Real Number Gradient Descent Decoding for Fault Tolerant Linear DSP Transforms

Oana Boncalo, Alexandru Amaricai
oana.boncalo@cs.upt.ro
Universitatea Politehnica Timisoara, Romania

Abstract

- We propose a fault tolerant mechanism for linear DSP applications, that relies on: (i) parity based real number error correction codes to store redundant information efficiently, (2) gradient descent correction loops for error estimation, and (3) fine grained check-pointing and roll-back for consistent redundant information encoding. The error correction is performed by a novel gradient descent symbol update method designed for real number parity based codes, which performs error extraction from the syndrome vector. We use the linear transform property of the application in order to compute the correction offset that is applied to the output data. The proposed solution is validated for a 512 point memory-based FFT architecture, augmented by the means of an LDPC code, for which an efficient way to embed the processing associated to the error correction code - encoding, syndrome check, decoding – is devised.

Iterative Decoding Method

- **Algorithm**

Algorithm GDSU decoding for Real number codes

Initialization:
set $x_k = \text{sign}(y_k)$ for k=[1,n]
Let $x = (x_1, x_2, ..., x_n)$
set $i = 0$
compute *syndrome*
while $(i \leq It_{max})$ and $(syndrome = false)$ **do**
Find symbol update position set F
$F = \{n' | n' = arg \max_{k \in [1,n]} \Delta_k^{GDR}\}$
for all $x_k \in F$ **do**
compute gradient reliability and $\delta_{n'}$ **update value acc. to (2)**
if $\delta_{n'} \neq 0$ **then**
Exit update loop in case single symbol update
compute new *syndrome*
set $i = i + 1$
Offloading: output $\leftarrow x$ =0

- **Ingredients**
  1. local energy function:

$$\Delta_k^{GDR} = zero(y_k - x_k) + \sum_{m' \in H(k)} zero(s_{m'}) \quad (1)$$

where $zero(x)$ is defined using bipolar representation.
  2. the decoder symbol updating rule is defined as:

$$j = arg \min_{m \in \Gamma_k} |s_m| \qquad \delta_k = sgnS(\Gamma_k)|s_j| \quad (2)$$

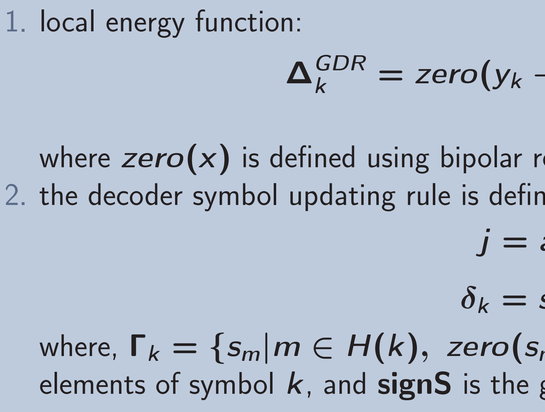
where, $\Gamma_k = \{s_m | m \in H(k),\ zero(s_m) = 1\}$, the set of all the syndrome vector elements of symbol k, and **signS** is the gradient direction reliability function.

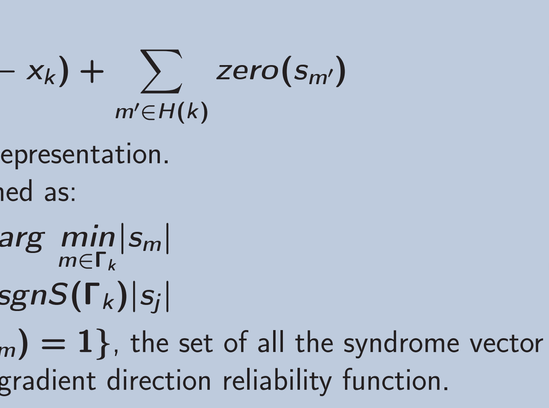
Decoding Performance Evaluation

- **Parity Check Code Selection: Quasi Cyclic Low Density Parity Check Code**
WiFi rate 1/2 standard code with expansion factor 32

$$\begin{pmatrix}
25 & 26 & 14 & -1 & 20 & -1 & 2 & -1 & 4 & -1 & -1 & 8 & -1 & 16 & -1 & 18 & 1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\
10 & 9 & 15 & 11 & -1 & 0 & -1 & 1 & -1 & -1 & 18 & -1 & 8 & -1 & 10 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\
16 & 2 & 20 & 26 & 21 & -1 & 6 & -1 & 1 & 26 & -1 & 7 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\
10 & 13 & 5 & 0 & -1 & 3 & -1 & 7 & -1 & -1 & 26 & -1 & -1 & 13 & -1 & 16 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 \\
23 & 14 & 24 & -1 & 12 & -1 & 19 & -1 & 17 & -1 & -1 & -1 & 20 & -1 & 21 & -1 & 0 & -1 & -1 & -1 & 0 & 0 & -1 & -1 \\
6 & 22 & 9 & 20 & -1 & 25 & -1 & 17 & -1 & 8 & -1 & 14 & -1 & 18 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 \\
14 & 23 & 21 & 11 & 20 & -1 & 24 & -1 & 18 & -1 & 19 & -1 & -1 & -1 & -1 & 22 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 \\
17 & 11 & 11 & 20 & -1 & 21 & -1 & 26 & -1 & 3 & -1 & -1 & 18 & -1 & 26 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & 0
\end{pmatrix}$$

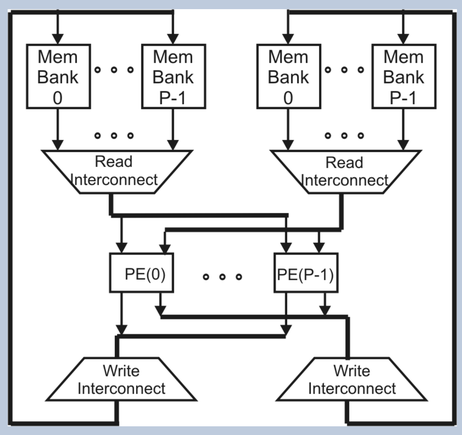
- **Performance**





Use Case

- Fault Tolerant FFT Architecture with Gradient Descent Correction Loop.



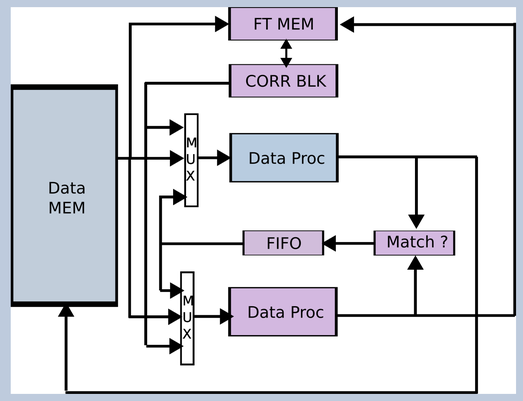


Figure: Hardening a memory based iterative FFT architecture with gradient descent correction loop

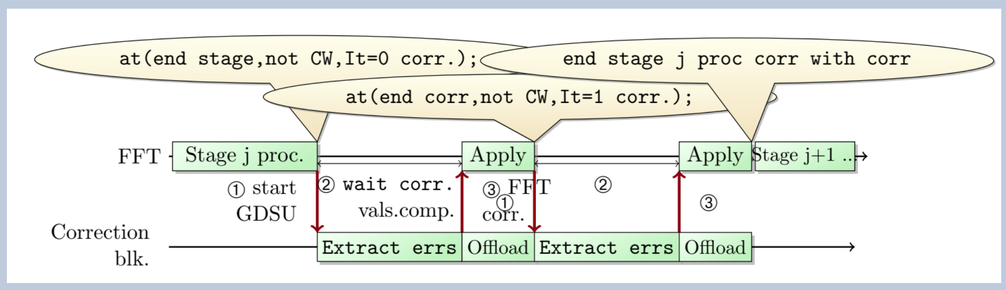
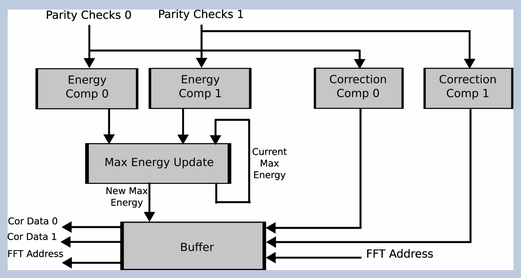


Figure: FFT with correction stage execution flow

- Fault tolerant ingredients:
  - Error correction block: computes correction information and position based on failed syndrome
  - Fault tolerant memory: ping-pong memory that stores parity information/syndrome and performs translation from FFT addresses to LDPC addresses



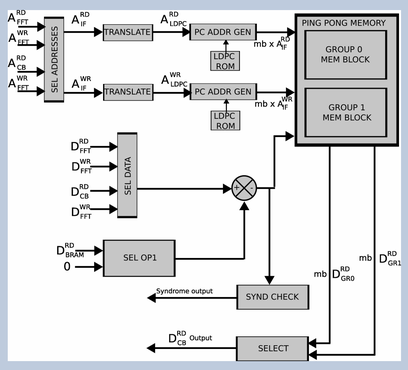


Figure: Error correction hardware ingredients: correction computation block and fault tolerant memory

Circuit	Slice LUT	Regs	DSP	BRAM (18 kb)
FT-FFT	6269	5566	4	3
GDSU	773	1364	0	0
FT Computation Block	1887	1979	0	0
Baseline FFT	307	206	4	3

Table: Synthesis results for the FT implementation of the 512 point memory based FFT

- Implementation: (1) RTL Verilog HDL (2) Matlab HLS with HDL Coder

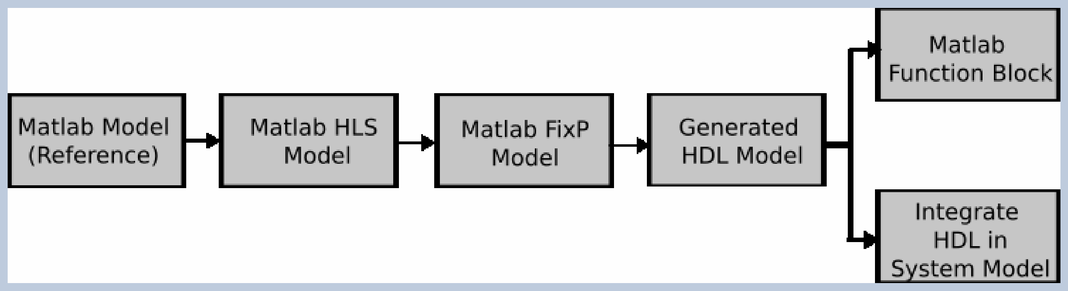


Figure: Matlab HLS Flow

Future work

- Extending to other field of applications, such as High Performance Computing or Image Processing
- Error correction code design: (i) higher rate codes (ii) real number codes (containing non-binary values in the parity-check matrix)
- Independent parity/redundant processing path
- Automation of the fault tolerant design process

ESA Early Technology Development
Project ESA 4000134184/21/NL/GLC