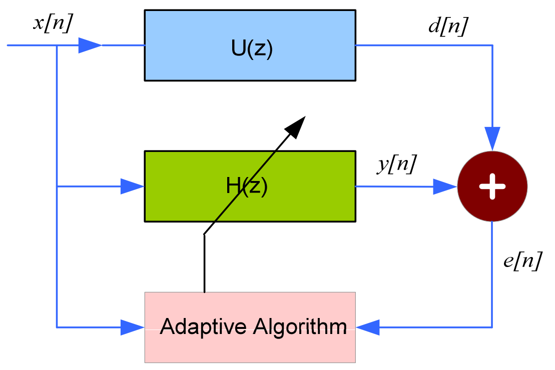
Micro-Programmed Adaptive Filtering Applications

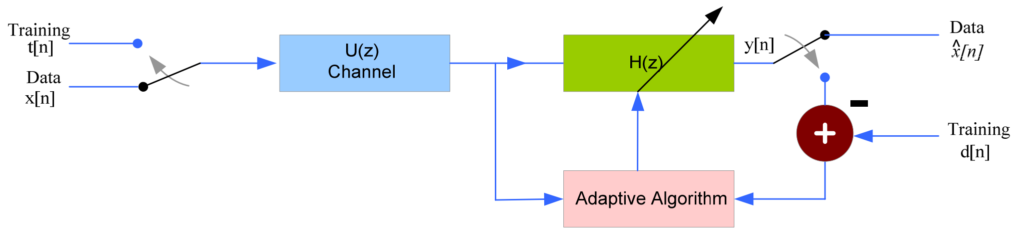
Lecture 11

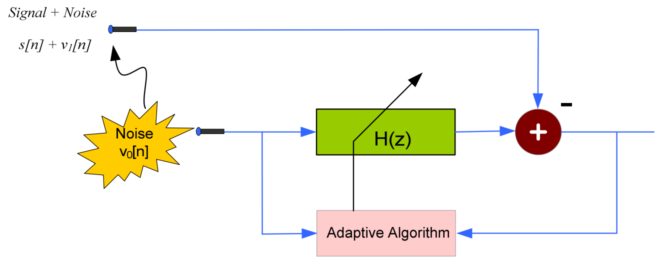
Dr. Shoab A. Khan

Adaptive Filter Configurations

- System Identification
- Inverse System Modeling
- Acoustic Noise Cancellation
- Linear Prediction

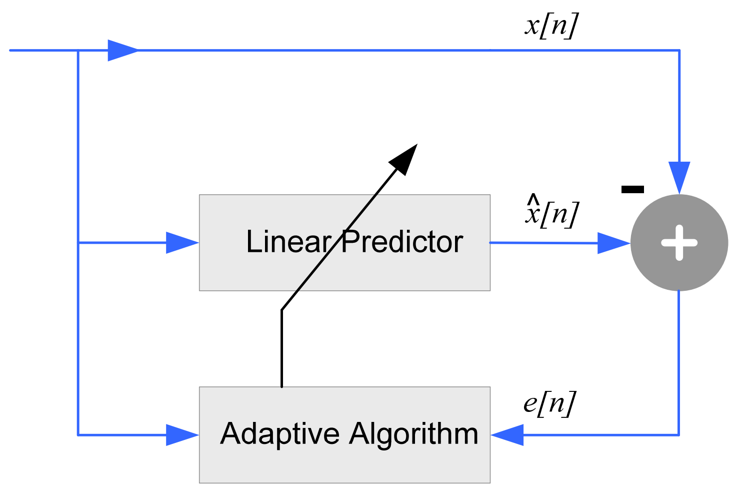






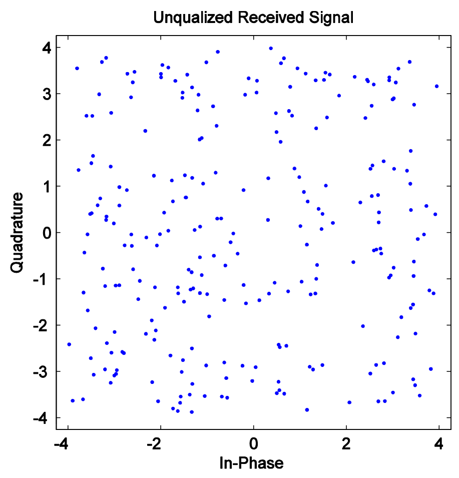
Adaptive Algorithms

- An ideal adaptive algorithm computes coefficients of the filter by minimizing an error criterion
- Least Mean Square (LMS) Algorithm
- Normalized LMS Algorithm
- Block LMS

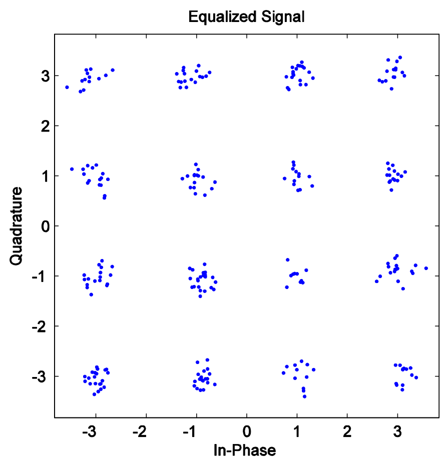


Channel Equalizer using NLMS

- Theory
- Example: NLMS Algorithm to Update Coefficients



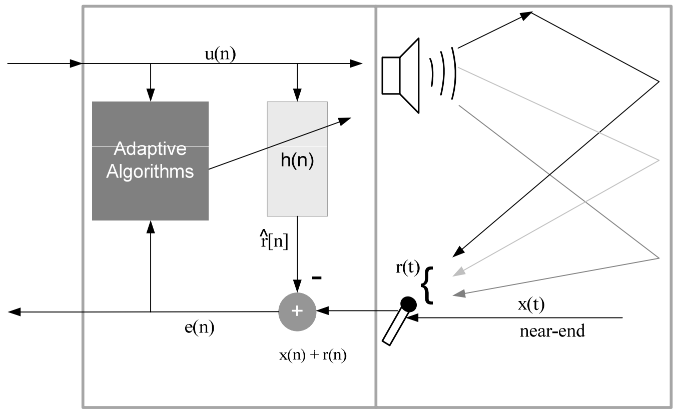
(a)

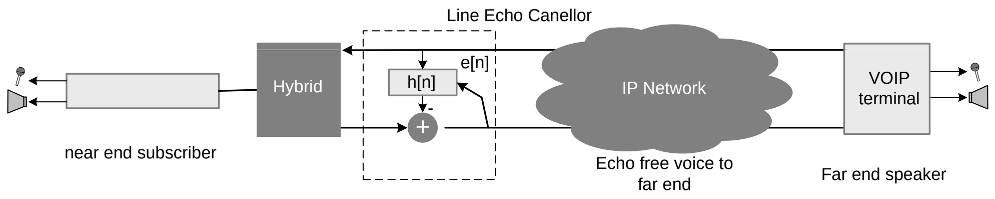


(b)

Echo Canceller

- Acoustic Echo Canceller
- Line Echo Cancellation (LEC)





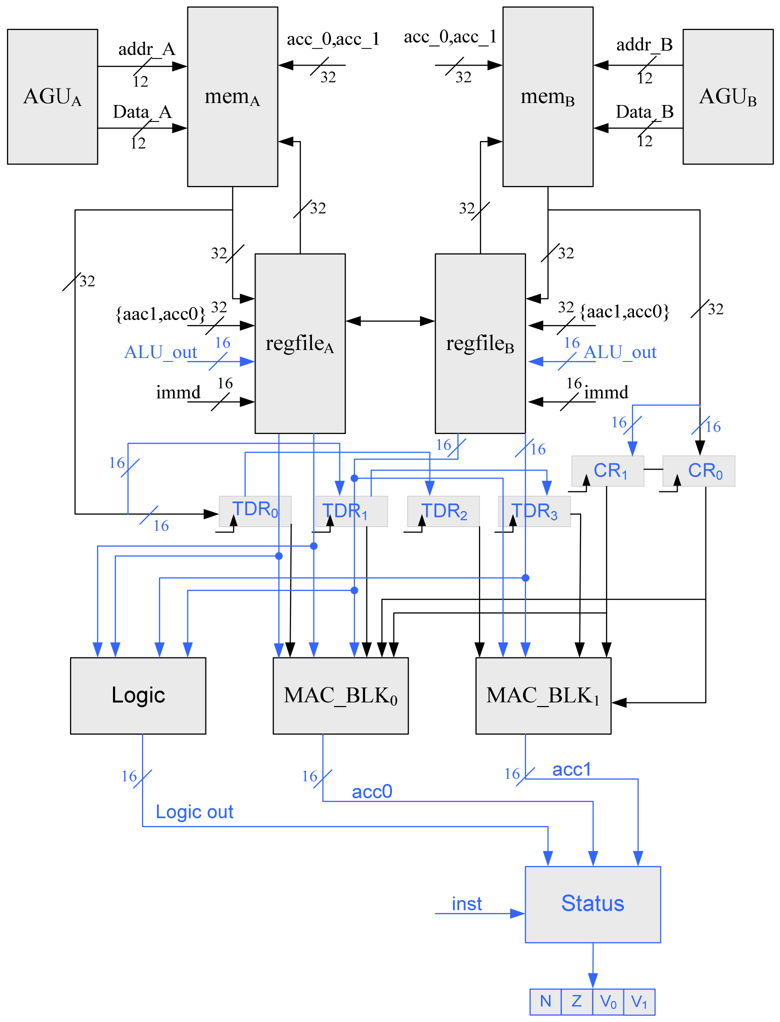
Adaptive Algorithms with Micro-programmed State Machines

- It is evident from the discussion so far in this chapter that, although coefficient adaptation is the main component of any adaptive filtering algorithm, several auxiliary computations are required for complete implementation of the associated application
- Example: LEC Micro-coded Accelerator
  - Top Level Design
  - Datapath/Registers
    - Register Files
    - Special Registers
    - Arithmetic and Logic Operations
    - Status Register
- Address Generation Unit
- Program Sequencer
- Repeat Instruction
- Conditional Branch Instruction
- Condition Multiplexer (C_MUX)

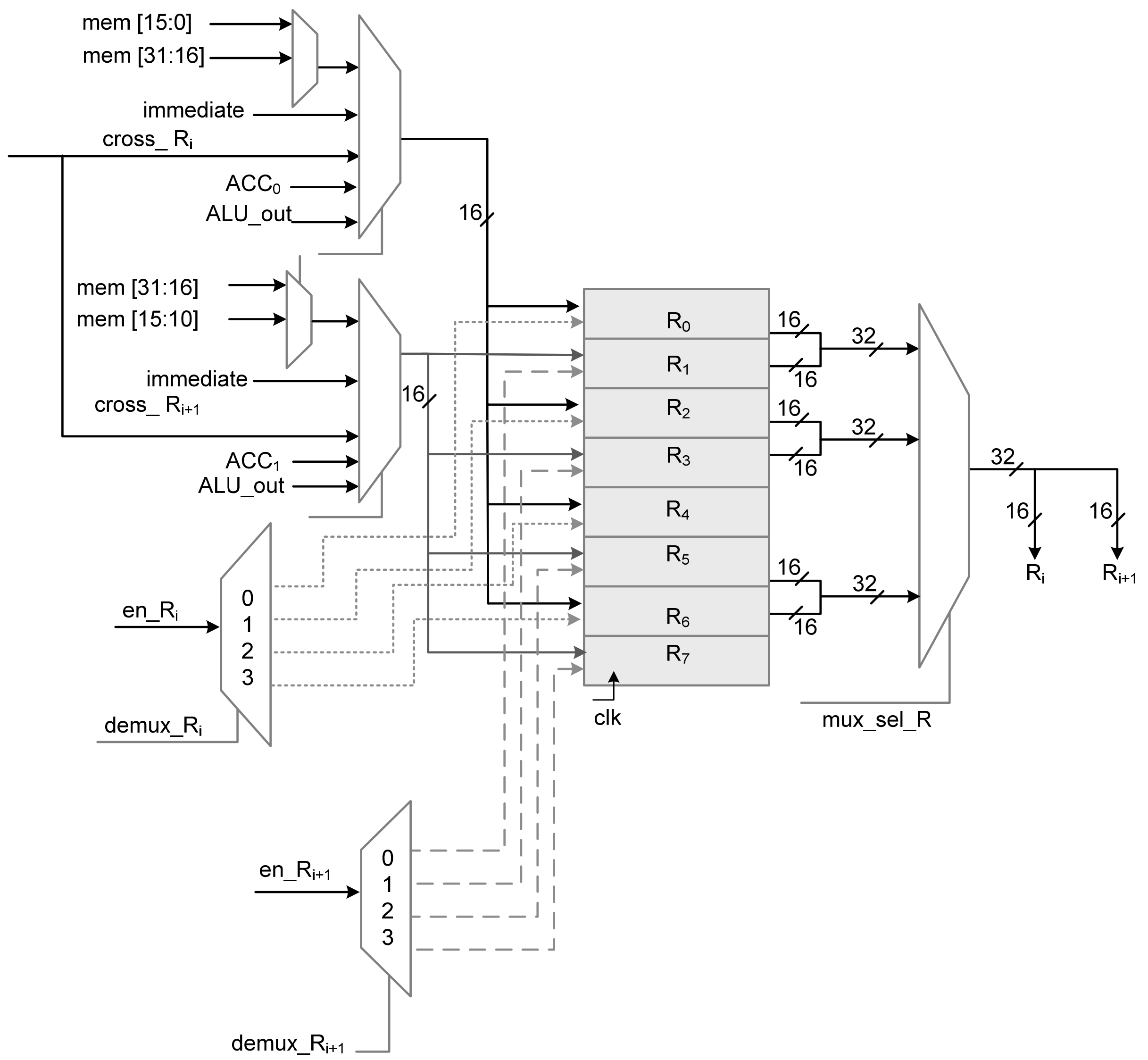
Contd….

- Next Address Logic
- Data Memories
- Instruction Set Design
- Single Load Long Instruction
- Parallel Load Long Instructions
- Single and Parallel Load Short Instruction
- Single Long Store Instruction
- Parallel Long Store Instruction
- Single and Parallel Load Short Instruction
- Pre-increment Decrement Instruction

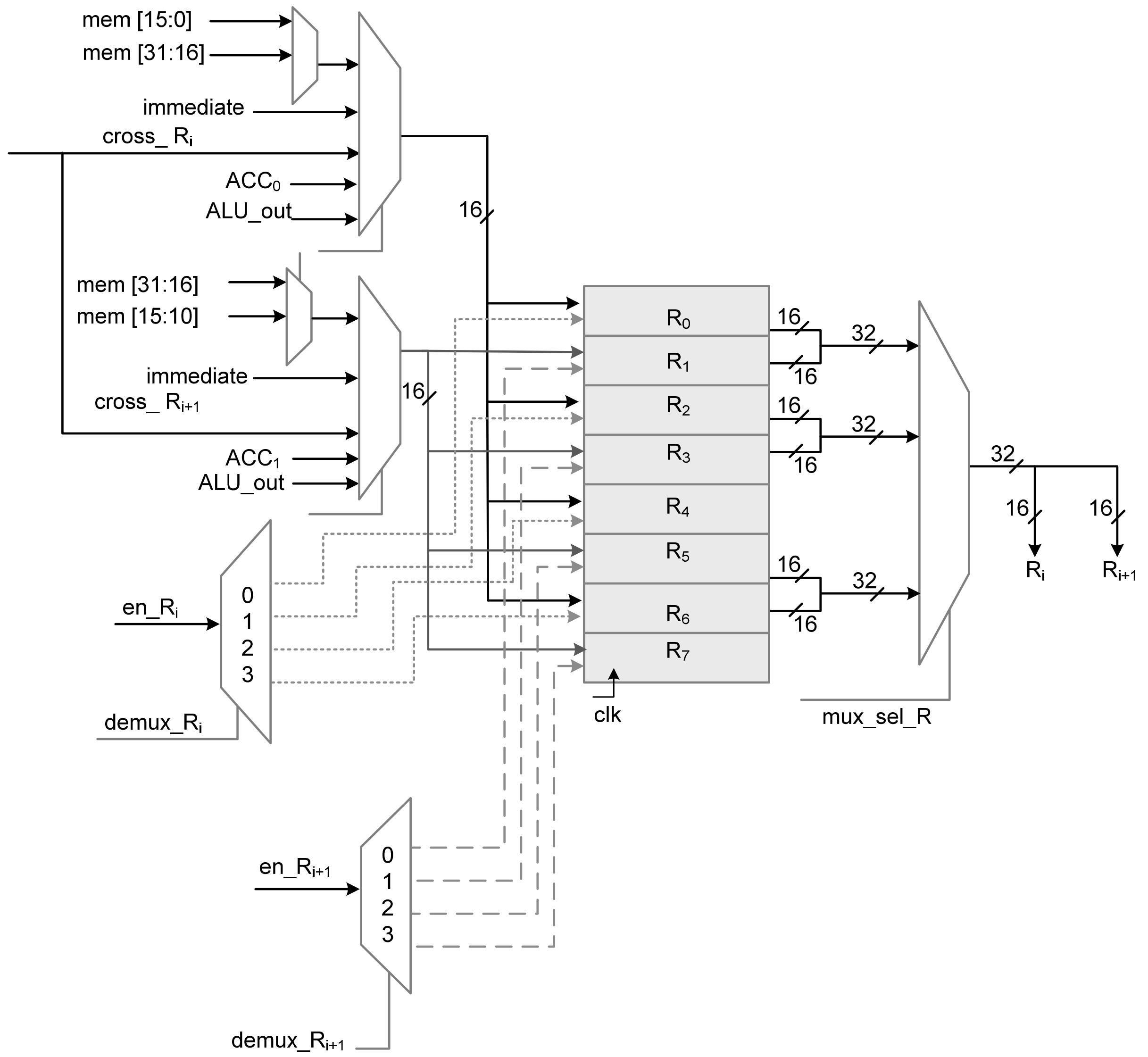
Datapath of the multi-channel line echo canceller



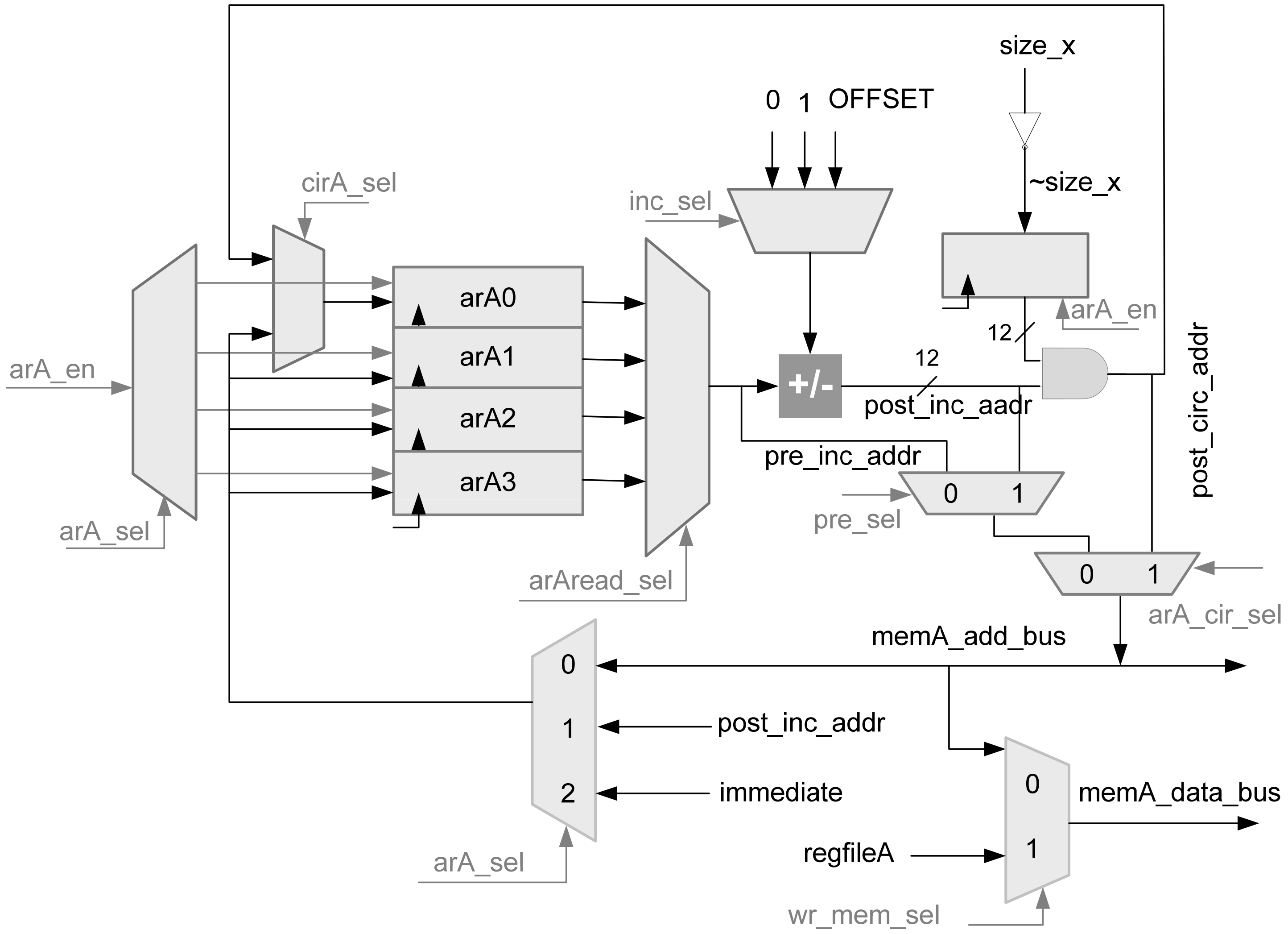
RTL design of one of the register files



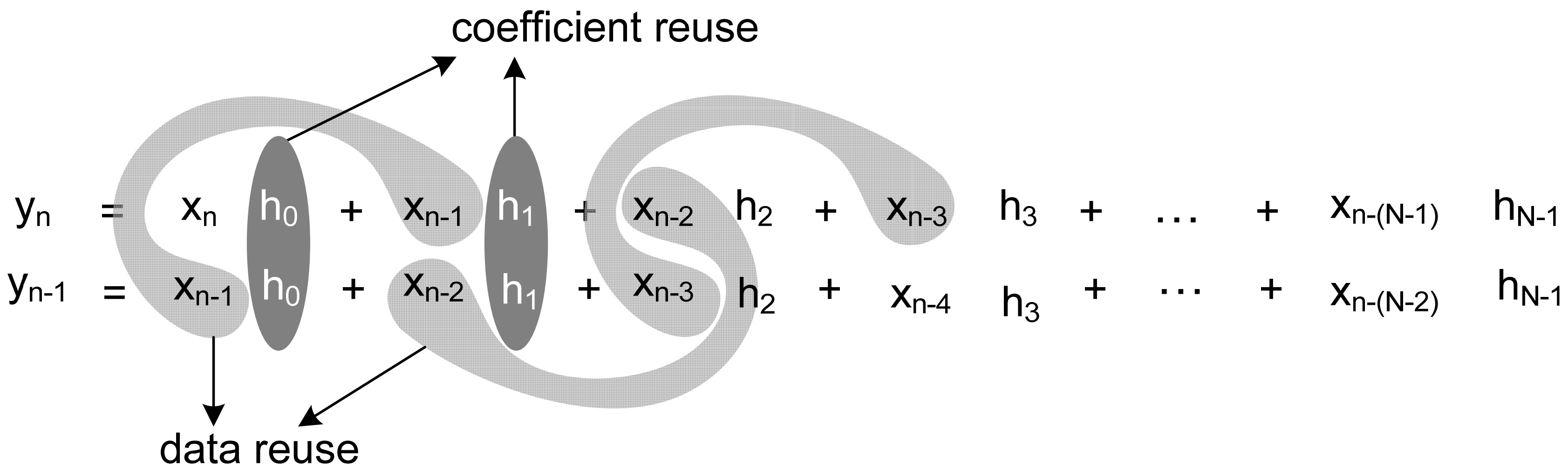
RTL design of one of the register files



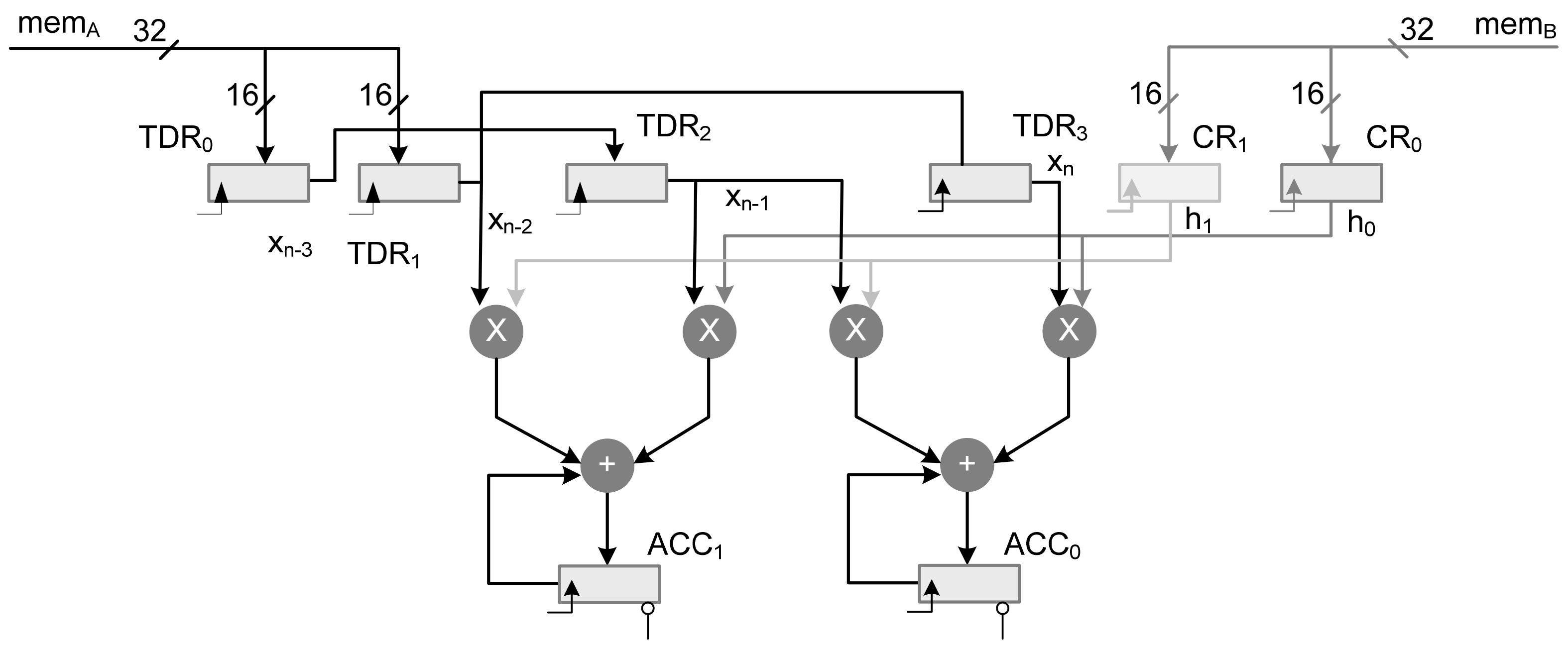
Address generation unit A



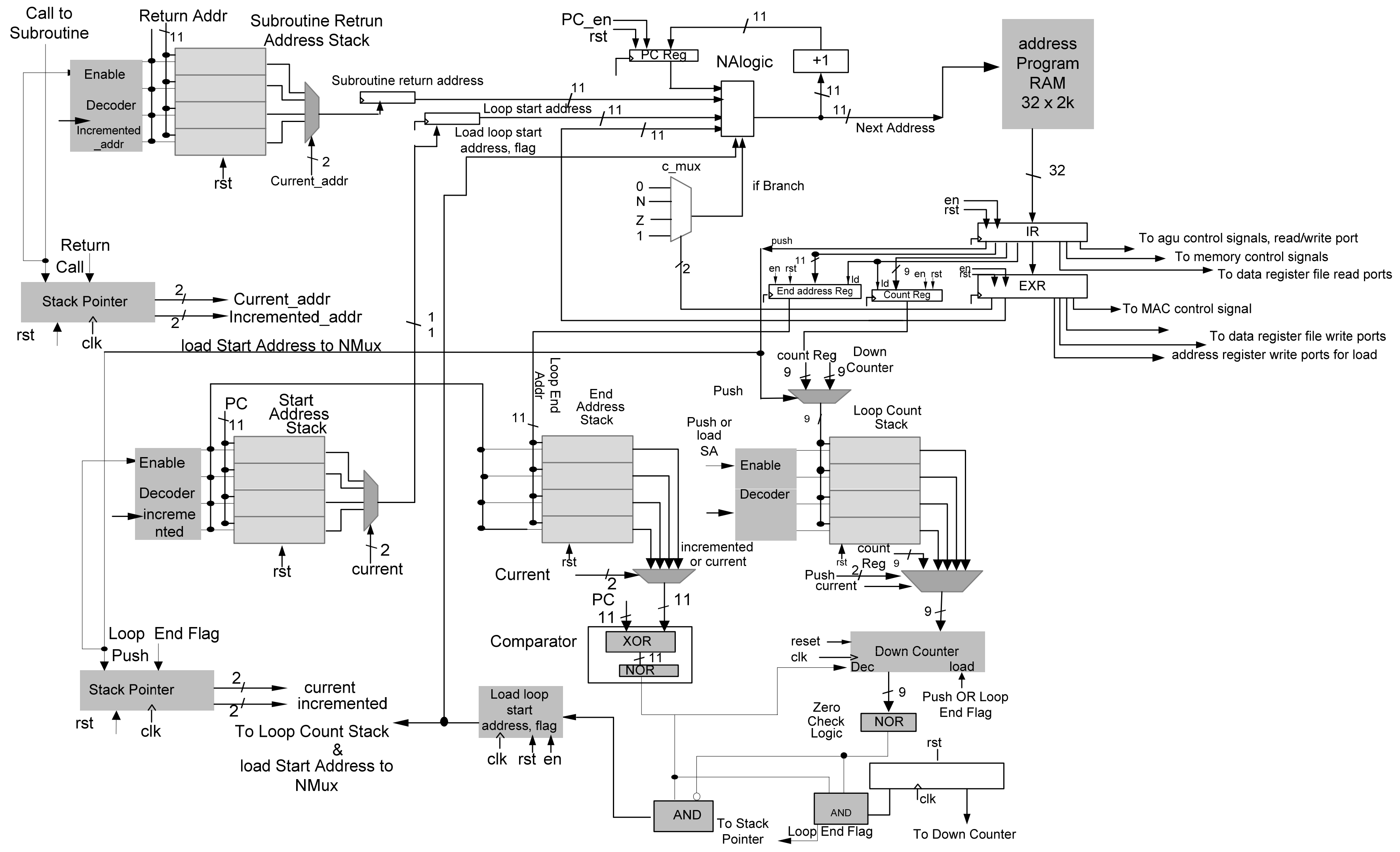
Reuse of data for computing two consecutive output values of convolution summation


coefficient reuse
$y_n = x_n h_0 + x_{n-1} h_1 + x_{n-2} h_2 + x_{n-3} h_3 + \dots + x_{n-(N-1)} h_{N-1}$
$y_{n-1} = x_{n-1} h_0 + x_{n-2} h_1 + x_{n-3} h_2 + x_{n-4} h_3 + \dots + x_{n-(N-2)} h_{N-1}$
data reuse

Two MAC blocks and specialized registers for maximum data reuse



Program sequences of the LEC accelerator



Logic implemented by C_MUX (see text)

Select signals		Output (cond_flag)()
N	**Z**	
0	0	FALSE
0	1	T/F depends on flag Z
1	0	T/F depends on flag N
1	1	TRUE unconditional

Address Registers Arithmetic

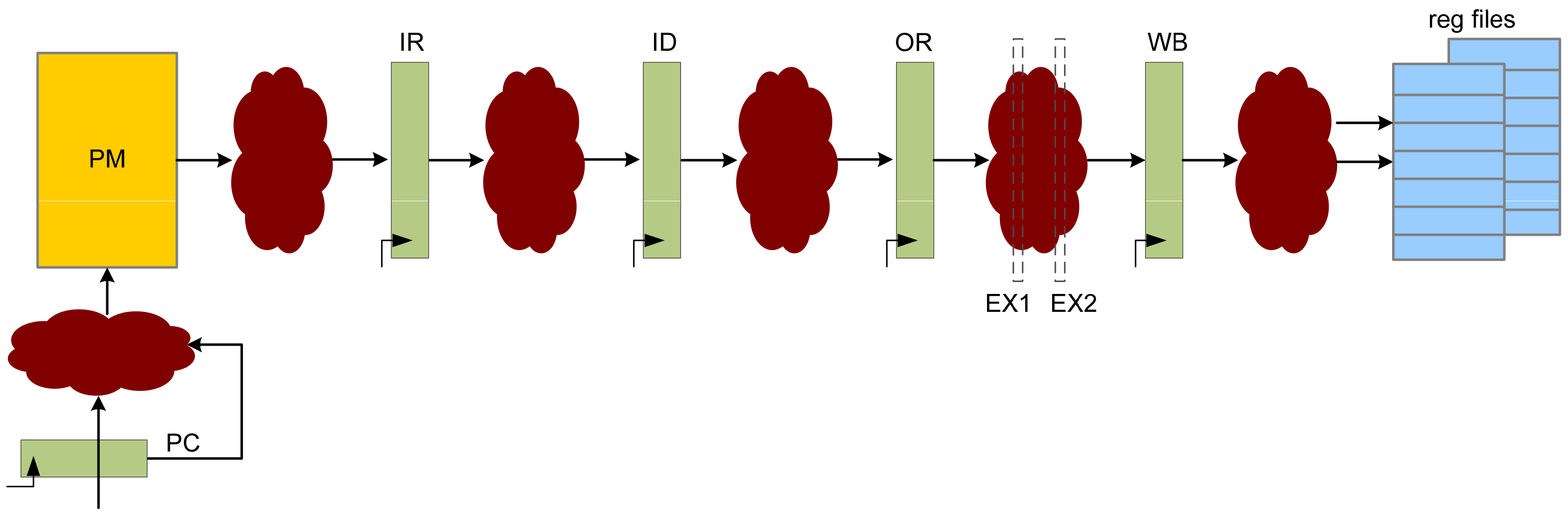
- Circular Addressing
- Arithmetic and Logic Instruction
- Accumulator Instructions
- Branch Instruction
- Application Specific Instructions
- Instruction Encoding

0	Orthogonal instructions			
0	15	15	10	10
0	Data Path A	Data Path B	AGU-A	AGU-B
1	Special and non-orthogonal instruction			

Pipelining Options

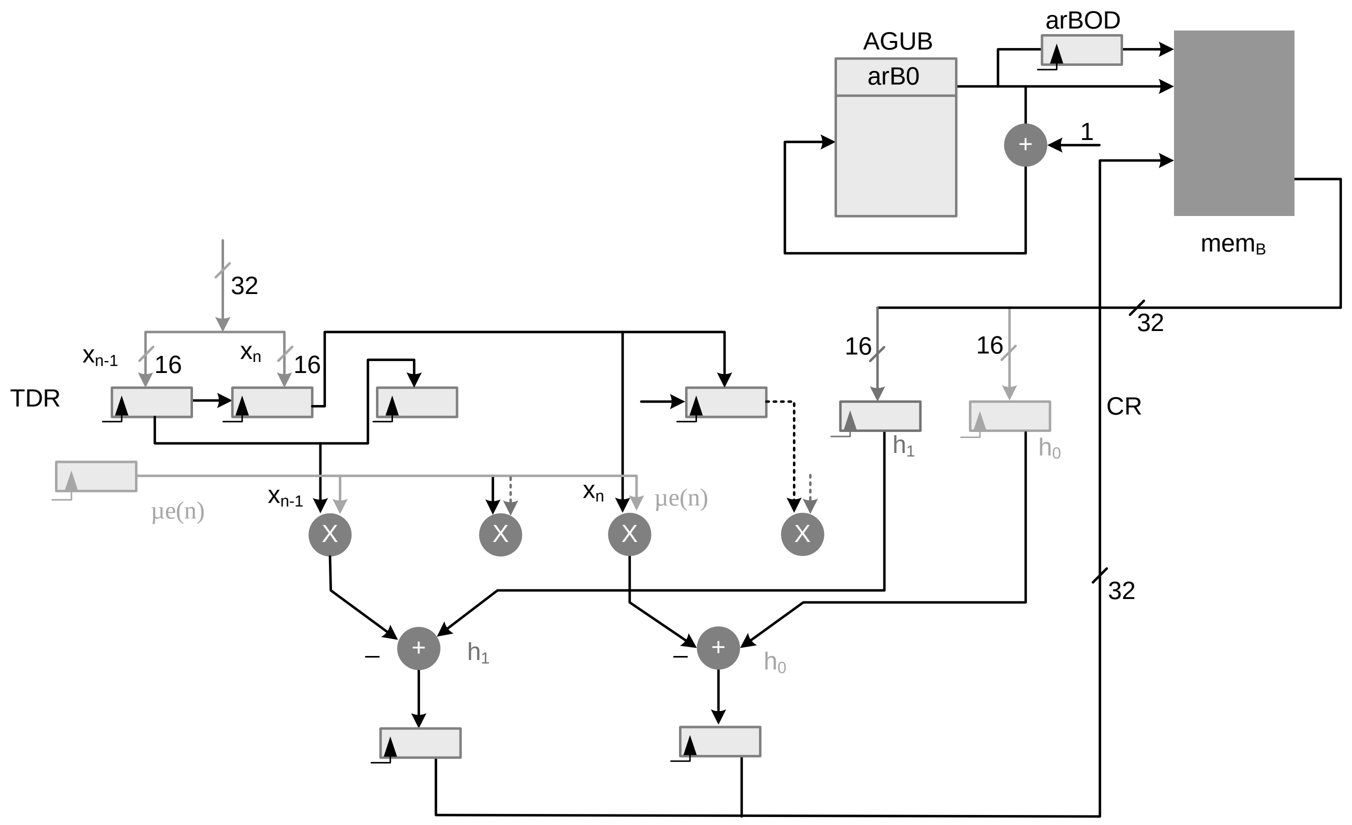
- The pipelining in the datapath is not shown but the accelerator can be easily pipelined
  - Delay Slot

    Depending upon the number of pipeline stages, the delay slot can be effectively used to avoid pipeline stalls



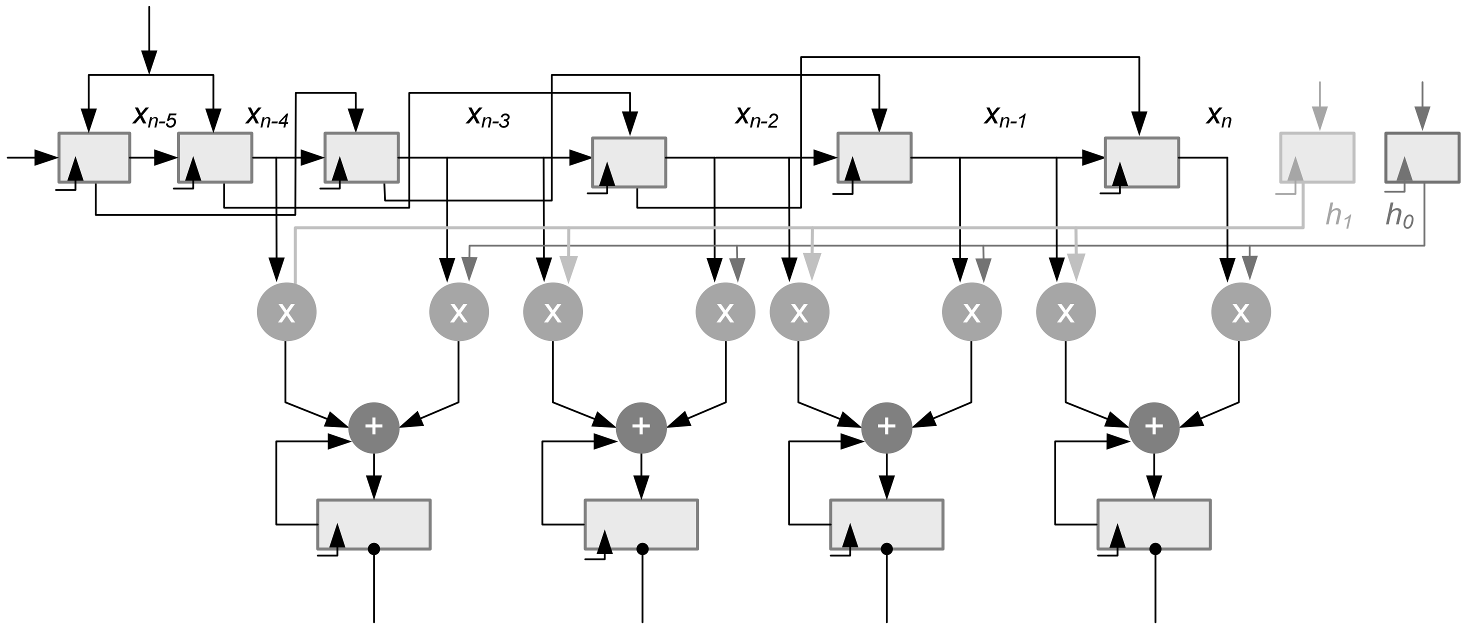
Optional Support for Coefficient Update

- In the design of micro-coded state machine, HW support can also be added to speedup coefficients update operation of (11–3). This requires addition of a special error register ER that saves the scaled error value me½n and few provisions in the datapath for effective use of multipliers, adders and memory for coefficient updates



Multi MAC Block Design Option

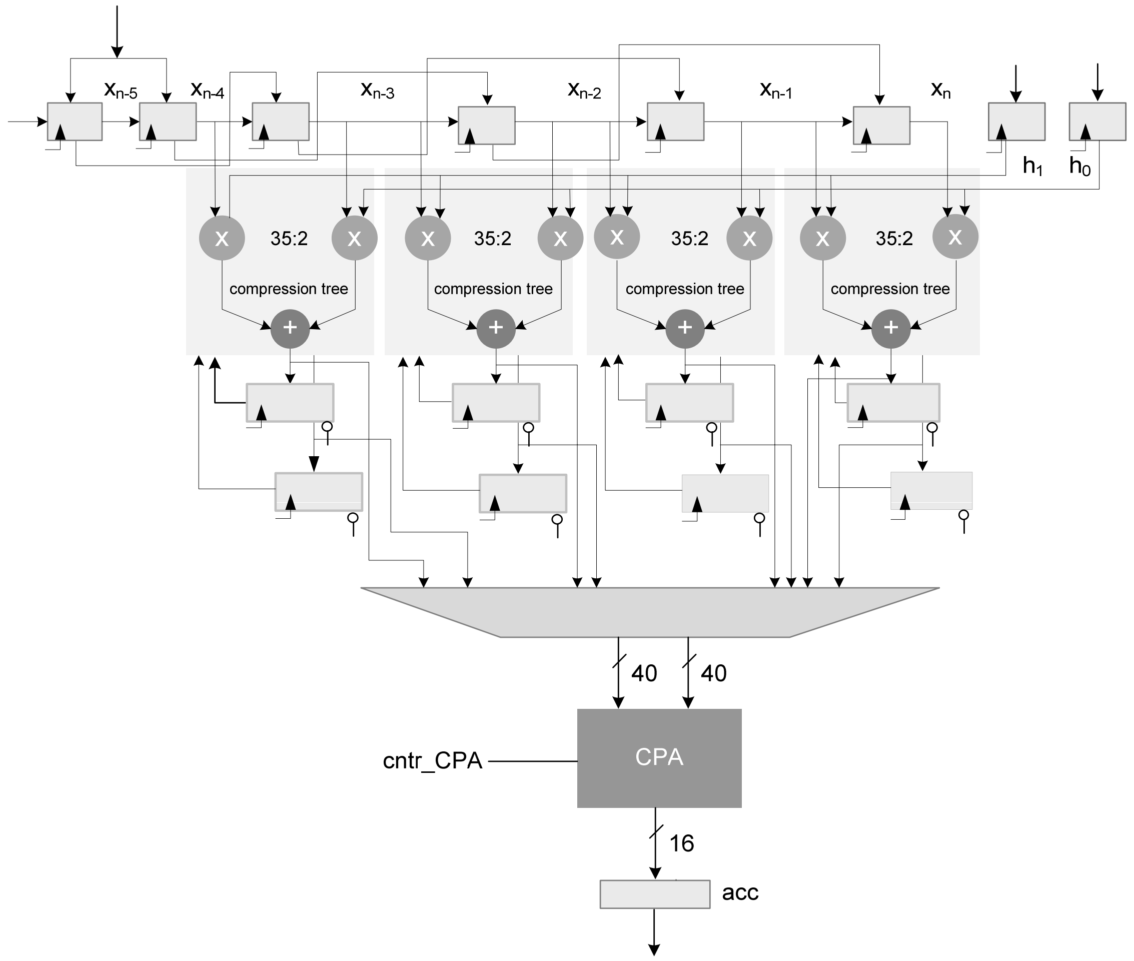
- The datapath can be easily extended to include more MAC blocks to accelerate the execution of filtering operation. A configuration of the accelerator with four MAC blocks is depicted in figure below:



Optional additional MAC blocks for accelerating convolution calculation

Compression Tree and Single CPA-based Design

- The design can be further optimized for area by using compression trees with a single CPA



Optional use of compression trees and one CPA for optimized hardware

Questions/Feedback