Turbo-coding: Merits of implementation in Actel RTAX FPGA

Charles Howard

Southwest Research Institute

Abstract

- The Kepler mission implements turbo-coding via an RTSX FPGA architecture employing external PROM and SRAM. With the general acceptance of the embedded features of the RTAX architecture, it is possible to implement the entire turbo code function within a single device. The maximum performance of the baseline design will be quantified, and the critical paths will be identified. By mapping the existing HDL/netlist to RTAX, a direct comparison will be drawn due to additional logic resources, such as I/O flops. Modifications to the design will be “limited” to utilization of internal RAM in place of external devices. Maximum performance will be illustrated by implementing the frame RAM and/or Pi ROM internally as embedded RTAX RAM blocks. Limitations of implementing large RAMs in the RTAX will be highlighted.

Telemetry FPGA

- Baseline approach is used on Kepler mission
  - VC Memory Management
  - First Header Pointer logic
  - CADU Formatter
    - CCSDS VCDU formatter
      - Round-robin selection, fill frame generation
  - Turbo Memory Manager
    - Memory management
  - Serializer
    - Encoder LFSR / Mux
    - Randomizer and ASM attach

CCSDS 101.0-B-6

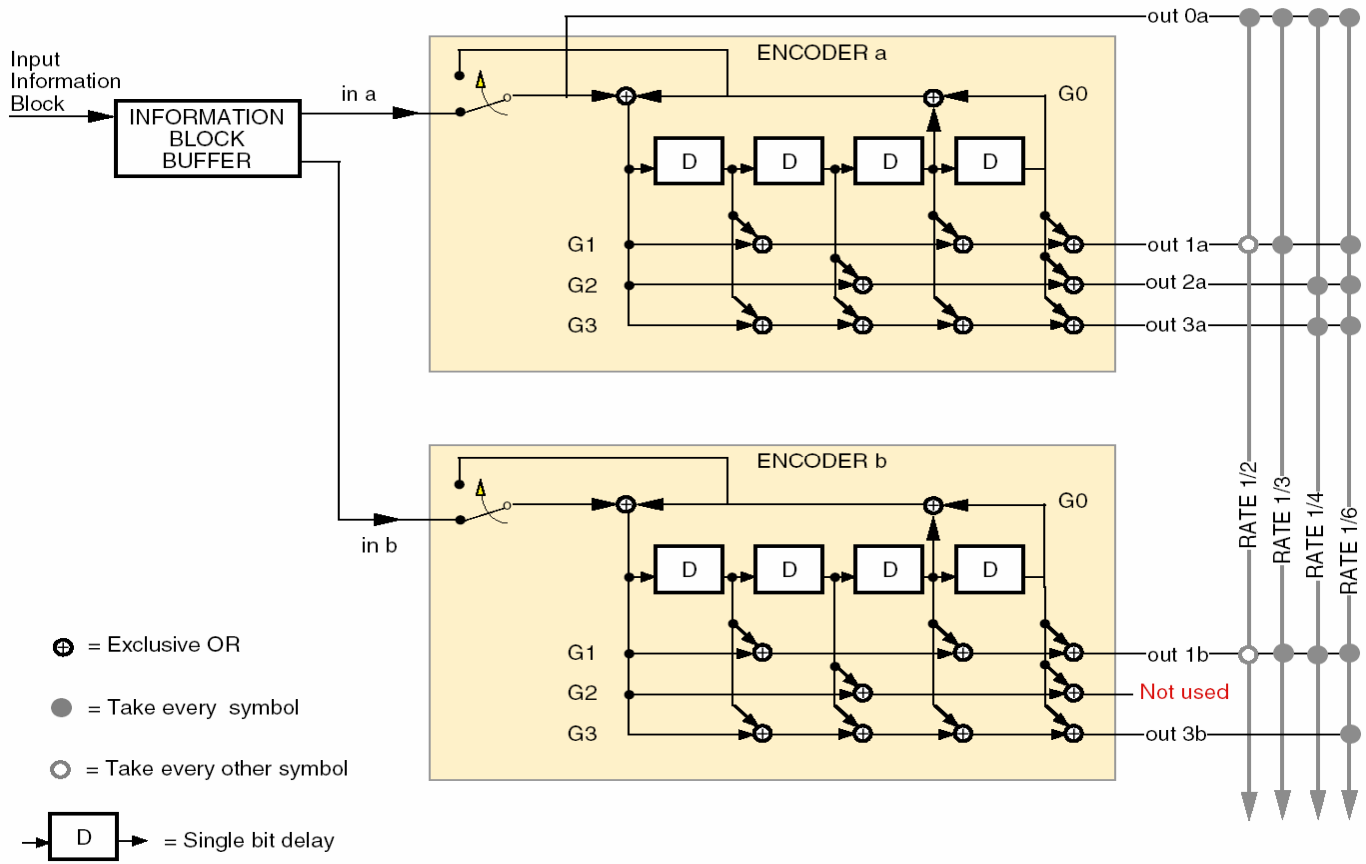


Figure 4-2: Turbo Encoder Block Diagram

CCSDS 101.0-B-6

$$
\begin{aligned}
m &= (s-1) \bmod 2 \\
i &= \left\lfloor \frac{s-1}{2\,k_2} \right\rfloor \\
j &= \left\lfloor \frac{s-1}{2} \right\rfloor - i\,k_2 \\
t &= (19i+1) \bmod \frac{k_1}{2} \\
q &= t \bmod 8 + 1 \\
c &= (p_q j + 21m) \bmod k_2 \\
\pi(s) &= 2\left(t + c\,\frac{k_1}{2} + 1\right) - m
\end{aligned}
$$

The interpretation of the permutation numbers is such that the sth bit read out on line 'in b' in figure 5-2 is the $\pi(s)$th bit of the input information block, as shown in figure 5-1.

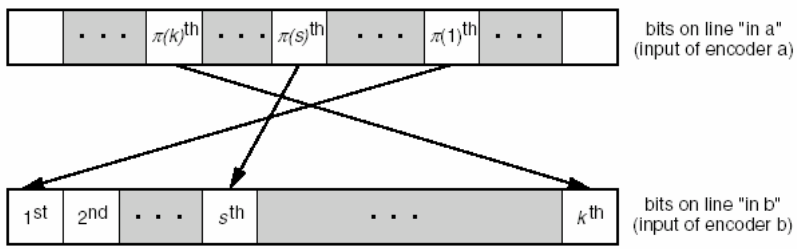
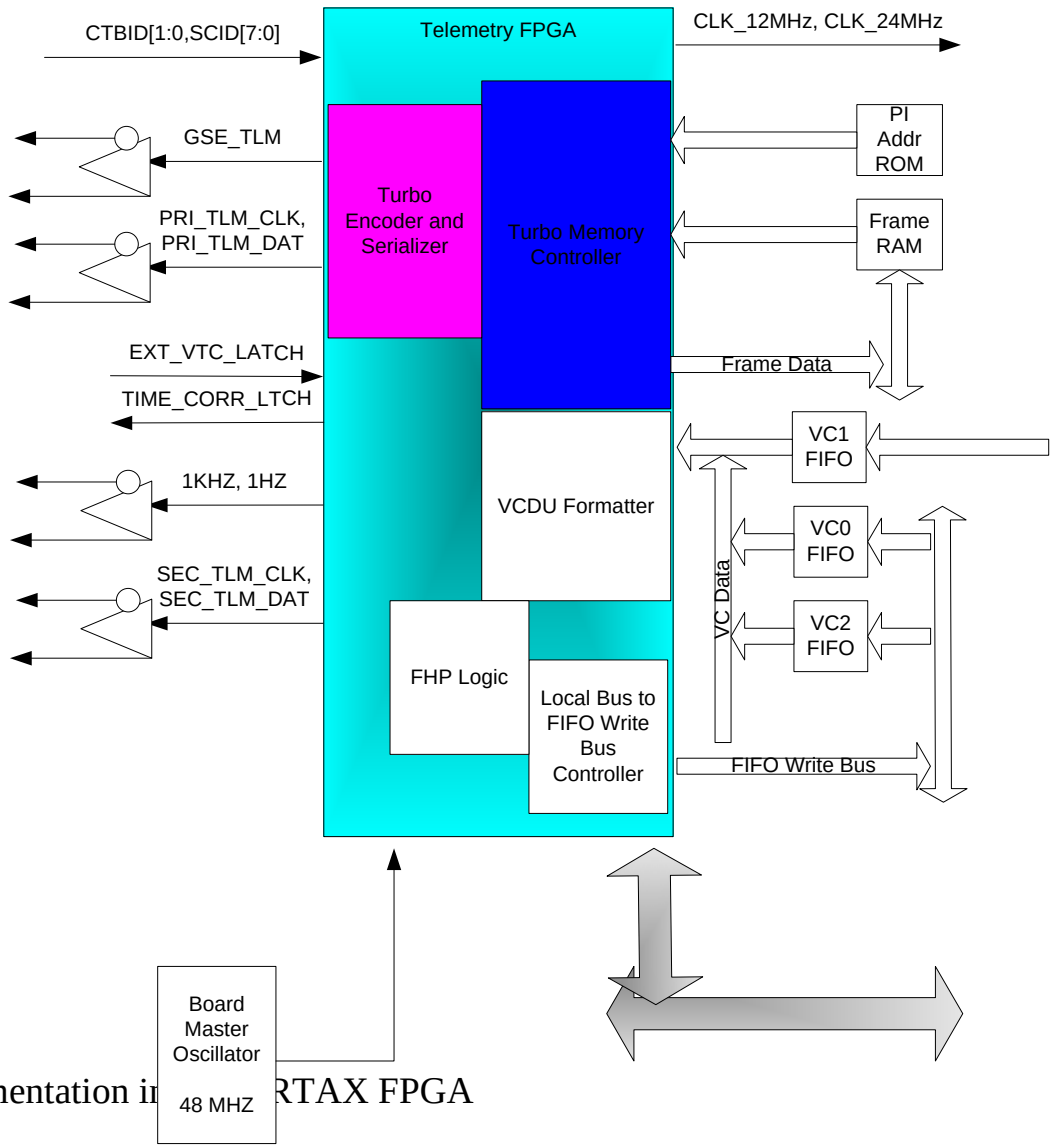


Figure 5-1: Interpretation of Permutation

- Subtract. Modulo. Multiply. Divide. Accumulate. Just to get an address...
  - That implies a lot of logic and cycles to find which bit to look up from the frame buffer
- Look Up Table friendly!!!

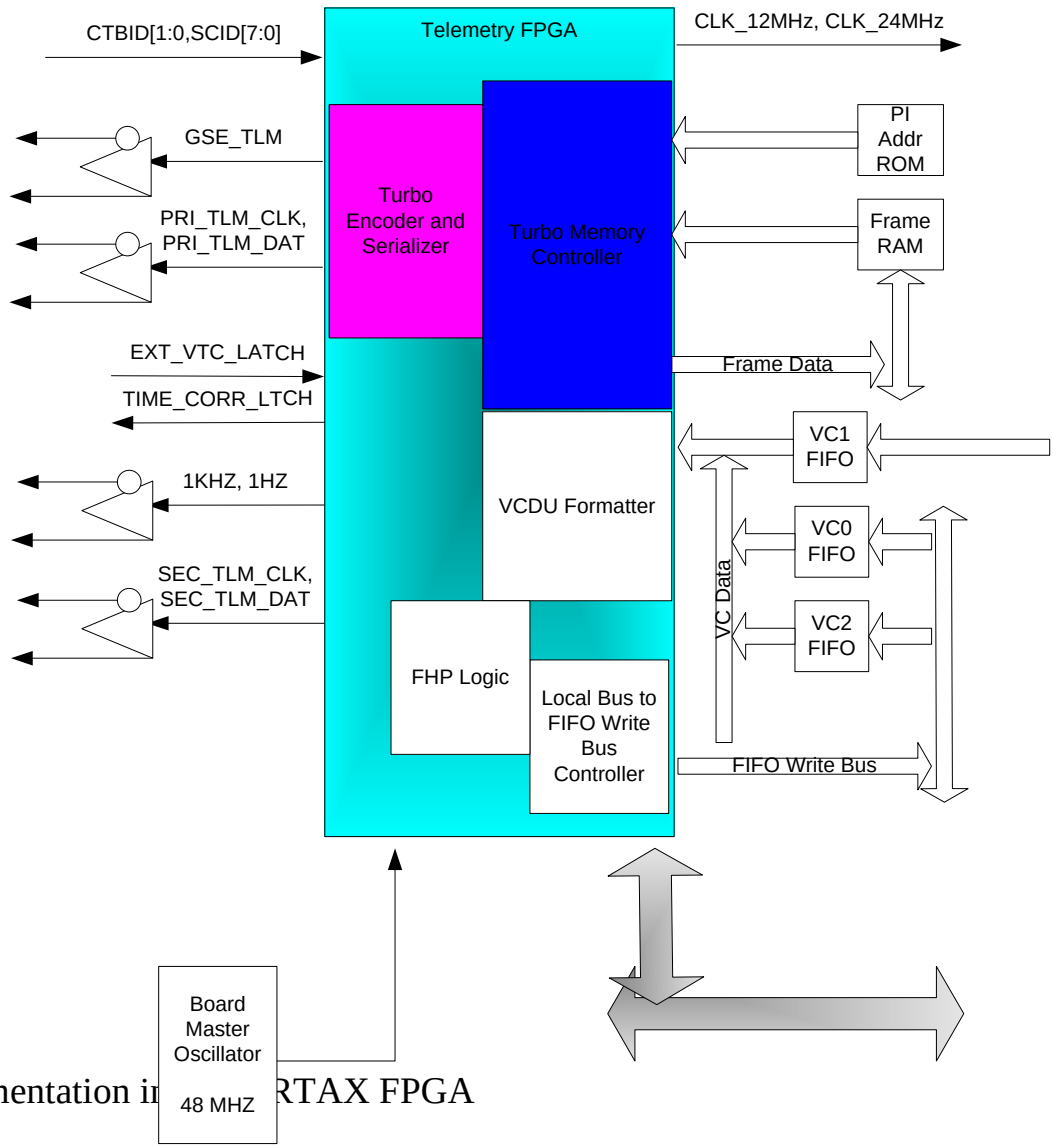
TM FPGA Block Diagram

- TSER_LFSR implements slide 3
- TMEM_CTRL implements math
- Support for two frame lengths
  - PROM is 32K
  - RAM is 128K
  - Parts commonality



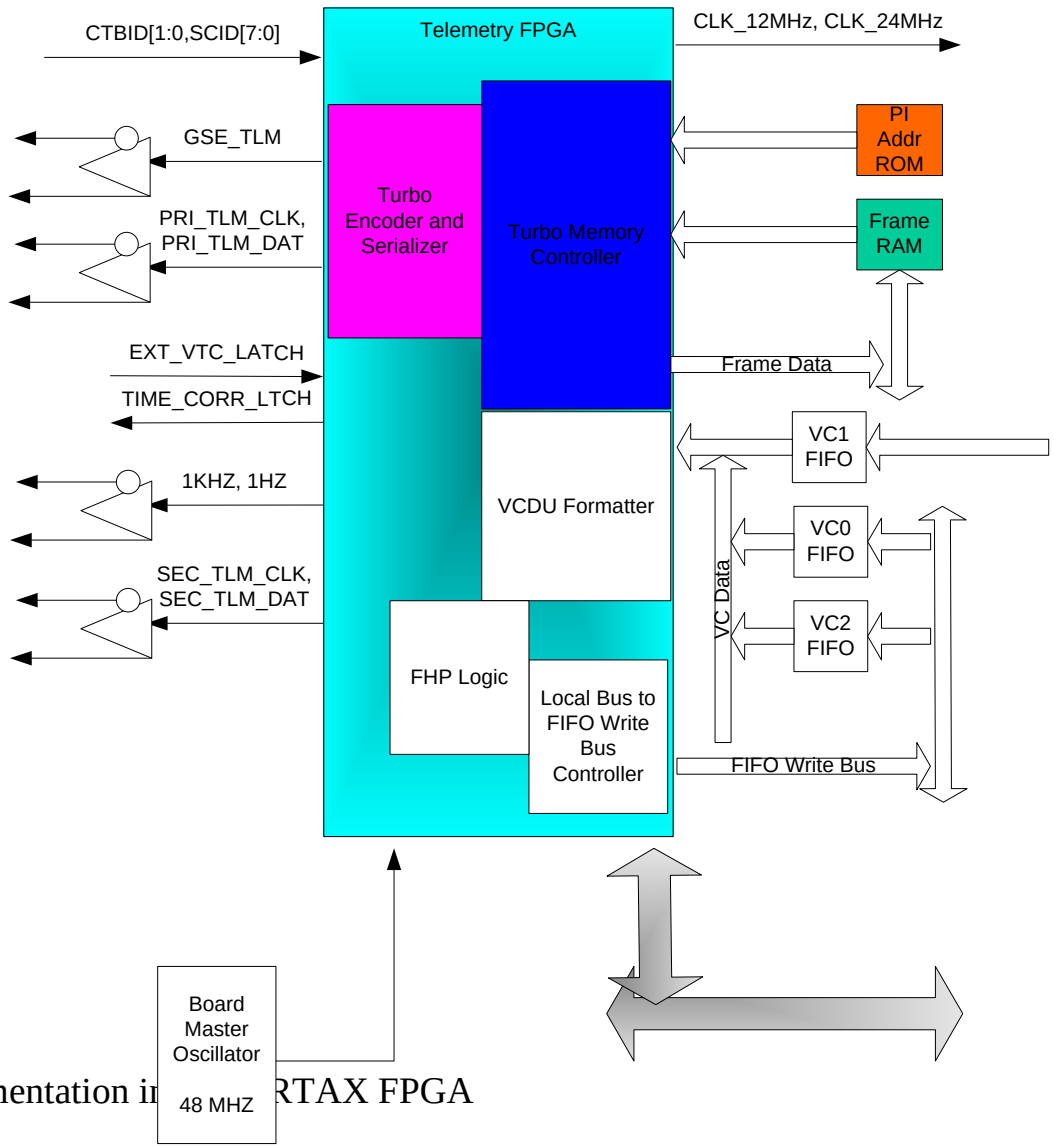
TM tm_tser_lfsr

- Turbo Encoder
  - Pi & true bit feed LFSR
  - Trellis termination
  - Pre-pend ASM / randomization
  - Rate configuration
  - Frame length logic



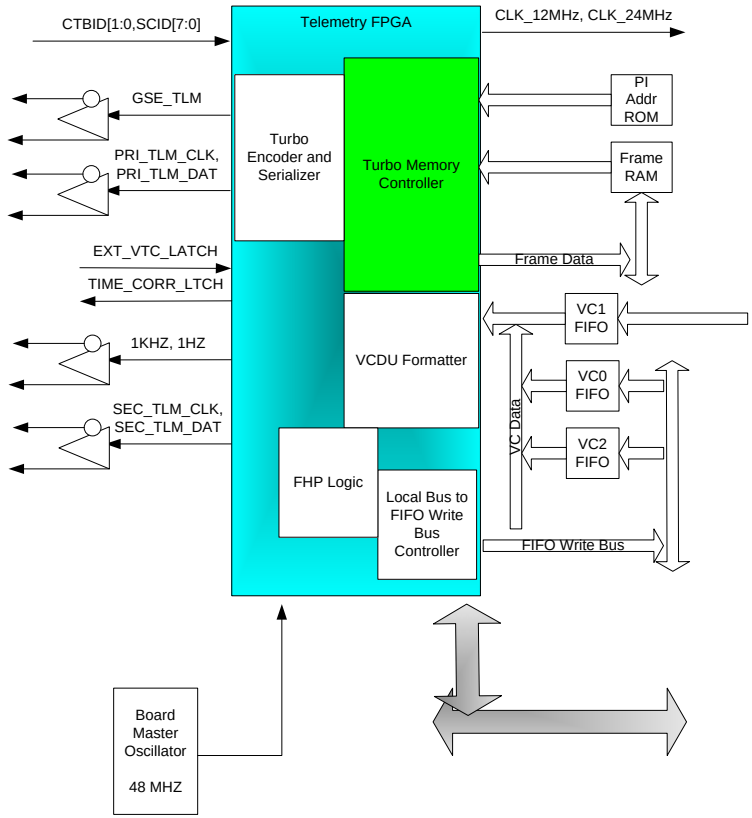
TM tm_tmem_ctrl

- Turbo Memory Controller
  - Temporary VCDU storage (Frame RAM)
  - Look Up Table! (Pi Addr ROM)
  - Pi & true bit submission to encoder



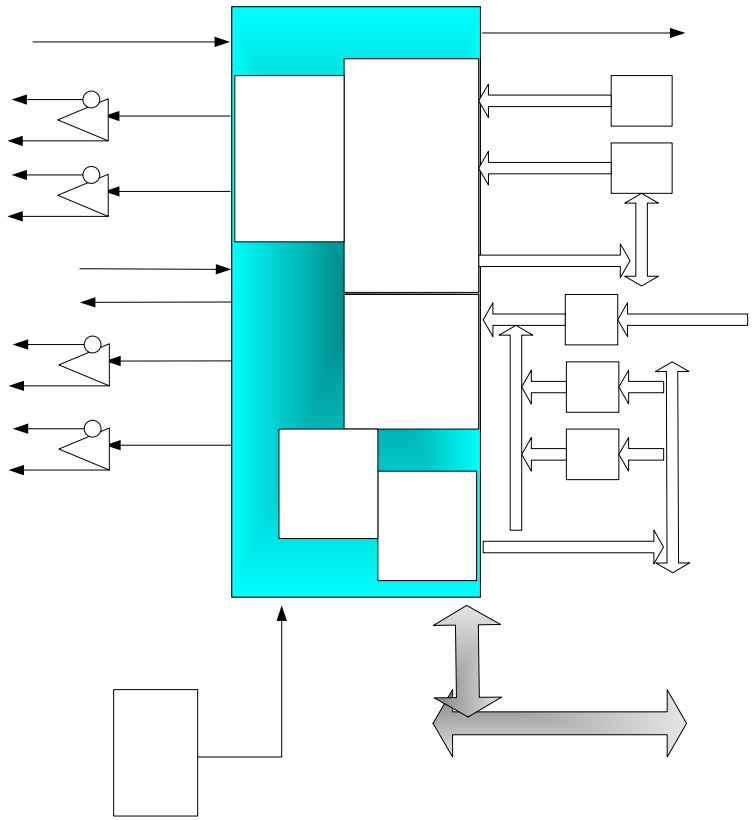
Turbo Memory Controller

- This module controls the turbo-coding “math” for the permutation function.
  - Storage of formatted frames (background task).
  - Collecting the next bit from the VCDU frame (tru_bit) and it's corresponding pseudorandom (pi_bit)
  - Handshake with tser_lfsr module for rate metering



AX vs SX Experiment

- Theme and Variation (RT parts)
  - Baseline (RTSX)
  - Baseline+ (free pins, Libero8.4)
  - Baseline RTAX1000
  - RTAX1000 (frame RAM)
  - RTAX2000 (frame RAM, ½ ROM)
  - RTAX4000 (frame RAM, ROM)



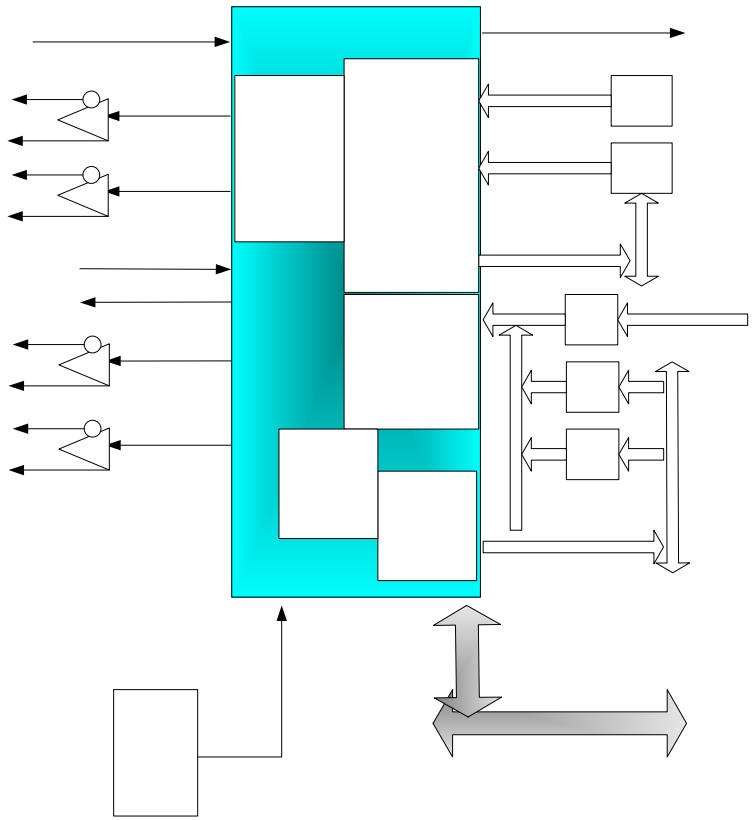
- "ROM" is misnomer...
  - Replaced ROM model with SRAM model
  - Loaded at runtime for sims
  - Provided write interface for synthesis...

CAVEAT STATEMENTS

- All metrics based on RAM blocks WITHOUT EDAC!
- Kepler design had specific requirements on serializer function that do not support maximum bit-rates
  - Yes, I designed it!
- Performance metrics are normalized
- Results are based on memory bandwidth rather than serializer rate
  - I did look at critical paths

FINDINGS

- Theme and Variation (RT parts)
  - Baseline (RTSX)
  - Baseline+ (free pins, Libero8.4)
  - Baseline RTAX1000
  - RTAX1000 (frame RAM)
  - RTAX2000 (frame RAM, ½ ROM)
  - RTAX4000 (frame RAM, ROM)

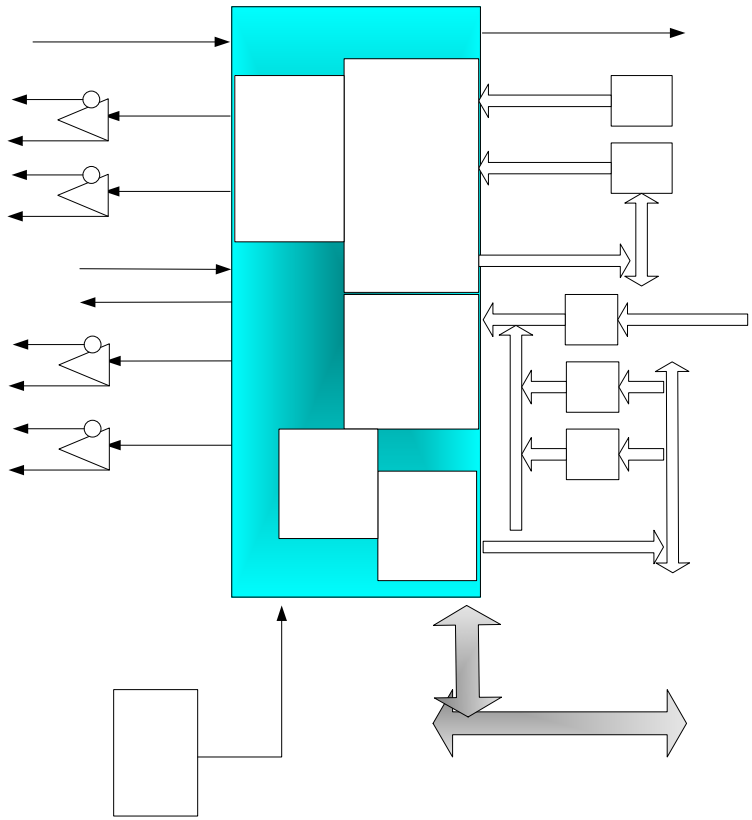


- BASELINE == 1.0

- 1.0

FINDINGS

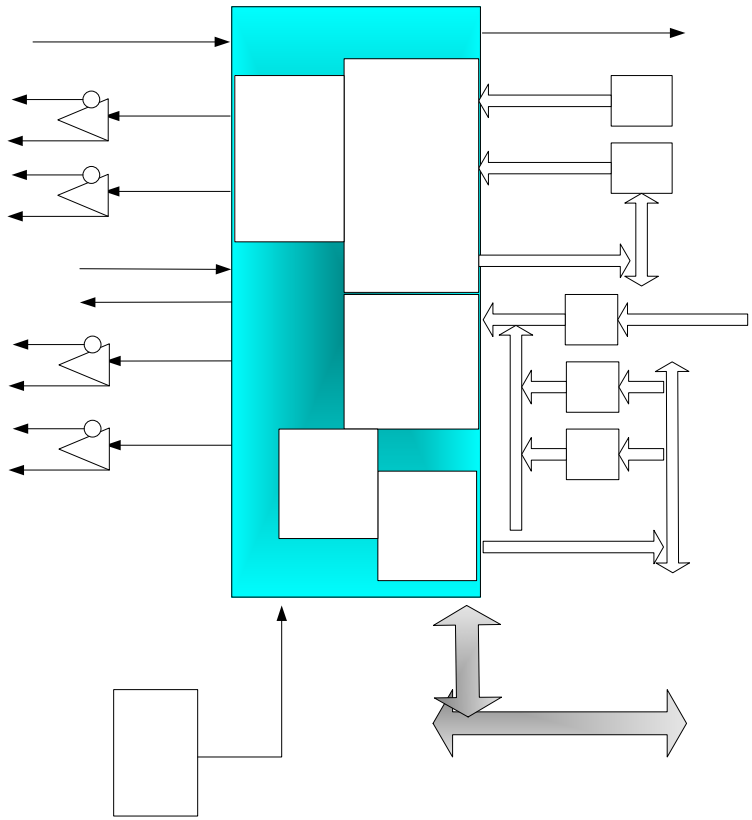
- Theme and Variation (RT parts)
  - Baseline (RTSX)
  - Baseline+ (free pins, Libero8.4)
  - Baseline RTAX1000
  - RTAX1000 (frame RAM)
  - RTAX2000 (frame RAM, ½ ROM)
  - RTAX4000 (frame RAM, ROM)



■1.16

FINDINGS

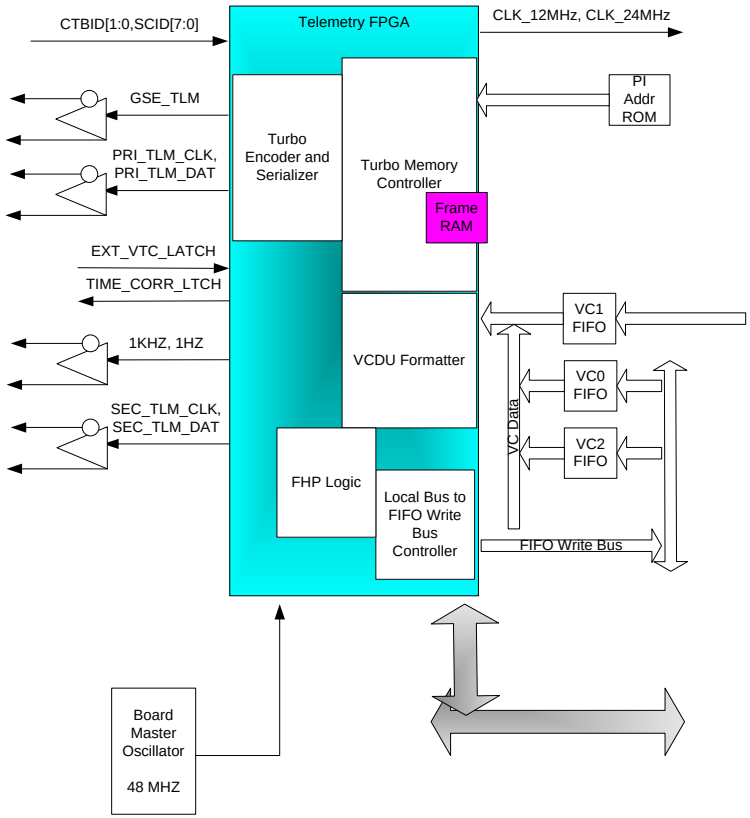
- Theme and Variation (RT parts)
  - Baseline (RTSX)
  - Baseline+ (free pins, Libero8.4)
  - Baseline RTAX1000
  - RTAX1000 (frame RAM)
  - RTAX2000 (frame RAM, ½ ROM)
  - RTAX4000 (frame RAM, ROM)



- 2.94

FINDINGS

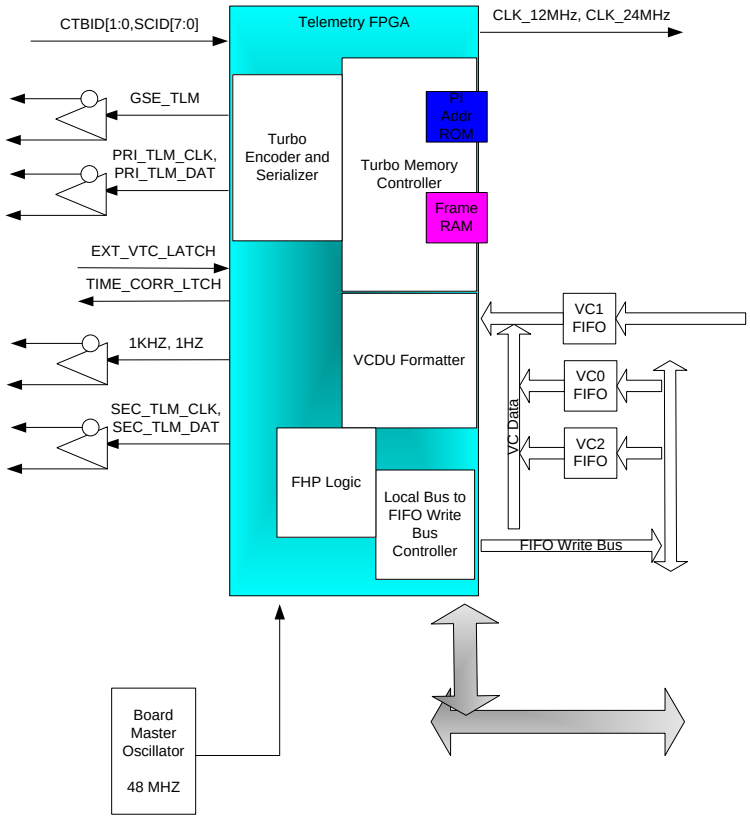
- Theme and Variation (RT parts)
  - Baseline (RTSX)
  - Baseline+ (free pins, Libero8.4)
  - Baseline RTAX1000
  - RTAX1000 (frame RAM)
  - RTAX2000 (frame RAM, ½ ROM)
  - RTAX4000 (frame RAM, ROM)



■3.02

FINDINGS

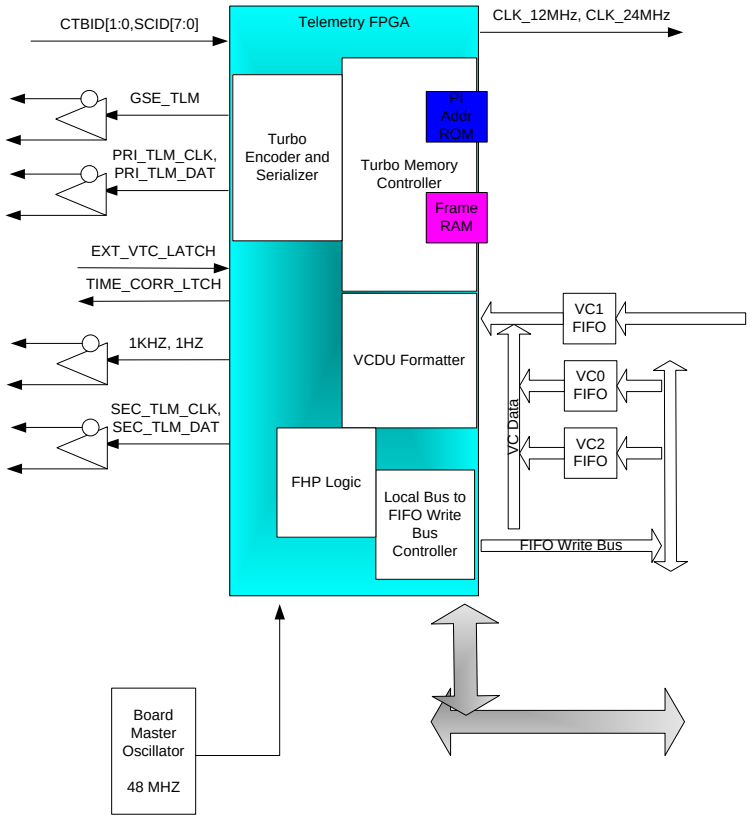
- Theme and Variation (RT parts)
  - Baseline (RTSX)
  - Baseline+ (free pins, Libero8.4)
  - Baseline RTAX1000
  - RTAX1000 (frame RAM)
  - RTAX2000 (frame RAM, ½ ROM)
  - RTAX4000 (frame RAM, ROM)



■1.63

FINDINGS

- Theme and Variation (RT parts)
  - Baseline (RTSX)
  - Baseline+ (free pins, Libero8.4)
  - Baseline RTAX1000
  - RTAX1000 (frame RAM)
  - RTAX2000 (frame RAM, ½ ROM)
  - RTAX4000 (frame RAM, ROM)



■1.13

AX vs SX Observations

- Baseline
- Baseline+
  - 116% faster due to releasing I/O, Libero 8.4
- Baseline RTAX1000
  - 294% faster due to RTAX
- RTAX1000 (RAM)
  - 302% faster, but only 8% faster than “RAM-less”; 128K SRAM gone
- RTAX2000 (RAM, ½ ROM)
  - 163% faster, functional change now requires FSW intervention, 2 parts gone
- RTAX4000 (RAM, ROM)
  - 113% faster, functional change now requires FSW intervention, 2 parts gone

RTAX vs RTSX Summary

- RTAX Strengths
  - Process point gives nice increase in speed
  - IO flops – GOTTA LOVE 'EM!
  - "Small RAMs" are GREAT!
  - ROM emulation is possible
- RTAX Weaknesses
  - "Large RAMs" incur a large performance penalty
    - FIFOs are much worse as size increases
  - Sizing AX to match "system" memory needs may be cost / performance prohibitive
    - i.e. External memories & I/O is cheaper option than AX4000!