A Multi-Dimensional Logarithmic Number System based CPU

Mahzad Azarmehr
Supervisor: Dr. R. Muscedere

University of Windsor
Electrical and Computer Engineering

Second Seminar, Fall 2006

Outline

- **Multidimensional Logarithmic Number System (MDLNS)**
  - **Introduction**
  - **Specifications**
  - **Calculations**
  - **Conversion**
- **Proposed Research**
- **2DLNS CPU Design and Implementation**
  - **Architecture**
  - **Instruction Set Architecture**
  - **Organization**
  - **Multiplier and Accumulator Unit**
- **Filterbank Application**
  - **Specifications**
  - **2DLNS Program and Results**
- **Future Work**
- **References**

MDLNS (Introduction)

- DSP systems manipulates signals as a sequence of numbers, and usually require massive arithmetic computations to perform algorithmic processing such as modulation or filtering
- Multipliers are fundamental units in most DSP applications and are the most hardware consuming components
- In order to simplify multiplication, some alternative number systems have been considered

MDLNS (Introduction)

- Desired characteristics of a number system used in DSP
  - Smaller size of corresponding representations
  - More error-free mapping approximations
  - Less complexity of arithmetic operations
  - More accurate representation of smaller values (like less than one coefficient values in a filter)

MDLNS (Representation)

- A representation of the real number X , in the form:

$$X = \sum_{i=1}^{n} s_i \prod_{j=1}^{b} P_j^{e_j^{(i)}}$$

- where s_i is sign (-1,0,1), p_j is a real, and $e_j^{(i)}$ are integers, is called an n digit multi-dimensional logarithmic representation of X

- b is the number of bases used (at least two) and the first one, p_1, is always be assumed to be 2

MDLNS (Properties)

- Larger dynamic range
- More degrees of freedom by virtue of having two or more orthogonal bases and the ability to use multiple digits
- A significant reduction in hardware complexity
- Simplified mathematical computation

MDLNS (Calculations)

- Multiplication and Division

Given a single-digit representation of (one-bit sign):

$x = \{s_x, a_x, b_x\}$ and $y = \{s_y, a_y, b_y\}$

$$x.y = \{\, s_x \ xor \ s_y \,,\, a_x + a_y \,,\, b_x + b_y \,\}$$

$$x \div y = \{\, s_x \ xor \ s_y \,,\, a_x - a_y \,,\, b_x - b_y \,\}$$

MDLNS (Calculations)

- Addition and Subtraction

$$2^{a}x \,.\, D^{b}x + 2^{a}y \,.\, D^{b}y = (2^{a}x \,.D^{b}x).(1 + 2^{a}y^{-a}x \,.\, D^{b}y^{-b}x\ )$$

$$\approx (2^{a}x \,.D^{b}x).\Phi(a_y - a_x\ ,\ b_y - b_x)$$

$$2^{a}x \,.\, D^{b}x - 2^{a}y \,.\, D^{b}y = (2^{a}x \,.D^{b}x).(\ 1 - 2^{a}y^{-a}x \,.\, D^{b}y^{-b}x\ )$$

$$\approx (2^{a}x \,.D^{b}x).\Psi(a_y - a_x\ ,\ b_y - b_x)$$

The operators Φ and Ψ are lookup tables that store the precomputed 2DLNS values

- MDLNS Addition and Subtraction are not implemented in the CPU

MDLNS (Conversion)

- There is no functional relationship between Binary and MDLNS representations
- Conversions between Binary and MDLNS representations are efficiently implemented with Range Addressable Look-up Tables (RALUT)
- The proper second base (optimal base) should be selected in accordance to the specific design considerations
- The virtue of using multiple digits makes appropriate size of RALUTs reasonably small

Proposed Research

- The main goal of this research project is design and implement a 2DLNS-based Central Processing Unit (CPU)
- This CPU in addition to performing most traditional arithmetic and logic operations, is also able to perform some particular tasks including conversions between 2DLNS and Binary
- This CPU will facilitate using 2DLNS in every other research work as well as practical DSP applications including filtering, modulation,….

Proposed Research

- A custom design for a Filterbank based on 2DLNS has already been implemented
- The developed Filterbank uses eight symmetric FIR filters in order to split and process the signal in different frequency bands
- Implementing a Filterbank application on the 2DLNS CPU effectively demonstrates its efficiency
- The 2DLNS CPU should also be capable of implementing the Filterbank application and other generic DSP architectures

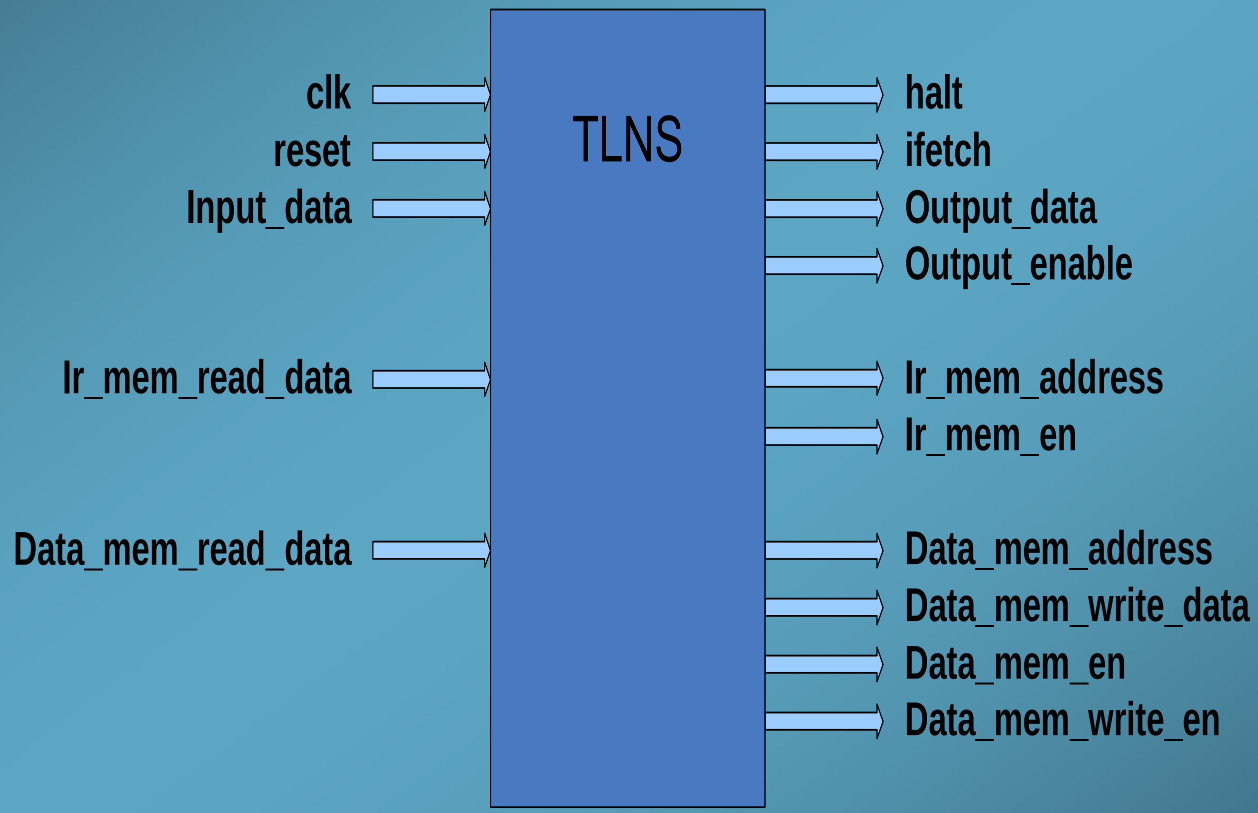
TLNS CPU (Architecture)

- This CPU has been developed based on a relatively simple Reduced Instruction Set Computer (RISC) architecture
- TLNS has 16 general purpose registers, as well as some special purpose registers including a Program Counter (PC) and a Memory Address Register (MAR)
- Separate Instruction memory and Data memory have been considered, both of which are also addressable through controller unit

TLNS CPU (Architecture)

- Based on custom design Filterbank work, considering B = 6 and R = 5 provides most error free mapping of binary data
- In order to achieve the desired precision, 2-digit 2DLNS representations are considered
- Considering one-bit sign representation, two-digit 2DLNS data can be represented with 24 bits
- Instruction length, register size, data buses, and memory words have all been designed in 24 bits

TLNS CPU (External Ports)



TLNS CPU (Instruction Set Architecture)

- Instruction Set includes Instructions for:
  - Transferring data to and from memory
  - Performing arithmetic and logical operations
  - Transferring control within a program
  - Some special instructions based on 2DLNS
- There are three different Instruction types

I - Type	opcode	r_s	r_d	Immed-10	
R - Type	opcode	r_{s1}	r_{s2}	r_d	Sp-func
J - Type	opcode	Immed-18			

TLNS CPU (Instructions)

- TLNS Instructions for arithmetic and logical operations

Instruction	Description
add, addu, sub, subu	Signed and unsigned add and subtract
addi, addui, subi, subui	an immediate value
sxx, sxxu	Set if condition (xx: eq,ne,lt,le,gt,ge)
sxxi, sxxui	an immediate value
lhi	Load high immediate
nop	No operation
and, or, xor	Bitwise logical and, or, exclusive-or
andi, ori, xori	an immediate value
sll, srl, sra	Shift left-logical, right-logical, right-arith
slli, srli, srai	an immediate value

TLNS CPU (Instructions)

- TLNS Instructions for transferring data

Instruction	Description
lw	Load word
sw	Store word

- TLNS control transfer Instructions

Instruction	Description
beqz	Branch if register equal to zero
bnez	Branch if register not equal to zero
j, jal	Jump (and link) unconditional
jr, jalr	Jump (and link) register
halt	Halt execution

TLNS CPU (Instructions)

- TLNS Special Instructions

Instruction	Description
tbc	2DLNS to Binary Conversion
btc	Binary to 2DLNS Conversion
inpt	Read data from input register to register file
oupt	Read data from register file to output register
mult	2DLNS multiplication
mac	2DLNS Multiply and Accumulation
filter	FIR filter (repeated MAC)

TLNS CPU (Filter Instruction)

- Filter Instruction

opcode	r_{s1}	r_{s2}	filter _sym	coef _sym	order

rs_1 is the register which contains the data and coefficient start addresses in memories

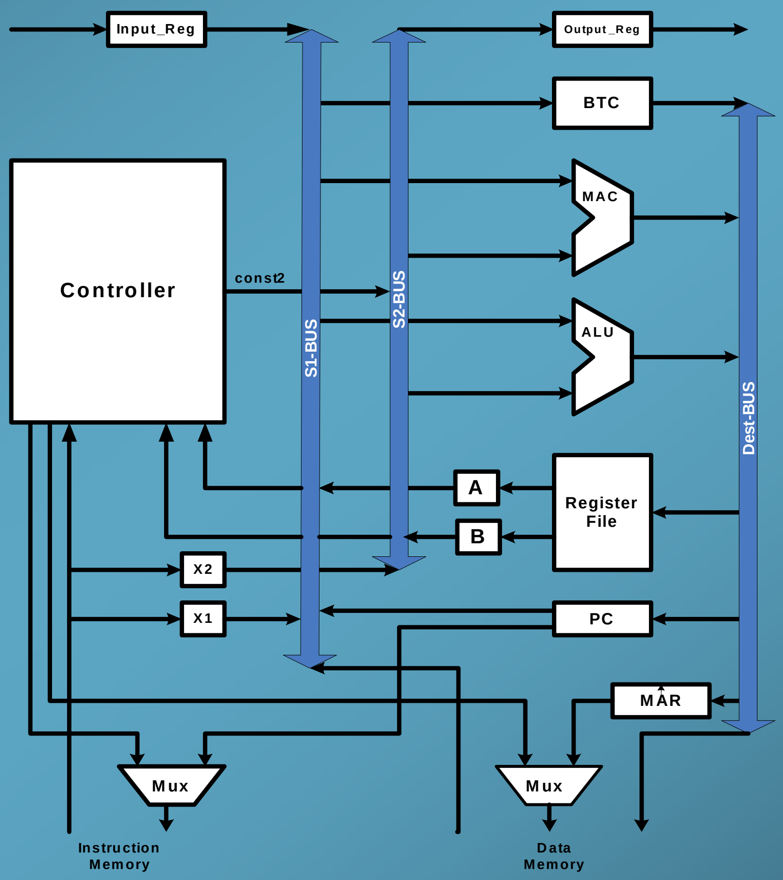
rs_2 is the register which contains the data address range in memory

filter_sym shows if the filter has a dual, and the type of symmetry

coef_sym shows if the coefficients are symmetric

order is the order of the filter, up to 128

TLNS CPU (Organization)



TLNS CPU (Components)

- ALU performs arithmetic and logical operation on register contents, or the contents of one register and an immediate value
- Register File consists of 16 registers, one can be written at a time, while two can be read simultaneously
- Extenders, are used to extend Instruction immediate values (10 or 18 bits) to 24 bits, as well as direct data from Instruction memory to CPU
- Data Memory is used for loading data to the CPU or storing data from the CPU, while Instruction Memory contains programmable Instructions and permanent data

TLNS CPU (Components)

- The Multiplexers, make it possible to address Data Memory and Instruction Memory from two different sources
- The Binary / 2DLNS converters, are used to convert data into the most efficient representation; TBC is embedded into MAC unit
- The Multiply and Accumulator unit, is used to multiply two 2DLNS values, as well as multiply and accumulate the result in case of multiple sequences of data
- The Controller, is a procedure based state machine which controls all CPU tasks in Fetch, Decode, and Execute stages

TLNS Multiplier and Accumulator (MAC) unit

- A MAC multiplies corresponding elements of two sequences of numbers $\{X_i\}$ and $\{Y_i\}$ and accumulates the sum of the products:

$$P = \sum_{i=1}^{n} X_i . Y_i$$

- The implementation of a MAC needs intensive computation and consumes a significant amount of hardware resources. There is always a traditional trade off of size versus speed.

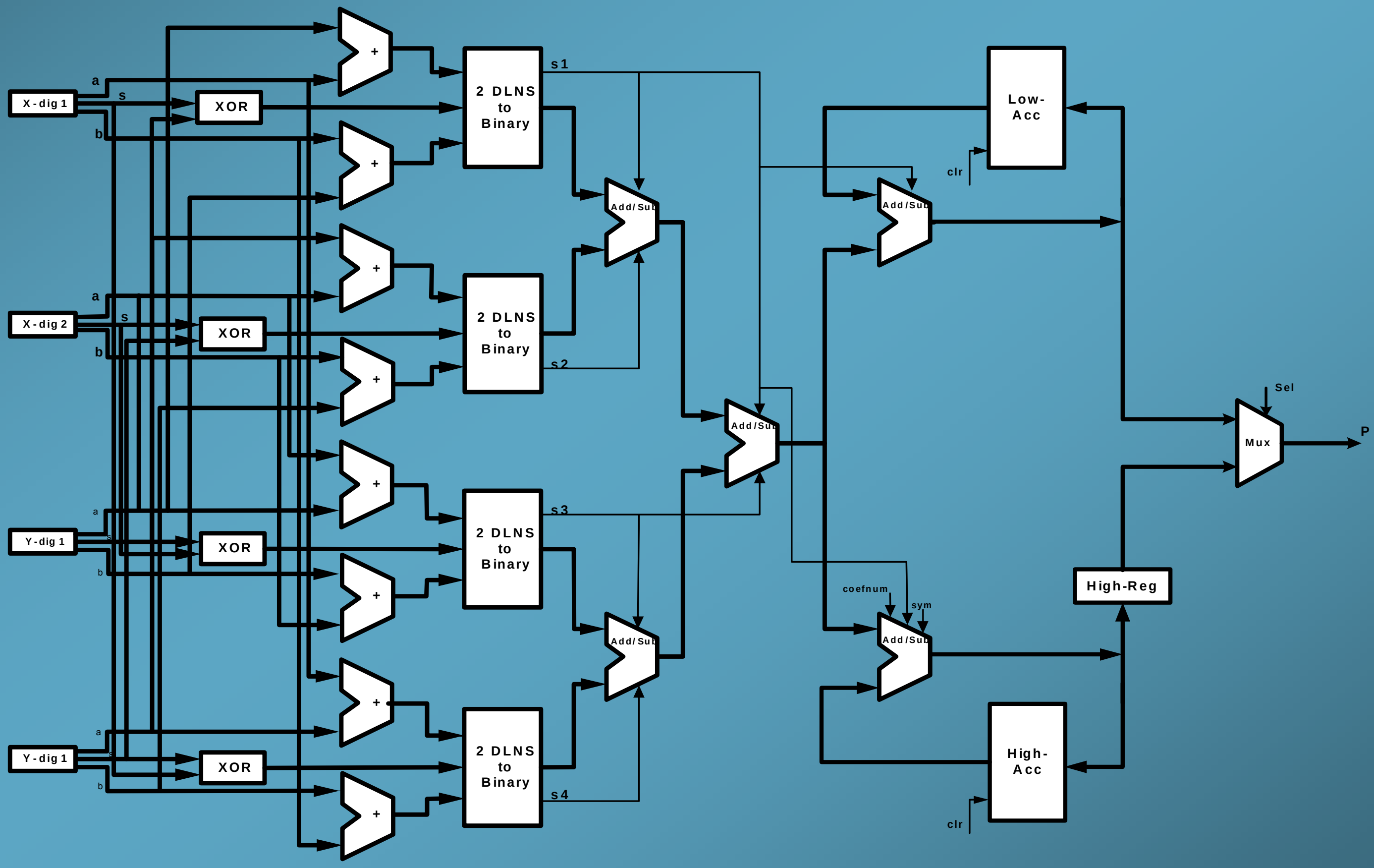
TLNS Multiplier and Accumulator (MAC) unit

- MAC is the fundamental unit in Finite Impulse Response (FIR) filters
- Multiplications are performed in 2DLNS, but partial products are converted to Binary for addition
- In filter applications, coefficients and data are stored in separate memories, therefore they can be read into the MAC unit, and processed in one clock cycle

TLNS Multiplier and Accumulator (MAC) unit

- In Symmetric filters coefficients are duplicate in magnitude and based on even or odd symmetry, every other coefficients should be negated
- The implemented MAC unit multiplies each pair of data/coefficients as absolute values, and accumulates the results with final sign separately. Therefore, it is capable to process dual filters at the same time
- One extra bit is considered for each Adder/Subtracter unit, so an overflow never occurs

TLNS MAC unit RTL Organization



Filterbank Application (Definition)

- As an application, a Filterbank architecture has been programmed and run on the 2DLNS CPU
- The function of a Filterbank is to split the input signal into several frequency bands, where custom algorithms are performed independently on the output of each band, and then merged back into one signal
- The program is dynamic, which will allow runtime loading of the parameters such as filter order, symmetry of filters, symmetry of coefficients, and the addresses of data and coefficients in memory

Filterbank Application (Specifications)

- The design specifications of the Filterbank are pass band ripple of 0.01 dB and stop band attenuation of 60 dB for all filters
- The advantage of generating symmetrical filters is the overall magnitude response is perfectly flat across the whole frequency range
- The digital Filterbank has been designed to obtain an arbitrary magnitude response with exact linear phase
- Using D = 0.92024380912663017, a pass band ripple of 0.0137 dB, and a stop band attenuation of 58.2 dB are obtained

Filterbank Application (TLNS Program)

```
X"201404",  --1         addi    r0, r5, dstart                 Preparing addresses
X"2429FF",  --2         addi    r0, r10, dend
X"3C11FF",  --3         lhi     r0, r4, dend
X"0114E5",  --4         or      r4 ,r5, r3
X"20044E",  --5         addi    r0, r1, data_address
X"200840",  --6   next  addi    r0, r2, coef_address
X"50600E",  --7         slli    r1, r8, E
X"0208A5",  --8         or      r8, r2, r2
X"401800",  --9         inpt    r0, r6, 0                      Entering Data
X"199C00",  --a         btc     r6, r7                         Converting Data
X"AC5C00",  --b         sw      M[r1], r7                      Storing Data
X"548DCB",  --c         filter  r2, r3, coef sym, even, 75     Filters 0,7
X"202400",  --d         addi    r0, r9, band_tag               Writing output
X"533004",  --e         slli    r12, r12, 4
X"027325",  --f         or      r9, r12, r12
X"443000",  --10        oupt    r0, r12, 0
X"202C07",  --11        addi    r0, r11, dual_band_tag
X"537404",  --12        slli    r13, r13, 4
X"02F765",  --13        or      r11, r13, r13
X"443400",  --14        oupt    r0, r13, 0
X"208826",  --15        addi    r2, r2, next_coef_address      Next set of Coefficients
X"548DCB",  --16        filter  r2, r3, coef sym, even, 75     Filters 1,6
X"202401",  --17        addi    r0, r9, band_tag               Writing output
X"533004",  --18        slli    r12, r12, 4
X"027325",  --19        or      r9, r12, r12
```

Filterbank Application (TLNS Program)

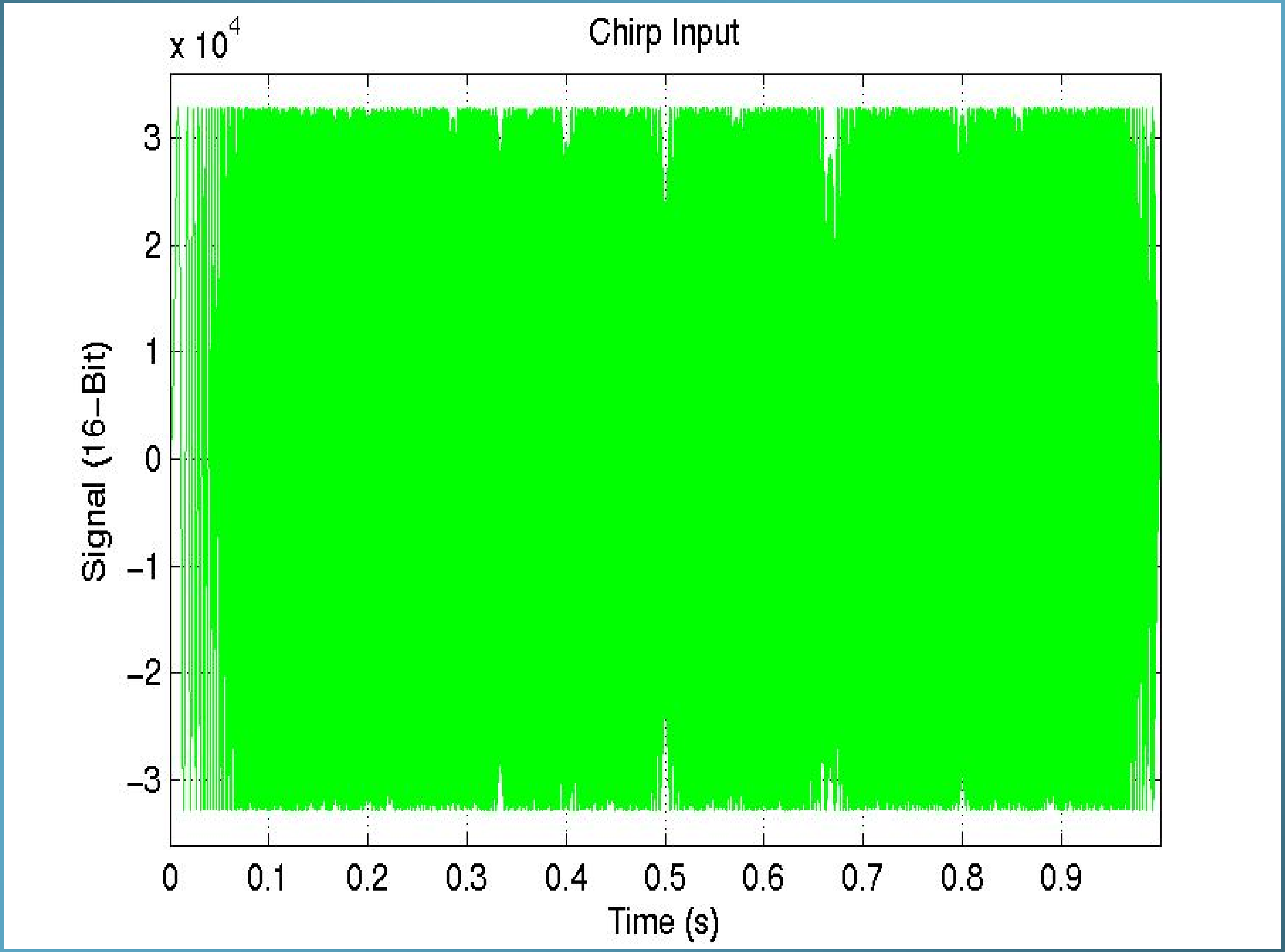
```
X"443000",   --1a    oupt    r0, r12, 0
X"202C06",   --1b    addi    r0, r11, dual_band_tag
X"537404",   --1c    slli    r13, r13, 4
X"02F765",   --1d    or      r11, r13, r13
X"443400",   --1e    oupt    r0, r13, 0
X"208826",   --1f    addi    r2, r2, next_coef_address     Next set of Coefficients
X"548DCB",   --20    filter  r2, r3, coef sym, even, 75    Filters 2,5
X"202402",   --21    addi    r0, r9, band_tag              Writing output
X"533004",   --22    slli    r12, r12, 4
X"027325",   --23    or      r9, r12, r12
X"443000",   --24    oupt    r0, r12, 0
X"202C05",   --25    addi    r0, r11, dual_band_tag
X"537404",   --26    slli    r13, r13, 4
X"02F765",   --27    or      r11, r13, r13
X"443400",   --28    oupt    r0, r13, 0
X"208826",   --29    addi    r2, r2, next_coef_address     Next set of Coefficients
X"548DCB",   --2a    filter  r2, r3, coef sym, even, 75    Filters 3,4
X"202403",   --2b    addi    r0, r9, band_tag              Writing output
X"533004",   --2c    slli    r12, r12, 4
X"027325",   --2d    or      r9, r12, r12
X"443000",   --2e    oupt    r0, r12, 0
X"202C04",   --2f    addi    r0, r11, dual_band_tag
X"537404",   --30    slli    r13, r13, 4
X"02F765",   --31    or      r11, r13, r13
X"443400",   --32    oupt    r0, r13, 0
```

Filterbank Application (TLNS Program)

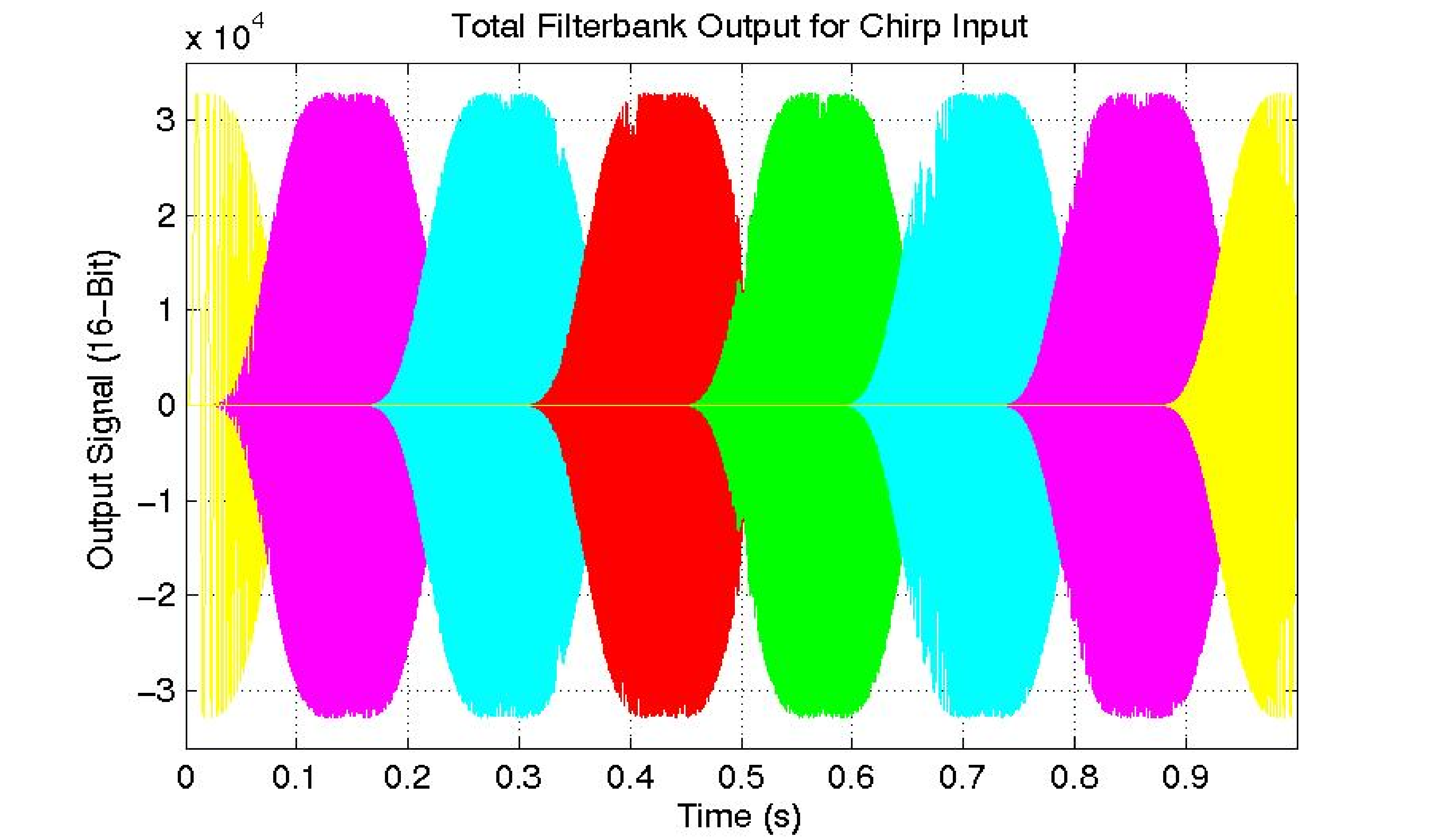
```
X"204401",   --33         addi   r1, r1, 1          Next Data address
X"0069AB",   --34         sgt    r1, r10, r6
X"118001",   --35         beqz   r6, cont
X"214400",   --36         addi   r5, r1, 0
X"0BFFCE",   --37   cont  j      next
```

- This program has been written in 55 words and just with 12 different instructions from the Instruction Set
- Filtering requires 82 clock cycles for 75th order filters
- The rest Instructions take 95 clock cycles in total
- In order to keep the CPU generic, no special purpose commands are used to write the Filterbank outputs

Filterbank Application (Input Signal)



Filterbank Application (Output Signal)



TLNS CPU (Synthesis Results)

Area	**Optimized for area**	**Clock-frequency 14 MHz**	**Clock-frequency 50 MHz**
***Total cell Area* (µm)^2**	**387928**	**580443**	**549216**

Future Work

- More Instructions can be added to Instruction Set, in order to speed up some particular applications
- Some special purpose instructions can be implemented
- The current Instruction Set may be optimized in order to reduce number of clock cycles, and increase the speed
- Micro codes in the procedures of the Controller state machine may be improved to reduce the number of states for each Instruction execution

References

[1] David A. Patterson, John L. Hennessy, Computer Organization and Design, Third edition, MORGAN KAUFMANN PUBLISHERS, 2005.

[2] John D. Carpinelli, Computer Systems Organization & Architecture, Addison Wesley Longman, Inc., 2001.

[3] Petter J. Ashenden, The Designer's Guide to VHDL, Second edition, MORGAN KAUFMANN PUBLISHERS, 2002.

[4] Douglas J. Smith, HDL Chip Design, Eighth edition, Doone Publications, 2000.

[5] Roberto Muscedere, "Difficult Operations in the Multi-Dimensional Logarithmic Number System ", PhD Thesis, University of Windsor, 2003

[6] G. A. Jullien, V. S. Dimitrov, B. Li, W. C. Miller, A. Lee, and M. Ahmadi, "A Hybrid DBNS Processor for DSP Computation", Proceeding of the 1999 IEEE International Symposium on Circuits and Systems, vol 1., pp. 5-8, Orlando, Florida, 1999

[7] V. S. Dimitrov, G. A. Jullien, and W. C. Miller, "Theory and Applications of the Double-Base Number System", IEEE Transactions on Computers, vol. 48, No. 10, pp. 1098-1107, October 1999

References

[8] R. Muscedere, G. A. Jullien, V. S. Dimitrov and W. C. Miller, “Non-Linear Signal Processing using Index Calculus DBNS Arithmetic”, Proceedings of the 2000 SPIE conference on Advanced Algorithms and Architectures in Signal Processing, pp. 247-257, San Diego, August 2000

[9] J. Eskritt, R. Muscedere, G. A. Jullien, V. S. Dimitrov and W. C. Miller, “A 2-digit DBNS filter architecture”, IEEE workshop on Signal Processing, Louisiana, October 2000

[10] V. S. Dimitrov, J. Eskritt, L. Imbert, G. A. Jullien and W. C. Miller, “The use of the Multi-Dimensional Logarithmic Number System in DSP Applications”, Proceedings of the 15th IEEE Symposium on Computer Arithmetic, pp. 247-256, June 2001

[11] R. Muscedere, G. A. Jullien, V. S. Dimitrov and W. C. Miller, “Efficient Conversion From Binary to Multi-Digit Multi-Dimensional Logarithmic Number Systems using Arrays of Range Addressable Look-Up Tables”, Proceedings of the 2002 IEEE conference on Application Specific Systems, Architectures, and Processors, pp. 130-138

[12] H. Li, “A 2-Digit Multi-dimensional Logarithmic Number System Filterbank Processor for a Digital Hearing Aid”, M.A.Sc. Thesis, University of Windsor, 2003

References

[13] H. Li, R. Muscedere, G. A. Jullien, and V. S. Dimitrov, "The Application of 2-D Logarithms to Low-Power Hearing-Aid Processors", Proc. 45th IEEE Int'l Midwest Symp. Circuits and Systems, vol. 3,pp. 13-16, Aug.2002

[14] V. S. Dimitrov, G. A. Jullien, and K. Walus, "Digital Filtering Using the Multidimensional Logarithmic Number System"

[15] S. J. Eskritt, 2001, "Inner Product Computational Architectures Using the Double Base Number System", M.A.Sc. Thesis, University of Windsor, 2001

A Multi-Dimensional Logarithmic Number System based CPU

Questions and Comments