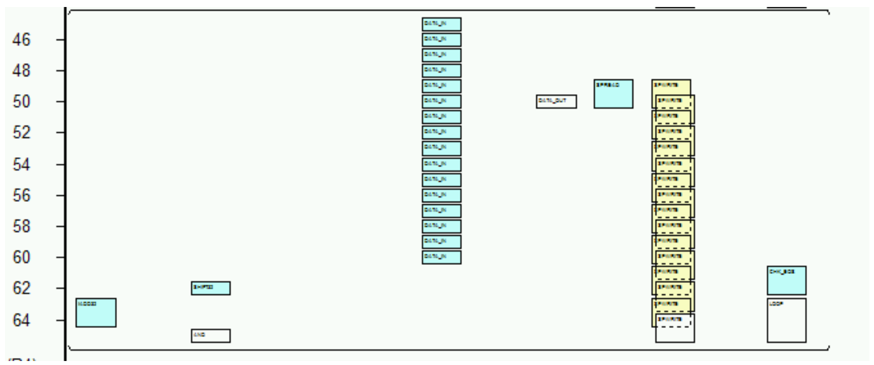
Imagine Hardware

- Segmented Arithmetic
  - Increase arithmetic density
- Comparator
  - Single cycle compares
- Explicit Communication
  - Share information between clusters

Imagine Schedule

- State Metric Update
- Traceback



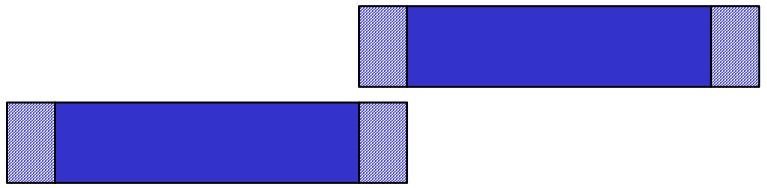


Imagine Results

- Input stream: 8 128 G0,G1 pairs
- Output stream: 8, 128 bits
- Cycle counts
  - Initialization: 49455 (bad)
  - State Metric Update: 11019 (7)
  - Traceback: ~3000 (3.5)

Feedback/forward & Dependencies

- Break the dependency chain!
- How?
  - Strip-mine/chunk data
  - Replicate or pad edges
- Drawbacks
  - Reduced throughput (must be able to amortize)



Conclusions

- Viterbi is a mature DSP application
  - Lots of documentation and tools
  - Compact Code
- Imagine – Not so mature
  - Feedback can be done
  - Viterbi can be streamed
  - Replication and Padding are a general tool

Imagine Limitations

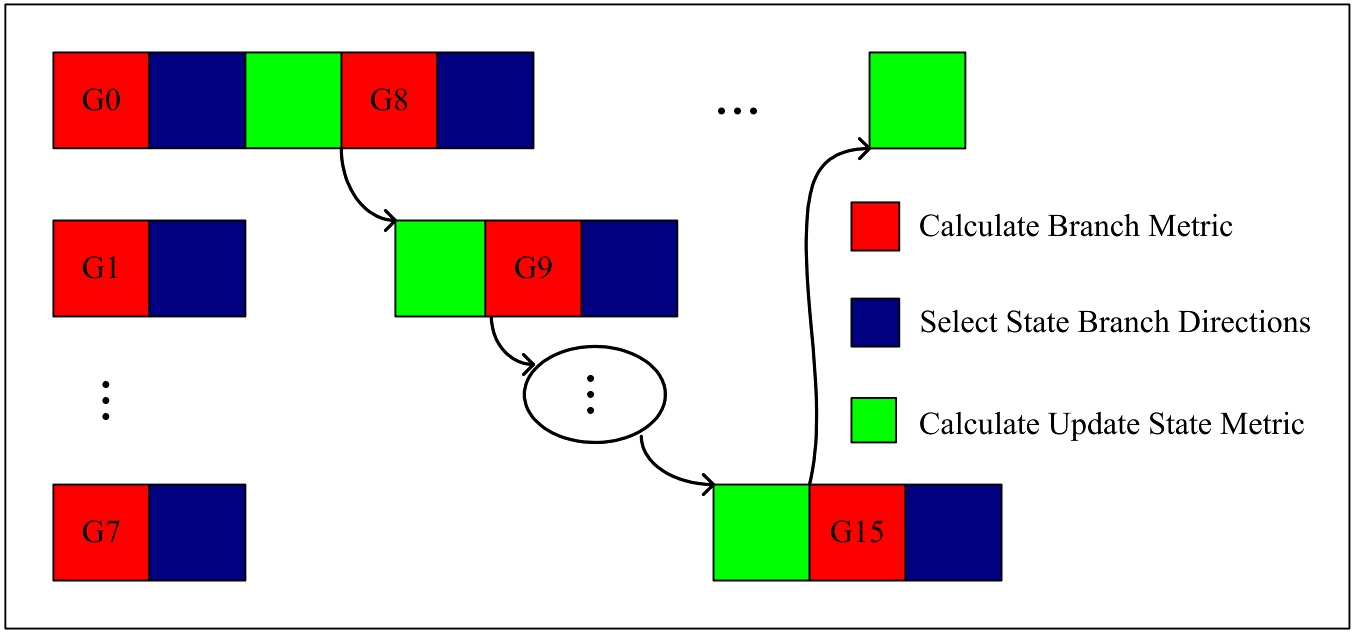
- Serial Access to the Scratchpad
- Large number of Intermediates values a problem $\rightarrow$ Larger LRF
- Code expansion: no nested loops
- Not enough comparators, ACS
- No explicit bit manipulation
- One word: ARRAYS!
- Bug: Indexed accesses with record streams, fixed soon

Questions and Comments

The Viterbi Decoder

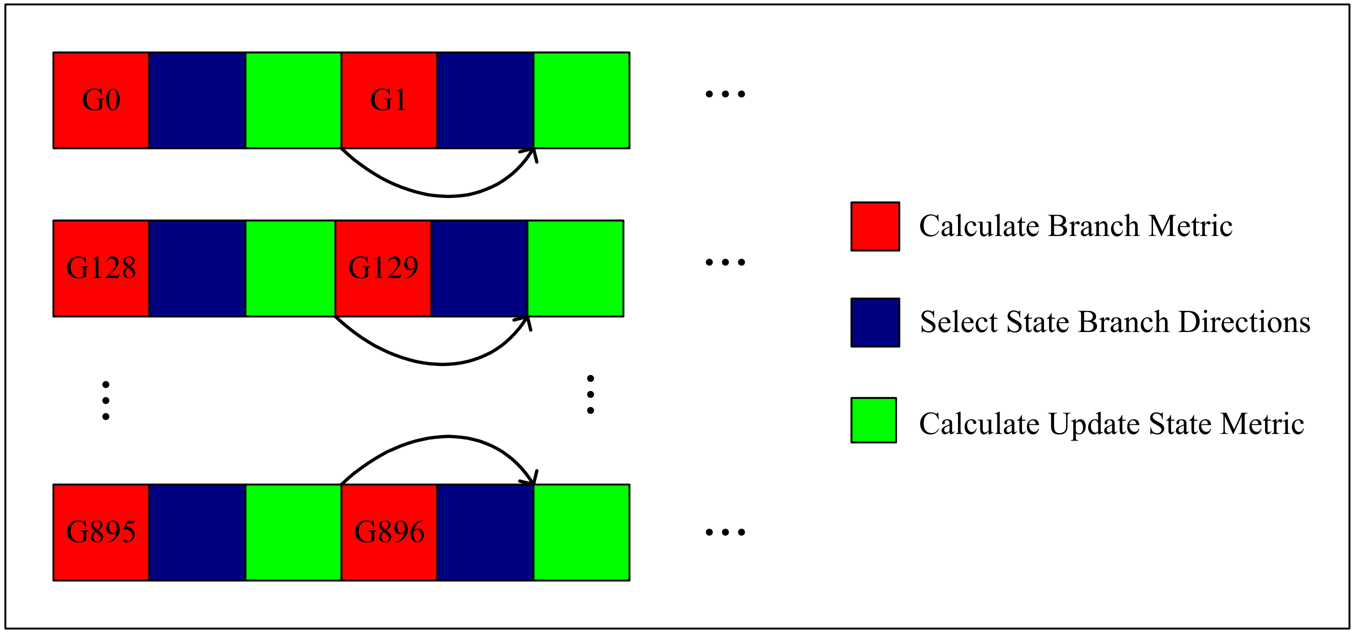
- Algorithms
  - Sliding Window
    - Dependency chain as long as data stream
    - Single state-wide compare for traceback
  - Frame
    - Dependency chain the length of the Frame
    - Added overhead and reduce throughput: $\approx$2K*Frames/Input Stream
    - Start in the same initial state, end in the same state

Exploit ILP?



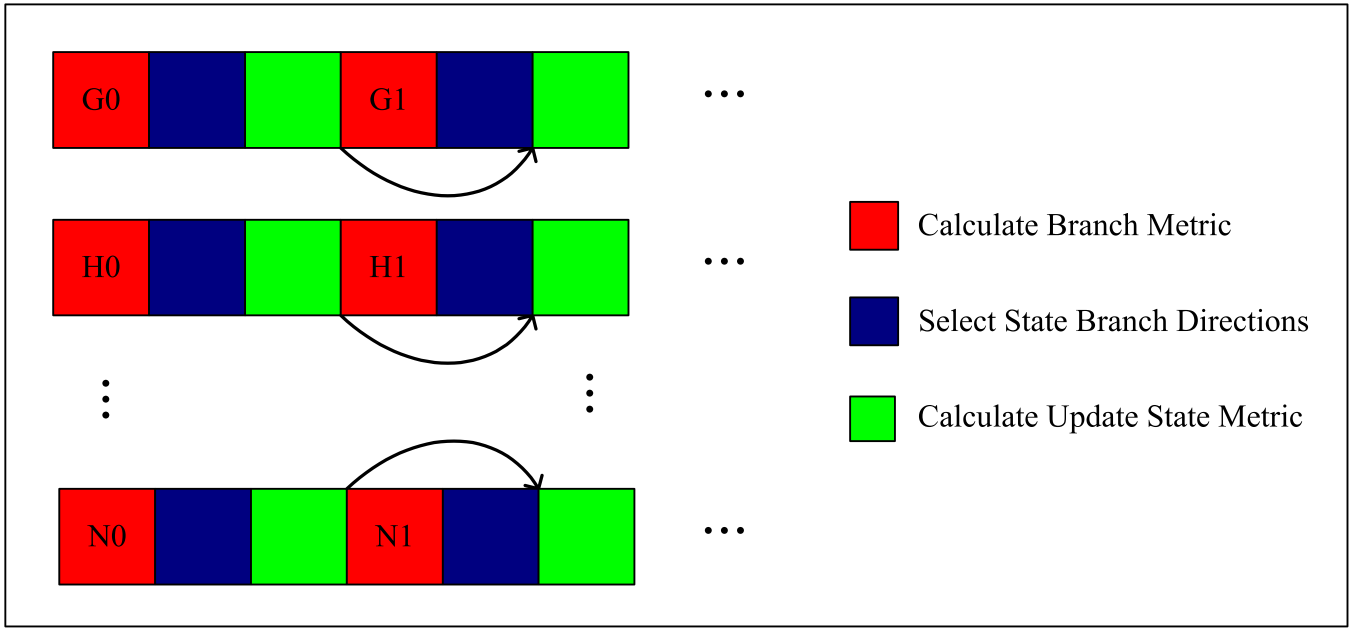
- Sliding window or Frame Algorithm
  - Next element dependency chains require intercluster communication – serializes the state metric update.

Exploit DLP?



- Frame algorithm – single input stream
  - Initial and End states known for every frame.
  - Added overhead and reduced throughput

Exploit TLP?



- Sliding window algorithm
  - Every clusters is a viterbi decoder
  - Feedback within a cluster, no intercluster communication