$$(1 \; 6t \; 0t+1) \times \times$$

USB INFORMACION Y DOCUMENTACION
SERVICIO DOCUMENTAL
CINVESTAV IPN

Centro de Investigación y de Estudios Avanzados del I.P.N.
Unidad Guadalajara

Diseño e Implementación de un Interleaver Multi-Estándar para los Estándares 802.11a, 802.11n, 802.16e y DVB

Tesis que presenta:

Carlos Rogelio Sánchez Ortiz

Para obtener el grado de:

Maestro en Ciencias

En la especialidad de:

Ingeniería Eléctrica

Directores de Tesis:

Dr. Ramón Parra Michel

Dr. Manuel Edgardo Guzmán Rentería

CINVESTAV
IPN
ADQUISICION
DE LIBROS

Guadalajara, Jalisco. Marzo de 2008

CLASIF.: IK165.C6 .S26 2008
ADQUIS.: BC - SOO
FECHA: 12 - XI - 2008
PROCED.. Don. - 2008
$
ID: 144224-1001

Diseño e Implementación de un Interleaver Multi-Estándar para los Estándares 802.11a, 802.11n, 802.16e y DVB

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:

Carlos Rogelio Sánchez Ortiz

Ingeniero Electrónica y
Sistemas de Comunicación Digital
Universidad Autónoma de Aguascalientes 2000-2005

Becario de CONACYT, expediente no. 199300

Directores de Tesis:

Dr. Ramón Parra Michel

Dr. Manuel Edgardo Guzmán Rentería

CINVESTAV del IPN Unidad Guadalajara, Marzo de 2008.

Centro de Investigación y de Estudios Avanzados del I.P.N.
Unidad Guadalajara

Design and Implementation of a Multi-Standard Interleaver for 802.11a, 802.11n, 802.16e & DVB Standards

A thesis presented by:

Carlos Rogelio Sánchez Ortiz

to obtain the degree of:

Master in Science

in the subject of:

Electrical Engineering

Thesis Advisors:

Dr. Ramón Parra Michel

Dr. Manuel Edgardo Guzmán Rentería

Guadalajara, Jalisco. March 2008

Design and Implementation of a Multi-Standard Interleaver for 802.11a, 802.11n, 802.16e & DVB Standards

Master of Science Thesis
In Electrical Engineering

By:

Carlos Rogelio Sánchez Ortiz

Engineer in Electronics

Universidad Autónoma de Aguascalientes 2000-2005

Scholarship granted by CONACYT, No. 199300

Thesis Advisors:

Dr. Ramón Parra Michel

Dr. Manuel Edgardo Guzmán Rentería

CINVESTAV del IPN Unidad Guadalajara, March, 2008.

Resumen

Los dispositivos electrónicos inalámbricos trabajando a altas velocidades son más comunes cada día. Las altas velocidades, no idealidades en el medio de transmisión entre otros, incrementan los problemas de errores en la recepción de los bits, por ello se han desarrollado muchas técnicas para reducir el porcentaje de error de los bits en la transmisión/recepción.

Una de las técnicas más utilizadas es el uso de interleavers después de la codificación de los bits.

Se ha probado que esto disminuye el porcentaje de error en los bits en muchos casos en un orden de magnitud [1].

Un interleaver (entrelazador) es un dispositivo que reacomoda el orden de una secuencia de símbolos (bits) uno a uno en un orden determinado.

Asociado con cualquier Interleaver está un Deinterleaver, el cual es un dispositivo que restaura la secuencia reordenada a su ordenamiento original.

El interleaver es utilizado en el transmisor y receptor en sistemas de comunicación inalámbrica, y por ello hay muchas variaciones de este bloque definidas en diferentes estándares.

La intención de esta tesis es la de crear un nuevo bloque reconfigurable Interleaver, capaz de implementar cualquiera de las operaciones de entrelazado en los estándares 802.11a, 802.11n, 802.16e y DVB (Digital Video Broadcasting). Esto permitirá diseños capaces de comunicarse con otros dispositivos utilizando cualquier configuración de interleaver en los estándares citados.

La arquitectura de interleaver reconfigurable llamada MSI (Multi Standard Interleaver), fue comparada en términos de reconfigurabilidad, área y consumo de potencia contra otras tres arquitecturas, las cuales son las implementaciones mas representativas de implementación de interleavers.

Los resultados de estas comparaciones muestran que el MSI logra el mejor desempeño en términos de reconfigurablilidad, área y consumo de potencia

Esta tesis sigue la tendencia de crear diseños multi-estándar.

Abstract

Wireless electronic devices at high speeds are more common each day. The high speeds, non idealities on the transmissions channel, between others, increase the problems of bit errors.

Many techniques have been used to reduce the percentage of bit error, one of the most useful techniques is the use of interleavers after encoding. This decreases the bit error percentage in many cases in one order of magnitude [1].

An Interleaver is a device that rearranges the ordering of a sequence of symbols in some one-to-one determined order. Associated with any Interleaver is a Deinterleaver, which is the device that restores the reordered sequence to its original ordering.

The Interleaver is used in the transmitter and receiver in wireless communication systems, and because of it, there are many variations of this block which are defined in wireless standards.

The intention of this thesis is to create a new reconfigurable Interleaver block, capable to implement any of the interleaving operations in the standards 802.11a, 802.11n, 802.16e y DVB (Digital Video Broadcasting). This allows designs capable of communicate with other devices using any configuration of Interleaver in the cited standards.

The proposed reconfigurable interleaver architecture MSI (Multi Standard Interleaver) was compared in terms of reconfigurability, area and power consumption versus three architectures, whose are the most representatives of the common approaches to implement interleavers. The comparison results show that the MSI approach achieves the best performance in terms of reconfigurability, area and power consumption.

This thesis follows the tendency to create multi-standard designs.

Agradecimientos:

A mis padres y hermana por su comprensión, amor y bendiciones.
A mi gran familia que siempre me apoyo a que siguiera mis metas.
A mis compañeros y amigos de la maestría por su convivencia y amistad.
A mis profesores por su dedicación y sus enseñanzas.
A los Doctores Ramón Parra y Manuel Guzmán por su confianza, apoyo lecciones de vida en mi formación profesional.
Al CONACYT por el apoyo económico.

INDEX

Resumen.....I
Abstract..... III
Agradecimientos..... V
Index..... VII
Figure Index..... IX
Table Index.....XI
Acronyms XIII
1 Introduction 1
1.1 Interleaver Definition 1
1.1.1 Block Interleaver 2
1.1.2 Convolutional Interleaver 3
1.2 Problem definition 4
1.3 Objectives 4
1.4 Document Structure 4
2 Interleavers in the standards 802.11a, 802.11n, 802.16e & DVB 5
2.1 802.11a standard LAN/MAN Wireless LANS 5
2.1.1 Permutation Definition 5
2.1.2 Interleaver Configurations 6
2.1.3 Interleaver Example.....7
2.2 802.11n standard LAN/MAN Wireless LANS 11
2.2.1 Permutation Definition 11
2.2.2 Interleaver Configurations 12
2.2.3 Interleaver Example.....13
2.3 802.16e LAN/MAN Broadband Wireless LANS 15
2.3.1 Permutation Definition 15
2.3.2 Interleaver Configurations 15
2.3.3 Interleaver Example.....16
2.4 DVB Digital Video Broadcasting 18
2.4.1 Convolutional Byte-wise Interleaving.....18
2.4.1.1 Permutation Definition 18
2.4.1.2 Interleaver Configurations 19
2.4.2 Bit-wise interleaving.....19
2.4.2.1 Permutation Definition 19
2.4.2.2 Interleaver Configurations 20
2.4.2.3 Interleaver Example 20
2.4.3 Symbol interleaver.....22
2.4.3.1 Permutation and Configurations Definition 22
3 Classes of Interleaver Implementations.....24
3.1 Interleaver with LUT.....25
3.2 Special matrix interleaver 26
3.3 Interleaver Design Using Crossbars 28
3.4 Arithmetic interleaver 31
3.5 Interleaver Proposal.....33
3.5.1 Optimized Arithmetical AGU.....33
3.5.2 STS structure.....35
3.5.3 Convolutional interleaver as block interleaver 38
4 Interleaver Proposal Technical Explanation.....39
4.1 Hardware Overview 39

4.1.1 Configuration Process 41
4.1.2 Interleaving Process 41
4.2 Functional Description 42
4.2.1 8AGU 1 42
4.2.1.1 AGU1 (0) 44
4.2.1.1.1 Mod Nconf2 Adder 46
4.2.1.1.2 Symbol Interleaver 48
4.2.1.2 AGU1 (1..7) 52
4.2.2 8AGU2 54
4.2.2.1 AGU2 (0) 56
4.2.2.1.1 Configuration Table 59
4.2.2.2 AGU2 (1..7) 60
4.2.3 Memory Structure 62
4.2.3.1 Convolutional Memory 64
4.2.3.2 802 Memory 65
4.2.3.2.1 ADU 67
4.2.3.2.2 802_Mem 68
4.2.3.3 Symbol Memory 69
4.2.3.3.1 MemoryS 70
4.2.4 Configuration Control 71
4.2.4.1 Control Manager 74
4.2.4.1.1 Control State Machine 76
4.2.4.1.2 Control Decoder 78
4.2.4.2 Param Memory 81
4.3 Architecture overview 83
4.4 Testing methodology 85
4.5 Configuration Modes 88
4.5.1 Configuration for 802.11a 90
4.5.2 Configuration for 802.11n 91
4.5.3 Configuration for 802.16e 99
4.5.4 Configuration for DVB 101
4.5.4.1 Outer Interleaver 101
4.5.4.1.1 Convolutional Interleaver 101
4.5.4.2 Inner Interleaver 102
4.5.4.2.1 Compact Bit Interleaver 102
4.5.4.2.2 Symbol Interleaver 103
5 Results 105
5.1 Reconfigurability 106
5.2 Size 107
5.3 Power Consumption 112
6 Conclusions and Future Work 115
6.1 Conclusions 115
6.2 Future Work 116
References 118

FIGURE INDEX

Figure 1-1 Burst Error Correction 1
Figure 1-2 Convolutional Interleaver 3
Figure 2-1 Interleaver Input/Output signals 5
Figure 2-2 Interleaver 802.11n In/Out signals 11
Figure 2-3 Interleaver 802.16e In/Out signals 15
Figure 2-4 Channel coding process for regular and repetition coding transmission 15
Figure 2-5 Convolutional interleaver in/out signals 18
Figure 2-6 Conceptual diagram of the outer Interleaver and deinterleaver 18
Figure 2-7 Bit-wise Interleaver In/Out signals 19
Figure 2-8 Symbol Interleaver In/Out signals 22
Figure 2-9 Symbol Interleaver address generation scheme for the 2K mode ... 23
Figure 2-10 Symbol Interleaver address generation scheme for the 2K mode . 23
Figure 3-1 Traditional block interleaver implementation 25
Figure 3-2 (a)Interleaver based on a matrix, write in row and read in columns,(b) Interleaved Stream output of the interleaver block 26
Figure 3-3 Special Matrix interleaver 27
Figure 3-4 Crossbar of X inputs and X outputs 28
Figure 3-5 Crossbar example 28
Figure 3-6 Crossbar implementation example 29
Figure 3-7 Reduced crossbar example 29
Figure 3-8 Conversion from 3 Stage Switching Matrix to a STS architecture ... 30
Figure 3-9 Hardware block organization of arithmetic interleaver 31
Figure 3-10 Block diagram implementation of Equation (1) 31
Figure 3-11 Block diagram implementation of Equation (2) 32
Figure 3-12 Block diagram implementation of Arithmetic AGU for 802.11a 32
Figure 3-13 Architecture optimization for the Equation (1) 33
Figure 3-14 Example of the generation addresses sequence of equation (1) ... 33
Figure 3-15 Architecture optimization for the Equation (2) 34
Figure 3-16 Example of the generation addresses sequence of equation (2) ... 34
Figure 3-17 AGU1 modifications to make 8 bits of parallelism 35
Figure 3-18 AGU2 modifications to make 8 bits of parallelism 36
Figure 3-19 architecture used to implement a parallelism of 8 bits. 37
Figure 3-20 Structure proposed for the convolutional interleaver 38
Figure 4-1 Input/Output signal diagram for Multi-Standard Interleaver (MSI) 39
Figure 4-2 Hardware block organization of Multi-Standard Interleaver (MSI) ... 40
Figure 4-3 Input/Output signal diagram for 8AGU1 42
Figure 4-4 Hardware block organization of 8AGU1 43
Figure 4-5 Input/Output signal diagram for AGU1(0) 44
Figure 4-6 Hardware block organization of AGU1(0) 45
Figure 4-7 Input/Output signal diagram for Mod Nconf2 Adder 46
Figure 4-8 Hardware block organization of Mod Nconf2 Adder 47
Figure 4-9 Input/Output signal diagram for Symbol Interleaver 48
Figure 4-10 Hardware block organization of Symbol Interleaver 49
Figure 4-11 Input/Output signal diagram for Configurable P-RSG 50
Figure 4-12 Hardware block organization of Configurable P-RSG 51
Figure 4-13 Input/Output signal diagram for AGU1(1..7) 52

Figure 4-14 Hardware block organization of AGU1(1..7) 53
Figure 4-15 Input/Output signal diagram for 8AGU2.....54
Figure 4-16 Hardware block organization of 8AGU2.....55
Figure 4-17 Input/Output signal diagram for AGU2(0) 56
Figure 4-18 Hardware block organization of AGU2(0) 58
Figure 4-19 Hardware block organization of Configuration Table 59
Figure 4-20 Input/Output signal diagram for AGU2(1..7) 60
Figure 4-21 Hardware block organization of AGU2(1..7) 61
Figure 4-22 Input/Output signal diagram for Memory Structure 62
Figure 4-23 Hardware block organization of Memory Structure.....63
Figure 4-24 Input/Output signal diagram for Convolutional Memory 64
Figure 4-25 Input/Output signal diagram for 802 Memory structure.....65
Figure 4-26 Hardware block organization of 802 Memory 65
Figure 4-27 Diagram block organization of ADU.....67
Figure 4-28 Input/Output signal diagram for 802_Mem.....68
Figure 4-29 Input/Output signal diagram for Symbol Memory.....69
Figure 4-30 Hardware block organization of Symbol Memory 69
Figure 4-31 Input/Output signal diagram for MemoryS 70
Figure 4-32 Input/Output signal diagram for Configuration Control.....71
Figure 4-33 Hardware block organization of Configuration Control.....72
Figure 4-34 Input/Output signal diagram for Control Manager.....74
Figure 4-35 Hardware block organization of Control Manager.....75
Figure 4-36 Input/Output signal diagram for Control State Machine 76
Figure 4-37 FSM of the Control State Machine.....77
Figure 4-38 Input/Output signal diagram for Control Decoder.....78
Figure 4-39 Hardware block organization of Control Decoder 80
Figure 4-40 Input/Output signal diagram for Param_Memory 81
Figure 4-41 Hardware block organization of Param Memory.....82
Figure 4-42 MSI architecture overview 83
Figure 4-43 MSI architecture overview with references to figure or table index 84
Figure 4-44 Testing methodology architecture for the MSI 85
Figure 4-45 Simulation example of no error configuration 86
Figure 4-46 Example how to configure MSI for Configuration process 88
Figure 4-47 Configuration parameters on the CONFIGURATION BUS 88
Figure 5-1 MSI architecture active on 802.11a, 802.11n, 802.16e and compact bit interleavers configurations 109
Figure 5-2 MSI architecture active on convolutional interleaver configuration 110
Figure 5-3 MSI architecture active on symbol interleaver configuration.....111
Figure 5-4 Comparative total power consumption.....114
Figure 5-5 Power theater tool screenshot 114

TABLE INDEX

Table 2-1 Rate-dependent parameters for 802.11a 6
Table 2-2 Input data bits 7
Table 2-3 First permutation 8
Table 2-4 Second permutation 9
Table 2-5 Output data bits 10
Table 2-6 Number of row and columns in the interleaver 12
Table 2-7 Interleaver Configurations for 802.11n 12
Table 2-8 input data bits 13
Table 2-9 First permutation 13
Table 2-10 Second permutation 14
Table 2-11 Third permutation 14
Table 2-12 Output data bits 14
Table 2-13 Rate-dependent parameters for 802.16e 15
Table 2-14 Input data bits 16
Table 2-15 First permutation 16
Table 2-16 Second permutation 17
Table 2-17 Output data bits 17
Table 2-18 Input data bits 20
Table 2-19 Permutation 20
Table 2-20 Output data bits 21
Table 2-21 Bit permutations for the 2K mode 22
Table 2-22 Bit permutations for the 8K mode 23
Table 3-1 Interleavers mapped by the interleaver implementations 24
Table 4-1 Input and Output organization for MSI 40
Table 4-2 Input and Output organization for 8AGU1 43
Table 4-3 Input and Output organization for AGU1(0) 45
Table 4-4 Input and Output organization of Mod Nconf2 Adder 46
Table 4-5 Input and Output organization of Symbol Interleaver 48
Table 4-6 Input and Output organization of Configurable P-RSG 50
Table 4-7 Input and Output organization of AGU1(1..7) 52
Table 4-8 Input and Output organization of 8AGU2 54
Table 4-9 Input and Output organization of AGU2(0) 57
Table 4-10 Selector signal configurations 57
Table 4-11 Input and Output organization of Configuration Table 59
Table 4-12 Input and Output organization of AGU2(1..7) 61
Table 4-13 Input and Output organization of Memory Structure 62
Table 4-14 Mem_Select signal configurations 63
Table 4-15 Input and Output organization of Convolutional Memory 64
Table 4-16 Input and Output organization of 802 Memory 66
Table 4-17 Input and Output organization of ADU 67
Table 4-18 Input and Output organization of 802_Mem 68
Table 4-19 Input and Output organization of Symbol Memory 69
Table 4-20 Input and Output organization of MemoryS 70
Table 4-21 Input and Output organization of Configuration Control 72
Table 4-22 Input and Output organization of Control Manager 74
Table 4-23 Configurations of CONFIG signal 76

Table 4-24 Input and Output organization of Control State Machine 77
Table 4-25 Input and Output organization of Control Decoder.....79
Table 4-26 Input and Output organization of Param Memory 81
Table 4-27 Test plan 87
Table 4-28 Configuration parameter list and description for MSI 89
Table 4-29 Configuration parameters for 802.11a 90
Table 4-30 Configuration parameters for 802.11n 20MHz iss=0.....91
Table 4-31 Configuration parameters for 802.11n 20MHz iss=1.....92
Table 4-32 Configuration parameters for 802.11n 20MHz iss=2.....93
Table 4-33 Configuration parameters for 802.11n 20MHz iss=3.....94
Table 4-34 Configuration parameters for 802.11n 40MHz iss=0.....95
Table 4-35 Configuration parameters for 802.11n 40MHz iss=1.....96
Table 4-36 Configuration parameters for 802.11n 40MHz iss=2.....97
Table 4-37 Configuration parameters for 802.11n 40MHz iss=3.....98
Table 4-38 Configuration parameters for 802.16e BPSK.....99
Table 4-39 Configuration parameters for 802.16e 16QAM & 64QAM.....100
Table 4-40 Configuration parameters for convolutional interleaver.....102
Table 4-41 Configuration parameters for compact bit interleaver 103
Table 4-42 Values and effects of EVEN\ODD signal 103
Table 4-43 Configuration parameters for symbol interleaver 104
Table 5-1 Interleavers mapped by the different interleaver implementations..106
Table 5-2 Total integration area of every architecture.....107
Table 5-3 Total integration active area for different interleaver configurations 108
Table 5-4 Power consumption results for all the interleaver configurations 113

ACRONYMS

ADU	Address Decoding Unit
AGU	Address Generator Unit
COFDM	Coded Orthogonal Frequency Division Multiplexing
CGU	Control Generator Unit
DVB	Digital Video Broadcasting
FEC	Forward Error Correction
FIFO	First In First Out
HF	High Frequency
MCS	Modulation of Coding Scheme
Mod	Modulus operation
MSI	Multi Standard Interleaver
N_{BPSC}	Number of Coded Bits per Subcarrier per Spatial Stream
N_{CBPS}	Number of Coded Bits per OFDM Symbol
N_{CBPSS}	Number of coded bits per symbol per spatial stream
N_{DS}	Number of Data Subcarriers
N_{SS}	Number of Spatial Streams
OFDM	Orthogonal Frequency Division Multiplexing
PRSG	Pseudo Random Sequence Generator
QAM	Quadrature Amplitude Modulation
QPSK	Quaternary Phase Shift Keying
ROM	Read Only Memory
SLAC	Single Load After Clear

1 Introduction

1.1 Interleaver Definition

An interleaver is a device that rearranges the order of a sequence of symbols in some one-to-one deterministic manner. Associated with any interleaver is a deinterleaver, which is the device that restores the reordered sequence into its original ordering. Interleavers and deinterleavers have a variety of applications in cryptography and communication technology.

In many applications of communication technology, interleaving is used as an adjunct to coding for error correction. One useful technique for some type of burst-error channels is to insert an interleaver between the channel encoder and the channel. The interleaver redistributes the channel symbols so that the symbols from a codeword are mutually separated by somewhat more than the length of a "typical" burst of errors. Thus, interleaving effectively makes the channel appear like a random-error channel to the decoder. For some HF (High Frequency) channels, this technique can improve the performance by one to three orders of magnitude [1].

As a example of the use of the interleaver see Figure 1-1, in this example the coder converts each 1 bit (a) into 3 bits (b) the coded data is interleaved (c) and at the transmission the burst errors affect the data (d), in this case to ensure the recover of all the bits, the maximum allowed burst error length is 3, that is the case in this example, we can see the 3 lost bits marked by a X. After Deinterleaving (e) the burst error is spread in the entire packet, the decoder get the coded bits with bit errors and recover the original data (f), because there is at most one bit error by coded bit.

Figure 1-1 Burst error correction

Many of the previous interleaver applications have been associated with block codes, so it is understandable that block interleavers have frequently been assumed for these applications. A common example of a block interleaver function is to divide symbol sequences into blocks corresponding to a two-dimensional array, and to conceptually read symbols in by rows and out by columns.

In some applications such as those dealing with convolutional codes, it is more natural to consider convolutional interleavers, in which a symbol is read out each time a symbol is read in. Convolutional interleavers are a more general class of interleavers than block interleavers, since any block interleaving function can be implemented by a convolutional interleaver [1].

Interleavers and deinterleavers are characterized by their encoding delay D, which is the maximum delay encountered by any symbol before it is inserted into the output sequence, and by the storage capacities S and S_d, which are the number of symbols stored by the interleaver and by the deinterleaver, respectively.

Interleaving is used in radio systems to distribute transmitted bits in time or frequency or both. Consecutive bits on the input stream should not be transmitted consecutively. Interleaving reduces the effect of burst errors caused by a fast fading channel or frequency selective fading in an OFDM (Orthogonal Frequency Division Multiplexing) system.

There are two major classes of interleavers, block interleavers and convolutional interleavers.

1.1.1 Block Interleaver

A block interleaver accepts a set of symbols and rearranges them, without repeating or omitting any of the symbols in the set. The number of symbols in each set is fixed for a given interleaver.

A block interleaver operates on one block of input bits at a time and there is no interleaving between the blocks. A block interleaver is often implemented by writing the bits into a matrix row by row and then reading them column by column. Deinterleaving is simply the reversed operation write column by column and read row by row.

1.1.2 Convolutional Interleaver

A convolutional interleaver consists of a set of shift registers, each with a fixed delay. In a typical convolutional interleaver, the delays are nonnegative integer multiples of a fixed integer (although a general multiplexed interleaver allows unrestricted delay values). Each new symbol from an input vector feeds into the next shift register and the oldest symbol in that register becomes part of the output vector. A convolutional interleaver has memory; that is, its operation depends not only on current symbols but also on previous symbols.

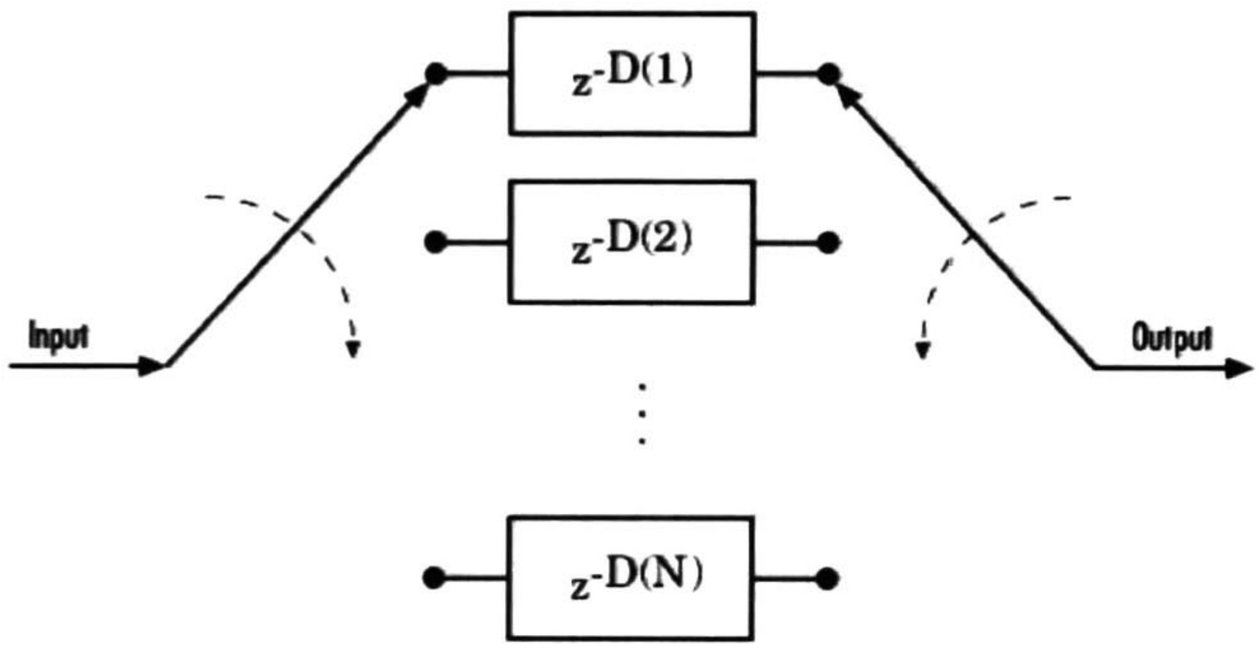


Figure 1-2 Convolutional Interleaver

The schematic in Figure 1-2 describes the structure of a general convolutional interleaver by showing the set of shift registers and their delay values D(1), D(2),..., D(N). The k^{th} shift register holds D(k) symbols, where k = 1,2,...,N.

1.2 Problem Definition

The interleaver is a processing block in the transmitter and receiver in many wireless communication systems, used to reduce the percentage of bit errors on the transmission, and because of it, there are many implementations of this block defined in different standards.

The intention of this thesis is to create a new reconfigurable interleaver block, capable to implement any of the interleaving operations in the selected standards. This tendency to reconfigurable blocks will allow designs capable of intercommunicate with other devices using different standards.

Currently at the market is working in the design of wireless devices with multi-standard orientation. With the idea to provide to its products the ability to change the standard used has were convenient, bringing more communication covertures for the user, moreover a wider market for the product.

Design like this brings to the designers, configurable blocks, capable to implement new variations in the standards, this reduces the design time of the product.

1.3 Objectives

The major objectives of the thesis are:

- Design reconfigurable architecture for an Interleaver/Deinterleaver block.
- Allow the implementation of the interleavers in the 802.11a, 802.11n, 802.16e and DVB standards.
- Make a power and area analysis of the proposed implementation vs. some Interleaver architectures.

1.4 Document Structure

This thesis is divided in five chapters. Chapter 1 is an introductory explanation of what an interleaver is and the way it works. Chapter 2 explains how the interleavers are declared in the 802.11a, 802.11n, 802.16e and DVB standards. Chapter 3 explains the use of some interleaver implementations using different approaches, and introduces the architecture that is proposed in this thesis. Chapter 4 is a technical explanation of the interleaver proposal; it describes the architecture, the use, testing methodology, and configuration of the interleaver proposal. Chapter 5 is a presentation of the comparative results obtained, by the proposal architecture versus other implementations in terms of reconfigurability, power and integration area.

2 Interleavers in the Standards 802.11a, 802.11n, 802.16e & DVB

This chapter is a recompilation of the interleaver definitions in the 802.11a, 802.11n, 802.16e and DVB standards. These standards were selected because it's importance in wireless communications.

- *IEEE 802.11a standard LAN/MAN Wireless LANS[25]*
- *IEEE 802.11n standard LAN/MAN Wireless LANS[26]*
- *IEEE 802.16e standard LAN/MAN Broadband Wireless LANS[27]*
- *DVB Digital Video Broadcasting standard[28]*

In this chapter the interleaver definitions are presented as described at standards. Starts with a small introduction of the interleaver, input/output block diagram, the permutation definition, and possible configurations for interleaving as described at the standards, and an example of interleaved data.

2.1 802.11a Standard LAN/MAN Wireless LANS

All encoded data bits shall be interleaved by a block interleaver with a block size corresponding to the number of bits in a single OFDM symbol, N_{CBPS} (number of coded bits per subcarrier per spatial stream). The interleaver is defined by a two-step permutation. The first permutation ensures that adjacent coded bits are mapped onto nonadjacent sub carriers. The second ensures that adjacent coded bits are mapped alternately onto less and more significant bits of the constellation [25]. The input/output signals are shown in Figure 2-1.

Figure 2-1 Interleaver Input/output signals

2.1.1 Permutation Definition

We shall denote by *k* the index of the coded bit before the first permutation; *i* shall be the index after the first and before the second permutation, and *j* shall be the index after the second permutation, just prior to modulation mapping.

The first permutation is defined by the rule

$$i = (N_{CBPS}/16)\,(k \bmod 16) + \text{floor}\,(k/16)\quad k = 0,1,\ldots,N_{CBPS} - 1 \qquad (1)$$

The function floor (x) denotes the largest integer not exceeding the parameter x.

The second permutation is defined by the rule:

$$j = s \cdot \text{floor}(i/s) + (i + N_{CBPS} - \text{floor}(16 \cdot i/N_{CBPS})) \bmod s \qquad (2)$$
$$i = 0,1,\ldots N_{CBPS} - 1$$

The value of s is determined by the N_{BPSC} (number of coded bits per sub-carrier), according to

$$s = \max(N_{BPSC}/2,1) \qquad (3)$$

The deinterleaver, which performs the inverse relation, is also defined by two permutations.

Here we shall denote by *j* the index of the original received bit before the first permutation; *i* shall be the index after the first and before the second permutation, and *k* shall be the index after the second permutation.

The first permutation is defined by the rule

$$i = s \cdot \text{floor}(j/s) + (j + \text{floor}(16 \cdot j/N_{CBPS})) \bmod s \qquad j = 0,1,\ldots N_{CBPS} - 1 \quad (4)$$

Where:
s is defined in Equation (3).

This permutation is the inverse of the permutation described in Equation (2).

The second permutation is defined by the rule

$$k = 16 \cdot i - (N_{CBPS} - 1)\text{floor}(16 \cdot i/N_{CBPS}) \; i = 0,1,\ldots N_{CBPS} - 1 \qquad (5)$$

This permutation is the inverse of the permutation described in Equation (1).

2.1.2 Interleaver Configurations

The modulation parameters dependent on the data rate used shall be set according to Table 2-1.

Data rate (Mbits/s)	Modulation	Coding rate (R)	Coded bits per subcarrier (N_{BPSC})	Coded bits per OFDM symbol (N_{CBPS})	Data bits per OFDM symbol (N_{DBPS})
6	BPSK	1/2	1	48	24
9	BPSK	3/4	1	48	36
12	QPSK	1/2	2	96	48
18	QPSK	3/4	2	96	72
24	16-QAM	1/2	4	192	96
36	16-QAM	3/4	4	192	144
48	64-QAM	2/3	6	288	192

Table 2-1 Rate-dependent parameters for 802.11a

2.1.3 Interleaver Example

The interleaver is defined as a two-permutation process. Table 2-2 shows a random input of 1's and 0's of 192 bits to the interleaver. We shall denote by *k* the index of the coded bit before the first permutation; *i* shall be the index after the first and before the second permutation; and *j* shall be the index after the second permutation, just prior to modulation mapping. The mapping from *k* to *i* is shown in Table 2-3, and the mapping from *i* to *j* is shown in Table 2-4.

As a specific example, consider the case of k = 17 (the #17 input bit). It is mapped by the first permutation to i = 13 and by the second permutation to j = 12 (the #12 output bit). The interleaved bits are shown in Table 2-5.

##	Bit	##	Bit	##	Bit	##	Bit	##	Bit	##	Bit
0	0	32	1	64	0	96	1	128	1	160	1
1	0	33	0	65	1	97	0	129	1	161	1
2	1	34	0	66	0	98	0	130	0	162	1
3	0	35	1	67	0	99	0	131	0	163	0
4	1	36	1	68	1	100	1	132	0	164	0
5	0	37	1	69	0	101	1	133	0	165	0
6	1	38	0	70	1	102	1	134	0	166	0
7	1	39	1	71	0	103	1	135	0	167	0
8	0	40	1	72	1	104	1	136	0	168	1
9	0	41	0	73	1	105	1	137	1	169	1
10	0	42	1	74	1	106	0	138	0	170	0
11	0	43	1	75	1	107	0	139	0	171	1
12	1	44	0	76	1	108	0	140	0	172	0
13	0	45	1	77	0	109	0	141	0	173	0
14	0	46	0	78	1	110	0	142	1	174	1
15	0	47	1	79	1	111	0	143	1	175	1
16	1	48	1	80	1	112	1	144	1	176	1
17	0	49	0	81	1	113	1	145	1	177	1
18	1	50	0	82	1	114	0	146	1	178	1
19	0	51	1	83	0	115	0	147	0	179	0
20	0	52	1	84	1	116	1	148	0	180	1
21	0	53	0	85	0	117	0	149	0	181	0
22	0	54	1	86	0	118	0	150	0	182	1
23	1	55	0	87	0	119	0	151	0	183	1
24	1	56	0	88	1	120	0	152	0	184	1
25	1	57	0	89	1	121	1	153	0	185	0
26	1	58	0	90	0	122	1	154	0	186	1
27	1	59	1	91	0	123	1	155	1	187	1
28	0	60	1	92	0	124	0	156	1	188	0
29	0	61	1	93	0	125	0	157	0	189	0
30	0	62	0	94	1	126	1	158	0	190	1
31	0	63	1	95	0	127	1	159	1	191	0

Table 2-2 Input data bits

k	i	k	i	k	i	k	i	k	i	k	l
0	0	32	2	64	4	96	6	128	8	160	10
1	12	33	14	65	16	97	18	129	20	161	22
2	24	34	26	66	28	98	30	130	32	162	34
3	36	35	38	67	40	99	42	131	44	163	46
4	48	36	50	68	52	100	54	132	56	164	58
5	60	37	62	69	64	101	66	133	68	165	70
6	72	38	74	70	76	102	78	134	80	166	82
7	84	39	86	71	88	103	90	135	92	167	94
8	96	40	98	72	100	104	102	136	104	168	106
9	108	41	110	73	112	105	114	137	116	169	118
10	120	42	122	74	124	106	126	138	128	170	130
11	132	43	134	75	136	107	138	139	140	171	142
12	144	44	146	76	148	108	150	140	152	172	154
13	156	45	158	77	160	109	162	141	164	173	166
14	168	46	170	78	172	110	174	142	176	174	178
15	180	47	182	79	184	111	186	143	188	175	190
16	1	48	3	80	5	112	7	144	9	176	11
17	13	49	15	81	17	113	19	145	21	177	23
18	25	50	27	82	29	114	31	146	33	178	35
19	37	51	39	83	41	115	43	147	45	179	47
20	49	52	51	84	53	116	55	148	57	180	59
21	61	53	63	85	65	117	67	149	69	181	71
22	73	54	75	86	77	118	79	150	81	182	83
23	85	55	87	87	89	119	91	151	93	183	95
24	97	56	99	88	101	120	103	152	105	184	107
25	109	57	111	89	113	121	115	153	117	185	119
26	121	58	123	90	125	122	127	154	129	186	131
27	133	59	135	91	137	123	139	155	141	187	143
28	145	60	147	92	149	124	151	156	153	188	155
29	157	61	159	93	161	125	163	157	165	189	167
30	169	62	171	94	173	126	175	158	177	190	179
31	181	63	183	95	185	127	187	159	189	191	191

Table 2-3 First permutation

i	j	i	j	i	j	i	j	i	j	i	j
0	0	32	32	64	65	96	96	128	128	160	161
1	1	33	33	65	64	97	97	129	129	161	160
2	2	34	34	66	67	98	98	130	130	162	163
3	3	35	35	67	66	99	99	131	131	163	162
4	4	36	37	68	69	100	100	132	133	164	165
5	5	37	36	69	68	101	101	133	132	165	164
6	6	38	39	70	71	102	102	134	135	166	167
7	7	39	38	71	70	103	103	135	134	167	166
8	8	40	41	72	72	104	104	136	137	168	168
9	9	41	40	73	73	105	105	137	136	169	169
10	10	42	43	74	74	106	106	138	139	170	170
11	11	43	42	75	75	107	107	139	138	171	171
12	13	44	45	76	76	108	109	140	141	172	172
13	12	45	44	77	77	109	108	141	140	173	173
14	15	46	47	78	78	110	111	142	143	174	174
15	14	47	46	79	79	111	110	143	142	175	175
16	17	48	48	80	80	112	113	144	144	176	176
17	16	49	49	81	81	113	112	145	145	177	177
18	19	50	50	82	82	114	115	146	146	178	178
19	18	51	51	83	83	115	114	147	147	179	179
20	21	52	52	84	85	116	117	148	148	180	181
21	20	53	53	85	84	117	116	149	149	181	180
22	23	54	54	86	87	118	119	150	150	182	183
23	22	55	55	87	86	119	118	151	151	183	182
24	24	56	56	88	89	120	120	152	152	184	185
25	25	57	57	89	88	121	121	153	153	185	184
26	26	58	58	90	91	122	122	154	154	186	187
27	27	59	59	91	90	123	123	155	155	187	186
28	28	60	61	92	93	124	124	156	157	188	189
29	29	61	60	93	92	125	125	157	156	189	188
30	30	62	63	94	95	126	126	158	159	190	191
31	31	63	62	95	94	127	127	159	158	191	190

Table 2-4 Second permutation

##	Bit	##	Bit	##	Bit	##	Bit	##	Bit	##	Bit
0	0	32	0	64	0	96	0	128	0	160	0
1	1	33	1	65	0	97	1	129	0	161	0
2	1	34	1	66	0	98	1	130	0	162	0
3	1	35	1	67	1	99	0	131	1	163	0
4	0	36	0	68	0	100	1	132	1	164	0
5	1	37	0	69	0	101	1	133	0	165	0
6	1	38	1	70	0	102	1	134	1	166	0
7	1	39	1	71	0	103	0	135	1	167	0
8	1	40	0	72	1	104	0	136	0	168	0
9	1	41	0	73	0	105	0	137	1	169	0
10	1	42	0	74	0	106	1	138	1	170	0
11	1	43	0	75	1	107	1	139	0	171	0
12	0	44	0	76	1	108	1	140	1	172	1
13	0	45	0	77	0	109	0	141	0	173	1
14	0	46	0	78	1	110	0	142	1	174	0
15	0	47	0	79	0	111	0	143	1	175	1
16	1	48	1	80	0	112	1	144	1	176	1
17	1	49	0	81	0	113	1	145	0	177	0
18	1	50	1	82	0	114	1	146	0	178	1
19	0	51	1	83	1	115	1	147	1	179	1
20	1	52	1	84	1	116	0	148	1	180	0
21	1	53	1	85	1	117	1	149	0	181	0
22	1	54	1	86	0	118	0	150	0	182	1
23	1	55	1	87	1	119	1	151	0	183	1
24	1	56	0	88	0	120	0	152	0	184	0
25	1	57	0	89	0	121	1	153	1	185	1
26	0	58	0	90	0	122	1	154	0	186	1
27	0	59	1	91	1	123	0	155	0	187	0
28	0	60	0	92	0	124	1	156	0	188	1
29	1	61	0	93	0	125	0	157	0	189	1
30	0	62	0	94	1	126	0	158	1	190	0
31	0	63	1	95	0	127	1	159	1	191	1

Table 2-5 Output data bits

2.2 802.11n Standard LAN/MAN Wireless LANS

Blocks of N_{CBPSS} (Number of coded bits per symbol per spatial stream) bits are interleaved by an interleaver that is a modification of the 802.11a interleaver.

Figure 2-2 Interleaver 802.11n in/out signals

2.2.1 Permutation Definition

The interleaving is defined using three permutations. The first permutation is defined by the rule:

$$i = N_{ROW}\,(k \bmod N_{COL}) + \text{floor}\,(k/\,N_{COL}) \quad k = 0,1,\ldots,N_{CBPSS} - 1 \qquad (6)$$

The second permutation is defined by the rule

$$j = s \cdot \text{floor}(i/s) + (i + N_{CBPSS} - \text{floor}(N_{COL}\ \ i/N_{CBPSS})) \bmod s \qquad (7)$$
$$i = 0,1,\ldots\ N_{CBPSS} - 1$$

The value of s is determined by the number of coded bits per sub carrier:

$$s = \max(N_{BPSC}/2,1) \qquad (8)$$

If more that one spatial stream exists, a frequency rotation is applied to the output of the second permutation

$$r = \{\, j - [\,(i_{ss}2) \bmod 3 + 3\ \text{floor}(i_{ss}/3)\,]\ N_{ROT}\ N_{BPSC}\,\} \bmod N_{CBPSS} \qquad (9)$$
$$j = 0,1,\ldots\ N_{CBPSS} - 1$$

Where $i_{ss}=0,1,\ldots,N_{ss}-1$ is the index of the spatial steam on which this interleaver is operating.

The deinterleaver uses the following operations to perform the inverse rotation. We denote by r the index of the bit in the received block (per spatial stream). The first permutation reverses the third (frequency rotation) permutation of the interleaver

$$j = \{r + [(i_{ss}2) \bmod 3 + 3\ \text{floor}(i_{ss}/3)\,]\ N_{ROT}\ N_{BPSC}\,\} \bmod N_{CBPSS} \qquad (10)$$
$$r = 0,1,\ldots\ N_{CBPSS} - 1$$

The second permutation reverses the second permutation in the interleaver.

$$i = s \cdot \text{floor}(j/s) + (j + \text{floor}(N_{COL} \cdot j / N_{CBPSS})) \bmod s \qquad (11)$$
$$j = 0, 1, \ldots N_{CBPSS} - 1$$

s is defined as above.
The third permutation reversed the first permutation of the interleaver:

$$k = N_{COL}\, i - (N_{CBPSS}-1)\ \text{floor}\,(i/ N_{ROW}) \quad i = 0,1,\ldots,N_{CBPSS} - 1 \qquad (12)$$

2.2.2 Interleaver Configurations

The bits at the output of the stream parser are divided into block of N_{CBPSS}, each block is interleaved by an interleaver based on the 802.11a interleaver.

This interleaver, which is based on entering the data in rows, and reading it out in columns, has a different number of columns and rows when a 20MHz channel is used and when a 40MHz channel is used. The numbers are described in the Table 2-6.

Parameter	20MHz	40MHz
N_{COL}	13	18
N_{ROW}	$4N_{BPSC}$	$6N_{BPSC}$
N_{ROT}	11	29

Table 2-6 Number of row and columns in the interleaver

After the 802.11a like operations have been applied, if more than one spatial stream exists, a third operation called frequency rotation is applied to the additional spatial streams. The parameter for the frequency rotation is N_{ROT}.

An additional parameter is the spatial stream index iss=0,..,Nss-1. The output of the third permutation is a function of the spatial stream index. Where Nss has values between 1 and 4.

Modulation	N_{BPSC}	N_{CBPSS}	Frequency	Iss
BPSK	1	52	20Mhz	0,1,2,3
QPSK	2	104	20Mhz	0,1,2,3
16-QAM	4	208	20Mhz	0,1,2,3
64-QAM	6	312	20Mhz	0,1,2,3
BPSK	1	108	40Mhz	0,1,2,3
QPSK	2	216	40Mhz	0,1,2,3
16-QAM	4	432	40Mhz	0,1,2,3
64-QAM	6	648	40Mhz	0,1,2,3

Table 2-7 Interleaver configurations for 802.11n

The parameters shown in Table 2-7 are the configurations for the interleaver in 802.11n and they are a total of 32 different configurations, because every value of Iss is a different interleaver scheme.

2.2.3 Interleaver Example

The interleaver is defined as a three-permutation process. Table 2-8 shows an random input of 1's and 0's of 52 bits to the interleaver for the BPSK modulation and iss=1. We shall denote by *k* the index of the coded bit before the first permutation; *i* shall be the index after the first and before the second permutation; and *j* shall be the index after the second permutation, just prior to modulation mapping. The mapping from *k* to *i* is shown in Table 2-9, and the mapping from *i* to *j* is shown in Table 2-10, and the mapping from j to r is shown in Table 2-11.

As a specific example, consider the case of $k = 1$ (the #1 input bit). It is mapped by the first permutation to $i = 4$, by the second permutation to $j = 4$, and by the third permutation to $r = 34$ (the #34 output bit).

The interleaved bits are shown in Table 2-12.

##	Bit	##	Bit	##	Bit	##	Bit
0	1	13	0	26	1	39	0
1	1	14	0	27	0	40	0
2	0	15	1	28	0	41	0
3	1	16	0	29	1	42	0
4	0	17	0	30	1	43	1
5	1	18	1	31	1	44	0
6	0	19	0	32	0	45	1
7	0	20	1	33	1	46	1
8	1	21	0	34	0	47	1
9	1	22	1	35	0	48	0
10	1	23	0	36	1	49	0
11	0	24	1	37	1	50	1
12	0	25	0	38	1	51	0

Table 2-8 input data bits

k	i	k	i	k	i	k	i
0	0	13	1	26	2	39	3
1	4	14	5	27	6	40	7
2	8	15	9	28	10	41	11
3	12	16	13	29	14	42	15
4	16	17	17	30	18	43	19
5	20	18	21	31	22	44	23
6	24	19	25	32	26	45	27
7	28	20	29	33	30	46	31
8	32	21	33	34	34	47	35
9	36	22	37	35	38	48	39
10	40	23	41	36	42	49	43
11	44	24	45	37	46	50	47
12	48	25	49	38	50	51	51

Table 2-9 First permutation

i	j	i	j	i	j	i	j
0	0	13	13	26	26	39	39
1	1	14	14	27	27	40	40
2	2	15	15	28	28	41	41
3	3	16	16	29	29	42	42
4	4	17	17	30	30	43	43
5	5	18	18	31	31	44	44
6	6	19	19	32	32	45	45
7	7	20	20	33	33	46	46
8	8	21	21	34	34	47	47
9	9	22	22	35	35	48	48
10	10	23	23	36	36	49	49
11	11	24	24	37	37	50	50
12	12	25	25	38	38	51	51

Table 2-10 Second permutation

j	r	j	r	j	r	j	r
0	30	13	43	26	4	39	17
1	31	14	44	27	5	40	18
2	32	15	45	28	6	41	19
3	33	16	46	29	7	42	20
4	34	17	47	30	8	43	21
5	35	18	48	31	9	44	22
6	36	19	49	32	10	45	23
7	37	20	50	33	11	46	24
8	38	21	51	34	12	47	25
9	39	22	0	35	13	48	26
10	40	23	1	36	14	49	27
11	41	24	2	37	15	50	28
12	42	25	3	38	16	51	29

Table 2-11 Third permutation

##	Bit	##	Bit	##	Bit	##	Bit
0	1	13	0	26	1	39	1
1	1	14	1	27	0	40	1
2	0	15	1	28	0	41	0
3	1	16	1	29	0	42	0
4	0	17	1	30	1	43	0
5	0	18	0	31	0	44	1
6	0	19	0	32	1	45	1
7	1	20	1	33	0	46	1
8	1	21	1	34	1	47	1
9	0	22	0	35	1	48	1
10	0	23	0	36	0	49	0
11	1	24	0	37	0	50	1
12	0	25	0	38	1	51	0

Table 2-12 Output data bits

2.3 802.16e LAN/MAN Broadband Wireless LANS

Figure 2-3 Interleaver 802.16e in/out signals

Channel coding procedures include randomization, FEC (Forward Error Correction) encoding, bit interleaving, repetition and modulation. As shown in Figure 2-4.

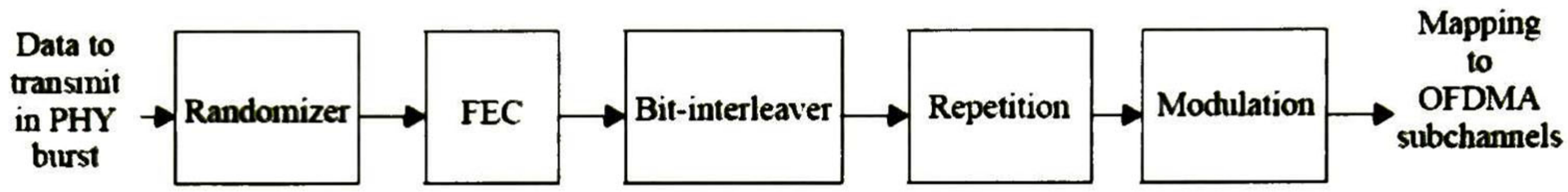


Figure 2-4 Channel coding process for regular and repetition coding transmission

2.3.1 Permutation Definition

All encoded data bits shall be interleaved by a block interleaver with a block size corresponding to the number of coded bits per encoded block size N_{CBPS}, has is done in the 802.11a, using the two permutation equations (13) and (14).

$$i = N_{ROW}\,(k \bmod N_{COL}) + (k/N_{COL}) \qquad k = 0,1,\ldots,N_{CBPS} - 1 \qquad (13)$$

$$j = S\,(i/S) + (i + N_{CBPS} - ((N_{COL} * i)/N_{CBPS})) \bmod S \qquad i = 0, 1,\ldots,N_{CBPS} - 1 \qquad (14)$$

2.3.2 Interleaver Configurations

The interleaver configurations used in 802.16e are shown in Table 2-13, there is a total of 10 different interleaver configurations for different sizes of interleaver blocks for the same modulation.

Modulation	Nrow	Ncol	S	N_{CBPS}
QPSK	6	16	1	96
QPSK	12	16	1	192
QPSK	18	16	1	288
QPSK	24	16	1	384
QPSK	30	16	1	480
QPSK	36	16	1	576
16-QAM	24	16	2	384
16-QAM	36	16	2	576
64-QAM	18	16	3	288
64-QAM	36	16	3	576

Table 2-13 Rate-dependent parameters for 802.16e

2.3.3 Interleaver Example

The interleaver is defined as a two-permutation process. Table 2-14 shows an input of 1's and 0's of 96 bits to the interleaver. We shall denote by *k* the index of the coded bit before the first permutation; *i* shall be the index after the first and before the second permutation; and *j* shall be the index after the second permutation. The mapping from *k* to *i* is shown in Table 2-15, and the mapping from *i* to *j* is shown in Table 2-16.

As a specific example, consider the case of $k = 22$ (the #22 input bit). It is mapped by the first permutation to $i = 37$ and by the second permutation to $j = 37$(the #37 output bit). The interleaved bits are shown in Table 2-17.

##	Bit	##	Bit	##	Bit	##	Bit	##	Bit	##	Bit
0	1	16	1	32	0	48	0	64	0	80	0
1	1	17	0	33	0	49	1	65	1	81	0
2	0	18	1	34	0	50	1	66	0	82	1
3	0	19	0	35	0	51	0	67	1	83	1
4	0	20	1	36	0	52	1	68	1	84	0
5	1	21	1	37	1	53	1	69	1	85	1
6	1	22	1	38	0	54	1	70	1	86	0
7	0	23	0	39	0	55	0	71	1	87	1
8	1	24	0	40	0	56	0	72	1	88	0
9	0	25	0	41	0	57	1	73	1	89	0
10	1	26	1	42	1	58	0	74	1	90	0
11	0	27	0	43	0	59	0	75	1	91	1
12	1	28	0	44	1	60	0	76	1	92	1
13	1	29	1	45	0	61	0	77	1	93	1
14	1	30	1	46	0	62	1	78	0	94	1
15	1	31	0	47	1	63	1	79	1	95	1

Table 2-14 Input data bits

k	i	k	i	k	i	k	i	k	i	k	i
0	0	16	1	32	2	48	3	64	4	80	5
1	6	17	7	33	8	49	9	65	10	81	11
2	12	18	13	34	14	50	15	66	16	82	17
3	18	19	19	35	20	51	21	67	22	83	23
4	24	20	25	36	26	52	27	68	28	84	29
5	30	21	31	37	32	53	33	69	34	85	35
6	36	22	37	38	38	54	39	70	40	86	41
7	42	23	43	39	44	55	45	71	46	87	47
8	48	24	49	40	50	56	51	72	52	88	53
9	54	25	55	41	56	57	57	73	58	89	59
10	60	26	61	42	62	58	63	74	64	90	65
11	66	27	67	43	68	59	69	75	70	91	71
12	72	28	73	44	74	60	75	76	76	92	77
13	78	29	79	45	80	61	81	77	82	93	83
14	84	30	85	46	86	62	87	78	88	94	89
15	90	31	91	47	92	63	93	79	94	95	95

Table 2-15 First permutation

i	j	i	j	i	j	i	j	i	j	i	j
0	0	16	16	32	32	48	48	64	64	80	80
1	1	17	17	33	33	49	49	65	65	81	81
2	2	18	18	34	34	50	50	66	66	82	82
3	3	19	19	35	35	51	51	67	67	83	83
4	4	20	20	36	36	52	52	68	68	84	84
5	5	21	21	37	37	53	53	69	69	85	85
6	6	22	22	38	38	54	54	70	70	86	86
7	7	23	23	39	39	55	55	71	71	87	87
8	8	24	24	40	40	56	56	72	72	88	88
9	9	25	25	41	41	57	57	73	73	89	89
10	10	26	26	42	42	58	58	74	74	90	90
11	11	27	27	43	43	59	59	75	75	91	91
12	12	28	28	44	44	60	60	76	76	92	92
13	13	29	29	45	45	61	61	77	77	93	93
14	14	30	30	46	46	62	62	78	78	94	94
15	15	31	31	47	47	63	63	79	79	95	95

Table 2-16 Second permutation

##	Bit	##	Bit	##	Bit	##	Bit	##	Bit	##	Bit
0	1	16	0	32	1	48	1	64	1	80	0
1	1	17	1	33	1	49	0	65	0	81	0
2	0	18	0	34	1	50	0	66	0	82	1
3	0	19	0	35	1	51	0	67	0	83	1
4	0	20	0	36	1	52	1	68	0	84	1
5	0	21	0	37	1	53	0	69	0	85	1
6	1	22	1	38	0	54	0	70	1	86	0
7	0	23	1	39	1	55	0	71	1	87	1
8	0	24	0	40	1	56	0	72	1	88	0
9	1	25	1	41	0	57	1	73	0	89	1
10	1	26	0	42	0	58	1	74	1	90	1
11	0	27	1	43	0	59	0	75	0	91	0
12	0	28	1	44	0	60	1	76	1	92	1
13	1	29	0	45	0	61	1	77	1	93	1
14	0	30	1	46	1	62	1	78	1	94	1
15	1	31	1	47	1	63	0	79	1	95	1

Table 2-17 Output data bits

2.4 DVB Digital Video Broadcasting

The DVB standard has 3 interleaver definitions classified in 2 kinds:

- Outer Interleaving: consist of convolutional byte-wise interleaver.
- Inner Interleaving: consists of bit-wise interleaving followed by symbol interleaving.

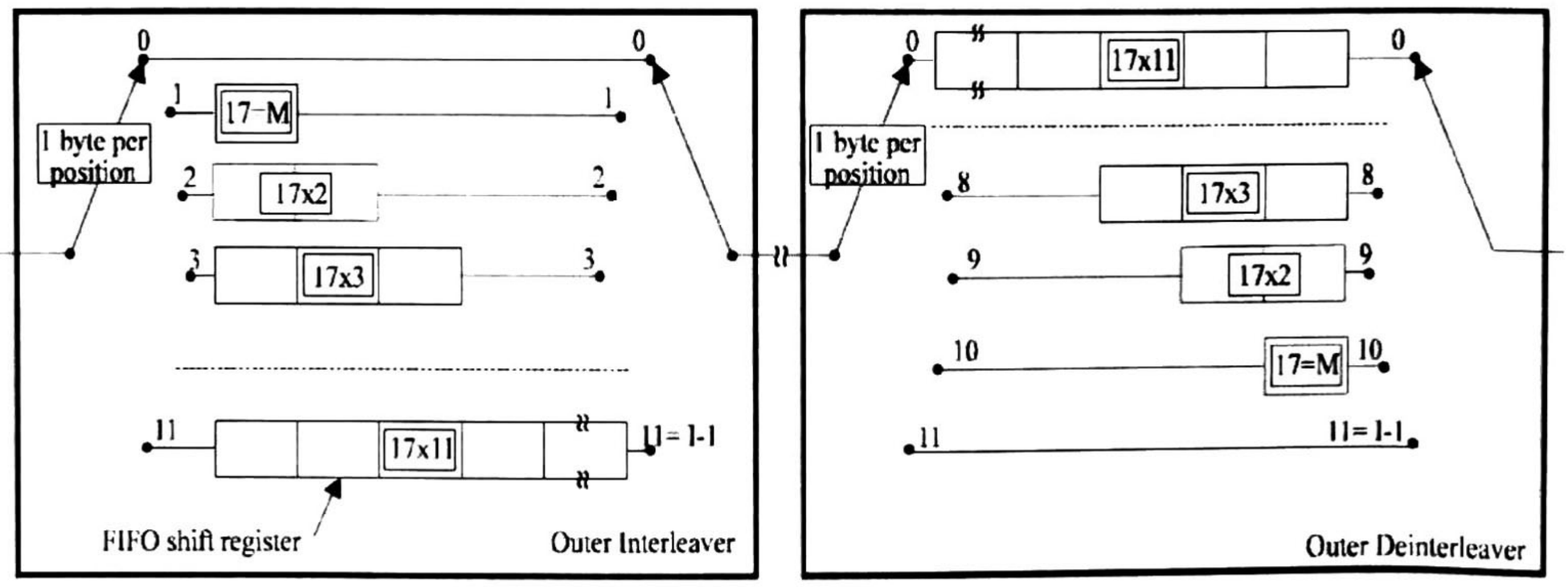
2.4.1 Convolutional Byte-Wise Interleaving

Following the conceptual scheme of Figure 2-6, the convolutional byte-wise interleaver with depth I = 12 is applied.

Figure 2-5 Convolutional interleaver in/out signals

2.4.1.1 Permutation Definition

The interleaver shall be composed of I = 12 branches, cyclically connected to the input byte-stream by the input switch. Each branch j shall be a First-In, First-Out (FIFO) shift register, with depth j x M cells where M = 17 = N/I, N = 204. The cells of the FIFO shall contain 1 byte, and the input and output switches shall be synchronized. For synchronization purposes, the SYNC bytes shall always be routed in the branch "0" of the interleaver (corresponding to a null delay) [28]. The SYNC bytes are the first bits of the block to interleave.



Sync byte always passes through branch 0

Figure 2-6 Conceptual diagram of the outer Interleaver and deinterleaver

The deinterleaver is similar in principle, to the interleaver, but the branch indices are reversed.

2.4.1.2 Interleaver Configurations

There is only one configuration declared for this Convolutional interleaver in this standard.

2.4.2 Bit-Wise Interleaving

6 bit-wise interleavers are declared with different configurations. The block size is the same for each interleaver, but the interleaving sequence is different in each case [28]. The bit interleaving block size is 126 bits.

Figure 2-7 Bit-wise Interleaver In/out signals

2.4.2.1 Permutation Definition

For each bit interleaver, the input bit vector is defined by:

$$B(e) = (b_{e,0}, b_{e,1}, b_{e,2}, \ldots, b_{e,125}) \quad (15)$$

Where e ranges from 0 to v-1.

The interleaved output vector

$$A(e) = (a_{e,0}, a_{e,1}, a_{e,2}, \ldots, a_{e,125}) \quad (16)$$

Is defined by:

$$a_{e,w} = b_{e,He(w)} \qquad w = 0, 1, 2, \ldots, 125 \quad (17)$$

Where $H_e(w)$ is a permutation function which is different for each interleaver.

$H_e(w)$ is defined as follows for each interleaver:

$$I0: H_0(w) = w \quad (18)$$
$$I1: H_1(w) = (w + 63) \bmod 126 \quad (19)$$
$$I2: H_2(w) = (w + 105) \bmod 126 \quad (20)$$
$$I3: H_3(w) = (w + 42) \bmod 126 \quad (21)$$
$$I4: H_4(w) = (w + 21) \bmod 126 \quad (22)$$
$$I5: H_5(w) = (w + 84) \bmod 126 \quad (23)$$

The outputs of the v bit interleavers are grouped to form the digital data symbols, such that each symbol of v bits will consist of exactly one bit from each of the v interleavers. Hence, the output from the bit-wise interleaver is a v bit word y' that has the output of I0 as its most significant bit, i.e.:

$$y'_w = (a_{0,w}, a_{1,w}, \ldots, a_{v-1,w}) \quad (24)$$

2.4.2.2 Interleaver Configurations

Three different bit interleaver configurations exist, depending the value of v, the values of v can be 2, 4 or 6. If v=2 then only the H_0 and H_1 interleavers are activated, if v=4 then the H_0, H_1, H_2 and H_3 are activated. If v=6 all the bit wise interleavers are activated.

2.4.2.3 Interleaver Example

The interleaver is defined has a permutation process. Table 2-18 shows an input of 1's and 0's of 126 bits to the interleaver with e=5, equation (23). We shall denote by w the index of the coded bit before the permutation; H(w) shall be the index after the permutation. The mapping from w to H(w) is shown in Table 2-19.

As a specific example, consider the case of k = 6 (the 6th input bit). It is mapped by the permutation to H(w) = 90 (the #90 output bit). The interleaved bits are shown in Table 2-20.

##	Bit	##	Bit	##	Bit	##	Bit	##	Bit	##	Bit	##	Bit	##	Bit
0	1	16	1	32	0	48	0	64	0	80	0	96	1	112	1
1	1	17	0	33	0	49	1	65	1	81	0	97	1	113	0
2	0	18	1	34	0	50	1	66	0	82	1	98	0	114	1
3	0	19	0	35	0	51	0	67	1	83	1	99	0	115	0
4	0	20	1	36	0	52	1	68	1	84	0	100	0	116	1
5	1	21	1	37	1	53	1	69	1	85	1	101	1	117	1
6	1	22	1	38	0	54	1	70	1	86	0	102	1	118	1
7	0	23	0	39	0	55	0	71	1	87	1	103	0	119	0
8	1	24	0	40	0	56	0	72	1	88	0	104	1	120	0
9	0	25	0	41	0	57	1	73	1	89	0	105	0	121	0
10	1	26	1	42	1	58	0	74	1	90	0	106	1	122	1
11	0	27	0	43	0	59	0	75	1	91	1	107	0	123	0
12	1	28	0	44	1	60	0	76	1	92	1	108	1	124	0
13	1	29	1	45	0	61	0	77	1	93	1	109	1	125	1
14	1	30	1	46	0	62	1	78	0	94	1	110	1		
15	1	31	0	47	1	63	1	79	1	95	1	111	1		

Table 2-18 Input data bits

w	H(w)	w	H(w)	W	H(w)	w	H(w)	w	H(w)	w	H(w)	w	H(w)	w	H(w)
0	84	16	100	32	116	48	6	64	22	80	38	96	54	112	70
1	85	17	101	33	117	49	7	65	23	81	39	97	55	113	71
2	86	18	102	34	118	50	8	66	24	82	40	98	56	114	72
3	87	19	103	35	119	51	9	67	25	83	41	99	57	115	73
4	88	20	104	36	120	52	10	68	26	84	42	100	58	116	74
5	89	21	105	37	121	53	11	69	27	85	43	101	59	117	75
6	90	22	106	38	122	54	12	70	28	86	44	102	60	118	76
7	91	23	107	39	123	55	13	71	29	87	45	103	61	119	77
8	92	24	108	40	124	56	14	72	30	88	46	104	62	120	78
9	93	25	109	41	125	57	15	73	31	89	47	105	63	121	79
10	94	26	110	42	0	58	16	74	32	90	48	106	64	122	80
11	95	27	111	43	1	59	17	75	33	91	49	107	65	123	81
12	96	28	112	44	2	60	18	76	34	92	50	108	66	124	82
13	97	29	113	45	3	61	19	77	35	93	51	109	67	125	83
14	98	30	114	46	4	62	20	78	36	94	52	110	68		
15	99	31	115	47	5	63	21	79	37	95	53	111	69		

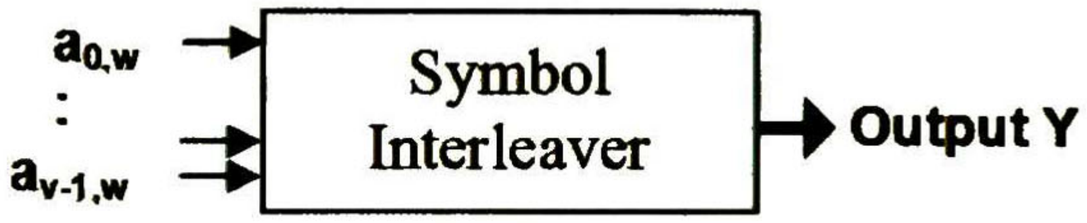
Table 2-19 Permutation

##	Bit	##	Bit	##	Bit	##	Bit	##	Bit	##	Bit	##	Bit	##	Bit
0	1	16	0	32	1	48	1	64	1	80	0	96	1	112	0
1	0	17	0	33	1	49	1	65	1	81	1	97	1	113	1
2	0	18	1	34	0	50	1	66	1	82	1	98	1	114	1
3	1	19	1	35	1	51	1	67	1	83	1	99	1	115	0
4	0	20	0	36	0	52	1	68	1	84	1	100	1	116	0
5	1	21	1	37	0	53	1	69	0	85	1	101	0	117	0
6	1	22	0	38	1	54	0	70	1	86	0	102	1	118	0
7	0	23	1	39	1	55	0	71	0	87	0	103	0	119	0
8	1	24	1	40	0	56	0	72	1	88	0	104	1	120	0
9	1	25	1	41	1	57	1	73	1	89	1	105	1	121	1
10	1	26	1	42	0	58	1	74	1	90	1	106	1	122	0
11	0	27	1	43	1	59	0	75	0	91	0	107	0	123	0
12	0	28	1	44	0	60	1	76	0	92	1	108	0	124	0
13	1	29	1	45	0	61	0	77	0	93	0	109	0	125	0
14	0	30	1	46	0	62	1	78	1	94	1	110	1		
15	0	31	1	47	1	63	0	79	0	95	0	111	0		

Table 2-20 Output data bits

2.4.3 Symbol Interleaver

The purpose of the symbol interleaver is to map v bit words onto the 1512 (2K mode) or 6048 (8K mode) active carriers per OFDM symbol. The symbol interleaver acts on blocks of 1512 (2K mode) or 6048 (8K mode) data symbols [28].



(V x Nmax) bits
$Y' = (y'_0, y'_1, y'_2, \ldots y'_{Nmax-1})$

(V x Nmax) bits
$Y = (y_0, y_1, y_2, \ldots y_{Nmax-1})$

Figure 2-8 Symbol Interleaver In/out signals

Thus in the 2K mode, 12 groups of 126 data words from the bit interleaver are read sequentially into a vector $Y' = (y'_0, y'_1, y'_2, \ldots y'_{1511})$. Similarly in the 8K mode, a vector $Y' = (y'_0, y'_1, y'_2, \ldots y'_{6047})$ is assembled from 48 groups of 126 data words.

2.4.3.1 Permutation and Configurations Definition

The interleaved vector $Y = (y_0, y_1, y_2, \ldots y_{Nmax-1})$ is defined by:

$y_{H(q)} = y'_q$ for even symbols for $q = 0, \ldots, N_{max}-1$ (25)
$y_q = y'_{H(q)}$ for odd symbols for $q = 0, \ldots, N_{max}-1$ (26)

Where $N_{max} = 1512$ in the 2K mode and $N_{max} = 6048$ in the 8K mode.

H(q) is a permutation function defined by the following.

An $(N_r - 1)$ bit binary word R'_i is defined, with $N_r = \log2\ M_{max}$, where $M_{max} = 2048$ in the 2K mode and $M_{max} = 8192$ in the 8K mode, where R'_i takes the following values:

$i = 0,1$: $R'_i[N_{r-2}, N_{r-3}, \ldots, 1, 0] = 0, 0, \ldots, 0, 0$
$i = 2$: $R'_i[N_{r-2}, N_{r-3}, \ldots, 1, 0] = 0, 0, \ldots, 0, 1$
$2 < i < M_{max}$: { $R' [N_{r-3}, N_{r-4}, \ldots, 1, 0] = R'_i\text{-}1 [N_{r-2}, N_{r-3}, \ldots, 2, 1]$;
in the 2K mode: $R'_i [9] = R'i\text{-}1 [0] \oplus R'_i\text{-}1 [3]$
in the 8K mode: $R'_i [11] = R'i\text{-}1 [0] \oplus R'_i\text{-}1 [1] \oplus R'_i\text{-}1[4] \oplus R'_i\text{-}1 [6]$ }

A vector R_i is derived from the vector R'_i by the bit permutations given in Table 2-21 and Table 2-22.

R'_i bit positions	9	8	7	6	5	4	3	2	1	0
R_i bit positions	0	7	5	1	8	2	6	9	3	4

Table 2-21 Bit permutations for the 2K mode

R'_i bit positions	11	10	9	8	7	6	5	4	3	2	1	0
R_i bit positions	5	11	3	0	10	8	6	9	2	4	1	7

Table 2-22 Bit permutations for the 8K mode

The permutation function H(q) is defined by the following algorithm:

q = 0;
for (i = 0; i < M_{max}; i = i + 1)

$$H(q) = (i \bmod 2) \cdot 2^{Nr-1} + \sum_{j=0}^{N_r-2} R_i(j) \cdot 2^j$$

if (H(q)<N_{max}) q=q+1; }

A schematic block diagram of the algorithm used to generate the permutation function is represented in Figure 2-9 for the 2K mode and in Figure 2-10 for the 8K mode.

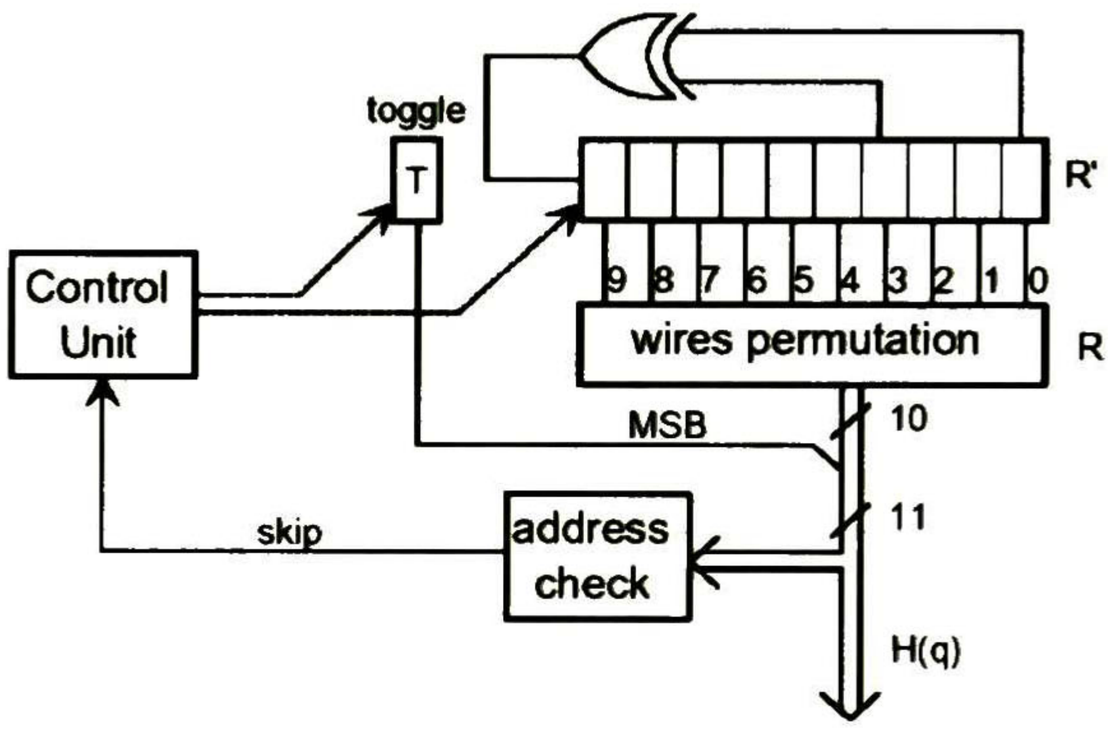


Figure 2-9 Symbol Interleaver address generation scheme for the 2K mode

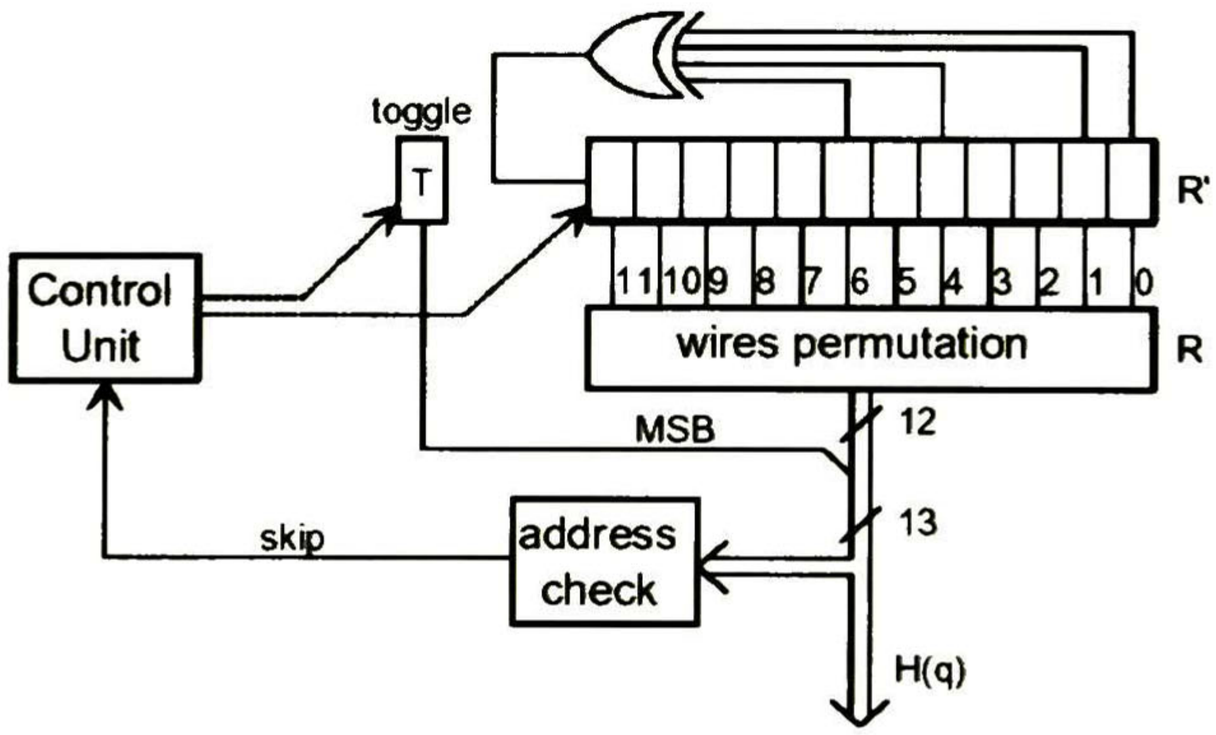


Figure 2-10 Symbol Interleaver address generation scheme for the 2K mode

In a similar way to y', y is made up of v bits:

$$y_{q'} = (y_{0,q'}, y_{1,q'}, \ldots, y_{v-1,q'}) \quad (27)$$

Where q' is the symbol number at the output of the symbol interleaver.

3 Classes of Interleaver Implementations

This chapter reviews four examples of implementation of interleavers. These implementations were selected because they are the state of the art in interleaver implementations, and they have some reconfigurability, and perform some interleaving schemes. The interleavers of the standards mapped by every Interleaver Implementation are shown in Table 3-1.

- The LUT articles are shown at [10],[14].
- Special Matrix article is shown at [14].
- Crossbar articles are shown at [19], [20], [21], [22], [23].

Interleaver Implementations	Interleavers Mapped					
	802.11a	802.11n	802.16e	Convolutional	Symbol	Compact Bit
MSI	✓	✓	✓	✓	✓	✓
LUT	✓	✓	✓	✓	✓	✓
Special Matrix	✓	X	✓	X	X	X
Crossbar	✓	✓	✓	X	✓	✓

Table 3-1 Interleavers mapped by the interleaver implementations

3.1 Interleaver with LUT

Interleaver with LUT (Look Up Table) is the easiest way to make an specific interleaver. The writing address are saved in a ROM (Read Only Memory), this address has the information of the interleaving scheme used to do the interleaving process. The write address is read sequentially from the ROM by a counter, and this address selects the memory location where the Input data will be written in the Memory.

The Figure 3-1 describes a traditional block interleaver implementation based on a bit addressable memory and a ROM (or look-up table) storing the interleaving sequence.

For interleaving operation the input bits are first written sequentially to the memory and then read in the order defined by the LUT.

For Deinterleaving the bits are written according to the LUT and then read sequentially. The number of bits in the data memory is equal to the largest block size and the number of words in the LUT is at least equal to the sum of the block sizes of all interleaving schemes it should handle. For example in the 802.11a interleaver the data memory size is 288 (maximum block size) bits and the LUT needs 624 (sum of all block sizes) 9-bit entries

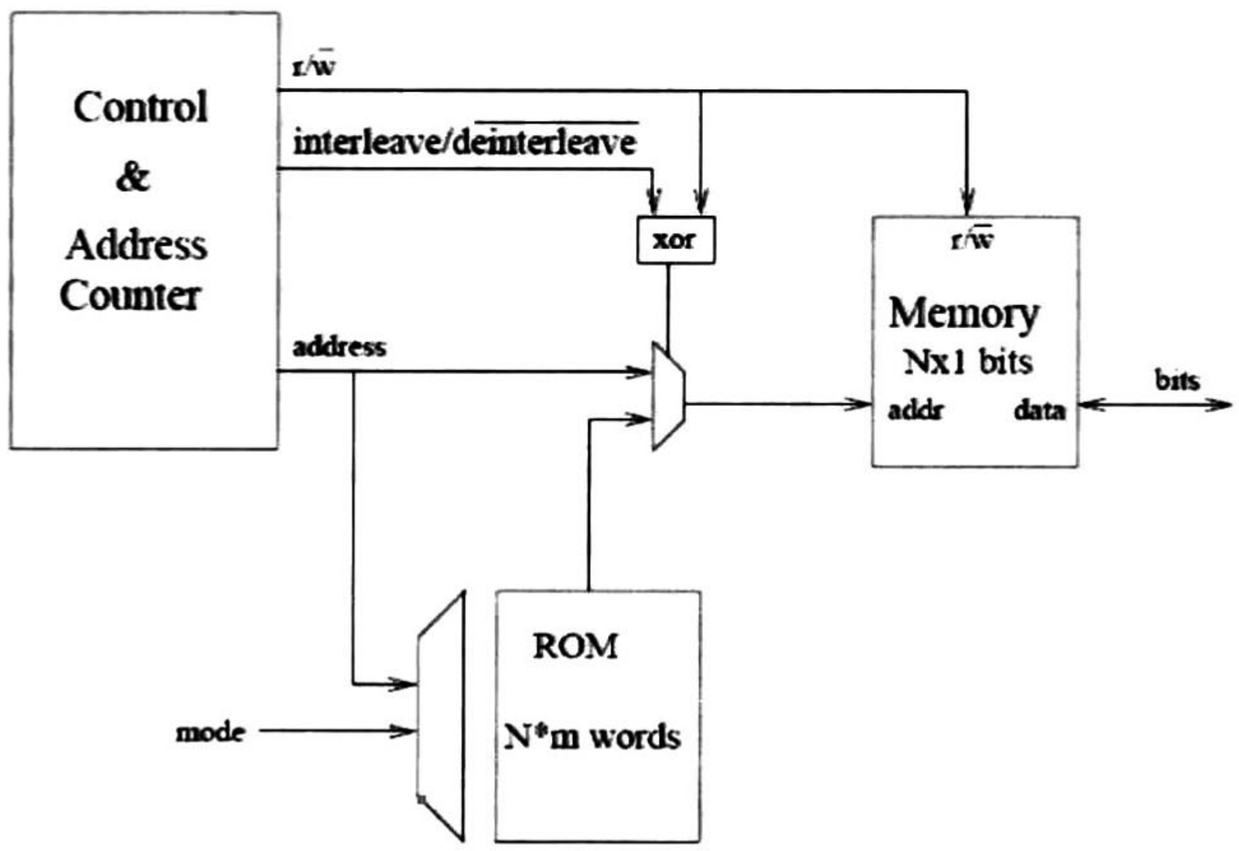


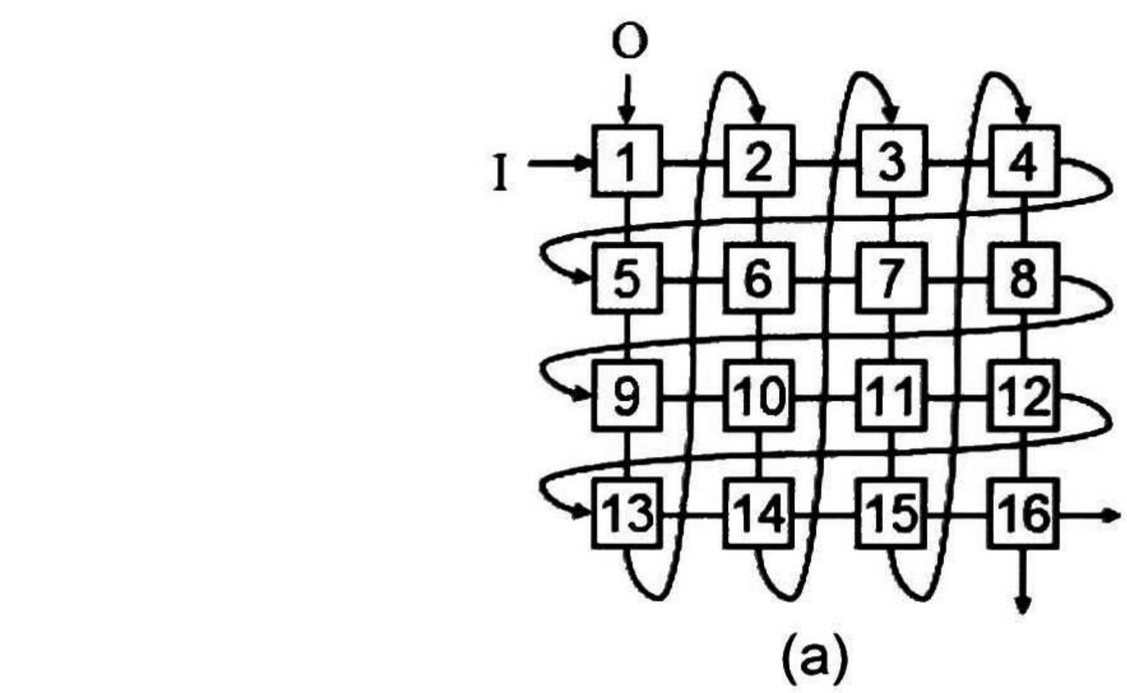
Figure 3-1 Traditional block interleaver implementation

The main advantages of this implementation are that it is very simple and that it is completely general as long as the block size is fixed for each mode.

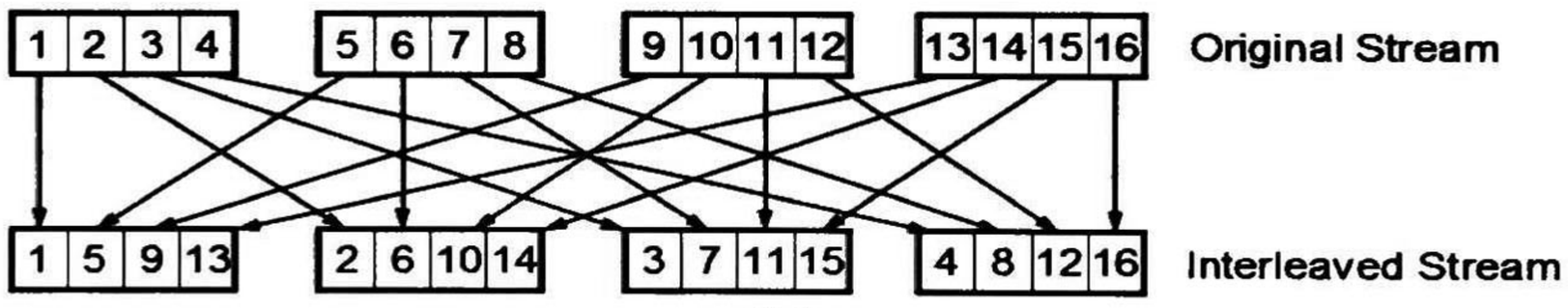
The main disadvantage is that the bits have to be written and read one at a time. Another disadvantage is that the complete interleaving sequence of every interleaving scheme has to be stored explicitly in the LUT, making it relatively large if many modes have to be supported.

3.2 Special Matrix Interleaver

This interleaver implementation comes from the idea of one of that an interleaver can be viewed as a special matrix where symbols are written into the matrix by rows, starting from the position labeled 'I', and read out by columns starting from the 'O' position [14], like shown in Figure 3-2.



(a)



(b)

Figure 3-2 (a)Interleaver based on a matrix, write in row and read in columns,(b) Interleaved Stream output of the interleaver block

The permutation function of the classical block interleaver in the Figure 3-2 is given by:

$$i = m * (k \bmod n) + \text{floor}(k/n) \qquad k = 0,1,\ldots, m*n - 1 \qquad (28)$$

Where "m" is the number of matrix columns and "n" the number of rows.

It should therefore be possible to make the interleaver compatible with the processor word length. Ideally the interleaver interface to the processor should look like an ordinary memory.

The interleaver could be used as a normal data buffer and it would be possible to carry out the interleaving at no extra instruction cost. The number of rows and columns should be configurable. Multiple intra-row and intra-column permutation schemes are supported. This solution can handle 802.11a and 802.11n standards. It might not have to be completely general but preferably the cost for supporting additional modes should be small compared to the corresponding cost (for increasing ROM size) in the traditional approach.

The Figure 3-3 shows the main idea of this architecture. The architecture is based on a special matrix memory block where words are written as rows but read as columns.

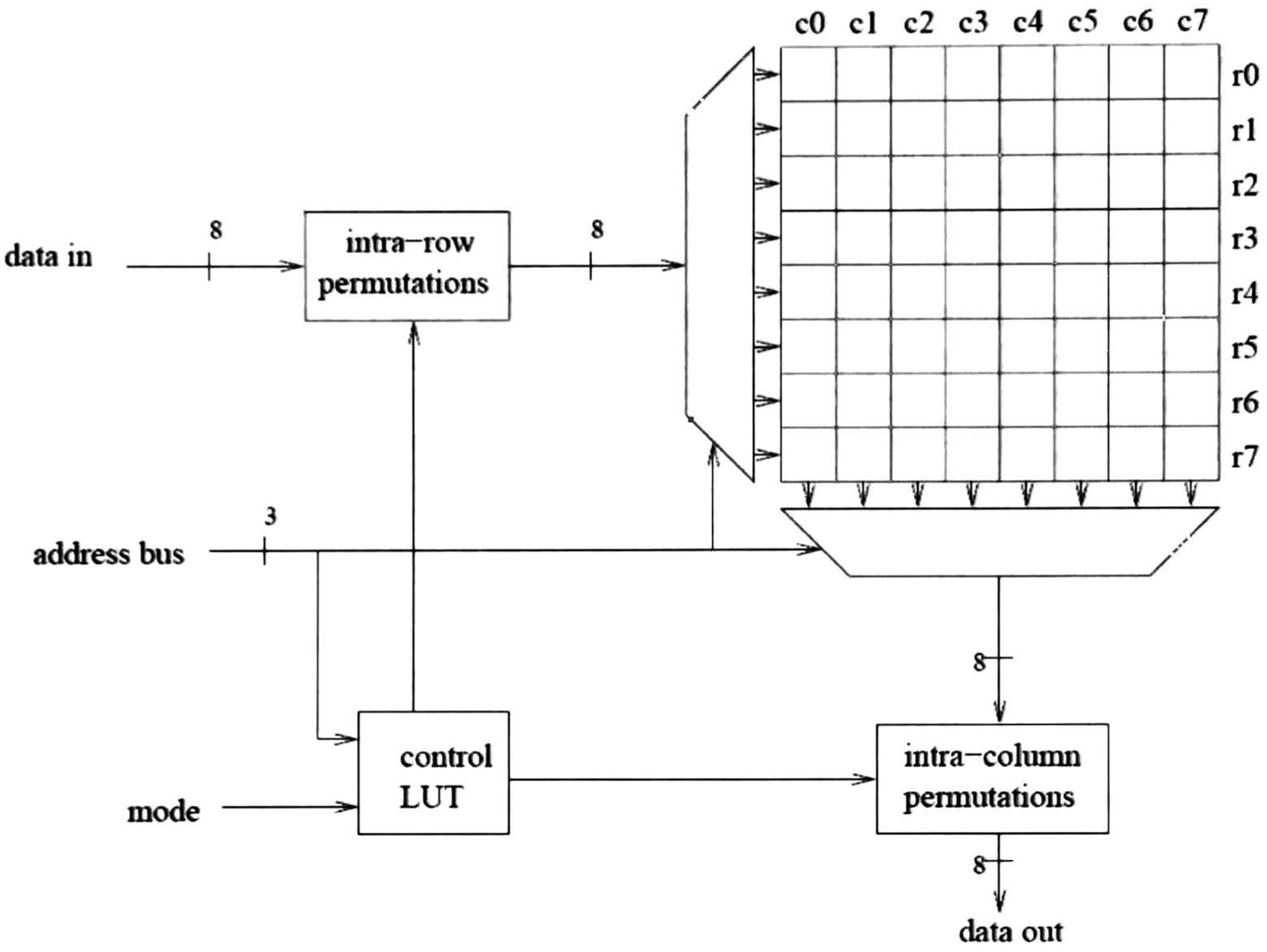


Figure 3-3 Special matrix interleaver

A complete row can be written in one clock cycle and a complete column can be read at one cycle. Intra-row permutations are carried out before the bits are stored to memory by simply reordering the bits on the input data bus. In the same way intra-column permutations are carried out by reordering the bits on the output data bus after the data has been read. If a small number of different permutation schemes are needed, the permutation blocks are just a set of small multiplexers using the same control signal.

The operation for deinterleaving is the same as for interleaving. The deinterleaving is seen as just another interleaving scheme where the row number has become column number and the intra-row permutations have become intra-column permutations and vice versa.

The matrix is dimensioned according to the largest number of rows or columns that has to be supported. If an interleaving scheme with smaller block size is used the extra rows and/or columns can be deactivated to save power.

The interleaver is configured by three control signals specifying the number of rows, the number of columns and the permutation scheme to use. A control block may be needed to decide the permutation mode according to the current mode (and address). This is implemented as a small look-up table

3.3 Interleaver Design Using Crossbars

An crossbar of X inputs and X outputs can interconnect each input to one output, depending on the configuration selected, this behavior can be view has an interleaver.

The crossbar uses configuration parameters to select witch paths are made; this is accomplished by the Control Generator Unit (CGU). The CGU has as many signals as the number of input with length (n_{bits}) dependent of $n_{bits} = Ln(N_{inputs})/Ln(2)$.

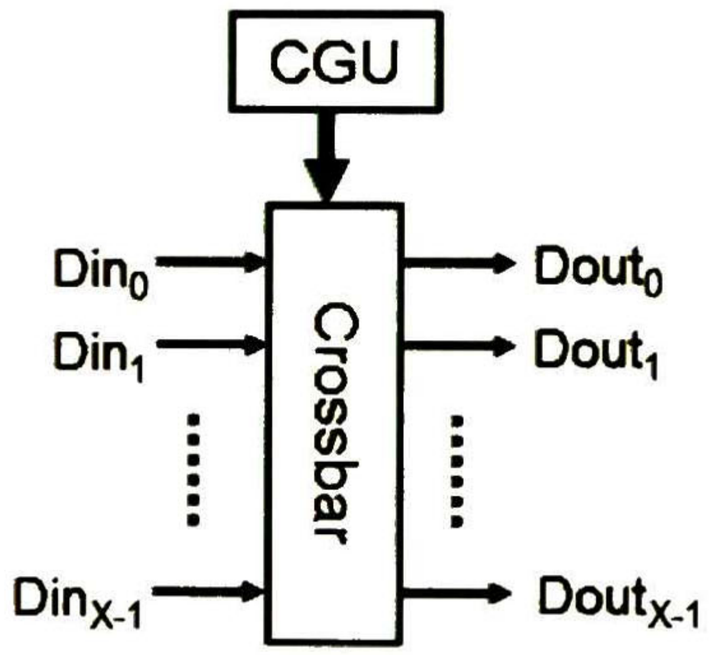


Figure 3-4 Crossbar of X inputs and X outputs

In a crossbar, a separate switch connects each input to each output; this switch is called cross point, each cross point is controlled by the CGU.

Using this implementation we can make a permutation from X inputs to X outputs in one single clock.

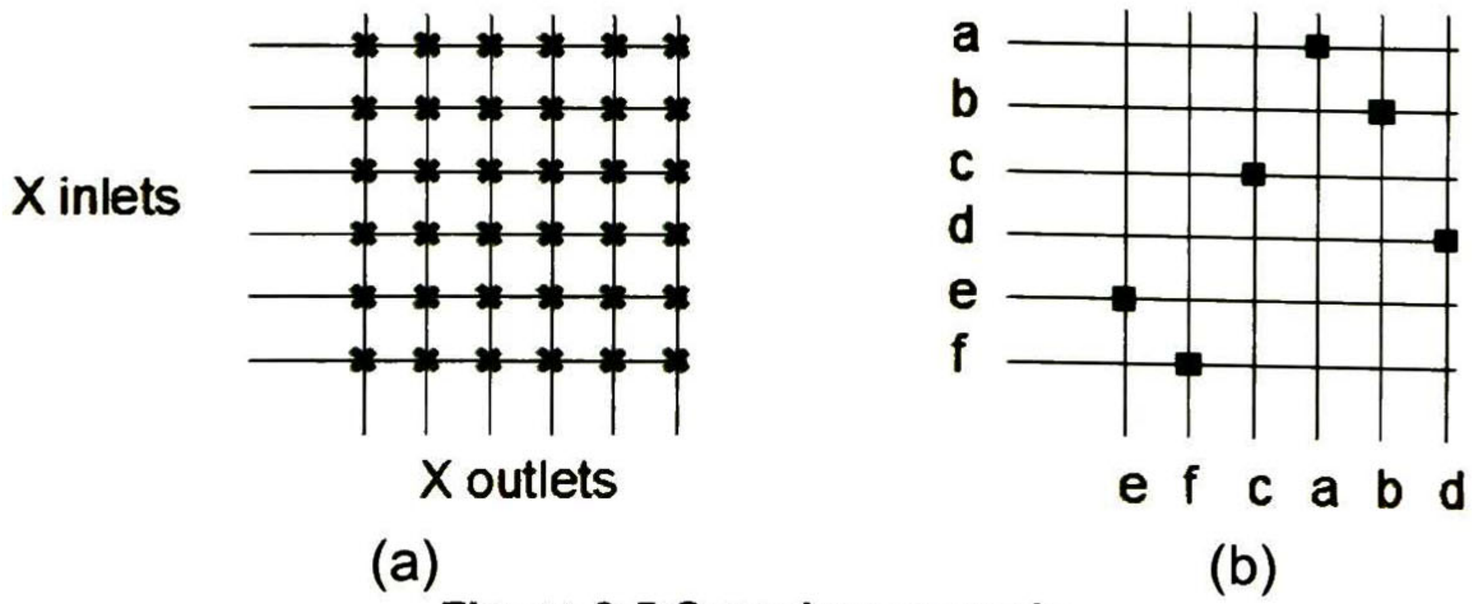


Figure 3-5 Crossbar example

A crossbar of X inputs and outputs can be built using X^2 switches per channel, a crossbar example is shown at Figure 3-6, this shows at (a) a generic programmable crossbar example of Xinlets and Xoutlets, note the use of X^2 cross points, (b) specific example of a crossbar of 6x6 with the programmed cross points.

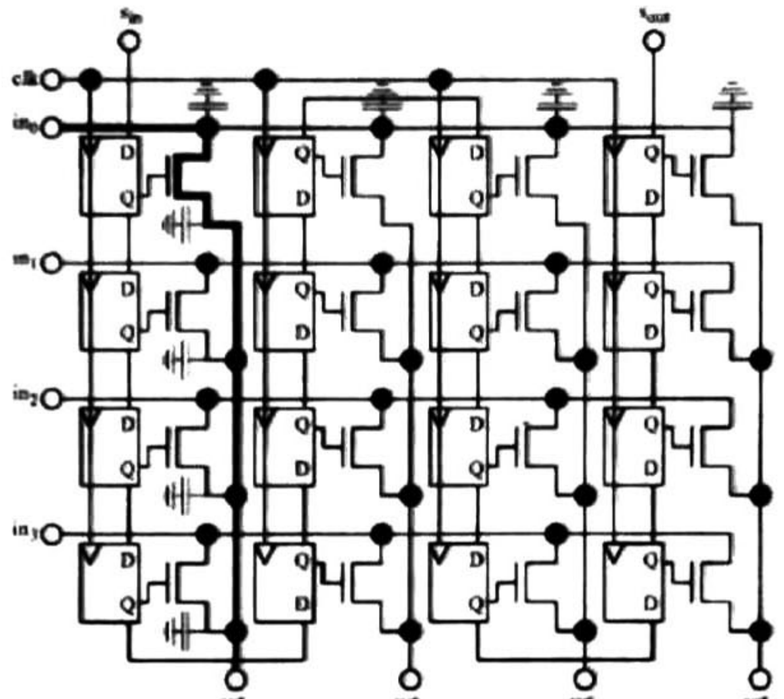


Figure 3-6 Crossbar implementation example

There are many techniques to design huge crossbars [13], [19], [20], [21] with the objective of minimize the load capacitance or the number of switches used.

The Figure 3-6 shows a crossbar with four inputs and four outputs. A path from input in to output out is selected by the flip-flop. The path crosses one pass transistor and is loaded by eight source or drain capacitances.

In this case we also need to create the address generator or save the address sequences in a memory or a micro-programmed address generator.

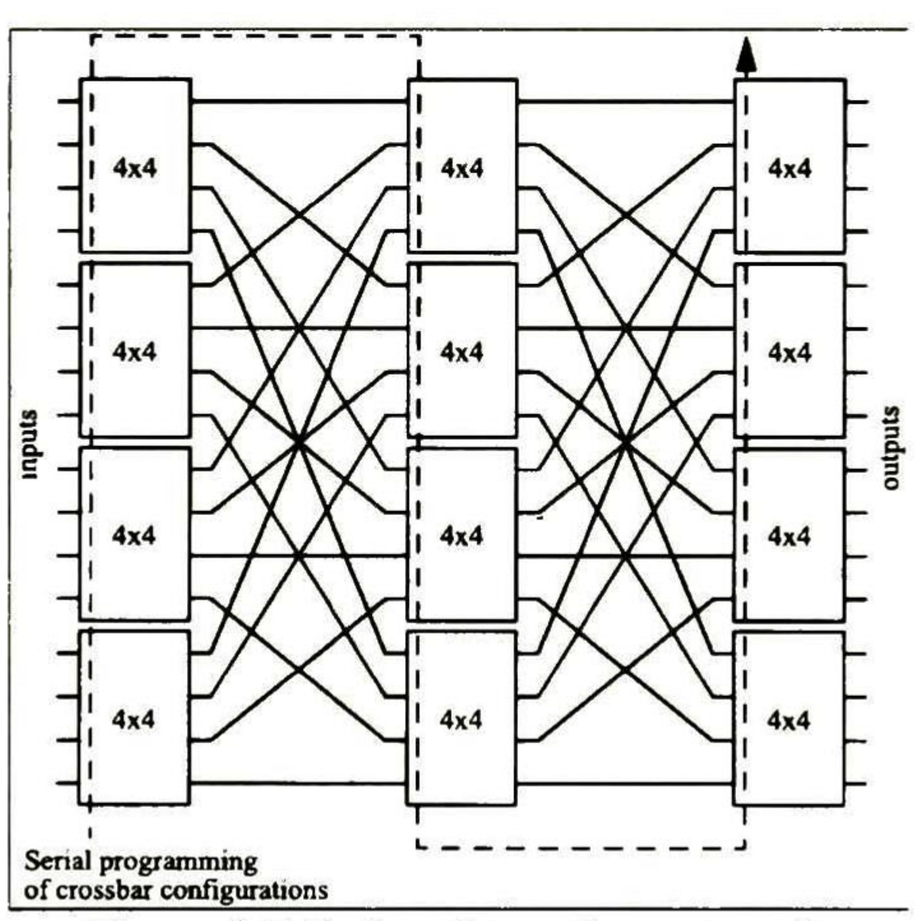


Figure 3-7 Reduced crossbar example

To make this architecture reconfigurable for different width of inputs bits, the crossbars increase in size by a quadratic way, so is necessary to use another alternative. The crossbar can be optimized by some ways like the one shown at Figure 3-7, this reduces the number of switches needed [13], [19], [20], [21].

Another kind of crossbar implementation is the use of a Spatial-Time-Spatial (STS) structure. The spatial crossbar array can be converted to a STS structure, this converts a multistage crossbar in 2 crossbars and a Time Slot Interchanger (TSI), and this is explained at space-time equivalents in connecting networks [22].

With the STS implementation we can make the same work of the crossbar, with the reduction on the first and third stage from N crossbars, to only one, the extra crossbar are mapped on the TSI as a TSI of N spaces. One example of this conversion is shown in Figure 3-8.

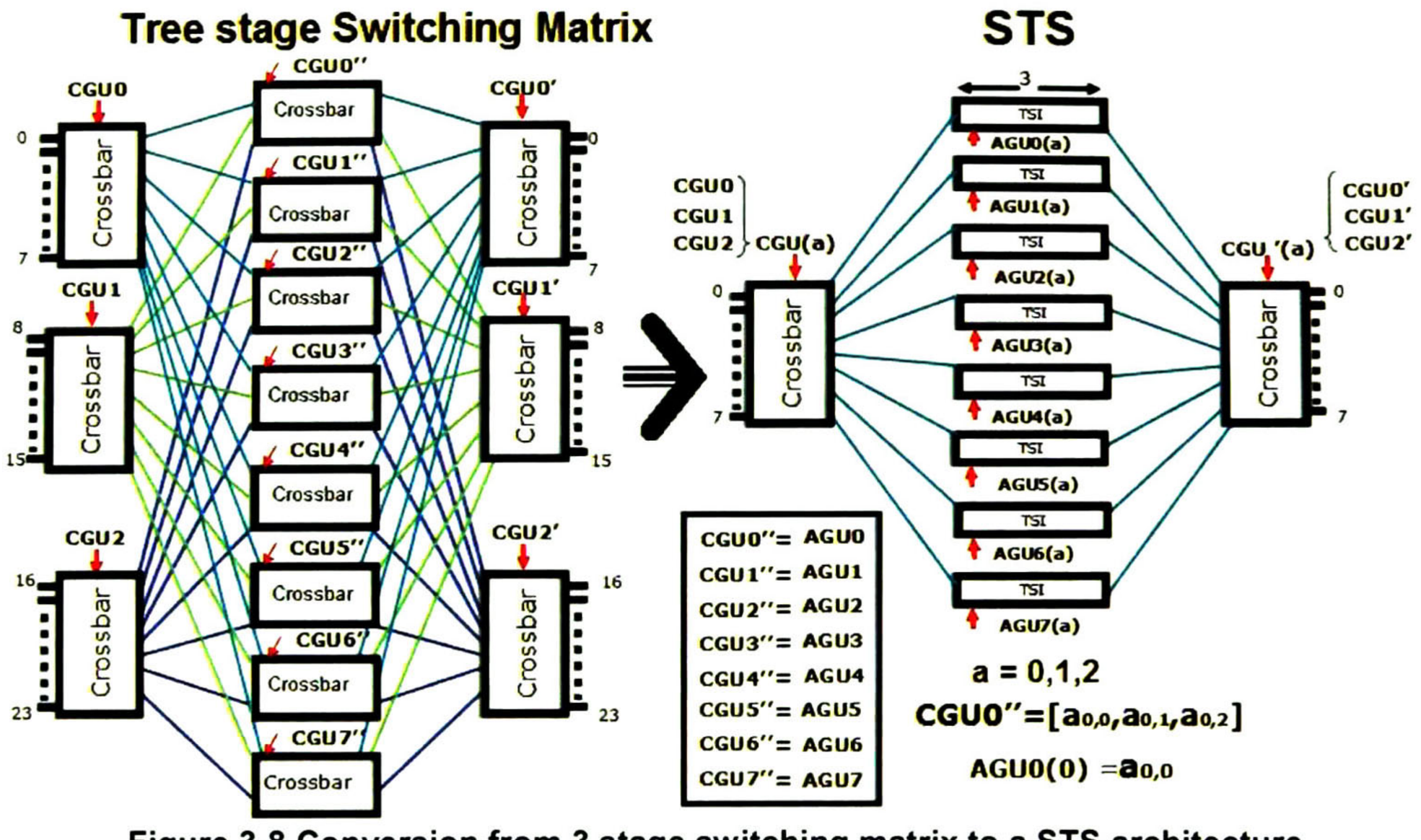


Figure 3-8 Conversion from 3 stage switching matrix to a STS architecture

3.4 Arithmetic Interleaver

This implementation obtain his name by the way the memory gets the Address sequence to save the interleaving data on it, the Write address has the information of the permutation sequence and is generated by an arithmetic AGU.

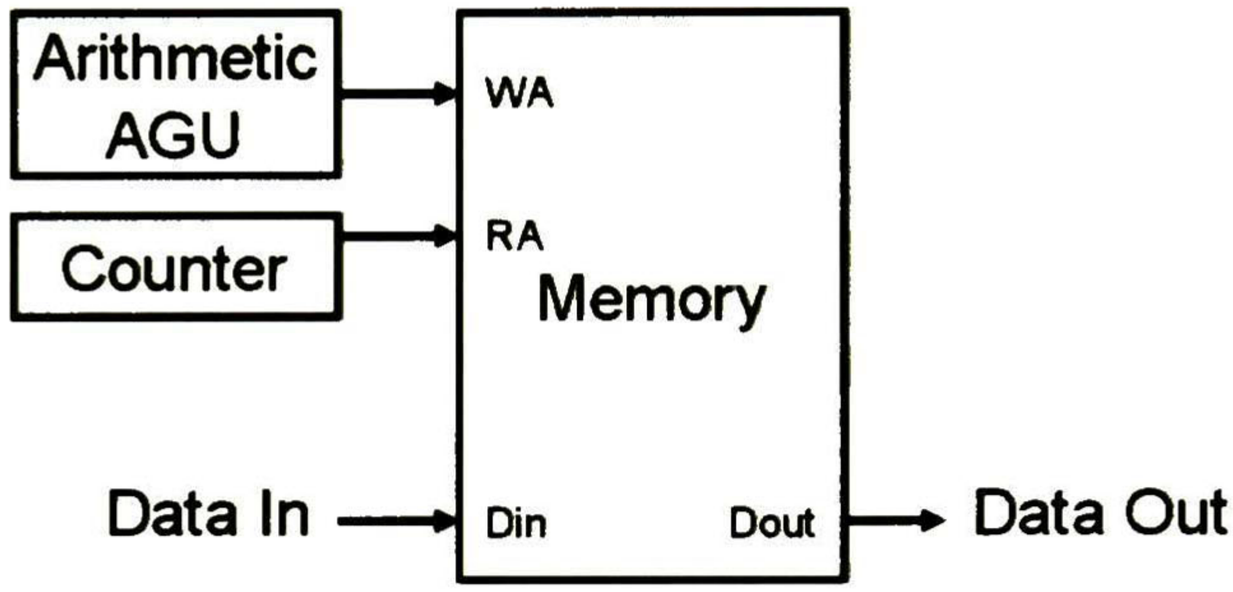


Figure 3-9 Hardware block organization of arithmetic interleaver

Arithmetic AGU has a multiplier, sum, and divider units to make the appropriated operations to generate the address to write the data. For example if we see the equations in the standard 802.11a, these are:

$$i = (\text{NCBPS}/16)\,(k \bmod 16) + \text{floor}\,(k/16) \quad k = 0,1,\ldots,\text{NCBPS} - 1 \qquad (1)$$

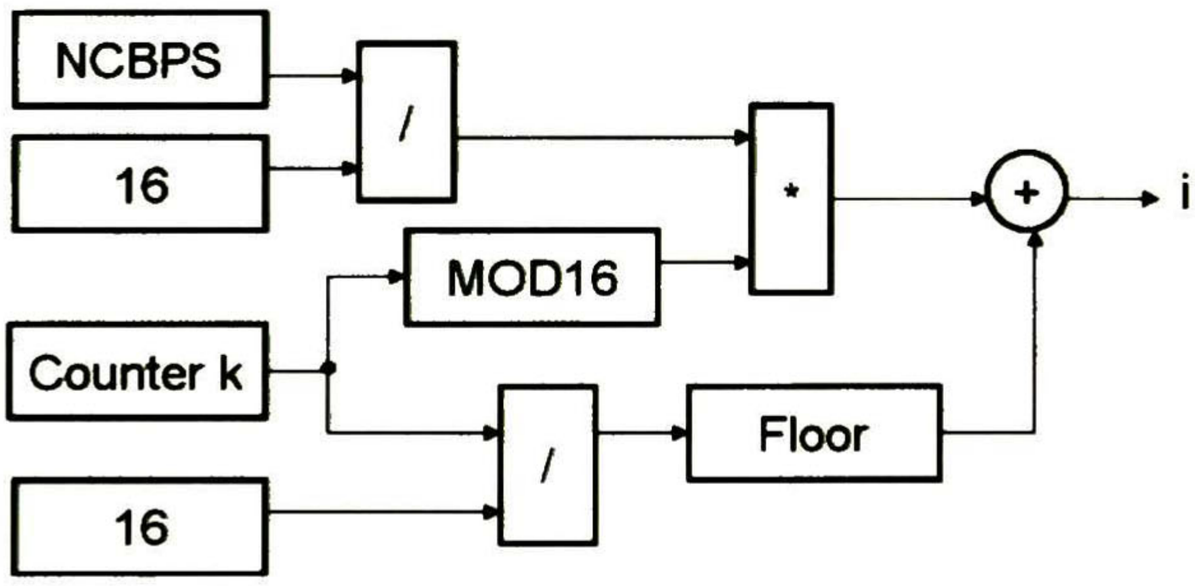


Figure 3-10 Block diagram implementation of equation (1)

So we need a counter for k, 3 dividers by 16, an adder and a multiplier, to implement

$$j = \text{s} \cdot \text{floor}(i/\text{s}) + (i + \text{NCBPS} - \text{floor}(16 \cdot i/\text{NCBPS})) \bmod \text{s} \qquad (2)$$
$$i = 0,1,\ldots\ \text{NCBPS} - 1$$

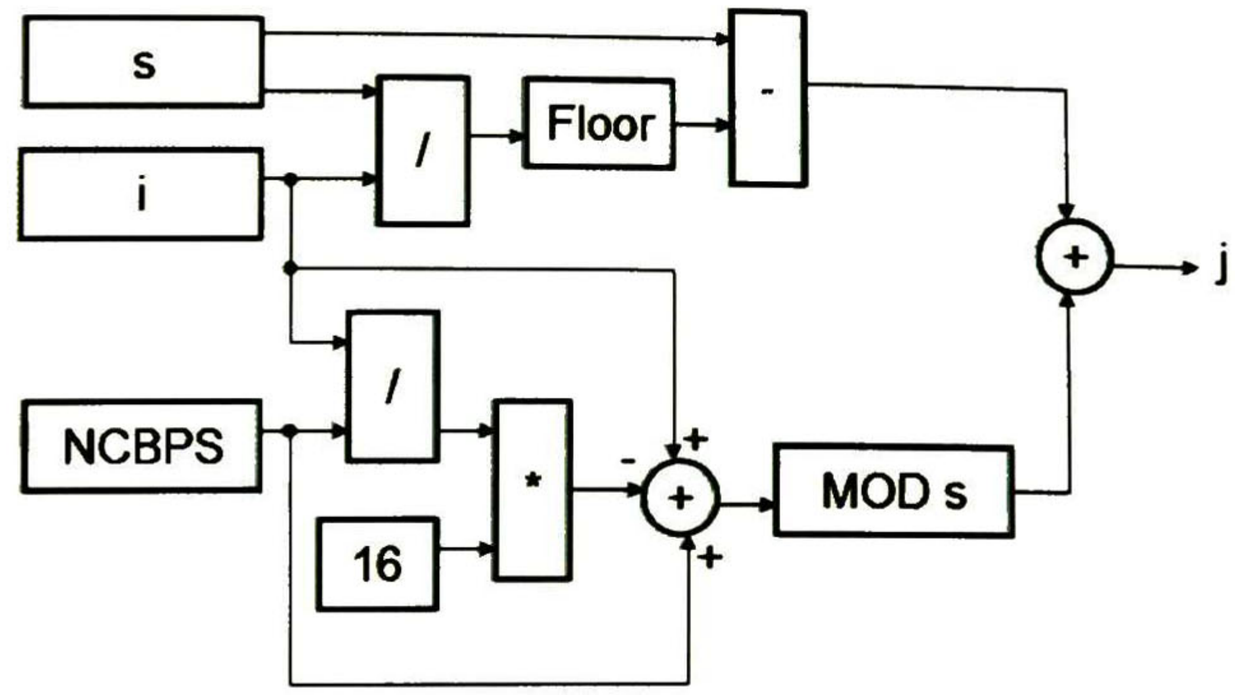


Figure 3-11 Block diagram implementation of equation (2)

So we need 3 dividers by s, NCBPS and s, 3 adders and a multiplier, to implement this address generator (AGU).

If we look inside the Arithmetic AGU for the 802.11a it will look like the junction of the Figure 3-10 and Figure 3-11, this is shown in Figure 3-12.

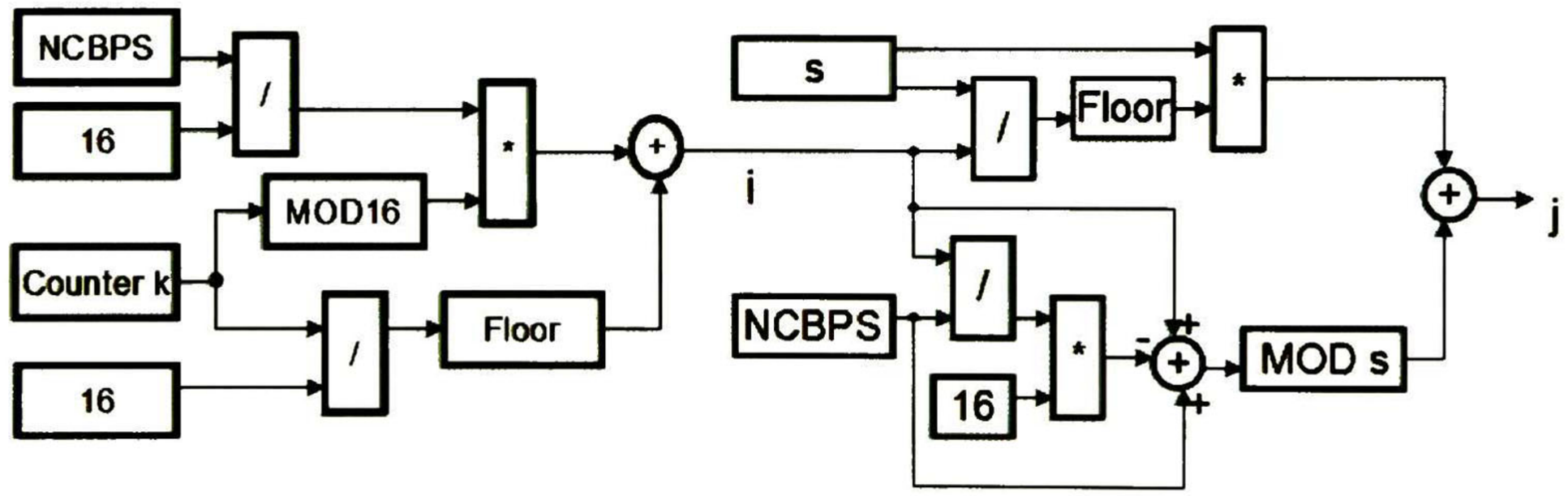


Figure 3-12 Block diagram implementation of Arithmetic AGU for 802.11a

This architecture can be implemented by using a microprocessor or the direct implementation of the dividers, adders and multipliers.

3.5 Interleaver Proposal

The interleaver proposal it's a combination of the STS structure and optimized arithmetical AGU.

3.5.1 Optimized Arithmetical AGU

The arithmetical AGU created, is a simplification of the AGU shown at arithmetic interleaver, the initial idea of the optimized arithmetical AGU implementation was given by Manuel E. Guzmán at the document [24] "Implementation of interleavers", written by Manuel E. Guzmán and Arturo Veloz.

The optimization of the arithmetical AGU starts from the process to eliminate the dividers blocks, and convert them to a more simplified version. This is done by the use of the architectures shown in Figure 3-13 and Figure 3-15, these implementations uses the advantage of feedback and a preload of small permutation sequences.

For the equation (1) we start by the analysis of the generated sequence, shown in Figure 3-14. Here we can see that equation (1) generates a periodic increment in one address to the next one, and it's limited by a modulus factor, that we call Nconf2.

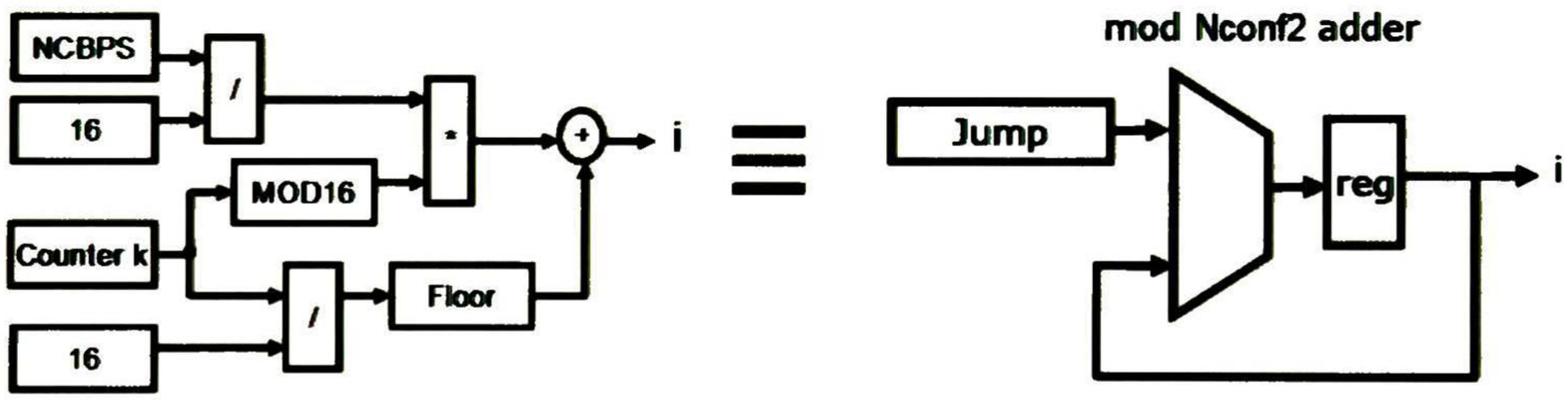


Figure 3-13 Architecture optimization for the equation (1)

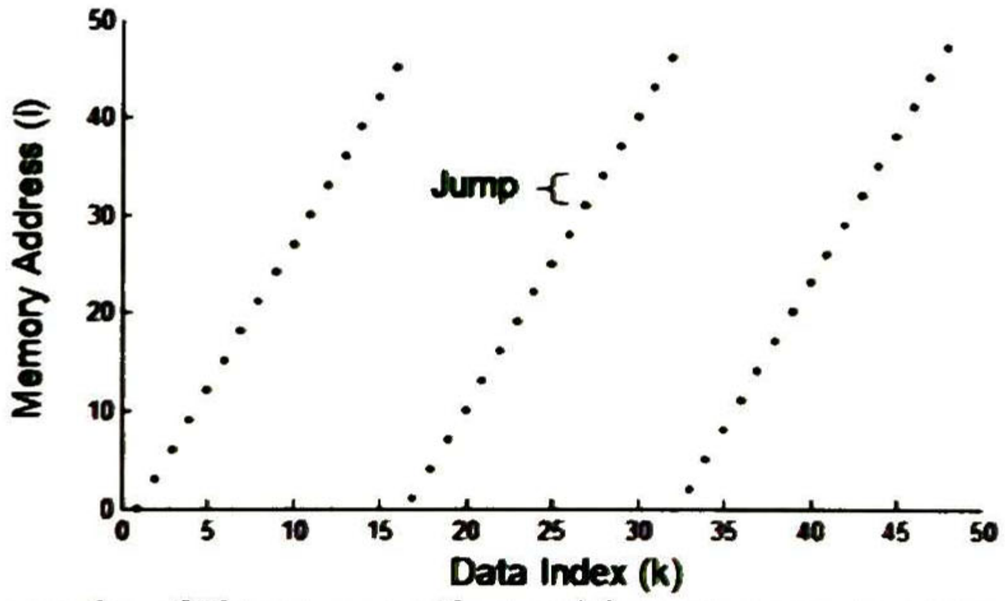


Figure 3-14 Example of the generation addresses sequence of equation (1) for the 802.11a BPSK configuration.

For the equation (2) we can see that the generated sequence, behaves like a counter when s=1, when s=2 the sequence has a periodic perturbation of increments of 1,-1, and when s=3 the sequence has a periodic perturbation of increments of 2,-1,-1, as we can see at Figure 3-16.

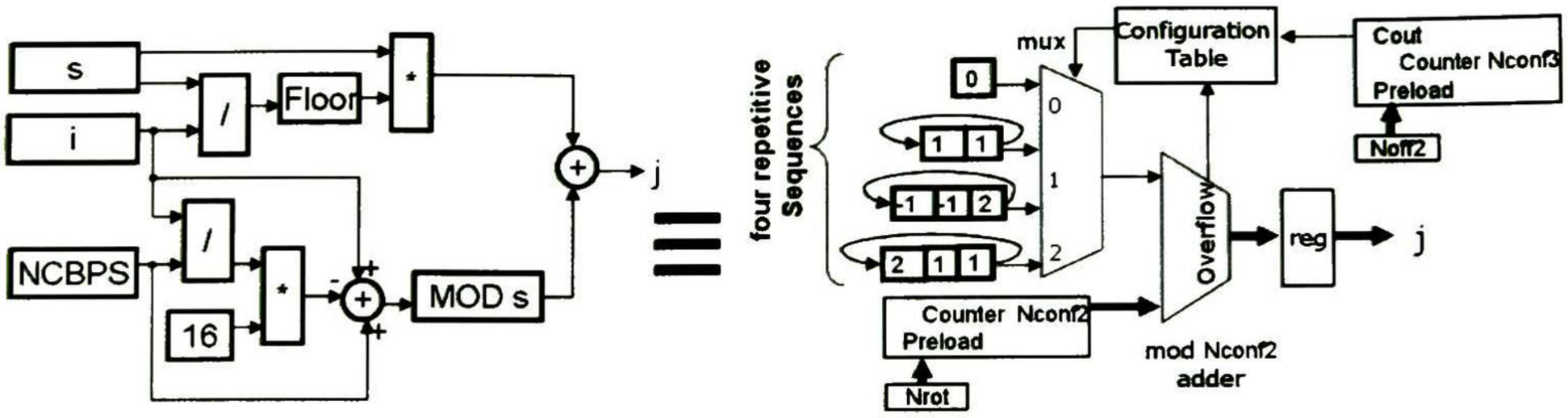


Figure 3-15 Architecture optimization for the equation (2)

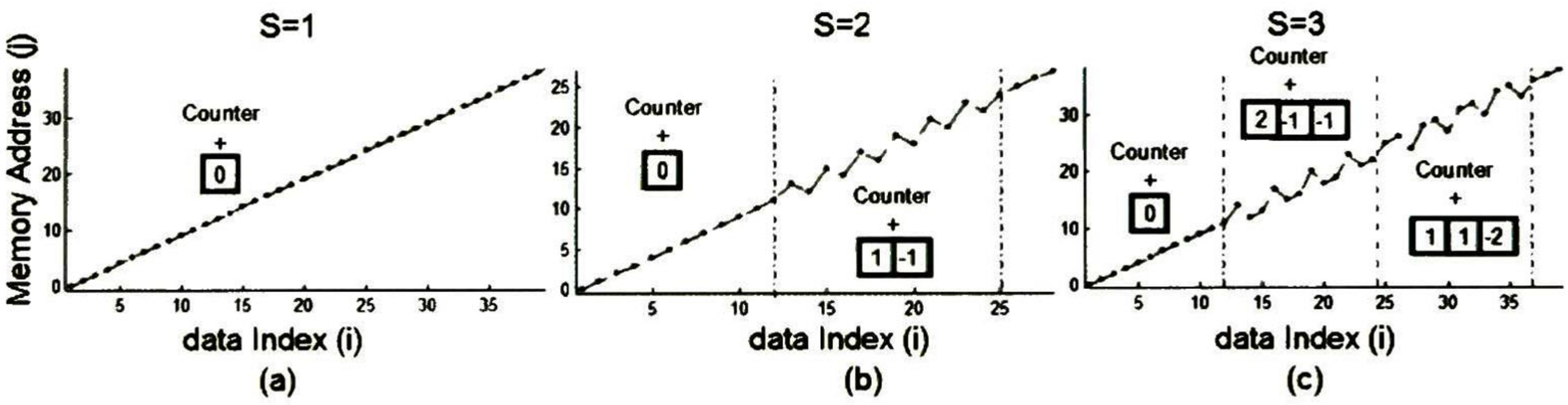


Figure 3-16 Example of the generation addresses sequence of equation (2) for the 16QAM configuration with different values of S.

The arithmetic interleaver writes the input sequence into the memory at the address generated by the arithmetic AGU, and reads the sequence from the memory using a counter. In this implementation we use the AGU from equation (1) to write the data to the memory and a modification of equation (2) to read the data from the memory.

To implement the deinterleavers we only need to switch the reading AGU for the writing AGU.

The rest of the interleaver implementations can be mapped into this architecture, except the symbol interleaver in the DVB standard that needs a Pseudo Random Sequence Generator (PRSG) structure, which is added to AGU0, this architecture is shown at Figure 4-12 Hardware block organization of Configurable P-RSG. This is a configurable feedback PRSG machine so that possible modifications or variations of the symbol interleaver are supported.

3.5.2 STS Structure

The intention of the thesis is to deliver an competitive product, in this case at the market of interleavers, one of the characteristics that are a plus value is the number of bits that can be processed in a clock cycle, this is called parallelism, this implies an increment in the logic necessary to implement the architecture, because the use of parallel AGU. Parallelism has advantages of reduction of clock cycles to process a block, and a reduction of the power consumption, mainly because the interleaver will be active a smallest time and then deactivated for power saving.

We decide to increase the parallelism to 8 and to allow this the STS structure was chosen. The STS structure has his name says it's a spatial – time – spatial structure, in other words, it's a crossbar – memory – crossbar structure, the Figure 3-8 shows an example of this implementation. Every memory of the STS structure needs an AGU to control the memory and the crossbar needs a CGU to control the crossbar, the CGU can be divided has a group of AGU, where every AGU controls one slot of the crossbar, and this AGU are not the same as the optimized arithmetical AGU.

The next step was to map the optimized arithmetical AGU to the STS structure, the idea was that having one optimized arithmetical AGU working well for all the standard, we can copy this AGU 8 times and make the parallelism of 8 bits, but we need to redefine the steps of the AGU instead of advance one address on one cycle it needs to advance 8 address on one cycle, this is done by the next modifications:

AGU1 Modifications:

- Initial value of register changes to the next formula:

$$INITIAL_VALUE_{NEW}(n) = mod(INITIAL\ VALUE_{OLD} + Nconf1*n, Nconf2)$$

Where n is the index of the AGU, from 0 to 7.

-Modified Input Parameter form Jump to Jump*8.

-Special ModNconf2 adder with double Cin (Carry in) and double overflow detection.

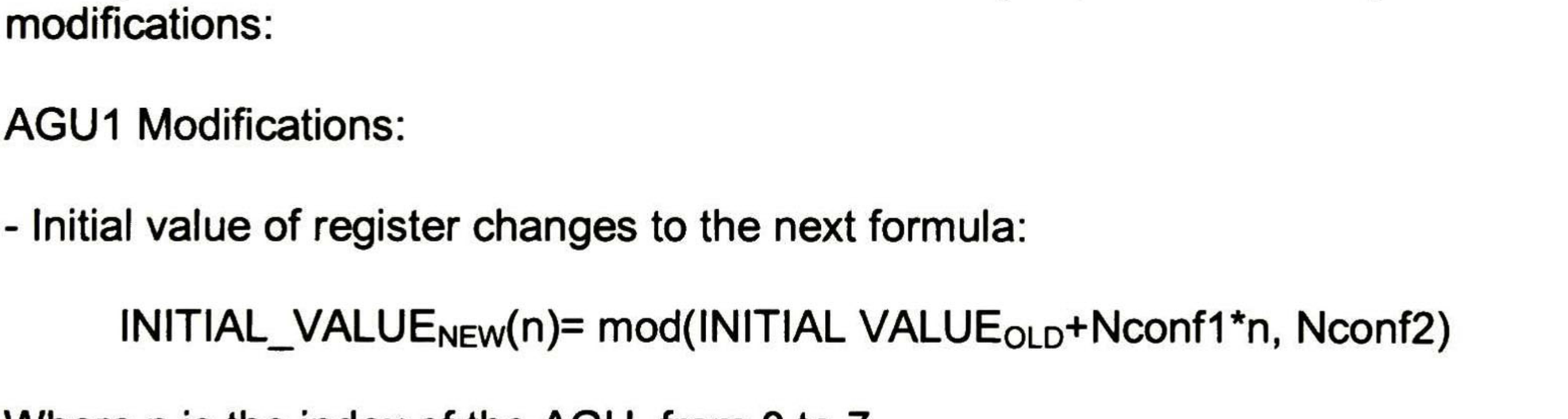


Figure 3-17 AGU1 modifications to make 8 bits of parallelism

AGU2 Modifications:

-Initial Value to Nrot and Noff2.

-Each counters increments 8 times instead of 1 every clock cycle.

-The addition of the Mod_6 block. Controlled by the signal count, that comes from the counter Nconf3.

-Exchange of the registers in the mux for small memory block, addressed by Conf_Mod block.

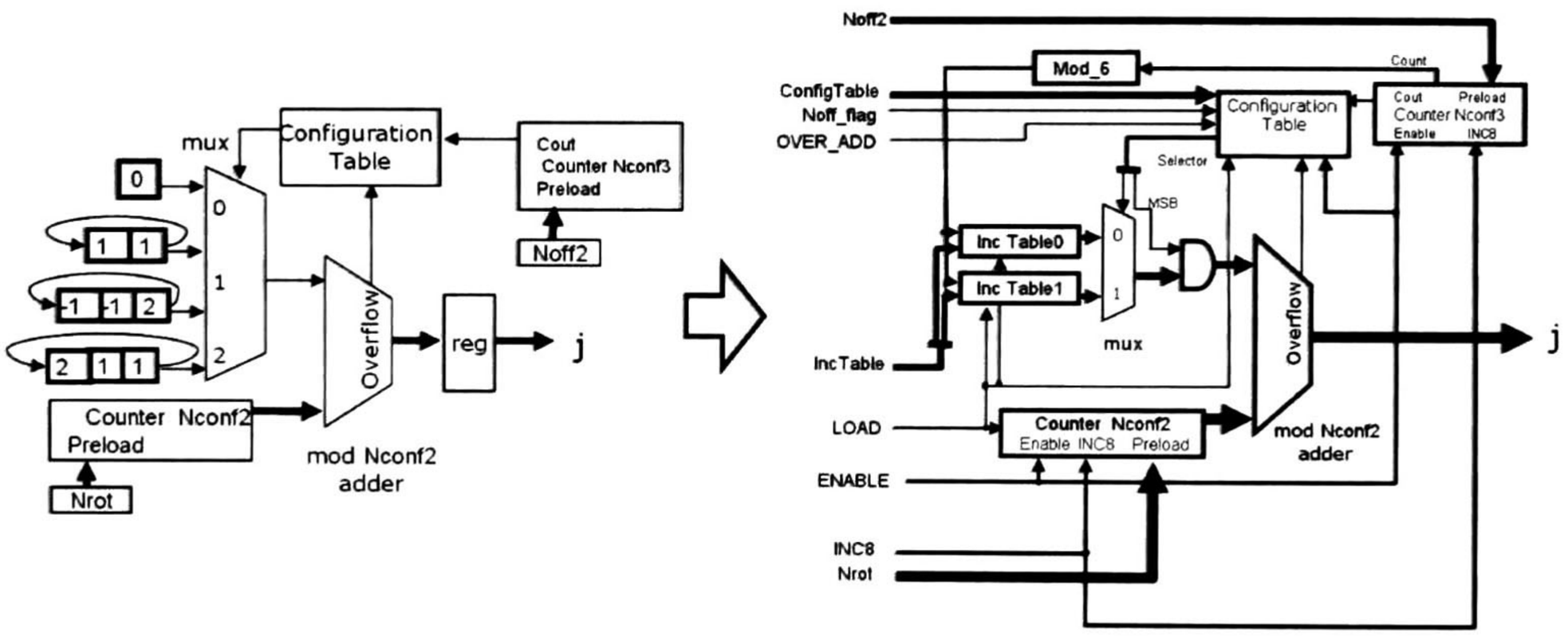


Figure 3-18 AGU2 modifications to make 8 bits of parallelism

From this step we only need to include the STS structure, and reuse the information generated by the parallel 8 bits AGU, to match both structures a new block is needed to generate the necessary information from the parallel 8 bits AGU this block is called ADU (Address Decoding Unit) this unit generates the necessary signals to control STS structure the signals CGU(a) and all the AGU0(a), AGU0(a), ... AGU7(a) are generated by the ADU, the internal structure is shown at Figure 4-27 Diagram block organization of ADU.

The STS structure is designed to manage 8 bits of parallelism. The implementation of 16 memories was chosen to avoid access conflicts to the memories. So only 8 memories are written at a clock cycle at same time, the other 8 memories are disabled.

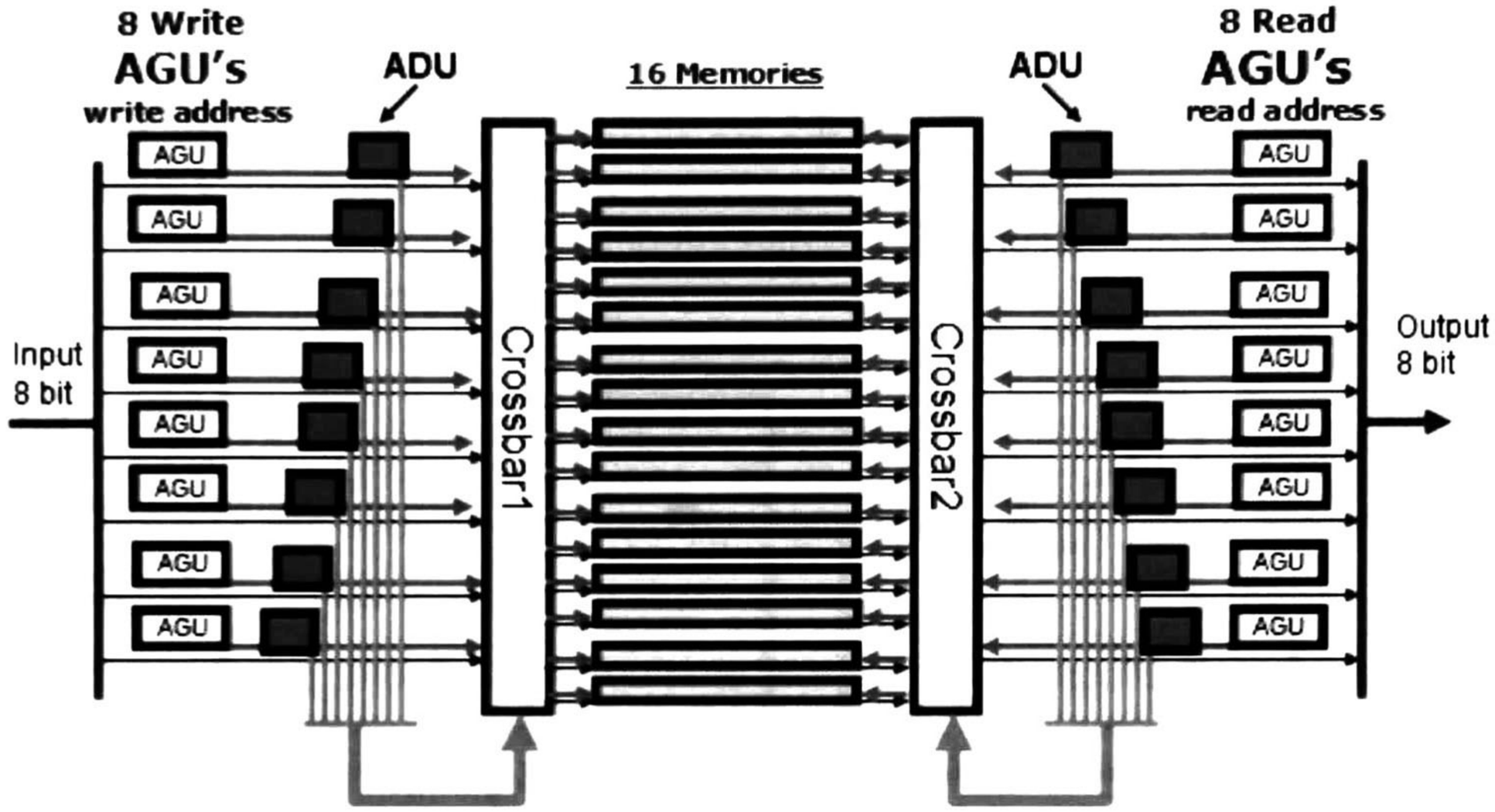


Figure 3-19 architecture used to implement a parallelism of 8 bits.

The interleaver implementation with the AGU, ADU and STS structure is shown at Figure 3-19. This picture shows at the left the "input 8 bit" this is the input to the interleaver the 8AGU from the left are the write AGU this ones generates the write address of the incoming bits, these address are decoded by the ADU.

The ADU generates the control signals for the crossbar and the selected address for the memory to save the incoming bit. After all the bits are written then the 8 AGU from the right are enabled, this AGU generates the read address to the ADU and the ADU generates the read address from the memory and the address for the crossbar, by this way the bits are read from the memory passed trough the crossbar and then ready in the "Output 8 bit"

The ADU generates the address for the memory "M_Address" and the control signals for the crossbar "CB_Ctrl", these are generated from the address from the AGU and a special parameter called PARAM1.

$$M_Address = \text{floor}(ADDRESS/16)$$

$$CB_Ctrl = \text{mod}(ADDRESS + \text{floor}(ADDRESS/PARAM1), 16)$$

PARAM1 was defined by experiment for every configuration to avoid access conflicts to the memories.

3.5.3 Convolutional Interleaver as Block Interleaver

The convolutional interleaver shown at Figure 2-6 can be viewed as a block interleaver and can be mapped by the structure shown in Figure 3-20. With this architecture we can implement the behavior of the convolutional structure as shown at 2.4.1 Convolutional Byte-Wise Interleaving.

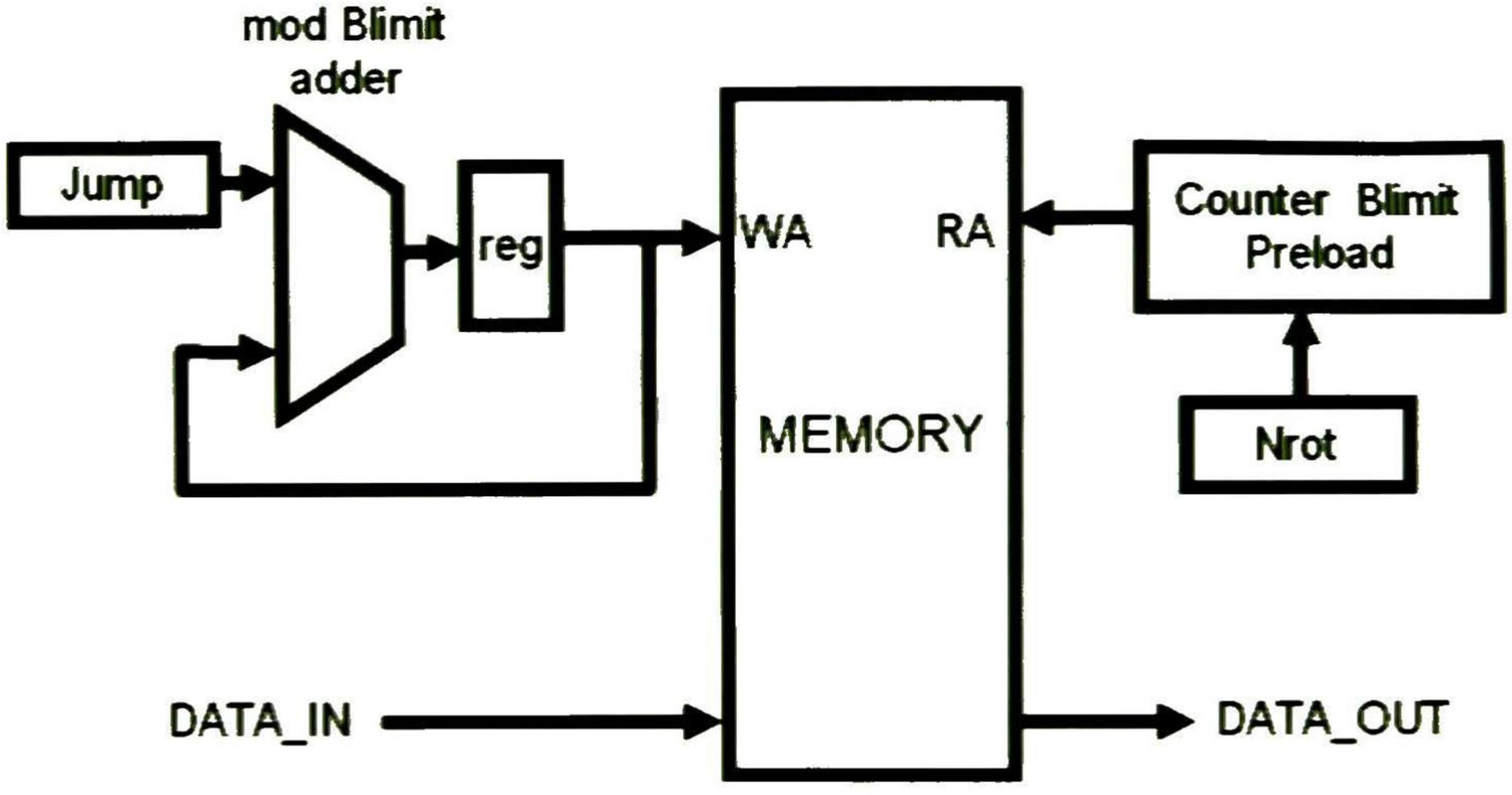


Figure 3-20 Structure proposed for the convolutional interleaver

One of the characteristic of the convolutional interleaver is that for every symbol that is written one symbol is read from the interleaver. This proposal of transformation for convolutional interleaver to block interleaver can perform this behavior and also allow writing "Number of rows" symbols before reading equal number of symbols.

The parameters Blimit and Jump are taken from the next equations:

$$\text{Jump} = \text{Number of rows} * M = 12*17+1 = 205$$

$$\text{Blimit} = \text{Number of rows} * (M * (\text{Number of columns}+1) = 12*17*(11+1) = 2448$$

This implementation of the convolutional interleaver can be deduced if we look at the behavior of the data that the original structure of the convolutional interleaver implies this denotes a periodic behavior just like the other interleaver schemes.

4 Interleaver Proposal Technical Explanation

This section describes the functional hardware blocks of the MSI in terms of functionality

4.1 Hardware Overview

The MSI is a Multi-Standard Interleaver that can perform the interleaver/deinterleaver operations of the following standards 802.11a, 802.11n, 802.16e & DVB, and support modifications to the interleaver schemes because of its re-configurability.

The Figure 4-1 shows the input/outputs of the Multi-Standard Interleaver (MSI). A brief explanation about his functional work is found in section 4.1.1 configuration process and 4.1.2 Interleaving Process.

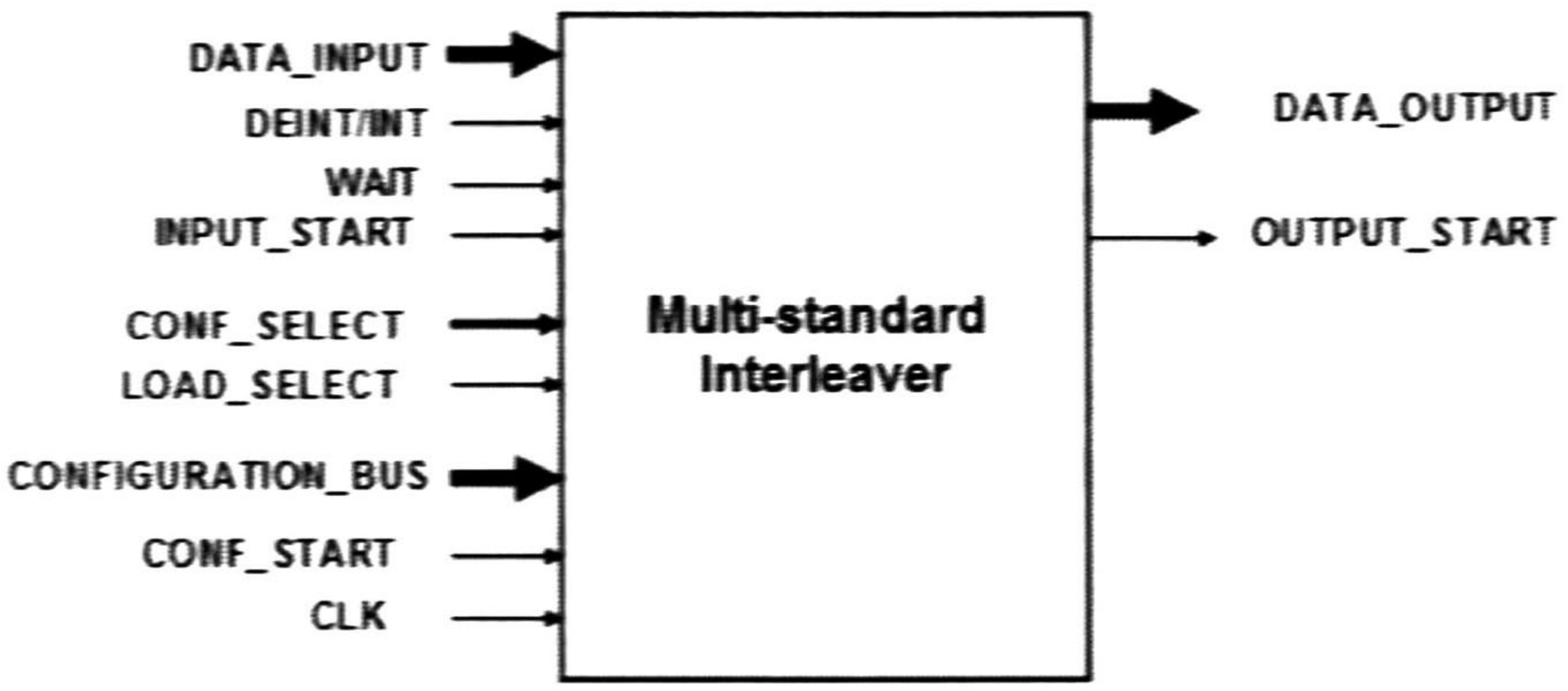


Figure 4-1 Input/Output signal diagram for Multi-Standard Interleaver (MSI)

Input	Description	Comes From
DATA_INPUT	data to be interleaved	Input Interface
DEINT/INT	Deinterleaver Interleaver selector When 1 = Deinterleaver When 0 = Interleaver	Input Interface
WAIT	Wait signal	Input Interface
INPUT_START	Indicates the start of send the input data to be interleaved	Input Interface
CONF_SELECT	Selection of the interleaver configuration on the configured on the fly	Input Interface
LOAD_SELECT	Load the selected parameters on the fly by	Input Interface

	CONF_SELECT	
CONFIGURATION_BUS	Configuration parameters to be loaded	uP Interface
CONF_START	Start of the Configuration mode	uP Interface
Output	**Description**	**Goes To**
DATA_OUTPUT	Data interleaved	Output Interface
OUTPUT_START	Indicates the start of send of output data	Output Interface

Table 4-1 Input and output organization for MSI

The Figure 4-2 shows the organization of the hardware blocks of the Multi-Standard Interleaver (MSI). Four principal blocks are shown:

- Memory Structure: This block has the memory arrays used to save all the data to interleave and allow a parallelism of 8 bits.

- 8AGU1 & 8AGU2: Each one of these blocks has inside eight AGU used to generate the read and write address of the memory structure.

- Configuration Control: This block has the control of the configuration and interleaving process, and generates all the control signals to manage the memory structure, and AGU configurations, also have the memories to save the parameters of the four configurations on the fly.

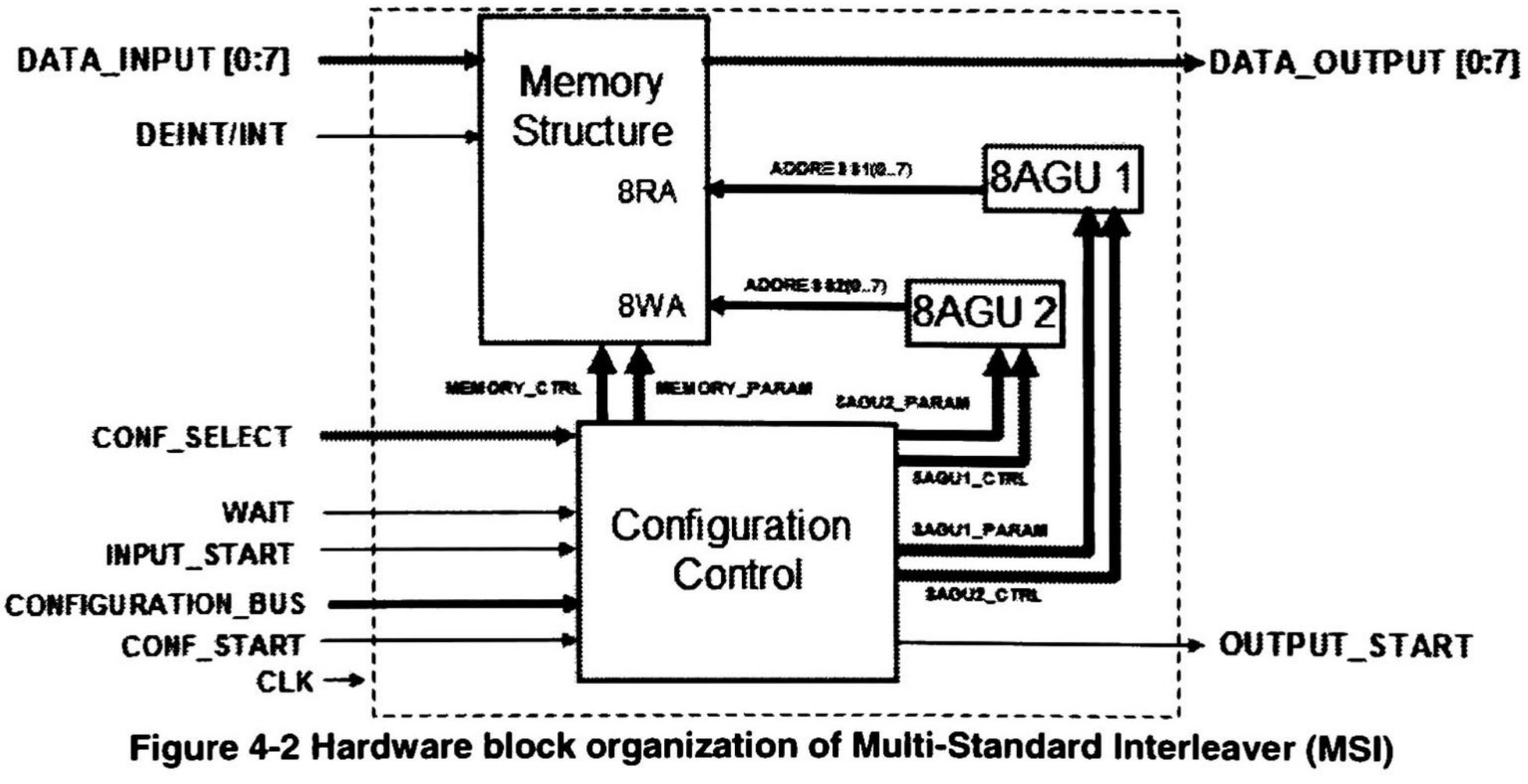


Figure 4-2 Hardware block organization of Multi-Standard Interleaver (MSI)

4.1.1 Configuration Process

In the configuration process of the MSI the user introduces the parameters necessary to implement the desired interleaving operation. We can have four different interleaver configured.

Looking at Figure 4-2 we see the CONF_START signal, the activation of this signal tells to the configuration control block that the configuration parameters are incoming in the CONFIGURATION_BUS.

Then the parameters of the 4 different interleaver configurations start to enter by a serial way to the configuration control block. The order of the sequence of parameters to be sent on the Configuration Bus can be followed at section 4.5 Configuration Modes .

When all the parameters are configured the MSI is enabled to implement any of the four interleavers configured.

4.1.2 Interleaving Process

This process starts when the LOAD_SELECT signal is asserted, at Figure 4-2, this load the configuration parameters on the fly of the selected configuration by CONF_SELECT.

At this moment the configuration control load the selected configuration registers to configure the memory structure, 8AGU1 and 8AGU2.

The next step is when the INPUT_START signal is asserted, this indicate that the DATA_INPUT[0:7] bus has the bits to be interleaved that are written to the address generated in the 8AGU1 or 8AGU2 structure has was selected for interleaving or de-interleaving process by the DEINT/INT signal. When the whole package to interleave is read, the configuration control asserts the OUTPUT_START signal.

At this moment the configuration control switch memory structure writing process and start the reading process with the address generated from 8AGU1 or 8AGU2, has was configured before. The AGU start the address generation to read from the Memory Structure and the interleaved bits appear in the DATA_OUTPUT[0:7], when the OUTPUT_START signal in no more asserted we have read the entire interleaved block.

4.2 Functional Description

4.2.1 8AGU 1

The 8AGU1 is an array of 8 AGU1 blocks from AGU1(0) to AGU1(7), this generate the address sequence for the different interleaving configurations.

The AGU1 has the necessary logic to implement a big variety of interleaving process, not only the ones in the standards here cited, and depending on the values of the configuration registers the write AGU have a different work. The Figure 4-3 shows the input/outputs signals of the AGU1, and the Table 4-2 shows its description

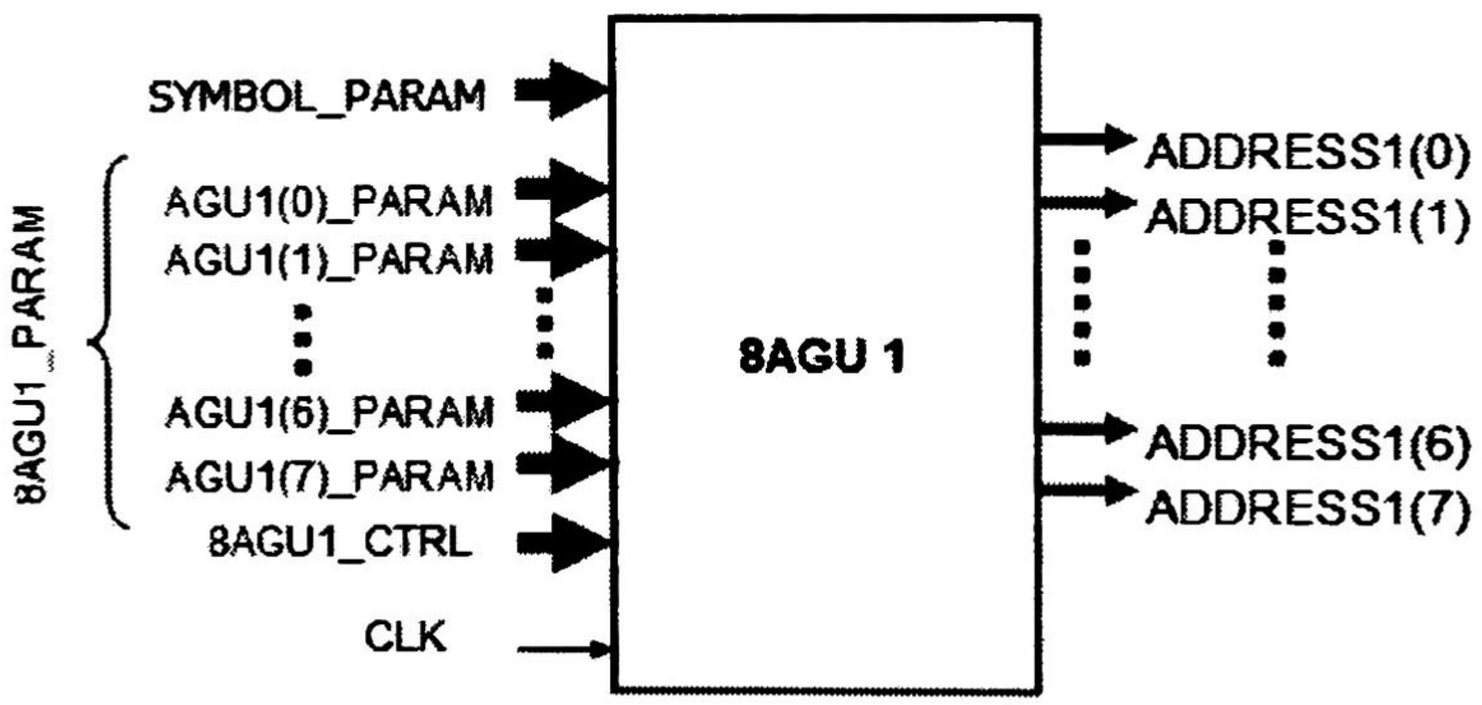


Figure 4-3 Input/Output signal diagram for 8AGU1

Input	Description	Comes From
SYMBOL_PARAM	Has the configuration parameters of the symbol interleaver	Configuration Control
AGU1(0)_PARAM	Has the parameters to configure the AGU1(0)	Configuration Control
AGU1(1..7)_PARAM	Has the parameters to configure the AGU1(1..7)	Configuration Control
8AGU1_CTRL	Has the control signals to manage the all the 8AGU1	Configuration Control
Output	**Description**	**Goes To**
ADDRESS1(0) [12:0]	Address bus of 13 bits width, for writing or reading on the memory structure, used for all the interleaver configurations	Memory Structure

ADDRESS1(1..7) [9:0]	Address bus of 10 bits width, for writing or reading on the Memory Structure, used for 802 & Compact Bit interleaving	Memory Structure

Table 4-2 Input and output organization for 8AGU1

The Figure 4-4 shows the organization of the hardware blocks of the 8AGU1, where we can see the 8 AGU1 blocks, from AGU1(0) to AGU1(7).

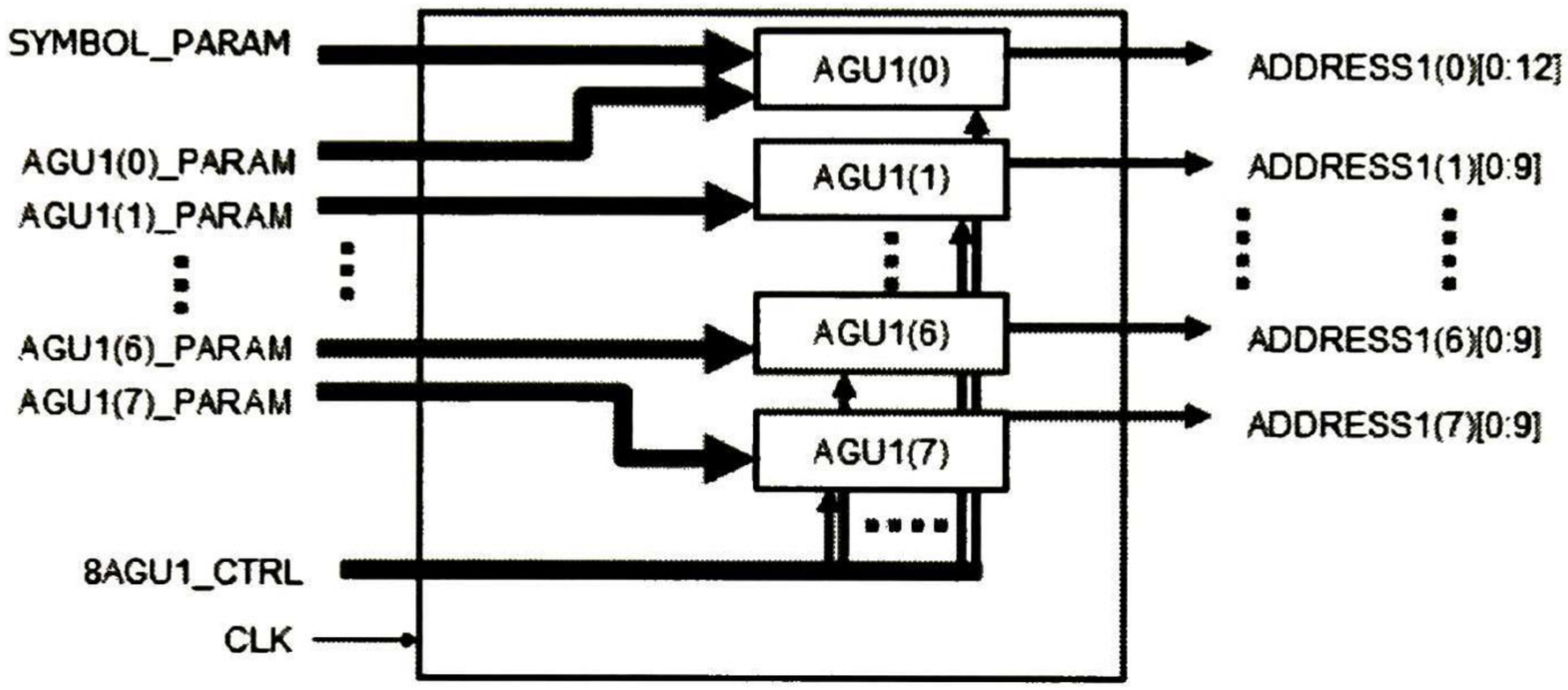


Figure 4-4 Hardware block organization of 8AGU1

4.2.1.1 AGU1 (0)

The AGU1(0) is different to the other AGU1(1..7), because AGU1(0) has an extra logic to generate the Symbol Interleaver sequence and the Convolutional Interleaving and has an address bus of 13 bits while the AGU1(1..7) has a address bus of 10 bits.

The Figure 4-5 shows the input/outputs signals of the AGU1, and the Table 4-3 shows its description

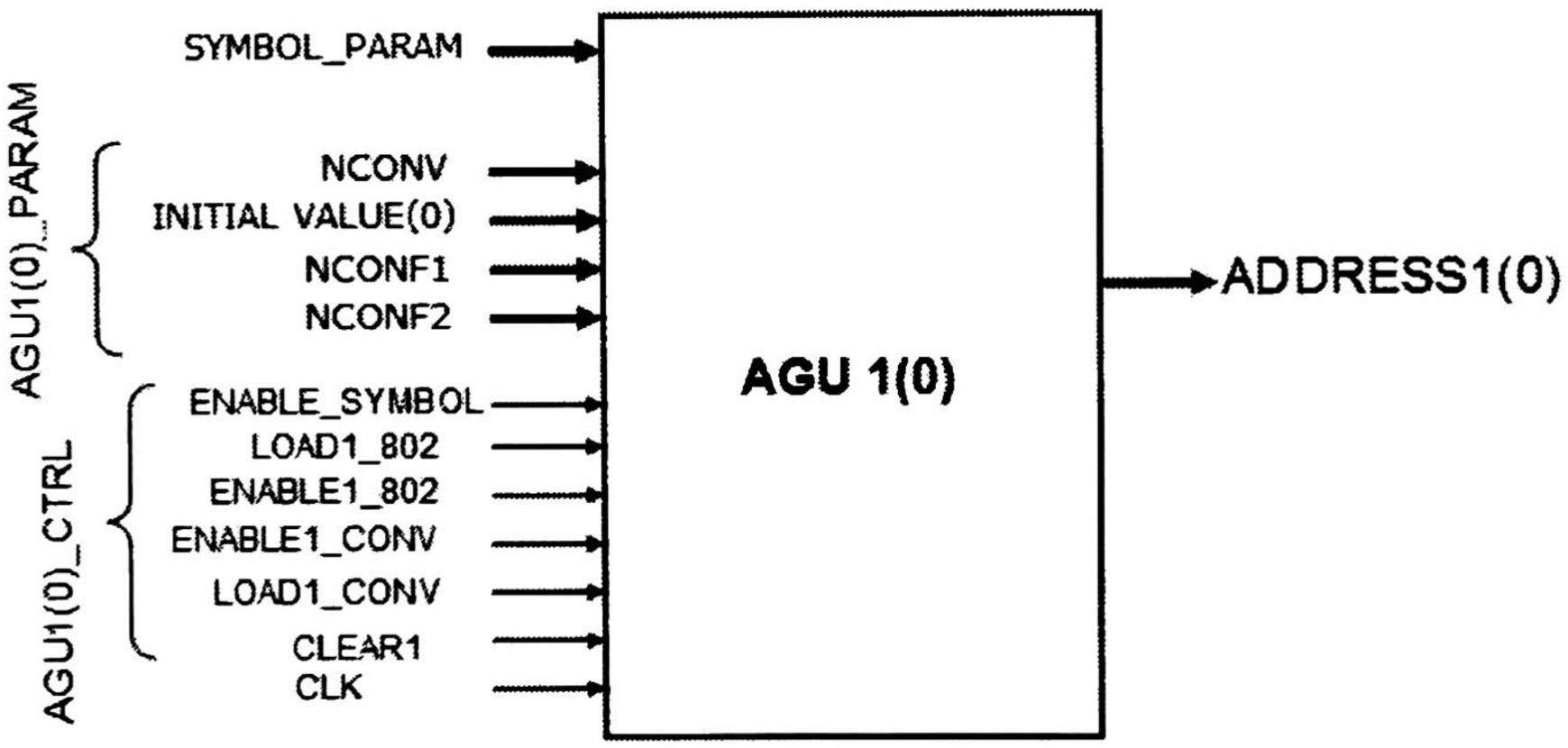


Figure 4-5 Input/Output signal diagram for AGU1(0)

Input	Description	Comes From
SYMBOL_PARAM	Parameters to configure the Symbol Interleaver	Configuration Control
NCONV	Enable of extra increment on the overflow	Configuration Control
INITIAL VALUE	Initial value of the AGU	Configuration Control
NCONF1 [9:0]	Parameter of the size of increment	Configuration Control
NCONF2 [12:0]		
ENABLE_SYMBOL	Enable of the Symbol Interleaver block	Configuration Control
LOAD1_802	Load of 802 Register	Configuration Control
ENABLE1_802	Enable of the 802 Register	Configuration Control
ENABLE_CONV	Enable of the Conv Register	
LOAD1_CONV	Load of Conv Register	Configuration Control
CLEAR1		
Output	Description	Goes To
ADDRESS1(0) [12:0]	Address bus of 13 bits width, for writing or	Memory Structure

	reading on the Memory Structure, used for all the interleaver configurations	

Table 4-3 Input and output organization for AGU1(0)

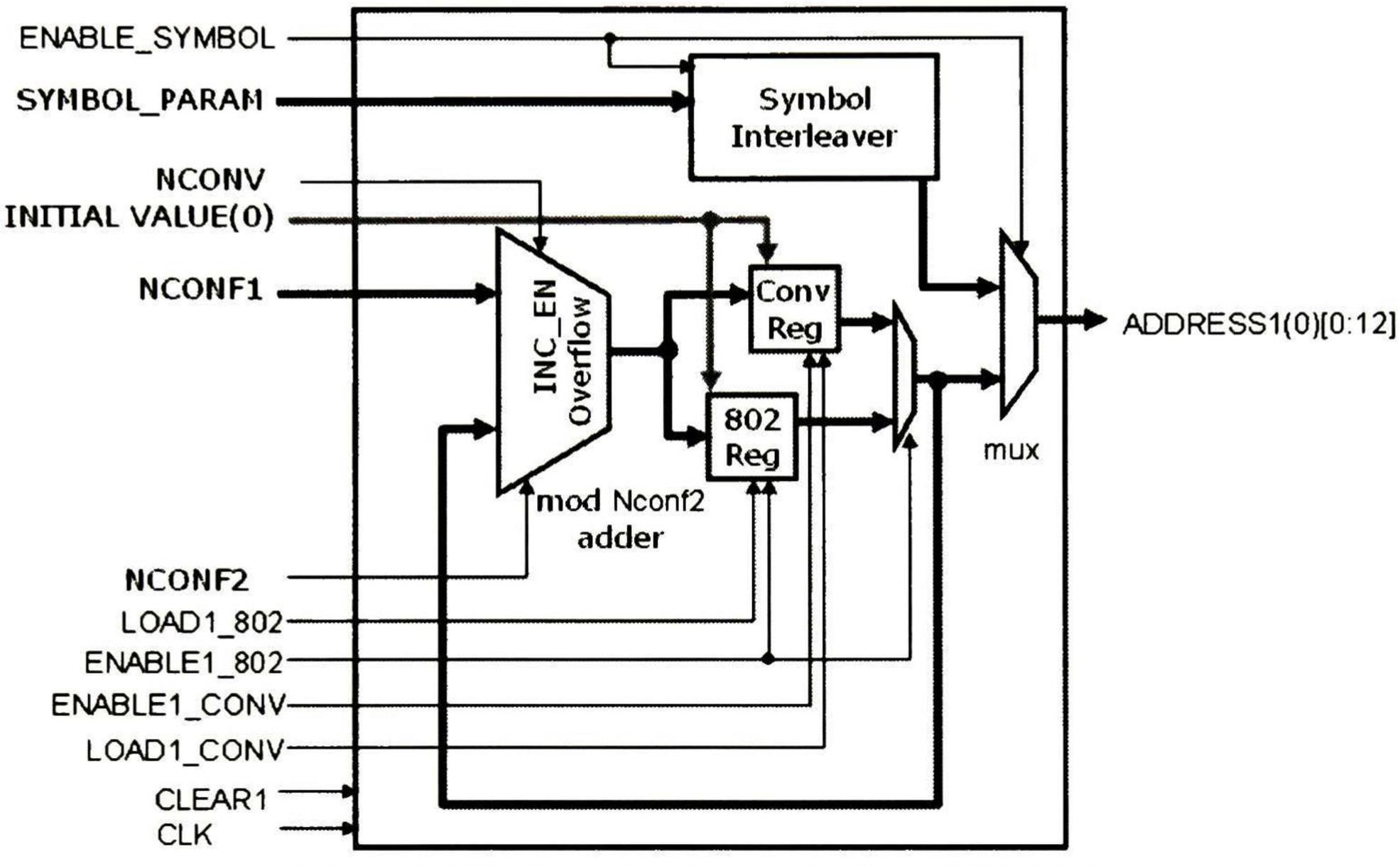


Figure 4-6 Hardware block organization of AGU1(0)

To start an interleaving process the configuration control block starts by placing the selected parameters in the P_RSG_CONF bus and the AGU1(0)_PARAM bus, this parameters are NCONV, INITIAL VALUE, NCONF2, NCONF1, then the appropriated ENABLE and LOAD signal in the AGU1(0)_CTRL bus are asserted, at this point the AGU is configured for a selected interleaver.

For the implementation of the convolutional interleaver, the AGU1(0) has an special register to store the information of the last value interleaved in this configuration, this is because the convolutional interleaver is a continuous interleaver and not a block Interleaver.

For the convolutional interleaver, the initial value is loaded by asserting the LOAD_CONV signal. This is done at the configuration time.

For symbol interleaver the Symbol_Param bus has the parameters to configure the symbol interleaver block.

4.2.1.1.1 Mod Nconf2 Adder

This block is a modified adder, which has a configurable modulus operator, and has an increment of one whenever the overflow signal is asserted and the INC_EN is enabled.

This block is used in all the AGU's. The Figure 4-7 shows the input/outputs signals of the AGU1, and the Table 4-4 shows its description

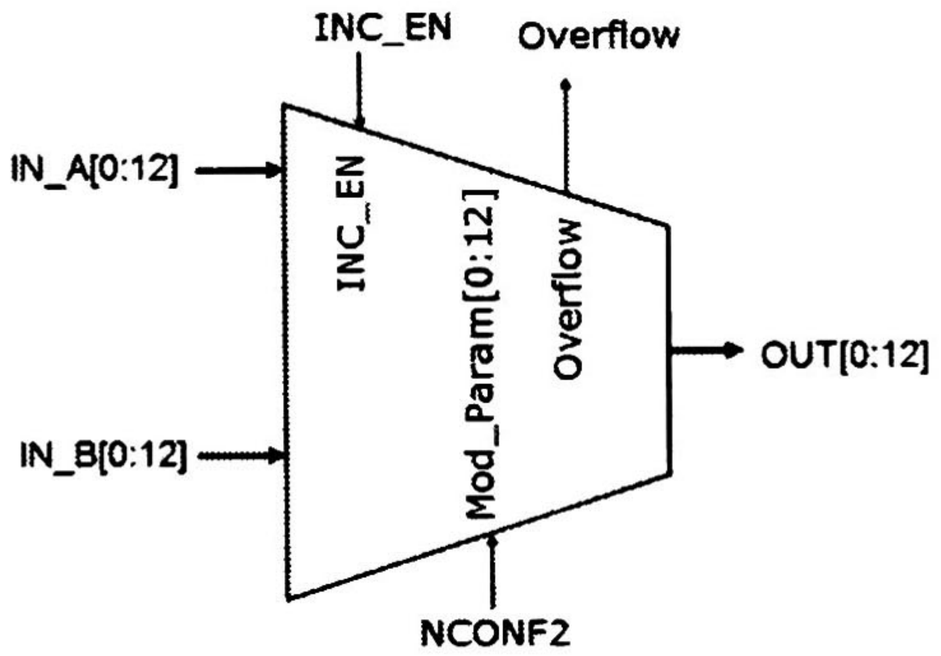


Figure 4-7 Input/output signal diagram for Mod Nconf2 Adder

This block is pure combinational and the configuration is done by simply placing the configuration signals on the inputs.

Input	Description	Comes From
INC_EN	Enable an extra increment when overflow is activated	Configuration Control
NCONF2	Modulus Parameter	Configuration Control
IN_A[0:12]	Input A to be added	Configuration Control
IN_B[0:12]	Input B to be added	Feedback from MUX
Output	**Description**	**Goes To**
Overflow	Overflow flag, whenever the sum exceeds the NCONF2 parameter	
OUT[0:12]	Data sum result	Conv Reg & 802 Reg

Table 4-4 Input and output organization of Mod Nconf2 Adder

The Figure 4-8 shows the organization of the hardware blocks of the Mod Nconf2 Adder, where we can see the special increment that is activated when an overflow occurs and the INC_EN flag is asserted.

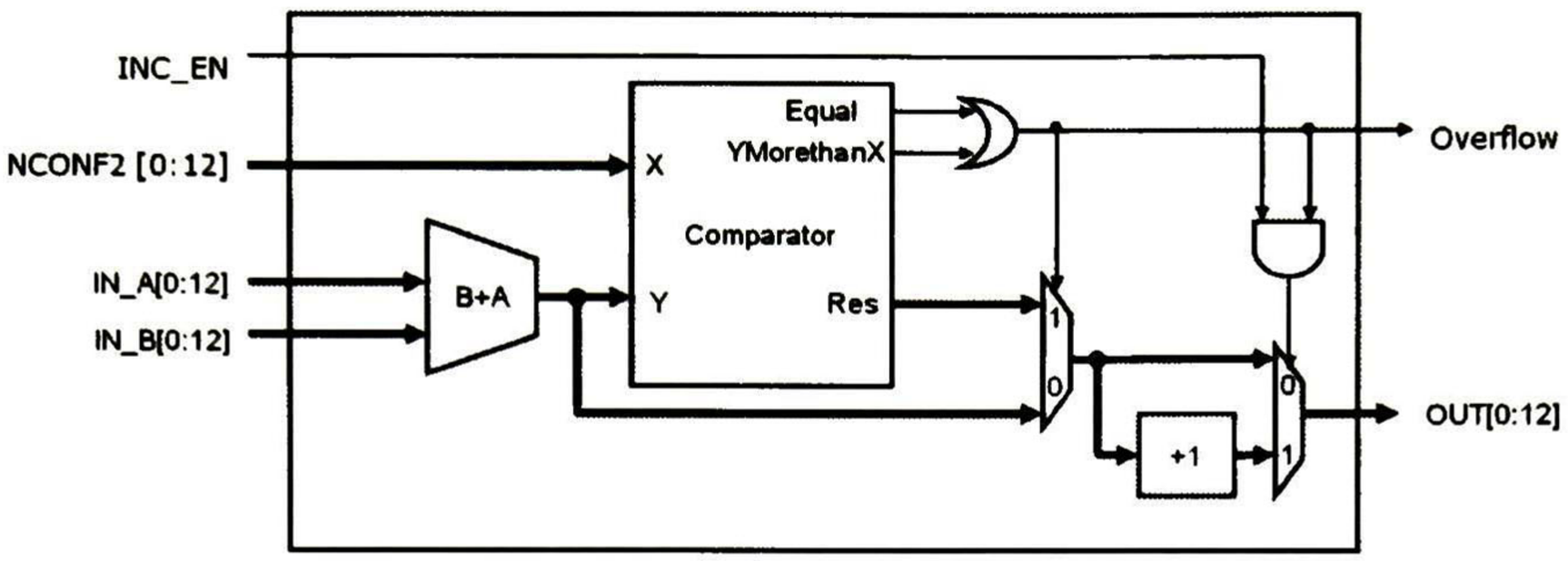


Figure 4-8 Hardware block organization of Mod Nconf2 Adder

4.2.1.1.2 Symbol Interleaver

The symbol interleaver block generates the address sequence for the configuration in DVB symbol interleaver. The Figure 4-9 shows the input/outputs signals of the symbol interleaver, and the Table 4-5 shows its description

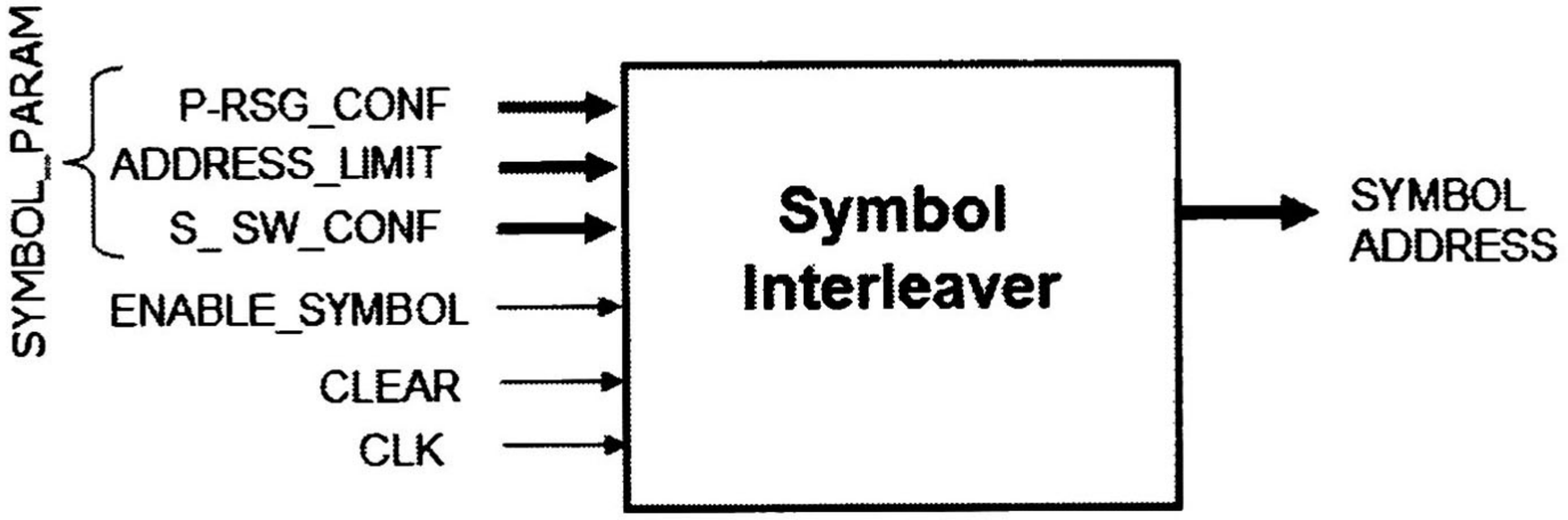


Figure 4-9 Input/output signal diagram for Symbol Interleaver

The SYMBOL_PARAM bus has the following signals on it:

- P-RSG_CONF: Has the configuration for the P-RSG block.
- S_SW_CONF: Configuration for the Configurable Spatial Switches.
- ADDRESS_LIMIT: Maximum address for the selected configuration.

This block has a PRSG that is the base for this address generation scheme, and is defined by the DVB standard in the Symbol Interleaver.

Input	Description	Comes From
P-RSG_CONF	Interconnection word of the Configurable P-RSG is of 12 bits	Configuration Control
S_SW_CONF	Configuration of the Spatial Switches	Configuration Control
ADDRESS_LIMIT	Address limit to the comparator of the Address Check block	Configuration Control
ENABLE_SYMBOL	Enable signal to the block	Configuration Control
CLEAR	Initialization signal of the block	Configuration Control
Output	**Description**	**Goes To**
SYMBOL ADDRESS	Address for write or read in the Memory Structure in the Symbol Interleaver configuration	Memory Structure

Table 4-5 Input and output organization of Symbol Interleaver

The Figure 4-10 shows the organization of the hardware blocks of the symbol interleaver.

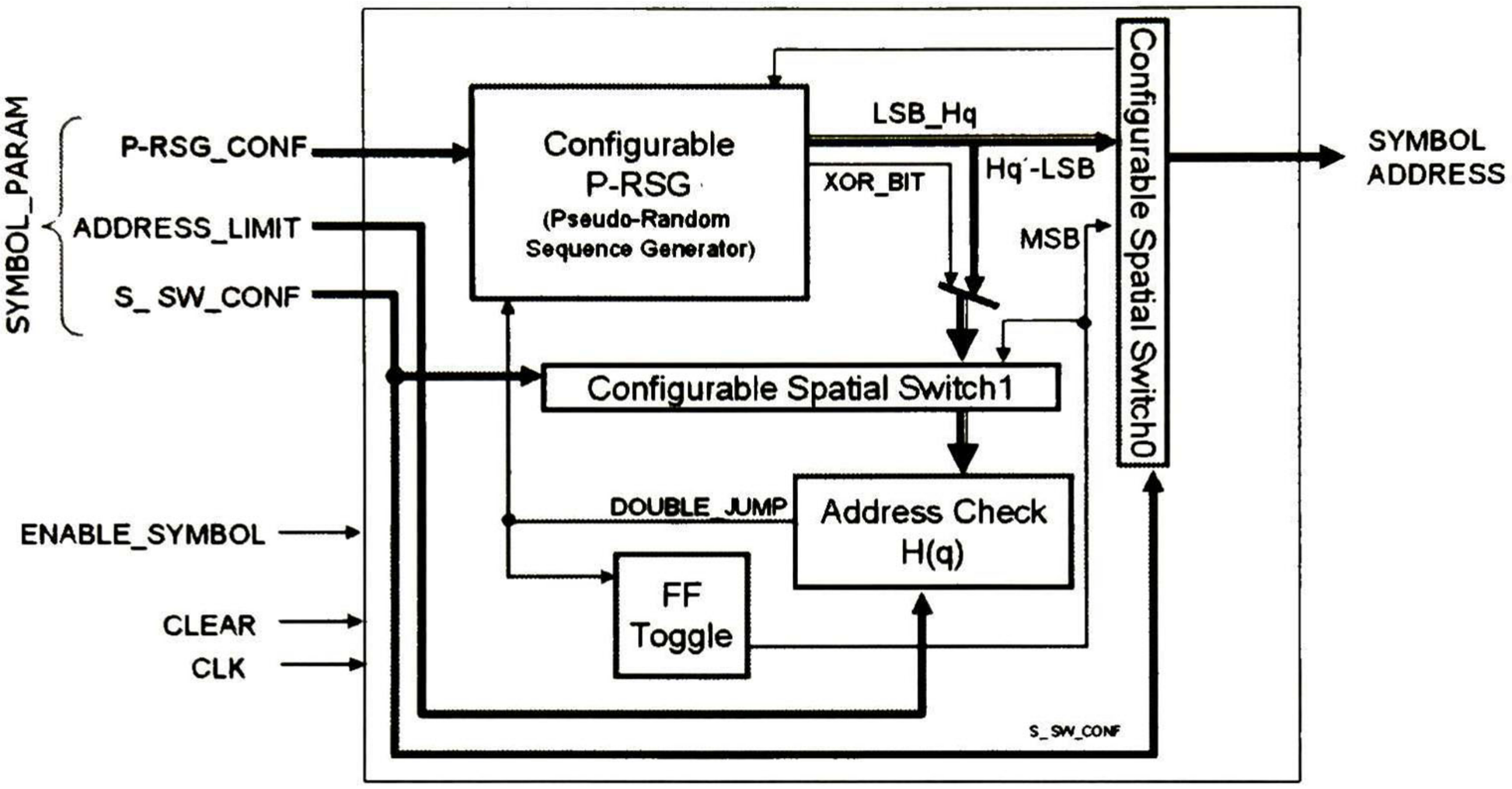


Figure 4-10 Hardware block organization of Symbol Interleaver

From the reconfigurable P-RSG block we can see the actual address Hq(n) and the next address Hq(n+1), the next address is generated by the Hq bits minus the LSB and using the XOR_BIT signal has the MSB of Hq(n+1), the rearrange is made by the configurable spatial switch1.

The address check block is a comparator which compares the next address and verifies if it's valid. An address is valid if it's less or equal than the ADDRESS_LIMIT, if the address exceeds the ADDRESS_LIMIT the address check block activate the double jump signal and this tells to the configurable P-RSG block to jump the incorrect address and also disables the FFToggle.

4.2.1.1.2.1 Configurable PRSG

The PRSG (Pseudo Random Sequence Generator) has an interconnection matrix that defines the interconnections of the shift register with the XOR gates that make the retro to the shift register.

This block allows the 2 configurations indicated in the DVB standard for symbol interleaver and also allows the implementation of any combination of interconnections in this kind of interleaving.

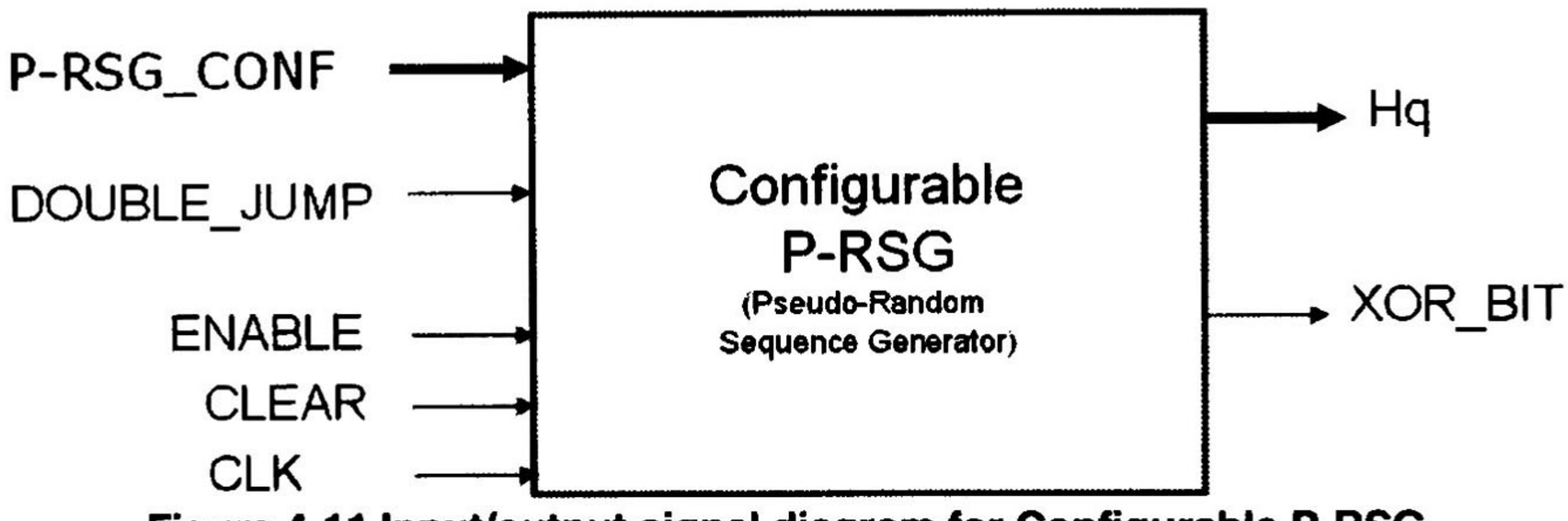


Figure 4-11 Input/output signal diagram for Configurable P-RSG

The Figure 4-11 shows the input/outputs signals of the configurable PRSG, and the Table 4-6 shows its description.

Input	Description	Comes From
P-RSG_CONF	Interconnection word of the Configurable P-RSG is of 12 bits	Configuration Control
DOUBLE_JUMP	Indicates a double jump of from the selected address, to avoid the dead time on the Address Generation	Address Check
ENABLE	Enable signal of the block	Configuration Control
CLEAR	Initialization signal of the block	Configuration Control
Output	**Description**	**Goes To**
Hq	Symbol Address bits	Configurable Spatial Switch0
XOR_BIT	Signal necessary to implement the Look ahead logic	Configurable Spatial Switch1

Table 4-6 Input and output organization of Configurable P-RSG

The Figure 4-12 shows the organization of the hardware blocks of the configurable PRSG.

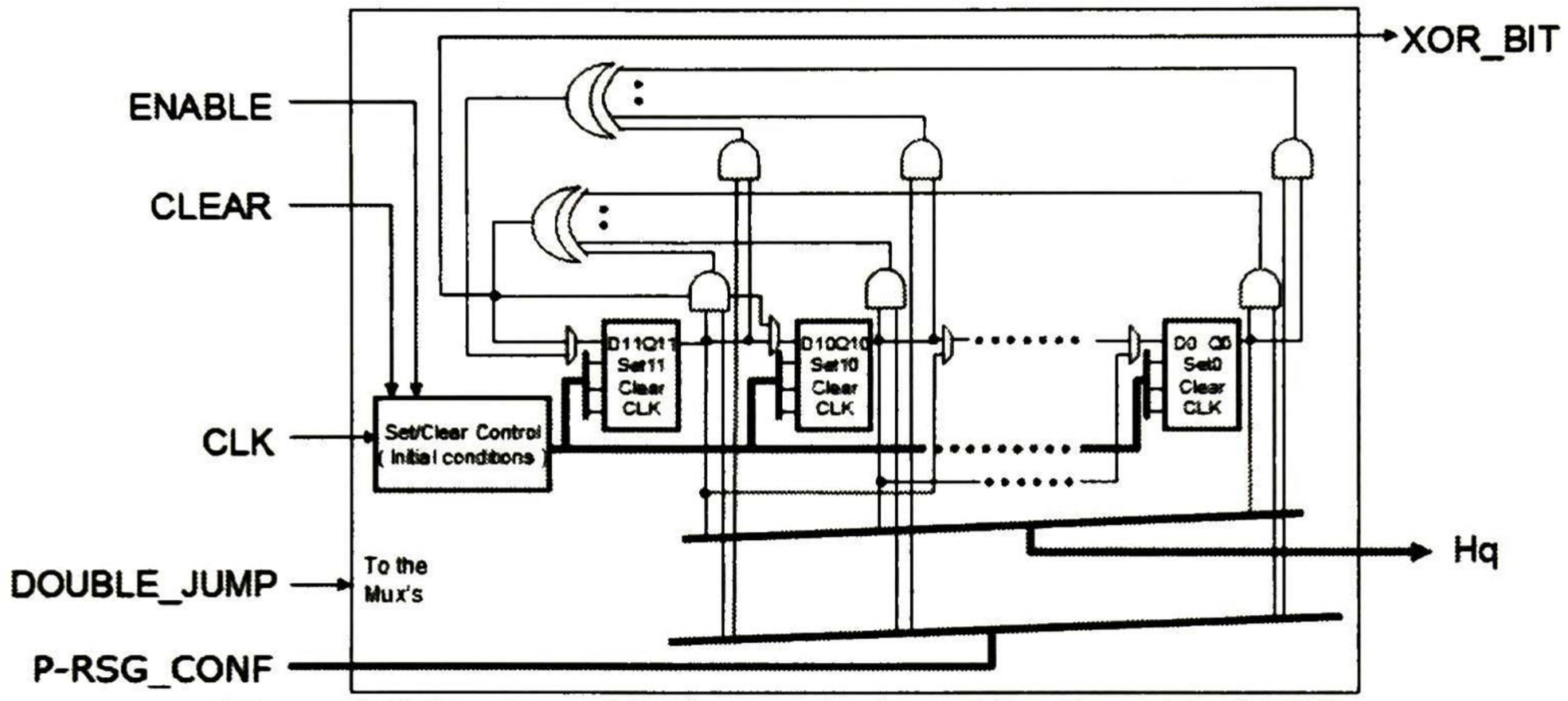


Figure 4-12 Hardware block organization of Configurable P-RSG

A PRSG is a mathematical function that generates all the possible combinations in the in n bits, if the shift register is used to implement 10 bits then the PRSG will generate all the possible combinations of these bits without repeating any of them until reaches the period of 2^n-1.

The DOUBLE_JUMP signal is the control of all the mux in this block.

The CLEAR signal indicates to the set/clear control to operate to configure the flip flop on the initial condition, and the ENABLE signal is the habilitation for the flip flop.

4.2.1.2 AGU1 (1..7)

The AGU1 (1..7) are the simplified form of the AGU1(0), this block are enabled for the 802 interleaving, and compact bit interleaving schemes.

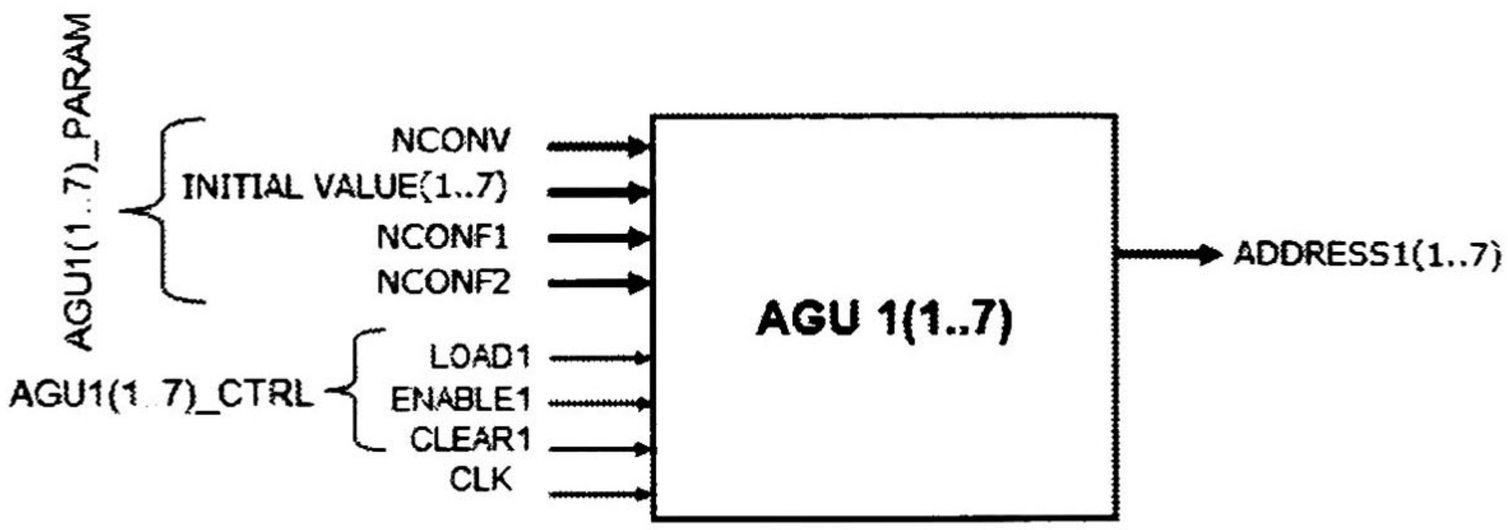


Figure 4-13 Input/output signal diagram for AGU1(1..7)

The Figure 4-13 shows the input/outputs signals of the AGU1(1..7), and the Table 4-7 shows its description

Input	Description	Comes From
NCONV	Enable of extra increment on the overflow	Configuration Control
INITIAL VALUE(1..7)	Initial value of the Adder	Configuration Control
NCONF1	Parameter of the size of increment	Configuration Control
NCONF2	Modulus of the Adder	Configuration Control
LOAD	Load of the Register	Configuration Control
ENABLE	Enable of the Register	Configuration Control
CLEAR	Clear of the Register	Configuration Control
Output	**Description**	**Goes To**
ADDRESS1(1..7)	Address bus of 10 bits width, for writing or reading on the Memory Structure, used for 802 & Compact Bit interleaving	Memory Structure

Table 4-7 Input and output organization of AGU1(1..7)

The Figure 4-14 shows the organization of the hardware blocks of the AGU1(1..7).

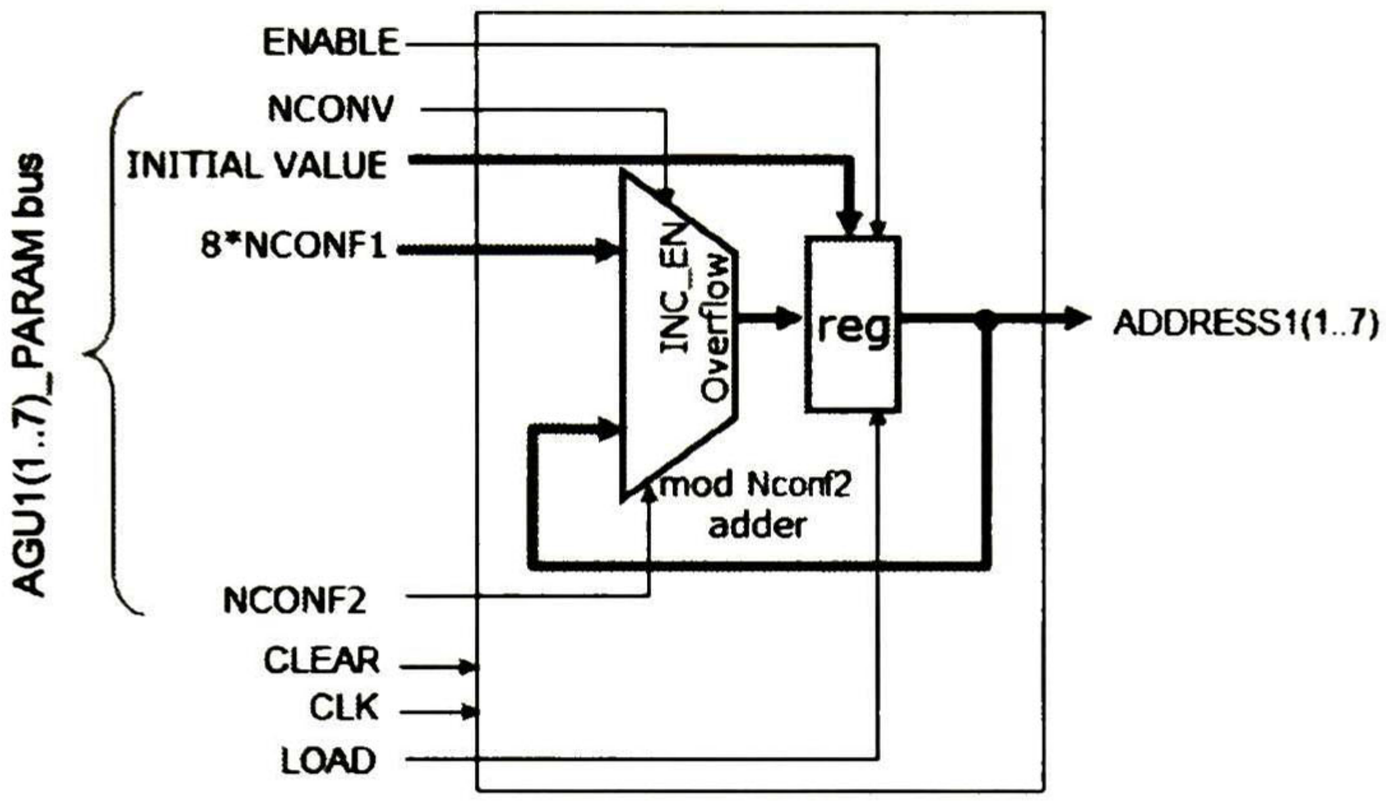


Figure 4-14 Hardware block organization of AGU1(1..7)

The configuration for this block is given by the AGU1(1..7)PARAM bus, and this parameters are INITIAL VALUE, NCONV,CONF2 and 8*NCONF1, that are given by the configuration control block.

Only the INITIAL VALUE signal has a different value for each AGU1(0..7).

4.2.2 8AGU2

The 8AGU2 is an array of 8 AGU2 blocks from AGU2(0) to AGU2(7), this ones generate the address sequence for the different interleaving configurations.

The AGU2 has the necessary logic to implement a big variety of interleaving process, not only the ones in the standards here cited, and depending on the values of the configuration registers the write AGU have a different work.

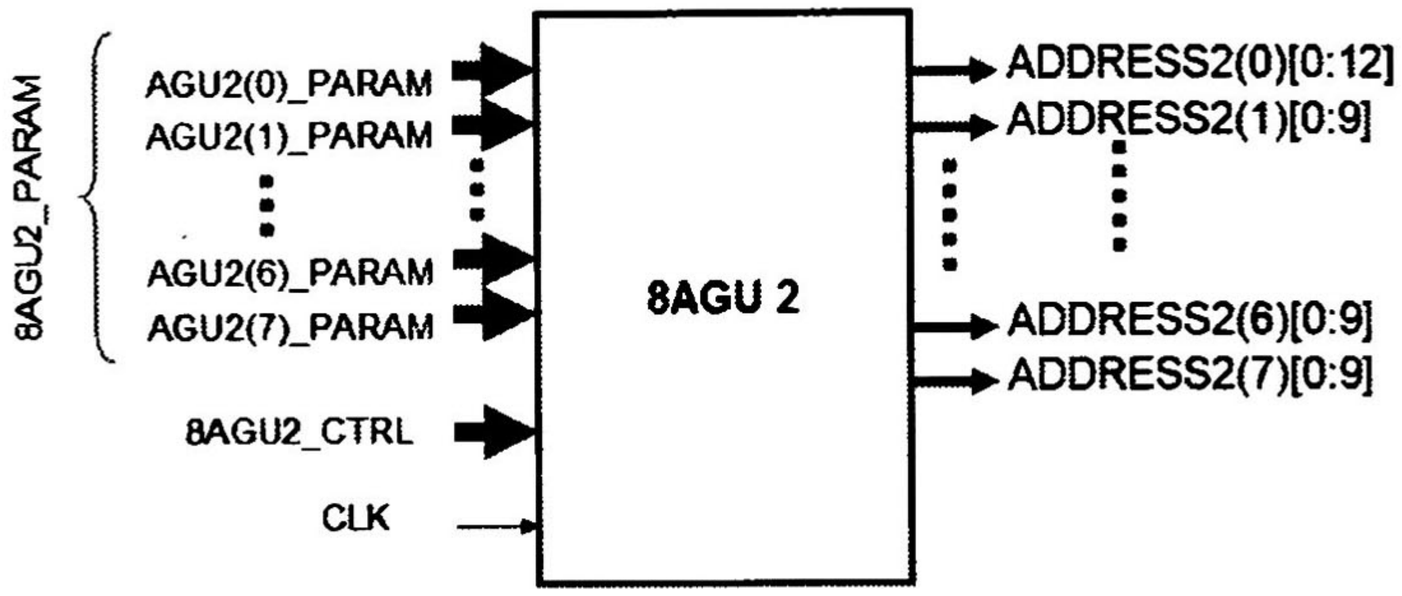


Figure 4-15 Input/output signal diagram for 8AGU2

The Figure 4-15 shows the input/outputs signals of the 8AGU2, and the Table 4-8 shows its description.

Input	Description	Comes From
AGU2(0)_PARAM	Has the Parameters to configure the AGU2(0)	Configuration Control
AGU2(1..7)_PARAM	Has the Parameters to configure the AGU2(1..7)	Configuration Control
8AGU2_CTRL	Has the Control signals to manage the all the 8AGU2	Configuration Control
Output	**Description**	**Goes To**
ADDRESS2(0)	Address bus of 13 bits width, for writing or reading on the Memory Structure, used for all the interleaver configurations	Memory Structure
ADDRESS2(1..7)	Address bus of 10 bits width, for writing or reading on the Memory Structure, used for 802 & Compact Bit interleaving	Memory Structure

Table 4-8 Input and output organization of 8AGU2

The Figure 4-16 shows the organization of the hardware blocks of the 8AGU2. There we can see the 8 blocks of AGU2, form AGU2(0) to AGU2(7).

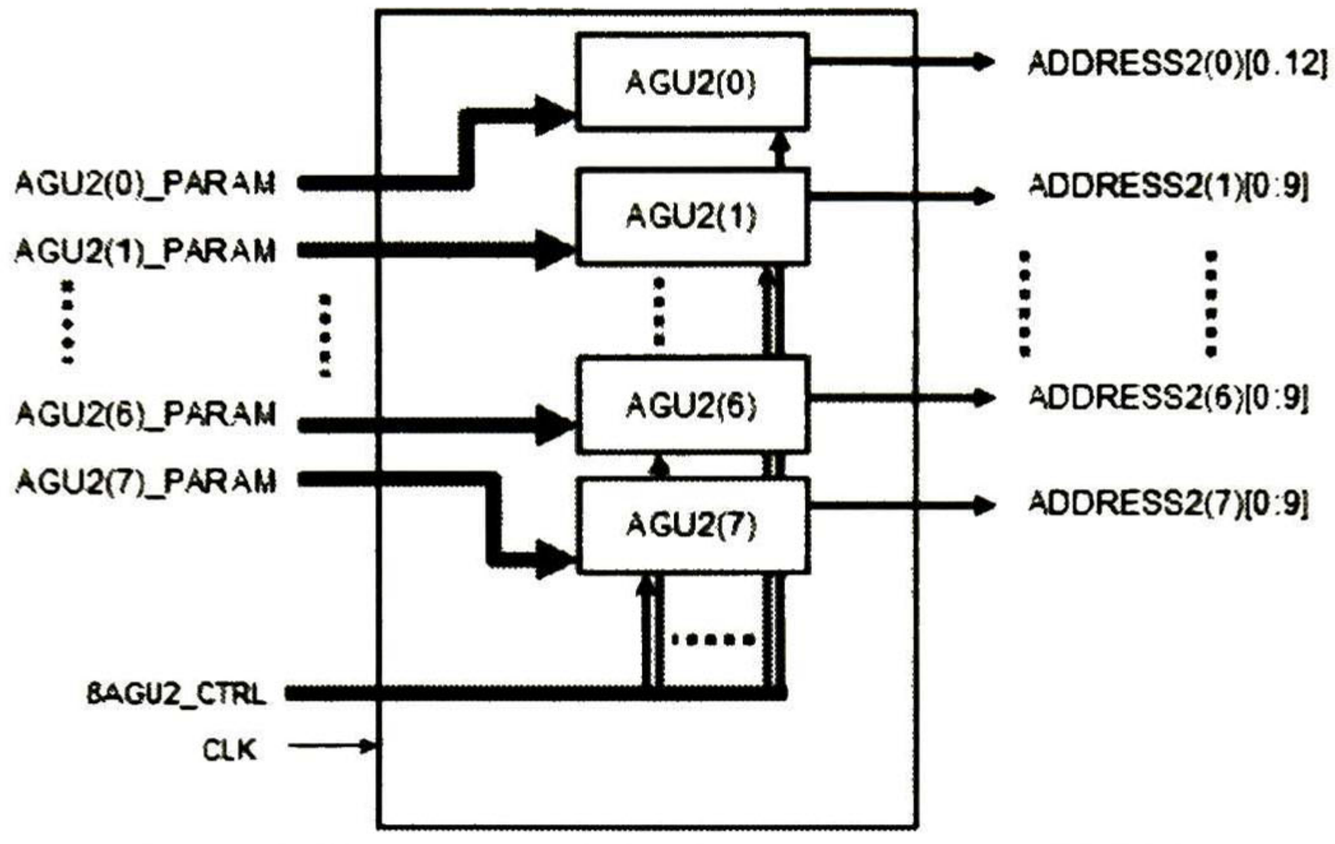


Figure 4-16 Hardware block organization of 8AGU2

4.2.2.1 AGU2 (0)

The AGU2(0) is different to the other AGU2(1..7), because AGU2(0) has an extra logic to generate the Symbol Interleaver sequence and the Convolutional Interleaving and has an address bus of 13 bits while the AGU2(1..7) has a address bus of 10 bits.

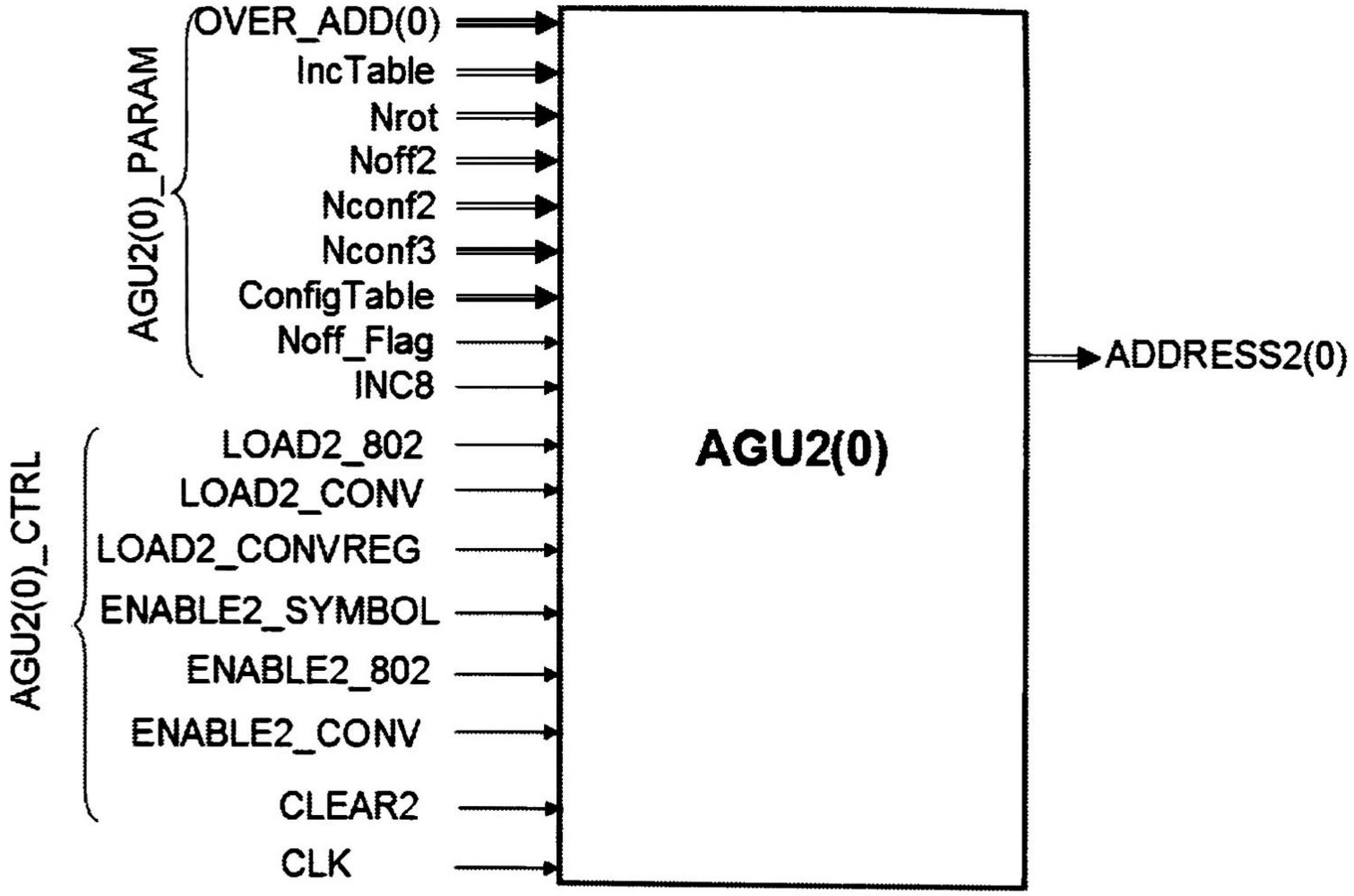


Figure 4-17 Input/output signal diagram for AGU2(0)

The Figure 4-17 shows the input/outputs signals of the AGU2(0), and the Table 4-9 shows its description

Input	Description	Comes From
OVER_ADD(0)	Extra increment in the Configuration table	Configuration Control
IncTable	Increment Values of then Inc Table0 and Inc Table1 registers	Configuration Control
NCONF2	Modulus of the Adder	Configuration Control
NCONF3	Modulus of the Counter Nconf3	Configuration Control
Noff2	Offset Preload of the Counter Nconf3	Configuration Control
ConfigTable	Configuration Values of the Table	Configuration Control
Noff_flag	Enables the shift of the configuration table on the overflow	Configuration Control

INC8	Increment by 8 enable	Configuration Control
NROT	Initial value of the Counter Nconf2	Configuration Control
CLEAR2	Clear of the counters and Registers except he CONV_REG.	Configuration Control
LOAD2_802	Load signal that loads the Counters values and the configuration table values	Configuration Control
LOAD2_CONV	Load signal that loads the Counters values and the configuration table values	Configuration Control
ENABLE2_CONV	Enable of the counters	Configuration Control
ENABLE2_SYMBOL	Enable of the counters	Configuration Control
ENABLE2_802	Enable of the counters and	Configuration Control
Nrot	Initial Value of the counter Nconf2	Configuration Control
LOAD_CONVREG	Load of the CONV_REG	Configuration Control
Output	**Description**	**Goes To**
ADDRESS2(0)	Address bus of 13 bits width, for writing or reading on the Memory Structure, used for all the interleaver configurations	Memory Structure

Table 4-9 Input and output organization of AGU2(0)

For the implementation of the convolutional interleaver, the AGU2(0) has an special register to store the information of the last value interleaved in this configuration, this register load the last address used in the convolutional interleaving process by the assert of the LOAD_CONVREG signal, this is because the convolutional interleaver is a continuous interleaver and not a block Interleaver.

The selector signal selects which memory will pass from the mux to the Nconf2 adder, Table 4-10, and the Mod_6 block indicates which data of the selected memory will pass.

Selector	Value to the Nconf2 Adder
0X	0
10	IncTable0
11	IncTable1

Table 4-10 Selector signal configurations

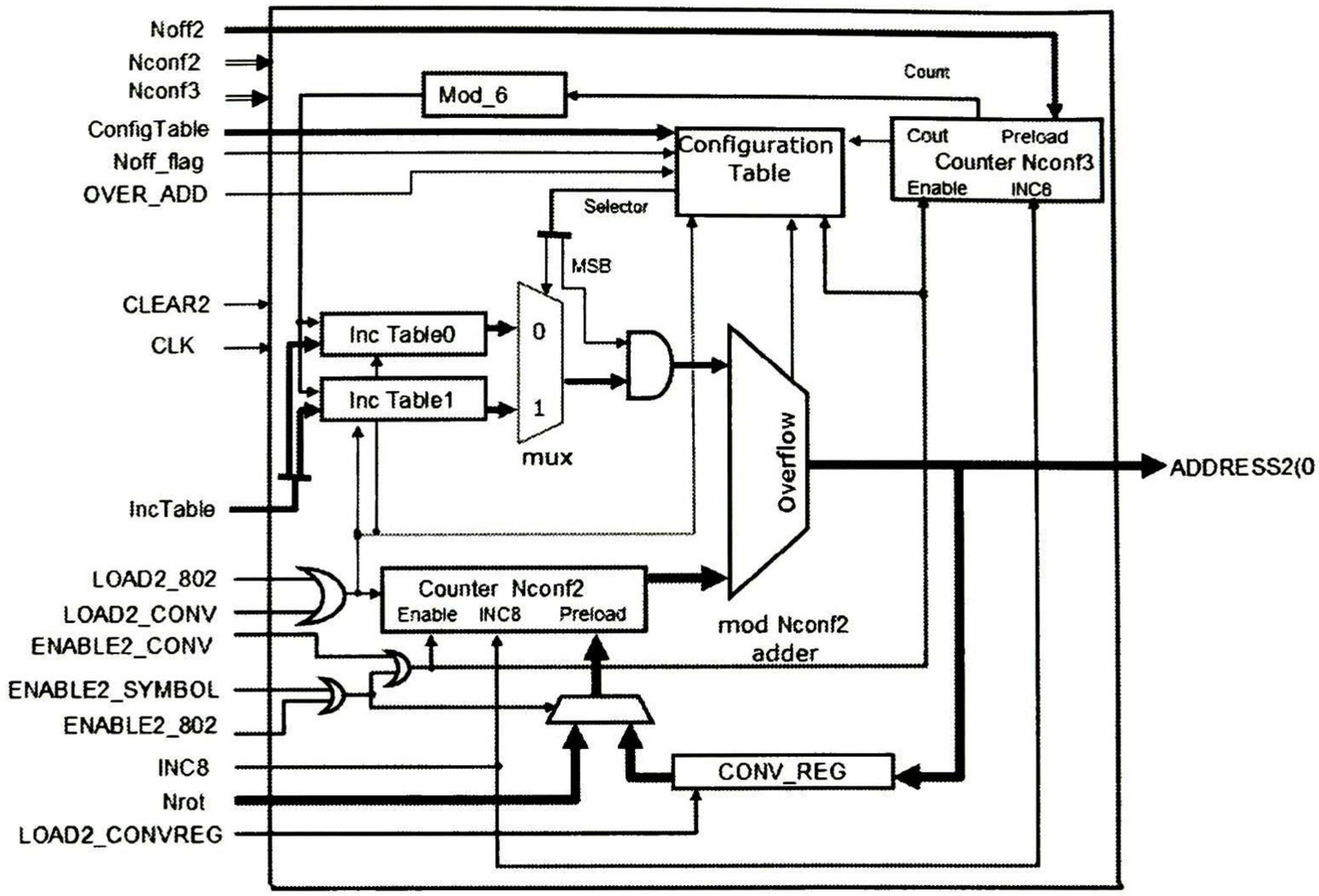


Figure 4-18 Hardware block organization of AGU2(0)

The Figure 4-18 shows the organization of the hardware block of the AGU2(0).

To start an interleaving process the Configuration Control block starts by placing the selected parameters in the AGU2(0)_PARAM bus, this parameters are Nrot, Noff2, Nconf2, Nconf3, ConfigTable, Noff_Flag & IncTable, then the appropriated ENABLE and LOAD signal in the AGU2(0)_CTRL bus are asserted, at this point the AGU is configured for a selected interleaver.

The counter Nconf3 start by the enable signal to generate the values of control of the configuration table, this block has the configuration logic to generate the right selection of data stored in the memories that access to the mux, by enabling the selected memory, selecting the appropriate data in the memory and controlling the mux to let pass the data to the mod Nconf2 Adder.

4.2.2.1.1 Configuration Table

The configuration table loads the values of Selector, in the shift register, and shifts them when the shift signal is asserted and the block is enabled.

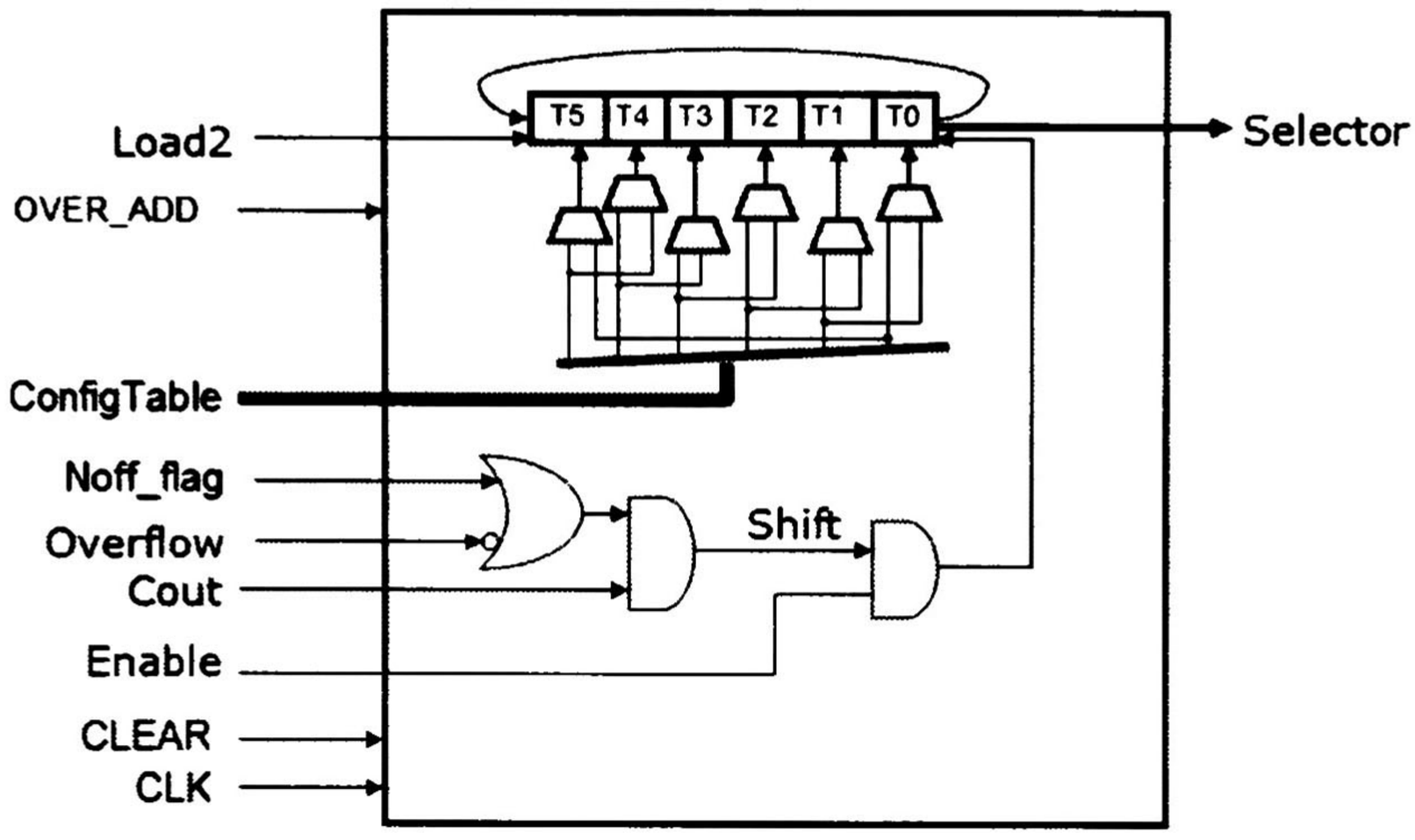


Figure 4-19 Hardware block organization of Configuration Table

The Figure 4-19 shows the organization of the hardware blocks of the Configuration Table, and the Table 4-11 shows its description of its inputs and outputs

Input	Description	Comes From
Load2	Loads the values of the circular Shift Register	Configuration Control
OVER_ADD	Extra increment in the Configuration table	Configuration Control
ConfigTable	Values of the circular Shift Register	Configuration Control
Noff_Flag	Enable of the shift on the Overflow	Configuration Control
Overflow	Overflow flag of the adder	Mod Nconf2 Adder
Cout	Carry out of the counter Nconf3	Counter Nconf2
Enable	Enable value for shift	
Clear	Clear the values of the Circular Shift Register	
Output	**Description**	**Goes To**
Selector	Configuration value. This value is the memory selected from the Mux	MUX

Table 4-11 Input and output organization of Configuration Table

4.2.2.2 AGU2 (1..7)

The AGU2 (1..7) are the simplified form of the AGU2(0), this blocks are enabled for the 802 interleaving, and compact bit interleaving schemes.

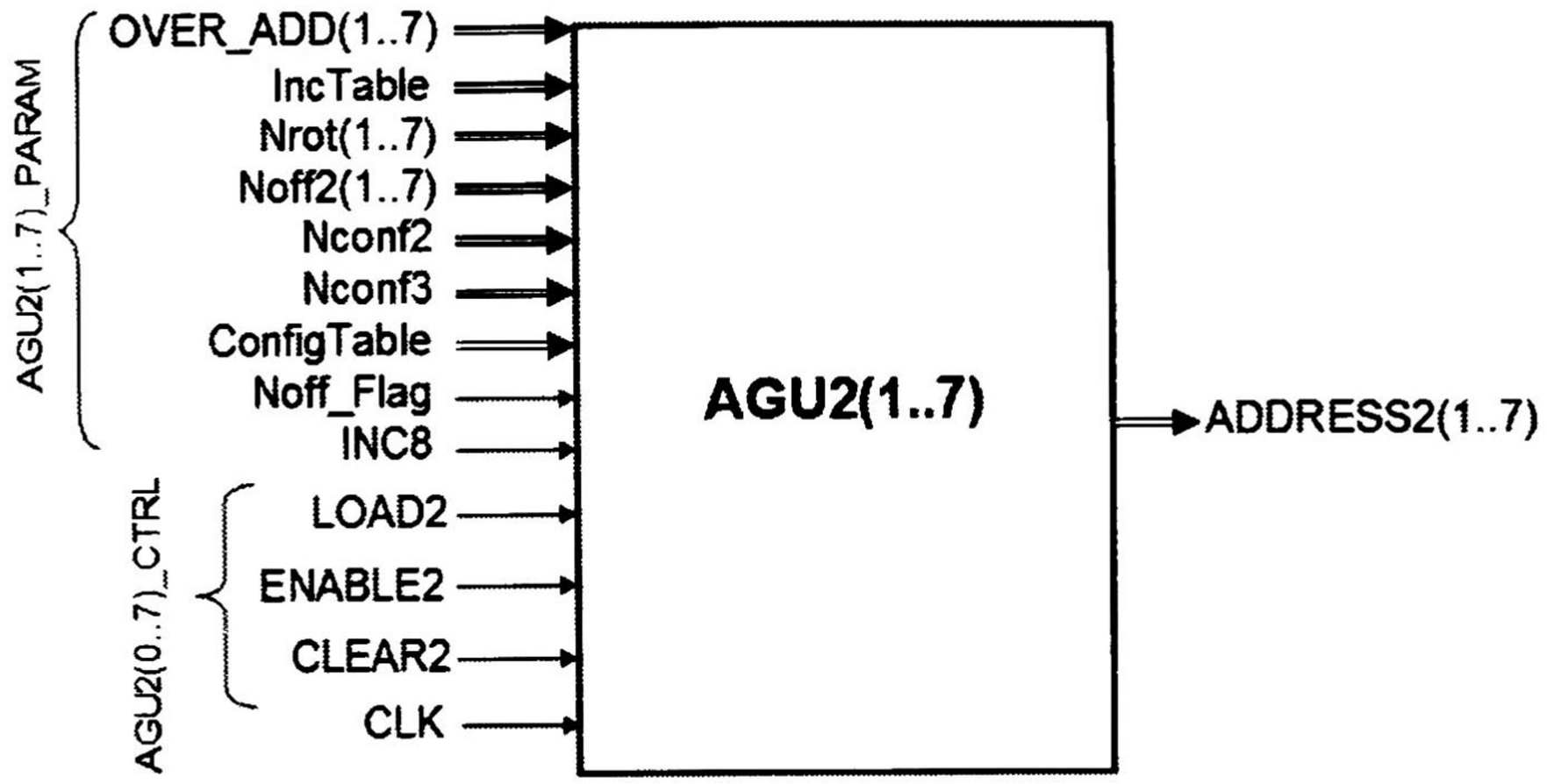


Figure 4-20 Input/output signal diagram for AGU2(1..7)

The Figure 4-20 shows the input/outputs signals of the AGU2(1..7), and the Table 4-12 shows its description

Input	Description	Comes From
Noff2(1..7)	Offset Preload of the Counter Nconf3	Configuration Control
OVER_ADD(1..7)	Extra increment in the Configuration table	Configuration Control
IncTable	Values of the increments to save in the registers	Configuration Control
NCONF2	Modulus of the Adder	Configuration Control
NCONF3	Modulus of the Counter Nconf3	Configuration Control
ConfigTable	Configuration Values of the Table	Configuration Control
Noff_flag	Enables the shift of the configuration table on the overflow	Configuration Control
CLEAR2	Clear of the counters and Registers except he CONV_REG.	Configuration Control
LOAD2	Load signal that loads the Counters values and the configuration table values	Configuration Control
ENABLE_CONV	Enable of the counters	Configuration Control
Nrot(1..7)	Initial Value of the	Configuration Control

	counter Nconf2	
Output	Description	Goes To
ADDRESS2(1..7)	Address bus of 10 bits width, for writing or reading on the Memory Structure, used for all the interleaver configurations	Memory Structure

Table 4-12 Input and output organization of AGU2(1..7)

The Figure 4-21 shows the organization of the hardware blocks of the AGU2(1..7)

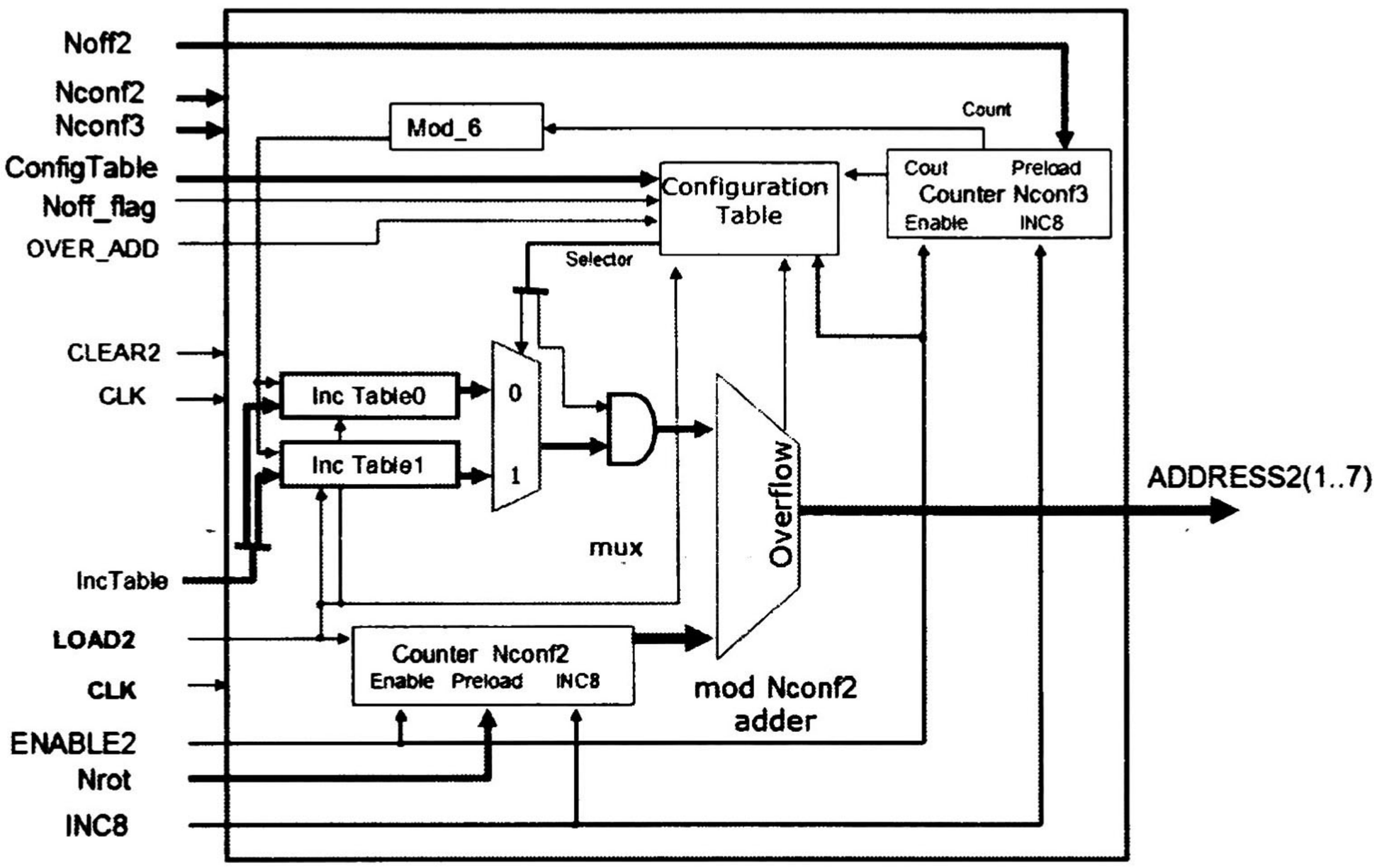


Figure 4-21 Hardware block organization of AGU2(1..7)

The configuration for this block is given by the AGU2 (1..7) PARAM bus, and this parameters are Noff2, Nconf2, Nconf3, ConfigTable, Noff_Flag & Nrot that are given by the configuration control block.

The INC8 signal enables the counters to increment 1 or 8 if is not asserted and if it's asserted, INC8 signal is asserted for the configurations of 802 interleaving and compact bit interleaving, and is not asserted for convolutional and symbol interleaver.

The Nrot & Noff2 signal has a different value for each AGU2 (0..7).

4.2.3 Memory Structure

The memory structure is a block of many memory blocks which are selected depending on the configuration for the interleaver process desired in the configuration control.

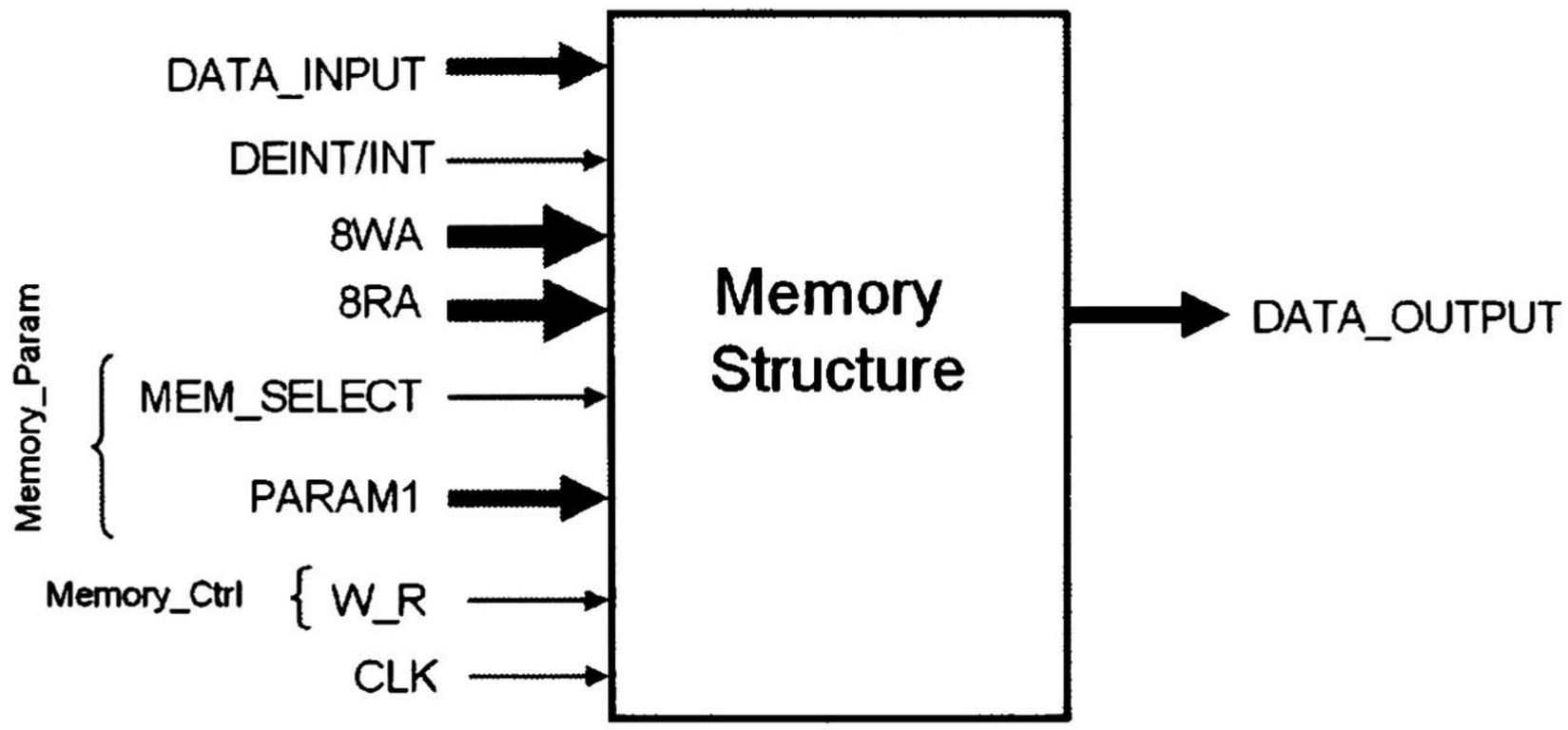


Figure 4-22 Input/output signal diagram for Memory Structure

The Figure 4-22 shows the input/outputs signals of the memory structure, and the Table 4-13 shows its description

Input	Description	Comes From
DATA_INPUT[7:0]	Data to be written in the memories	Input Interface
DEINT/INT	Deinterleaver Interleaver selector When 1 = Deinterleaver When 0 = Interleaver	Input Interface
8WA	8 Write address	8AGU1 & 8AGU2
8RA	8 Read address	8AGU1 & 8AGU2
MEM_SELECT[2:0]	This signal controls the selection of the memories in the Memory Structure	Configuration Control
PARAM1[7:0]	Special parameter for 802Memory	Configuration Control
W/R	If 1 then is in write mode if 0 then is in read mode	Configuration Control
CLEAR	Clear signal	Configuration Control
Output	**Description**	**Goes To**
DATA_OUTPUT[7:0]	Data read from the Memory Structure	Output Interface

Table 4-13 Input and output organization of Memory Structure

The memory structure has 3 blocks of memory:

- Convolutional memory.
- 802 Memory.
- Symbol Memory.

The Figure 4-23 shows the organization of the hardware blocks of the Memory Structure.

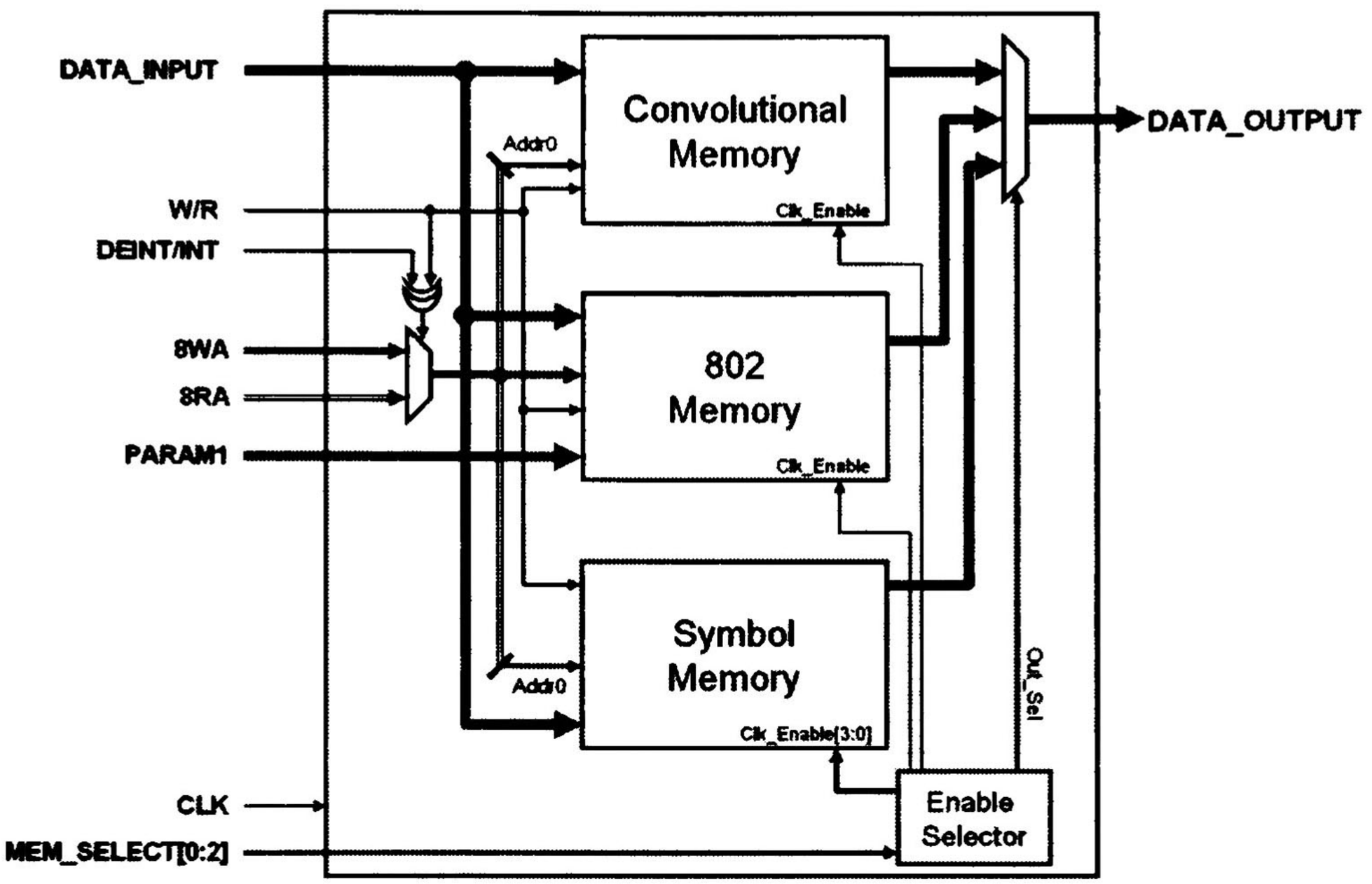


Figure 4-23 Hardware block organization of Memory Structure

The interleavers of the 802.11a, 802.11n, 802.16e and the compact bit interleaver (DVB) are mapped in the 802 memory. The symbol interleaver is mapped in the symbol memory, and the convolutional interleaving is mapped in the convolutional memory

Mem_Select	Enabled
000	None
001	Convolutional Memory
01X	802 Memory
100	Symbol Memory (2 bit)
101	Symbol Memory (4 bit)
110	Symbol Memory (6 bit)
111	Symbol Memory (8 bit)

Table 4-14 Mem_Select signal configurations

The Mem_Select signal has the control to enable the selected memory block as indicated in the Table 4-14.

4.2.3.1 Convolutional Memory

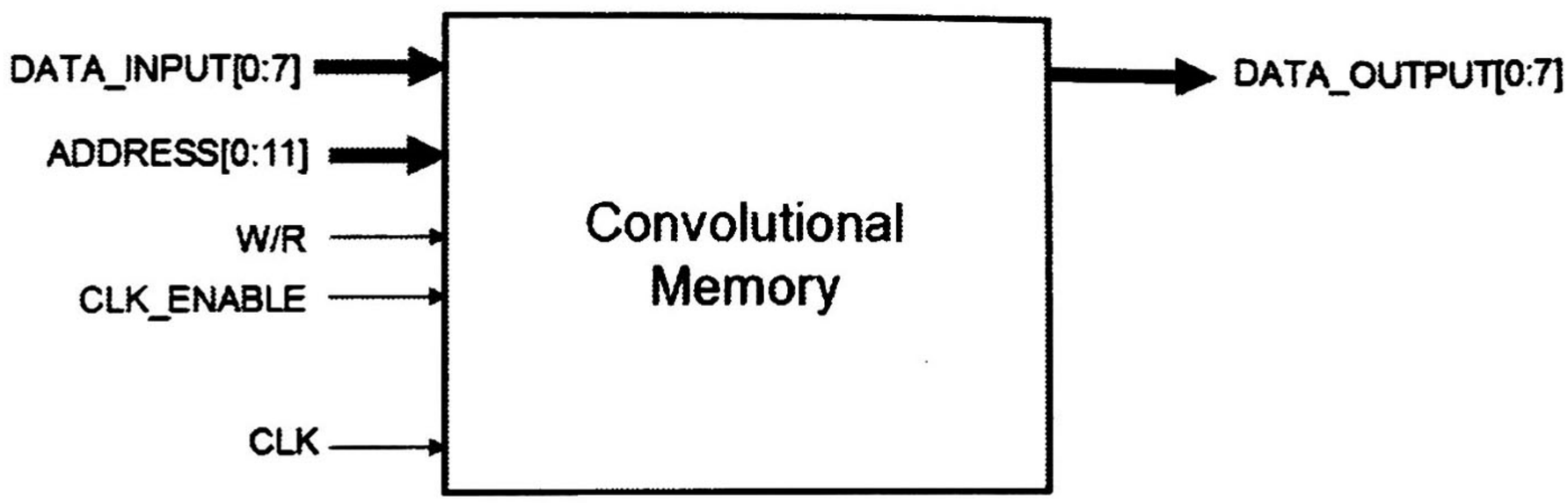


Figure 4-24 Input/output signal diagram for Convolutional Memory

The memory is of 2048x8 bits size.

The Figure 4-24 shows the input/outputs signals of the convolutional memory, and the Table 4-15 shows its description.

Input	Description	Comes From
DATA_INPUT[7:0]	Data to be written in the memories	Input Interface
ADDRESS[11:0]	address to write or read of 12 bits	8AGU1 & 8AGU2
W/R	If 1 then is in write mode if 0 then is in read mode	Configuration Control
CLK_ENABLE	Memory Enable	Configuration Control
Output	**Description**	**Goes To**
DATA_OUTPUT[7:0]	Data read from the memories	Output Interface

Table 4-15 Input and output organization of Convolutional Memory

4.2.3.2 802 Memory

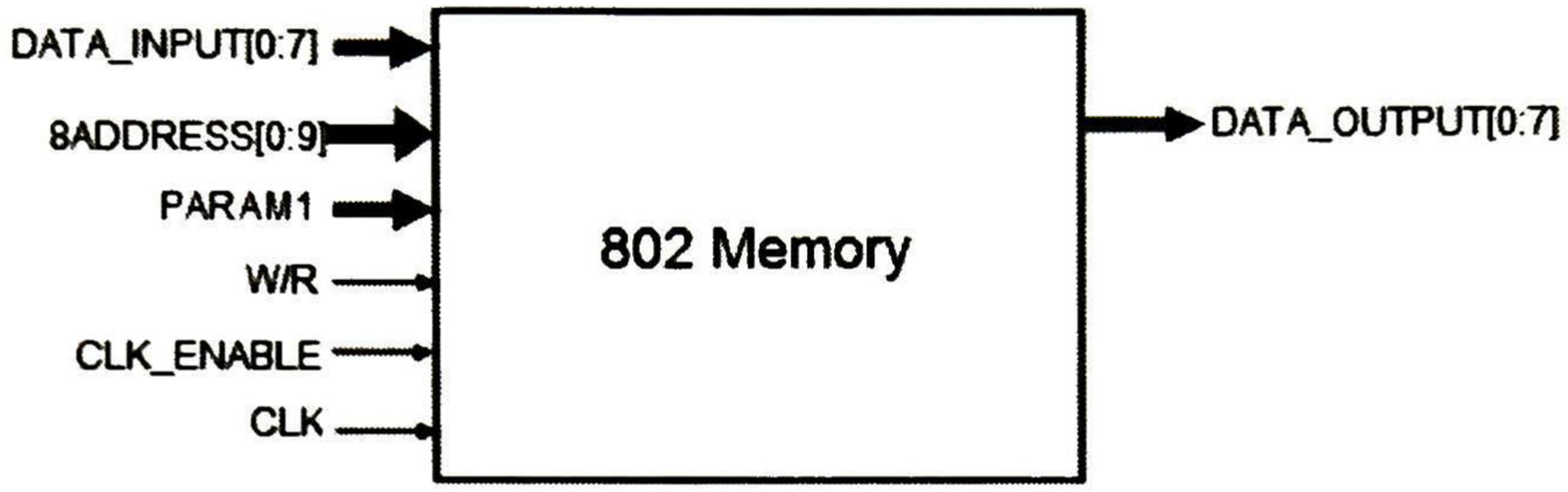


Figure 4-25 Input/output signal diagram for 802 Memory structure

This is an special memory array with the STS structure, with some decoding logic at the Address Decoding Units (ADU), uses 16 memories and 2 crossbar to implement an 8bit parallelism without access conflicts, the 8ADDRESS signals are the 8 address generated by the 8AGU1 or 8AGU2. Each address from the 8ADDRESS bus enters to an ADU, the ADU decodes the CB_Ctrl and the memory address position in which the data will be saved.

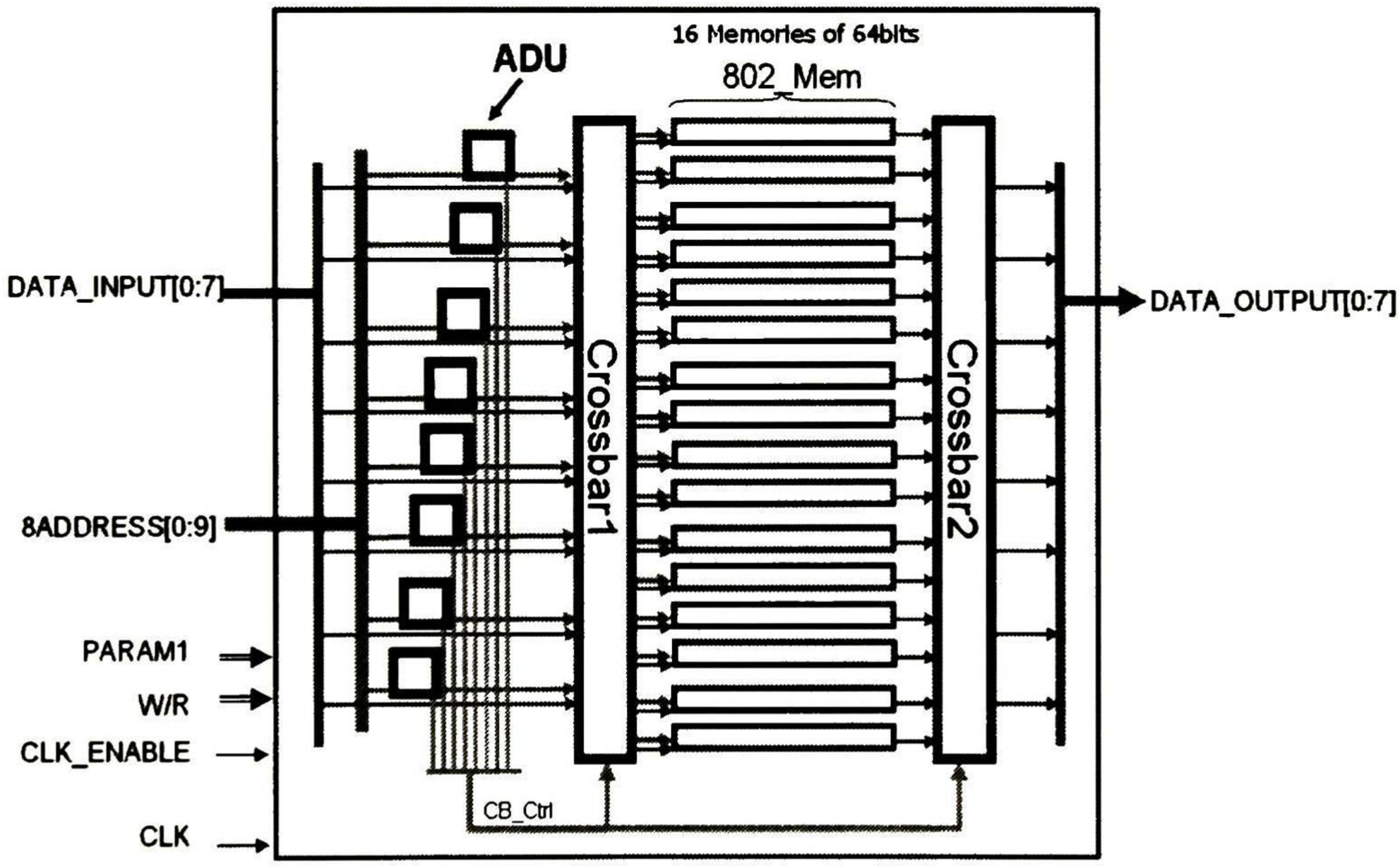


Figure 4-26 Hardware block organization of 802 Memory

The Figure 4-26 shows the block organization and input/outputs of the 802 Memory, and the Table 4-16 shows its description.

Input	Description	Comes From
DATA_INPUT[7:0]	Data to be written in the memories	Input Interface
8ADDRESS[9:0]	8 Write address of 10 bits each	8AGU1 & 8AGU2
W/R	If 1 then is in write mode if 0 then is in read mode	Configuration Control
CLK_ENABLE	Memory Enable	Configuration Control
PARAM1[7:0]	Parameter to configure the ADU	Configuration Control
Output	**Description**	**Goes To**
DATA_OUTPUT[7:0]	Data read from the memories	Output Interface

Table 4-16 Input and output organization of 802 Memory

4.2.3.2.1 ADU

This block is the address decoding unit, and generates the values for addressing the memories and to control the Crossbar.

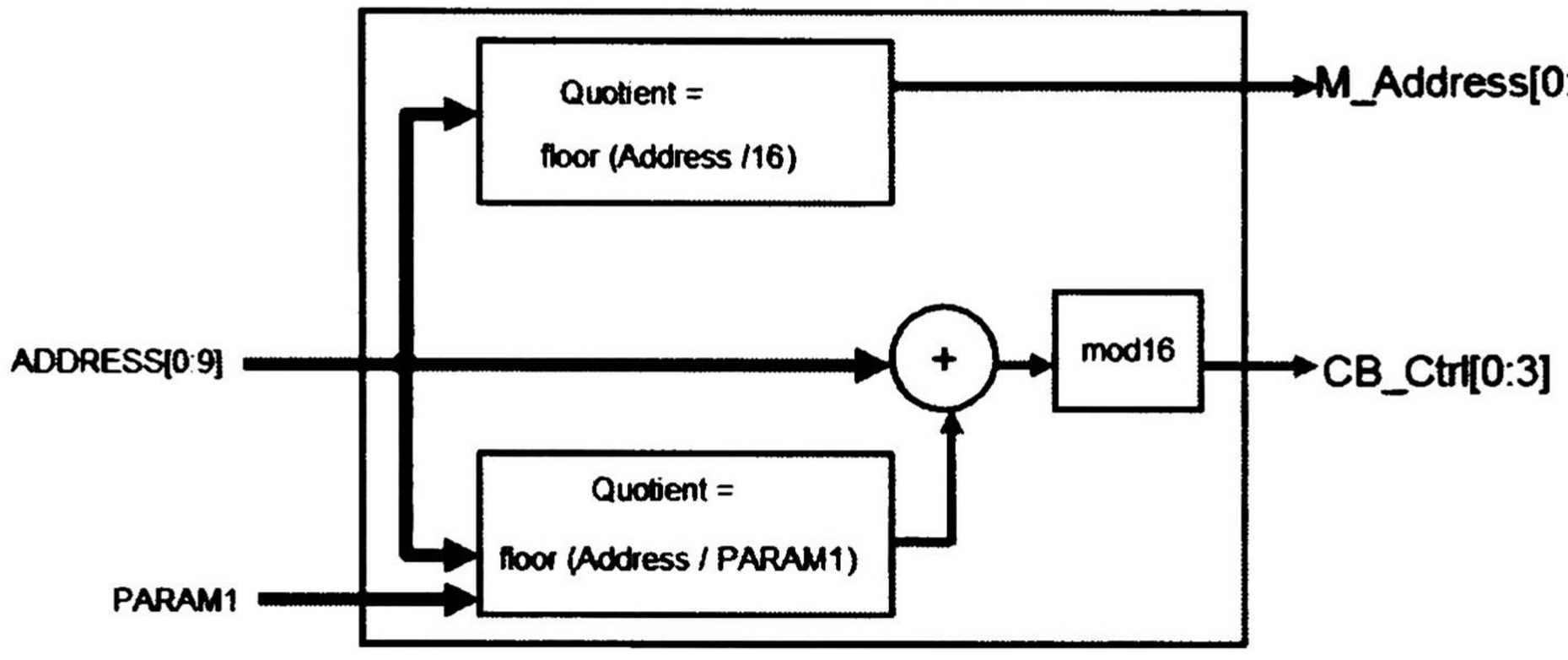


Figure 4-27 Diagram block organization of ADU

The operation Quotient=floor (Address/16) is accomplish by simply make a 4 shift operation to the LSB. The Mod16 block can be made by only take the first 4LSB bits.

The operation Quotient=floor (Address/PARAM1), needs an divider, because the values of PARAM1 are not multiples of 2^n. This divider has an extra logic to fix when the PARAM1 value is '0' the output value will be '0', this tells to the divider block to be disabled.

The Figure 4-27 shows the Block Organization and input/outputs signals of the ADU, and the Table 4-17 shows its description.

Input	Description	Comes From
ADDRESS[0:9]	Address Generated by the AGU to decode	AGU1 and AGU2
PARAM1[7:0]	Parameter for special increment in the Decodification, this signal comes in the MEMORY_PARAM bus	Configuration Control
Output	**Description**	**Goes To**
M_Address[0:5]	Address for memory Addressing	802_Mem thru Crossbar
CB_Ctrl[0:3]	Control signal of the Crossbar	Crossbar

Table 4-17 Input and output organization of ADU

4.2.3.2.2 802_Mem

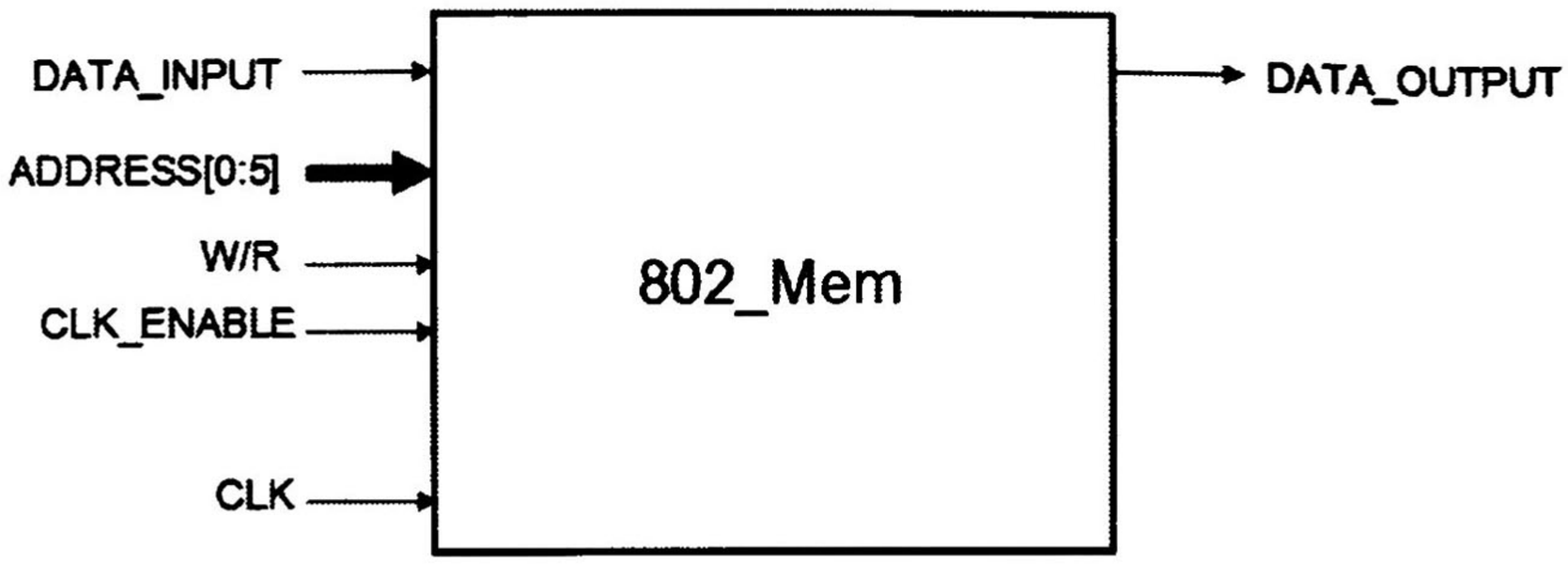


Figure 4-28 Input/output signal diagram for 802_Mem

The 802_Mem is an memory of 64x1 bits.

The Figure 4-28 shows the input/outputs signals of the 802_Mem, and the Table 4-18 shows its description.

Input	Description	Comes From
DATA_INPUT	Input Data to the memory	Input Interface
ADDRESS[0:5]	Write or Read Address	Crossbar1 or Crossbar2
W/R	Write or Read Control, if 1 Write, if 0 Read	Configuration Control
CLK_ENABLE	Enable signal for the memory block	Configuration Control
Output	**Description**	**Goes To**
DATA_OUTPUT	Data Output of the memory	Output Interface

Table 4-18 Input and output organization of 802_Mem

4.2.3.3 Symbol Memory

The symbol memory is an array of 4 memories, that are activated as is necessary for the configuration selected, the symbol interleaver can work over 2, 4, 6 or 8 bits for each word of interleaving.

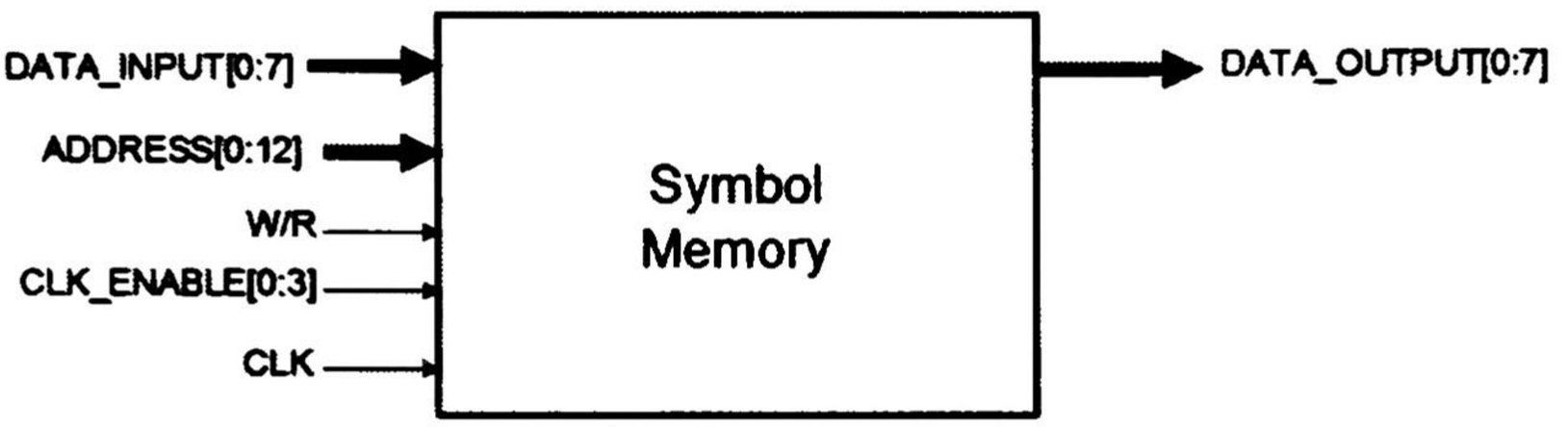


Figure 4-29 Input/output signal diagram for Symbol Memory

The Figure 4-29 shows the input/outputs signals of the symbol memory, and the Table 4-19 shows its description.

Input	Description	Comes From
DATA_INPUT[0:7]	Input Data to the memory	Input Interface
ADDRESS[0:12]	Write or Read Address of 13bits	8AGU1 & 8AGU2
W/R	Write or Read Control, if 1 Write, if 0 Read	Configuration Control
CLK_ENABLE[0:3]	Enable signal for each MemoryS	Configuration Control
Output	**Description**	**Goes To**
DATA_OUTPUT[0:7]	Data Output of the memory	Output Interface

Table 4-19 Input and output organization of Symbol Memory

The Figure 4-30 shows the organization of the hardware blocks of the symbol memory.

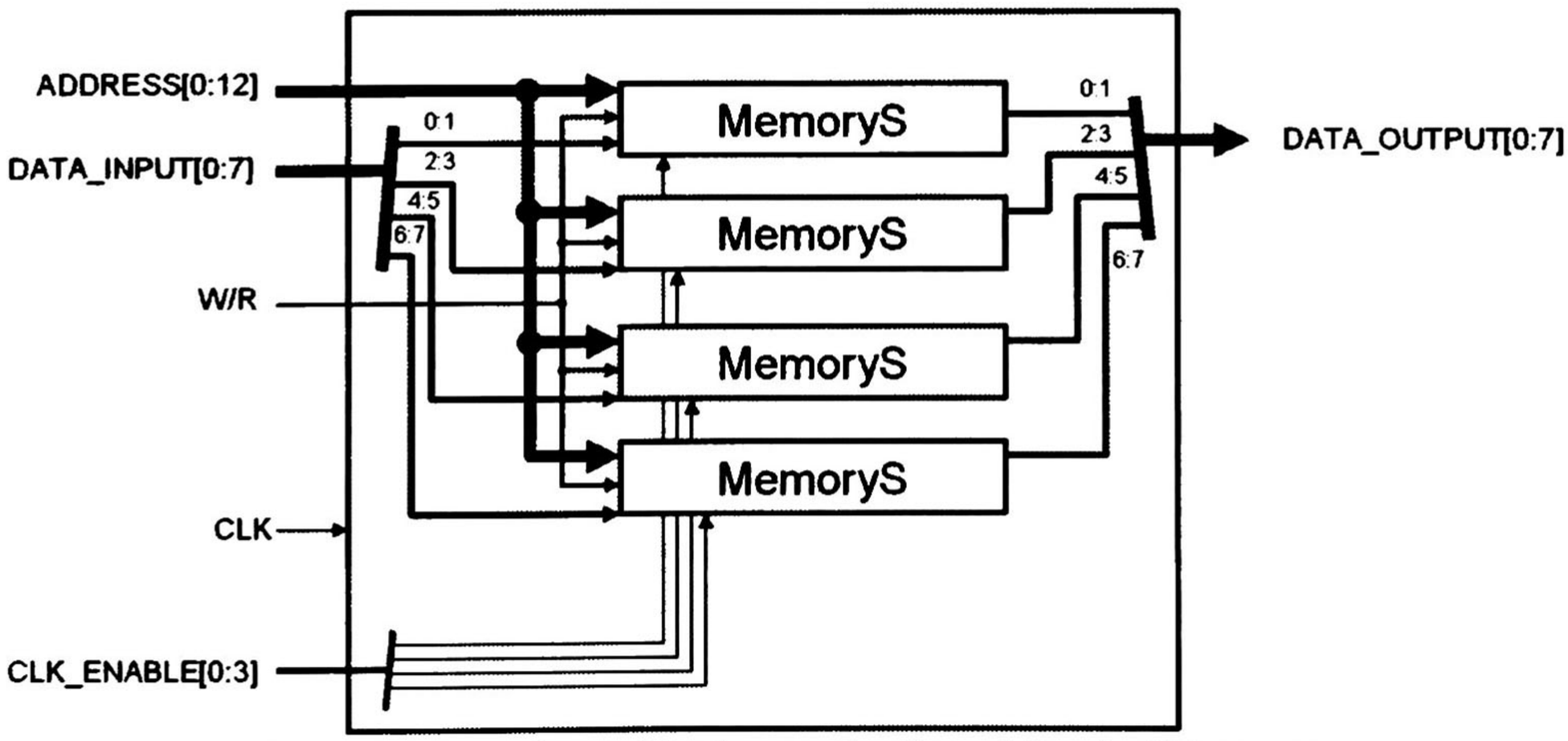


Figure 4-30 Hardware block organization of Symbol Memory

4.2.3.3.1 MemoryS

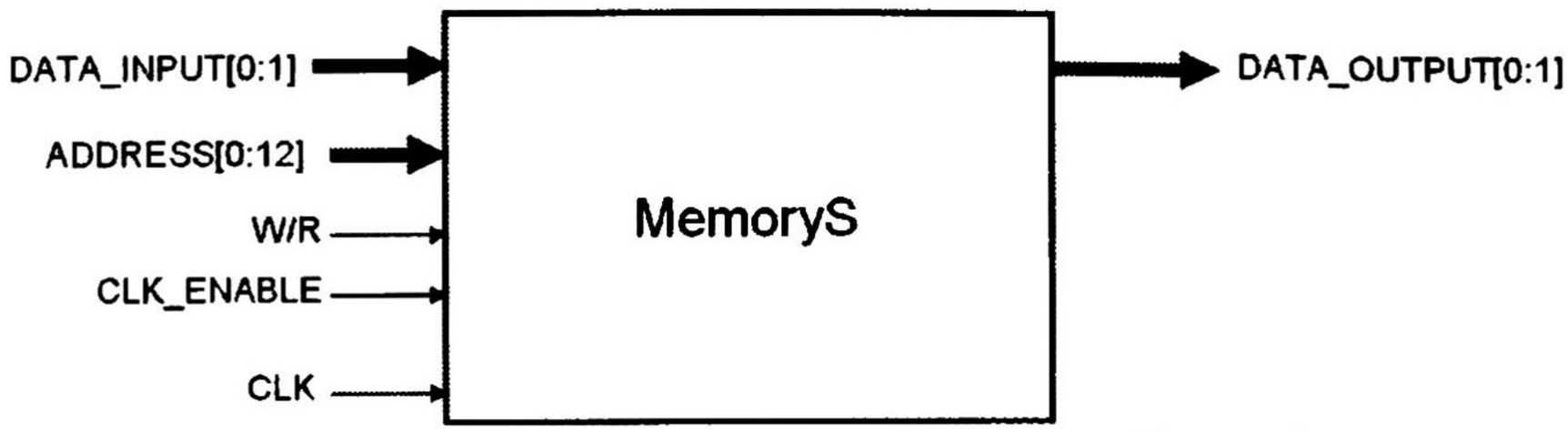


Figure 4-31 Input/output signal diagram for MemoryS

The memory is of 8Kx2bits size.

The Figure 4-31 shows the input/outputs signals of the MemoryS, and the Table 4-20 shows its description.

Input	Description	Comes From
DATA_INPUT[0:1]	Input Data to the memory	Input Interface
ADDRESS[0:12]	Write or Read Address of 13bits	8AGU1 & 8AGU2
W/R	Write or Read Control, if 1 Write, if 0 Read	Configuration Control
CLK_ENABLE	Enable signal for the memory block	Configuration Control
Output	**Description**	**Goes To**
DATA_OUTPUT[0:1]	Data Output of the memory	Output Interface

Table 4-20 Input and output organization of MemoryS

4.2.4 Configuration Control

The configuration control has two operation modes depending on the CONF_START signal, configure mode and operation mode.

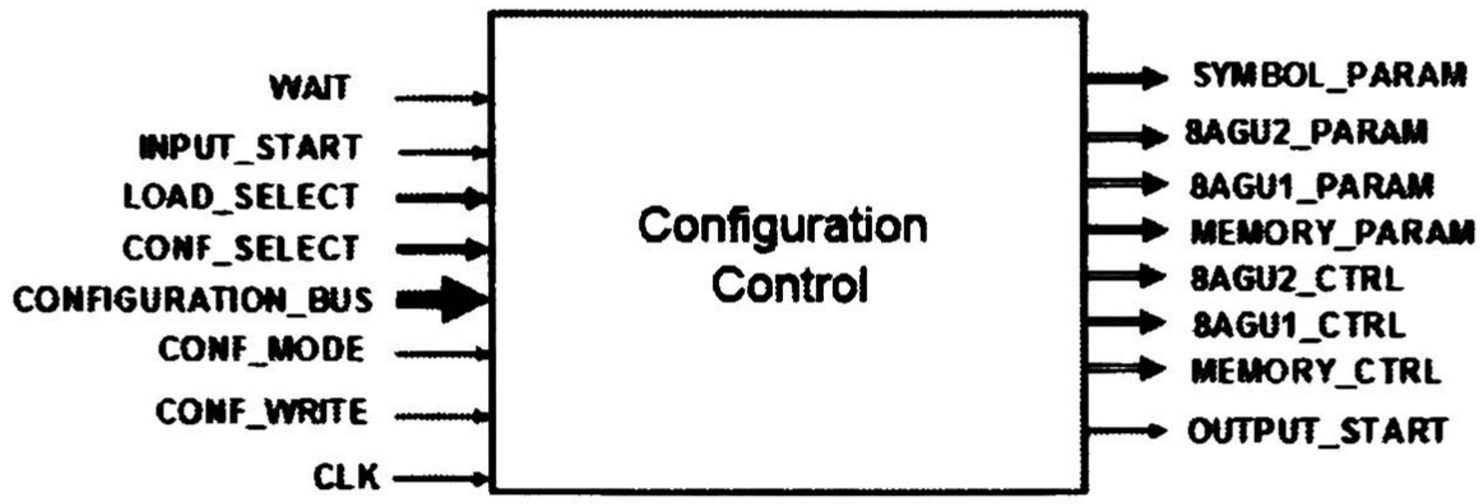


Figure 4-32 Input/output signal diagram for Configuration Control

The Figure 4-32 shows the input/outputs signals of the configuration control, and the Table 4-21 shows its description.

Input	Description	Comes From
WAIT	This signal tells to the Configuration Control to suspend the processing	Input Interface
INPUT_START	This signal indicates when a block of data is ready to be interleaver	Input Interface
CONF_SELECT	This signal selects one of the configurations of interleaver on the fly	Input Interface
LOAD_SELECT	Load the selected parameters on the fly by CONF_SELECT	Input Interface
CONFIGURATION_BUS	This bus is used to introduce the entire configuration parameters to the MSI	Conf Interface
CONF_MODE	This signal selects between the 2 modes of operation of the MSI If 1 then is in Configuration Mode If 0 then is in Operation Mode	Conf Interface
CONF_WRITE	This signal when is activated in the configuration mode and indicates that the word in the	Conf Interface

	CONFIGURATION_BUS is ready to be written	
Output	**Description**	**Goes To**
SYMBOL_PARAM	Parameters to configure the Symbol Interleaver	AGU1(0)
8AGU2_PARAM	Parameters to configure the 8AGU2	8AGU2
8AGU1_PARAM	Parameters to configure the 8AGU1	8AGU1
MEMORY_PARAM	Parameters to configure the Memory Structure	Memory Structure
8AGU2_CTRL	Control signals of the 8AGU2	8AGU2
8AGU1_CTRL	Control signals of the 8AGU1	8AGU1
MEMORY_CTRL	Control signals of the Memory Structure	Memory Structure
OUTPUT_START	Signal that indicate when the output is ready to be read	Output Interface

Table 4-21 Input and output organization of Configuration Control

When the CONF_START is asserted the configuration control is in configure mode, and starts the reading of the configuration parameters in the CONFIGURATION BUS.

The parameters are read sequentially into the Param_Memory, till the configuration CONF_START signal is no longer asserted.

The Figure 4-33 shows the organization of the hardware blocks of the configuration controll

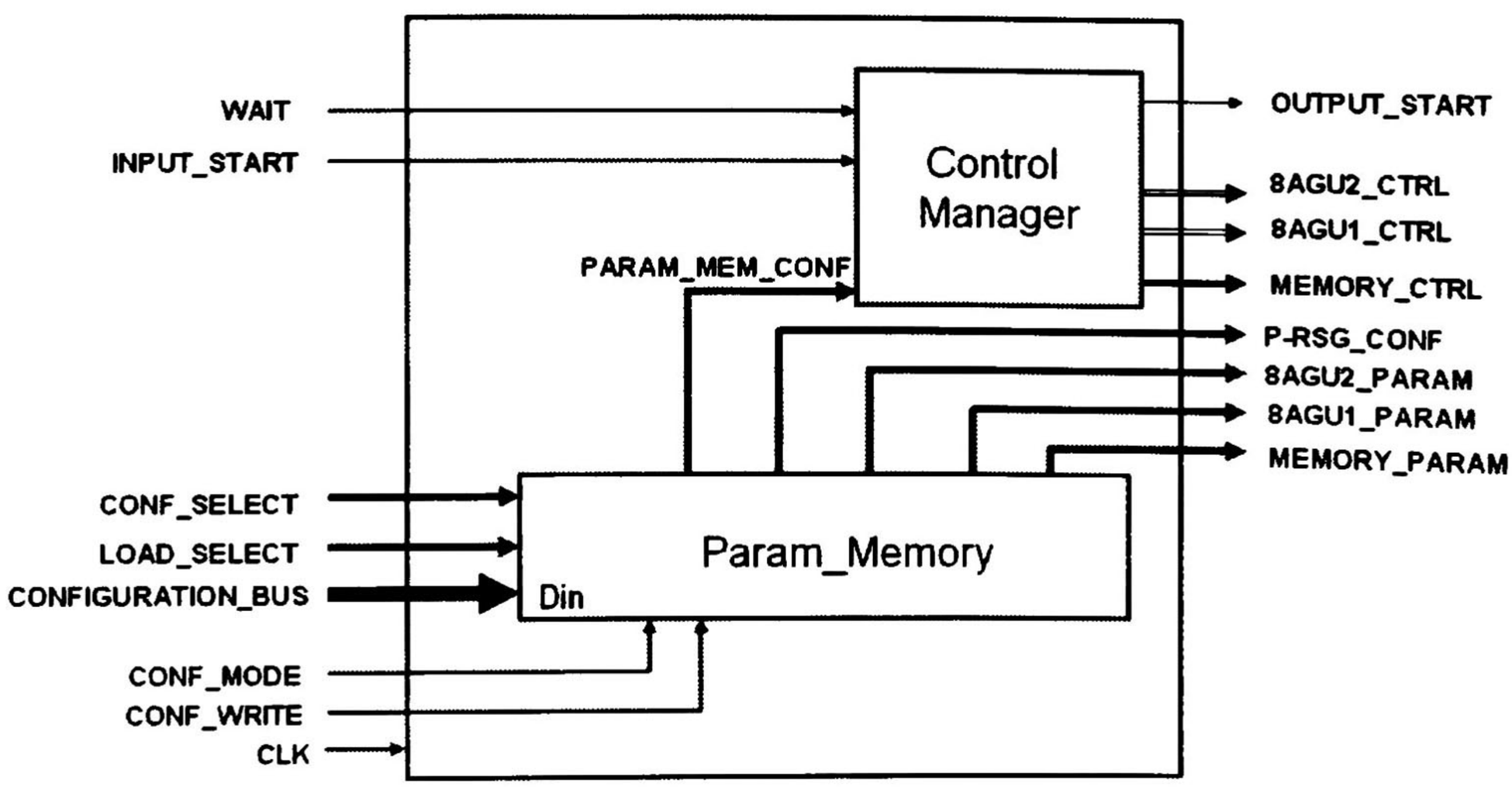


Figure 4-33 Hardware block organization of Configuration Control

The configuration parameters are the parameters used to implement the different interleaving process by configuration on the fly.

When the CONF_START signal is not asserted the configuration control assumes the operation mode of operation, in this mode the configuration control reads the CONF_SELECT signal and selects the desired parameters to implement the interleaving process selected.

The parameters SYMBOL_PARAM, 8AGU2_PARAM, 8AGU1_PARAM, MEMORY_PARAM, are read directly from the Param_Memory.

The 8AGU2_PARAM bus has the following signals on it Nrot(0..7), Noff2, Nconf2, Nconf3, ConfigTable, Noff_flag, INC8.

The 8AGU1_PARAM bus has the following signals on it INITIAL_VALUE(0..7), NCONV, Nconf1,Nconf2.

The MEMORY_PARAM bus has the following signals on MEM_SELECT, PARAM1

The MEMORY_CTRL bus has the following signals on it W_ENABLE.

The 8AGU2_CTRL bus has the following signals on it LOAD_802, LOAD_CONV, LOAD_CONVREG, LOAD, ENABLE_802, ENABLE_CONV, ENABLE_SYMBOL, ENABLE, CLEAR

The 8AGU1_CTRL bus has the following signals on it LOAD_802, LOAD_CONV, LOAD, ENABLE_802, ENABLE_CONV, ENABLE_SYMBOL, ENABLE, CLEAR

The MEMORY_CTRL bus manipulates the memory structure block and configures it for writing or reading.

The 8AGU2_CTRL, 8AGU1_CTRL and MEMORY_CTRL are signals that manipulate

The configuration control has the values of the parameters necessary to configure the whole MSI to implement any interleaving process in the standards 802.11a, 802.11n, 802.16e, DVB.

A description of the parameters list is shown in Table 4-28.

The parameters are read one by one on the CONFIGURATION_BUS.

The reconfigurability of this architecture makes possible an adaptation to future modifications in the interleavers.

4.2.4.1 Control Manager

The control manager has the necessary information to organize the input output access to the memory structure, and the necessary control for the 8AGU1 and 8AGU2.

Figure 4-34 Input/output signal diagram for Control Manager

The Figure 4-34 shows the input/outputs signals of the control manager, and the Table 4-22 shows its description.

Input	Description	Comes From
WAIT	This signal disables the interleaver processing while is asserted	Input Interface
INPUT_START	Indicates when the data to interleave is ready to be read, and start the Control State Machine	Input Interface
PARAM_MEM_CONF	Parameters to configure the proper work of the Control Manager	Memory Param
CLEAR	Clear Signal	Input Interface
Output	**Description**	**Goes To**
MEMORY_CTRL	Control parameters for the Memory Structure, the W/R signal	Memory Structure
OUTPUT_START	Output Is ready signal	Output Interface
8AGU1_CTRL	Control parameters for the 8AGU1	8AGU1
8AGU2_CTRL	Control parameters for the 8AGU2	8AGU2

Table 4-22 Input and output organization of Control Manager

The Figure 4-35 shows the organization of the hardware blocks of the control manager.

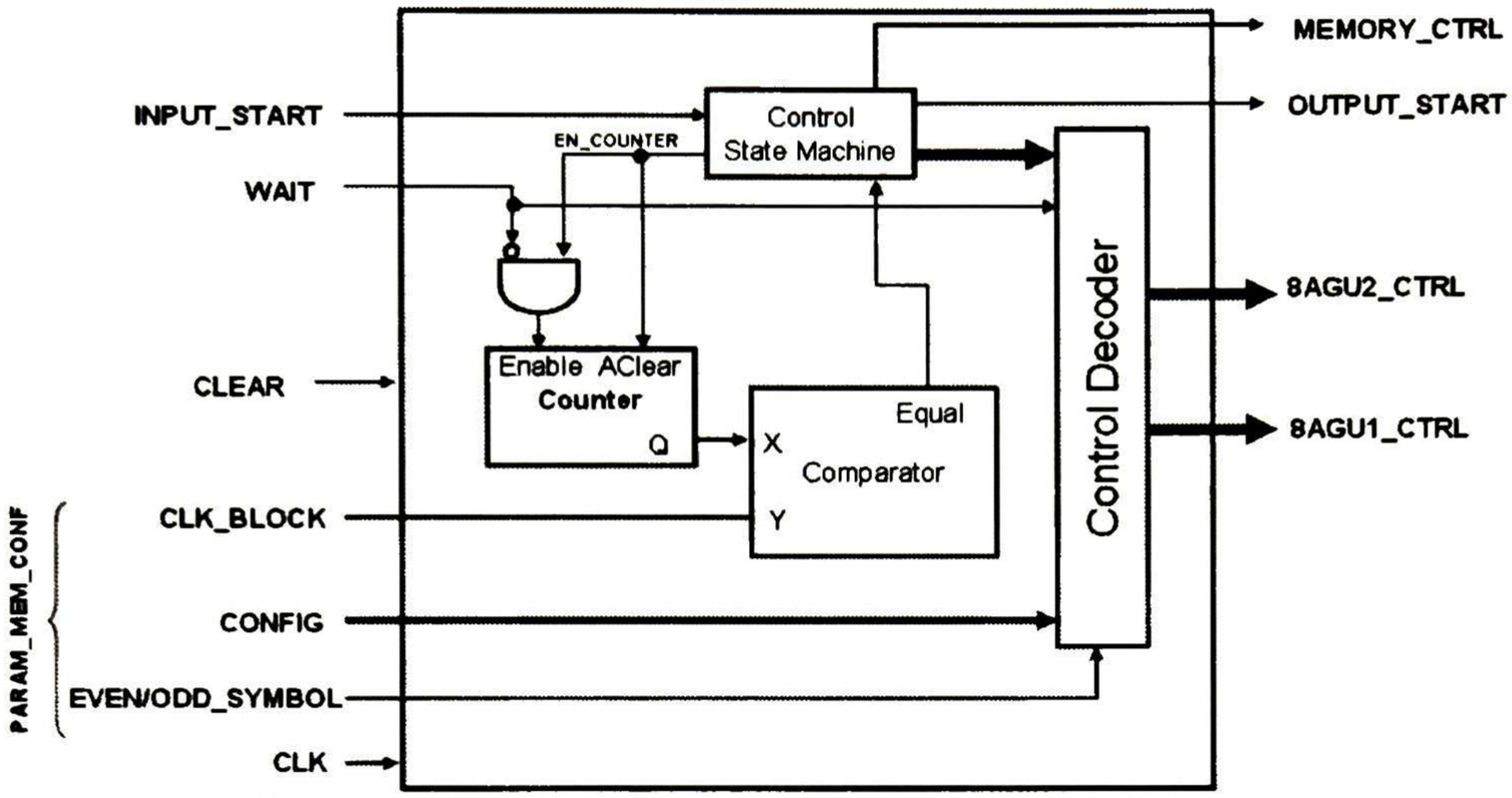


Figure 4-35 Hardware block organization of Control Manager

This block configures the connections to make possible the different configurations of input/ outputs necessary in the different configurations of interleavers in the standards 802.11a, 802.11n, 802.16e, DVB.

The EVEN/ODD_SYMBOL signal is only used in the configuration of symbol interleaver, and specifies that the current OFDM symbol is even if is asserted of odd if not.

The WAIT signal unset the enables to momentary stop the processing of the interleaving data.

The INPUT_START signal starts the control state machine and the interleaver process.

The PARAM_MEM_CONF bus has following signals on it:

- CLK_BLOCK: this parameter set the number of active cycles that MSI will be reading or writing the Interleaving data.
- EVEN/ODD SYMBOL: this parameter specifies if the OFDM frame is even or odd.
- CONFIG: this parameter specifies one of the 8 configuration modes for the different interleaver/deinterleaver schemes, listed in Table 4-23.

CONFIG	Configuration	Standard
000	802 Interleavers	802.11a, 802.11n, 802.16e
001	Compact Bit Interleaver	DVB
010	Symbol Interleaver	DVB
011	Convolutional Interleaver	DVB

Table 4-23 Configurations of CONFIG signal

The MEMORY_CTRL signal is the W/R signal.

4.2.4.1.1 Control State Machine

This block is an FSM that controls the memory structure, and 8AGU1 & 8AGU2, has where configured on the Param_Memoy, and selected by the CONF_SELECT signal.

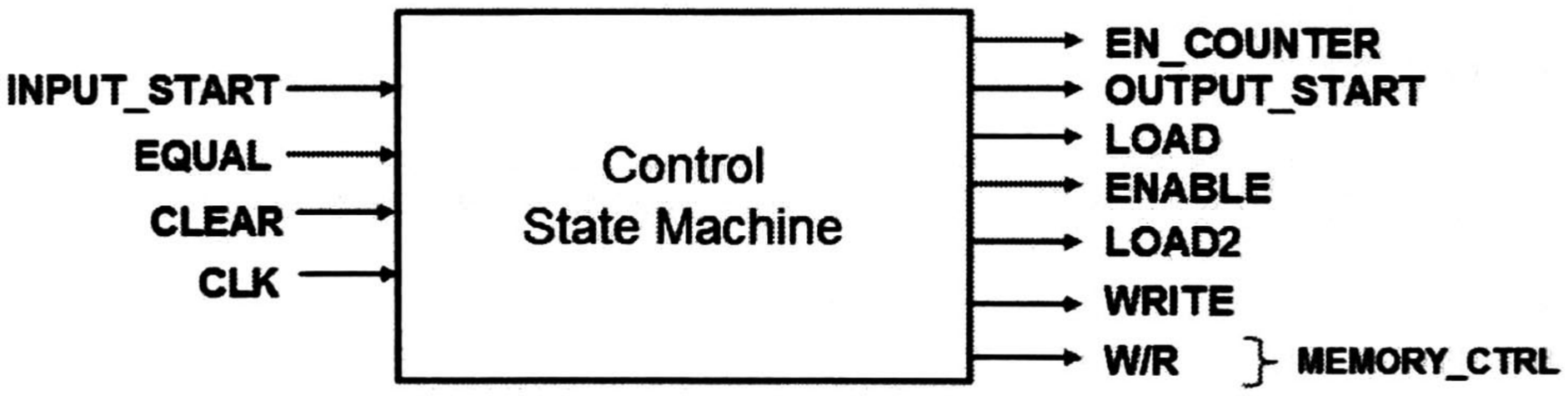


Figure 4-36 Input/output signal diagram for Control State Machine

The Figure 4-36 shows the input/outputs signals of the control state machine, and the Table 4-24 shows its description.

Input	Description	Comes From
INPUT_START	This signal starts the Control State Machine and indicate that the data is ready to be interleaved	Input Interface
EQUAL	This signal is asserted when the counter in the Control Manager reaches the count in CLK_BLOCK signal,	Comparator
CLEAR	Clear Signal	Input Interface
Output	**Description**	**Goes To**
EN_COUNTER	This signals enables the counter in Control Manager	Counter in Control Manager
OUTPUT_START	Output ready to be read signal	Output Interface
LOAD	Generic load for the Control Logic	Control Decoder

ENABLE	Generic enables of the Control Logic	Control Decoder
LOAD2	Load signal used to save the last state of the Convolutional interleaver	Control Decoder
WRITE	Defines if the control is in reading process or in write process.	Control Decoder
W/R	Defines when the memory is in writing or reading.	Memory Structure

Table 4-24 Input and output organization of Control State Machine

The Finite State Machine (FSM) type MOORE diagram is presented in Figure 4-37.if the input clear signal is asserted at any moment the FSM goes to Idle state.

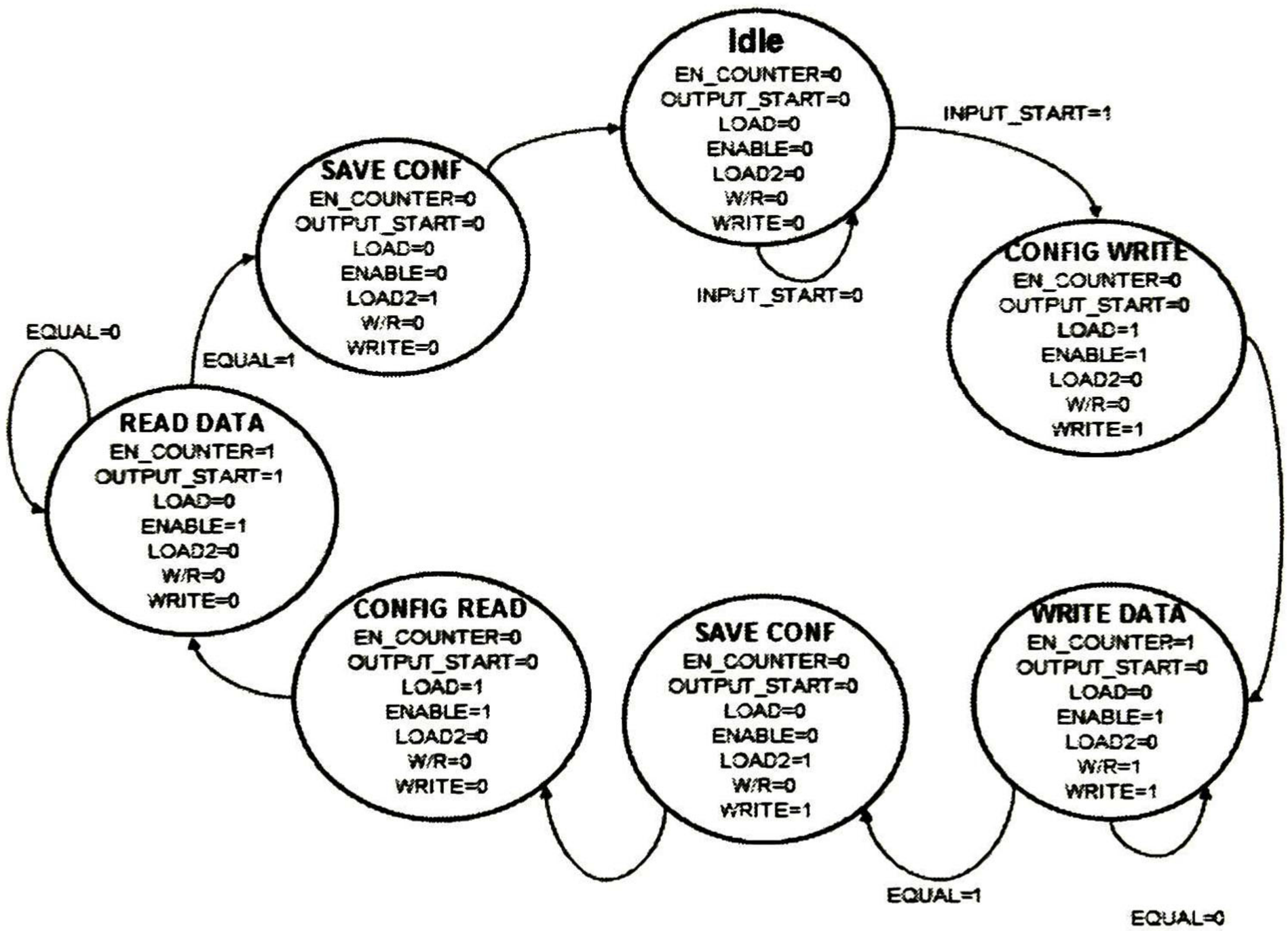


Figure 4-37 FSM of the Control State Machine

4.2.4.1.2 Control Decoder

This block decodes all the control signals of the MSI and control the wait habilitations, this take the information generated by the control state machine and rearranges the control to the selected interleaving/ deinterleaving scheme selected by the CONFIG signal as shown in Table 4-23 Configurations of CONFIG signal.

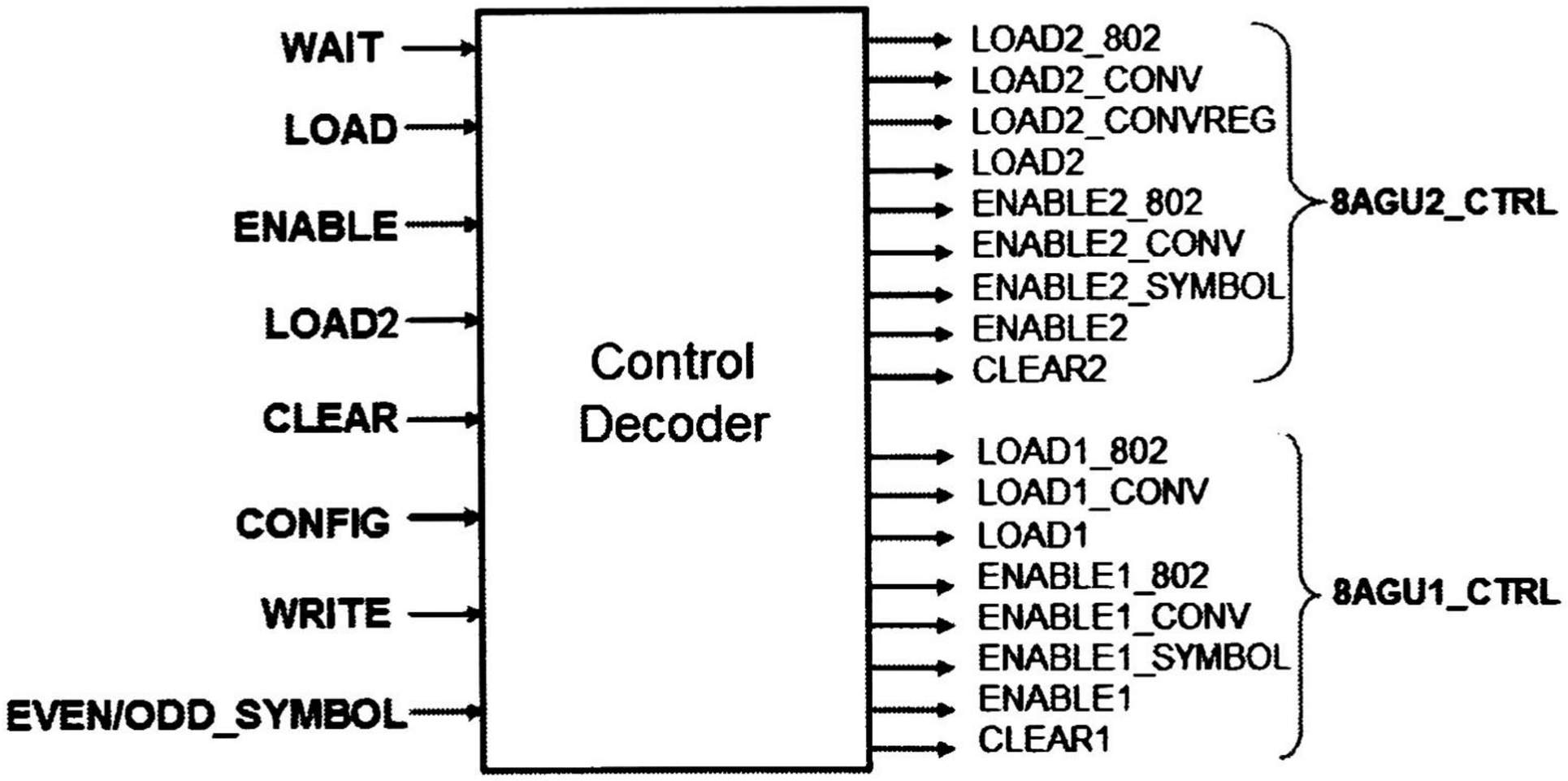


Figure 4-38 Input/output signal diagram for Control Decoder

The Figure 4-38 shows the input/outputs signals of the control decoder, and the Table 4-25 shows its description.

Input	Description	Comes From
WAIT	This signal disables the interleaver processing while is asserted	Input Interface
LOAD	Generic load for the Control Logic	Control Decoder
ENABLE	Generic enables of the Control Logic	Control Decoder
LOAD2	Load signal used to save the last state of the Convolutional interleaver	Control Decoder
CLEAR	Clear Signal	Input Interface
CONFIG	Configures one of the Interleaver schemes	Param Memory
WRITE	Defines if the control is in reading process or in write process.	Control Decoder
EVEN/ODD SYMBOL	Indicates if The ODFM symbol is Even or odd	Param Memory

Output	Description	Goes To
LOAD2_802	Load for AGU2(0) 802 configuration	AGU2(0)
LOAD2_CONV	Load for AGU2(0) convolutional configuration	AGU2(0)
LOAD2_CONVREG	Load for the Convolutional Register	AGU2(0)
LOAD2	Load for the AGU2(1..7)	AGU2(1..7)
ENABLE2_802	Enable for the AGU2(0) 802 configuration	AGU2(0)
ENABLE2_CONV	Enable for the AGU2(0) Convolutional configuration	AGU2(0)
ENABLE2_SYMBOL	Enable for the AGU2(0) Symbol configuration	AGU2(0)
ENABLE2	Enable for the AGU2(1..7)	AGU2(1..7)
CLEAR2	Clear for the AGU2	8AGU2
LOAD1_802	Load for AGU1(0) 802 configuration	AGU1(0)
LOAD1_CONV	Load for AGU1(0) convolutional configuration	AGU1(0)
LOAD1	Load for the AGU1(1..7)	AGU1(1..7)
ENABLE1_802	Enable for the AGU1(0) 802 configuration	AGU1(0)
ENABLE1_CONV	Enable for the AGU1(0) Convolutional configuration	AGU1(0)
ENABLE1_SYMBOL	Enable for the AGU1(0) Symbol configuration	AGU1(0)
ENABLE1	Enable for the AGU1(1..7)	AGU1(1..7)
CLEAR1	Clear for the AGU1	8AGU1

Table 4-25 Input and output organization of Control Decoder

The SLAC (Single Load After Clear) Block in Figure 4-39 is used to generate the LOAD1_CONV only once, after every clear, the SLAC has a flag bit that is cleared when the CLEAR signal is asserted and is set when the LOAD1_CONV signal is generated, when this flag is set the LOAD1_CONV is disabled.

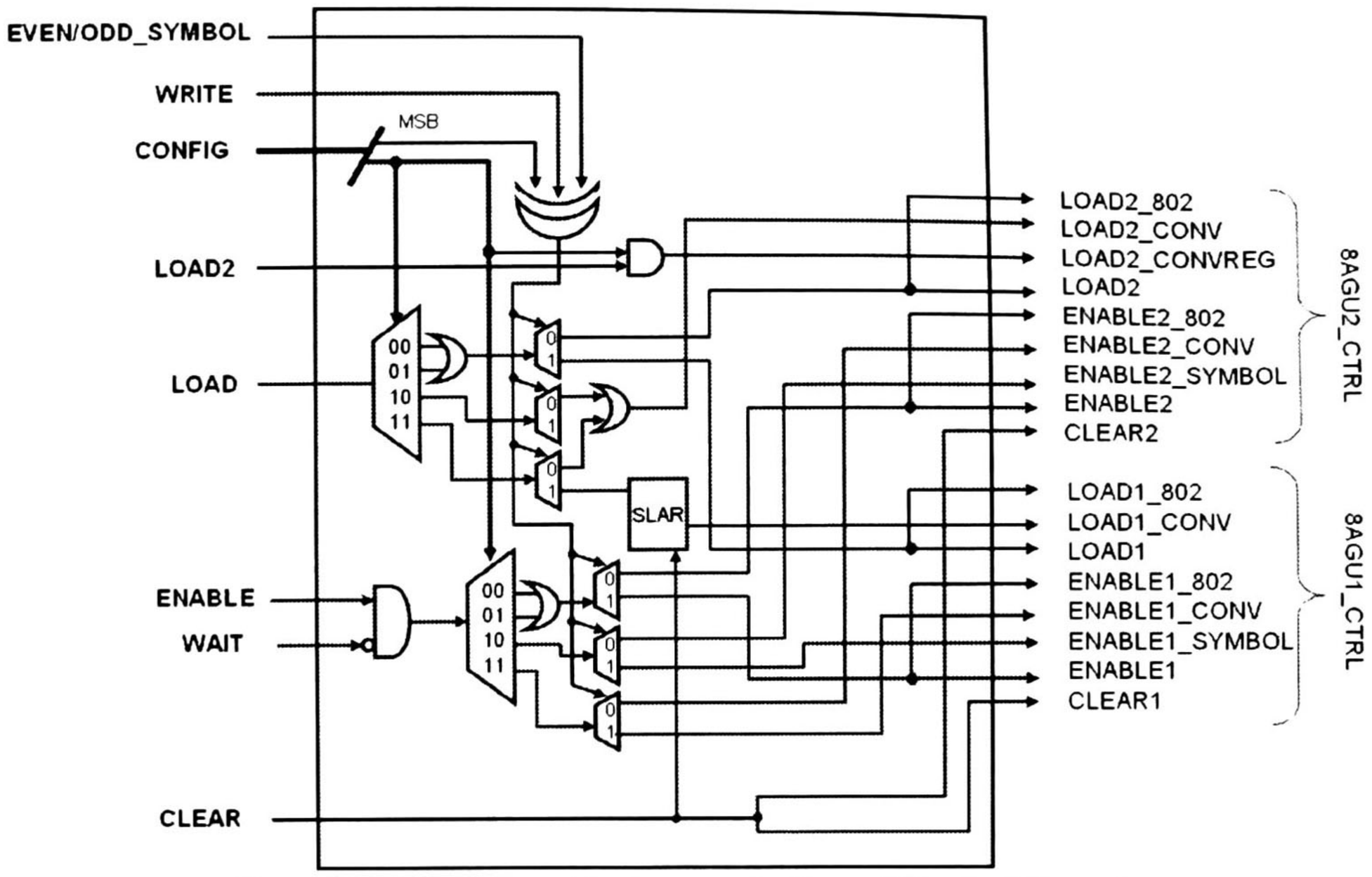


Figure 4-39 Hardware block organization of Control Decoder

4.2.4.2 Param Memory

This block is has the configuration parameters saved on the parallel shift registers, this configurations are the configurations on the fly, that are loaded by the μP Interface at the initialization.

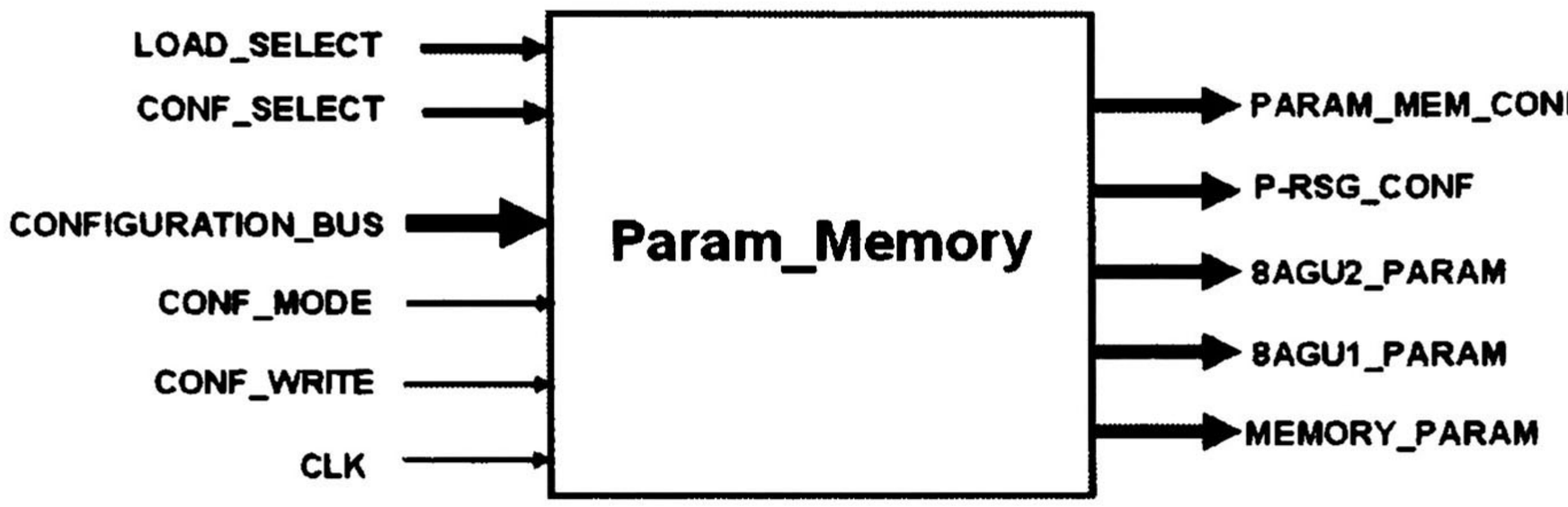


Figure 4-40 Input/output signal diagram for Param_Memory

The Figure 4-40 shows the input/outputs signals of the Param_Memory, and the Table 4-26 shows its description.

Input	Description	Comes From
CONF_SELECT[1:0]	This signal is the selector for the configurations on the fly saved in the Parallel Shift Registers	Input Interface
CONFIGURATION_BUS[7:0]	Input data to configure the Parallel Shift Registers	μP Interface
CONF_MODE	Tells to the Param Memory to change from Configuration Mode to Operation Mode	μP Interface
CONF_WRITE	Tells that data is ready to be written in the Parallel Shift Register	μP Interface
Output	**Description**	**Goes To**
PARAM_MEM_CONF	Parameters for the Control Manager	Control Manager
P-RSG_CONF	Parameters for the 8AGU1	8AGU1
8AGU2_PARAM	Parameters for the 8AGU1	8AGU1
8AGU1_PARAM	Parameters for the 8AGU2	8AGU2
MEMORY_PARAM	Parameters for the Memory Structure	Memory Structure

Table 4-26 Input and output organization of Param Memory

The Figure 4-41 shows the organization of the hardware blocks of the Param_Memory

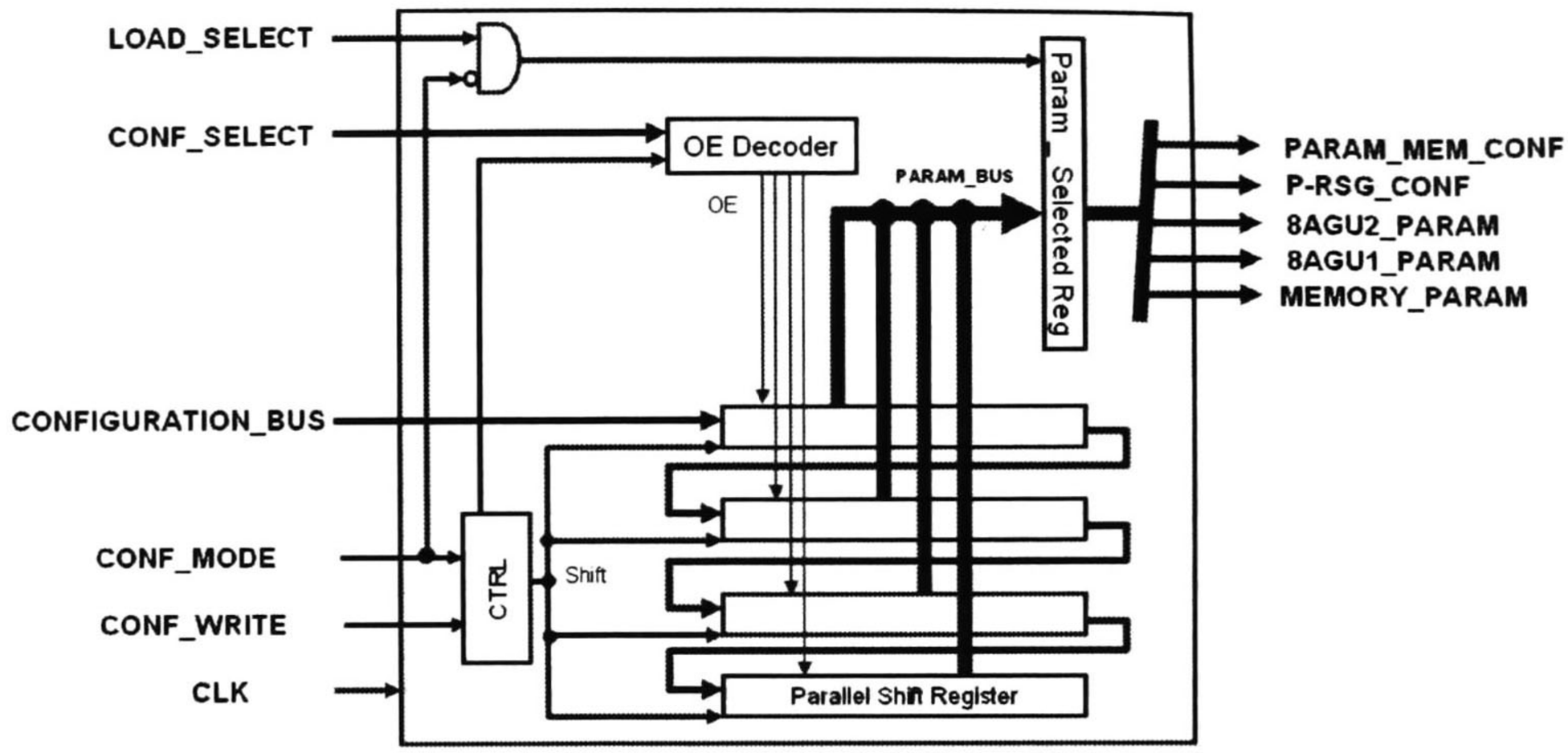


Figure 4-41 Hardware block organization of Param Memory

For the configuration mode the µP Interface starts by putting the CONF_MODE signal in 1. The CONFIGURATION_BUS is ready when the CONF_WRITE signal is asserted this tell to the parallel shift registers to start shifting the data on the CONFIGURATION_BUS.

In Configuration Mode the OE Decoder disables all the OE.

When all the parameters are shifted into the correct position, and the CONF_MODE signal is 0, this ends the Configuration Mode. At this time the MSI is ready to implement any Configuration Saved in the Configuration Mode by selecting it by the CONF_SELECT signal, this enables one of the 4 Parallel Shift Registers.

The OE decoder is an output enables decoder that decodes the CONF_SELECT signal and generates the OE selecting one of the Parallel Shift Register or better said configurations on the fly.

The selected configuration takes place on the PARAM_BUS and is saved in the Param_Selected reg that makes this configuration stable for the MSI, and allows the Configuration Mode while the MSI is working on the interleaving of a data block.

4.3 Architecture Overview

This part is an overview of the MSI architecture in terms of black box sub-circuits. The Figure 4-42 shows the MSI architecture overview. We can see two levels of sub blocks inside the MSI black box. The references to subparts of the system are to figure or table index found in Figure 4-43.

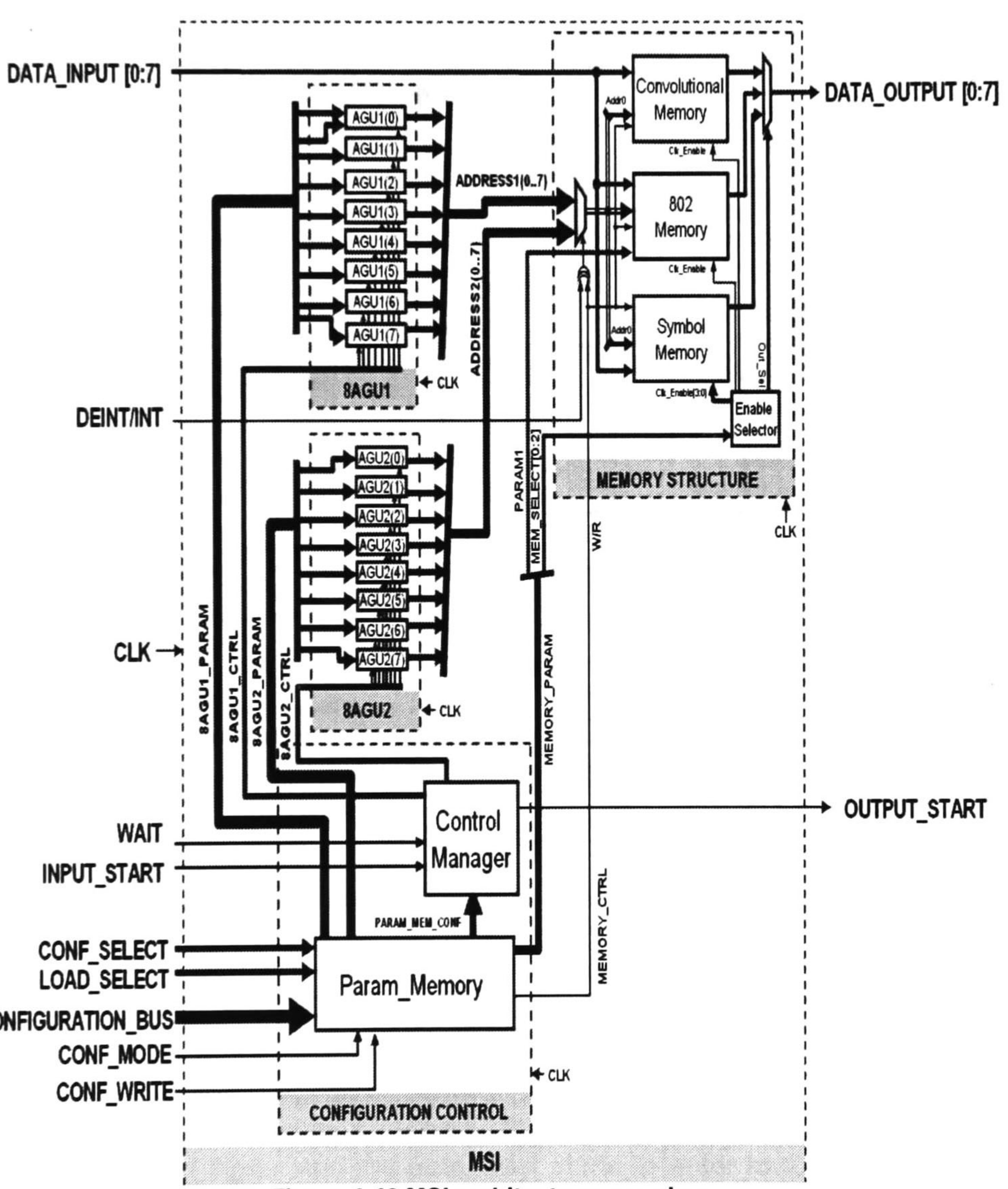


Figure 4-42 MSI architecture overview

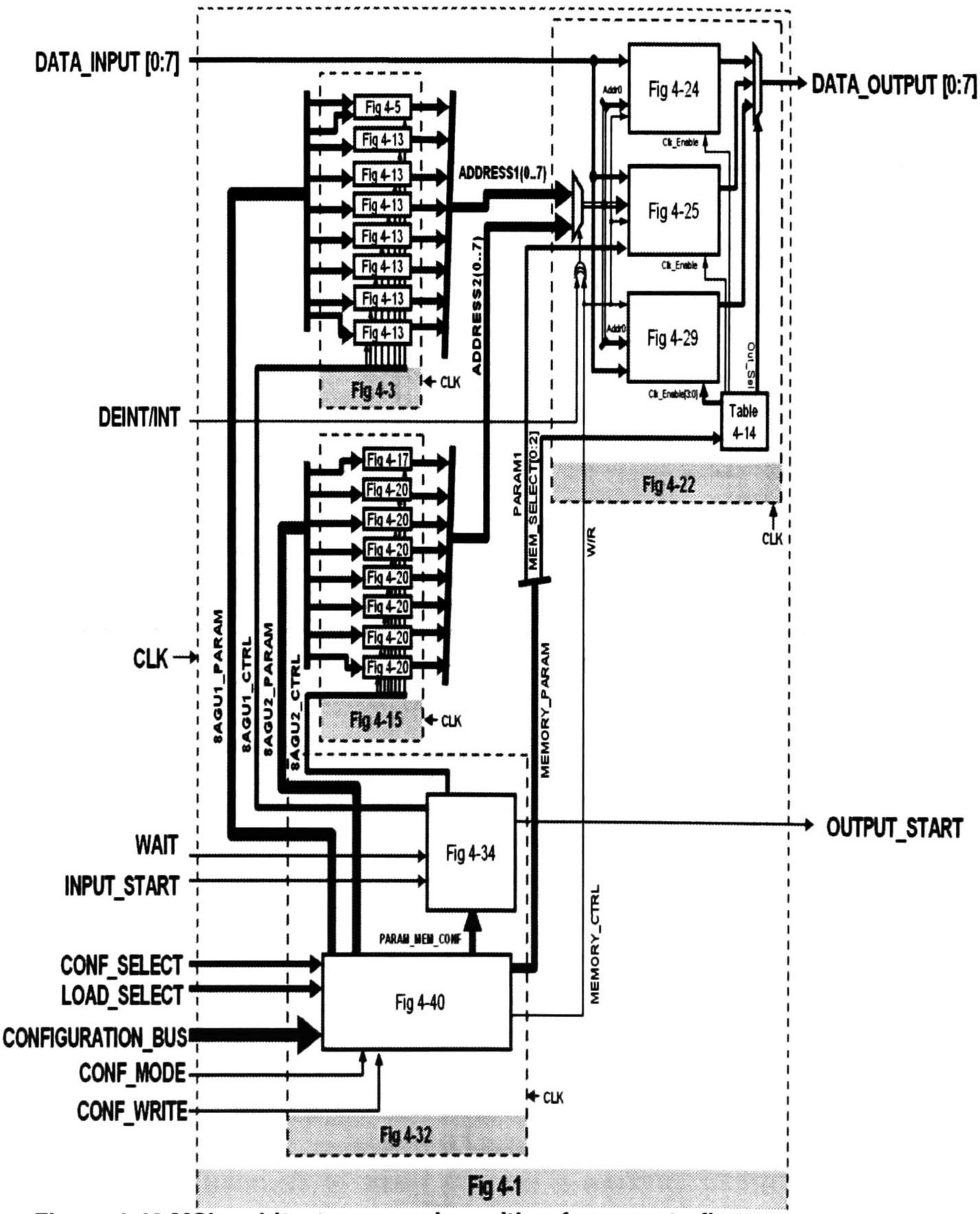


Figure 4-43 MSI architecture overview with references to figure or table index

4.4 Testing Methodology

A testing methodology was designed to ensure that the MSI is having good results.

At this project we need to test a lot of configurations. The testing process needs to be automatic to reduce the time to generate the input, output and configuration files, this is done by the use of Matlab.

Matlab is used to generate the golden models. The golden models are ideal results generated using the formulas of the standards through programming it on Matlab. All the interleavers were programmed according to the standard formulas or configurations. The Figure 4-44 shows the testing methodology architecture for the MSI. Matlab is used to generate the configuration file, input file and expected output files. The input file is randomly generated. These files are loaded in the test bench and the configuration and input files are loaded to the MSI through the test bench. When the output of the MSI is ready the test bench starts to compare the generated output of the MSI versus the expected output of the matlab output file writing a report of the errors on the output. The Report File contains the errors and the location of the errors on the output sequence.

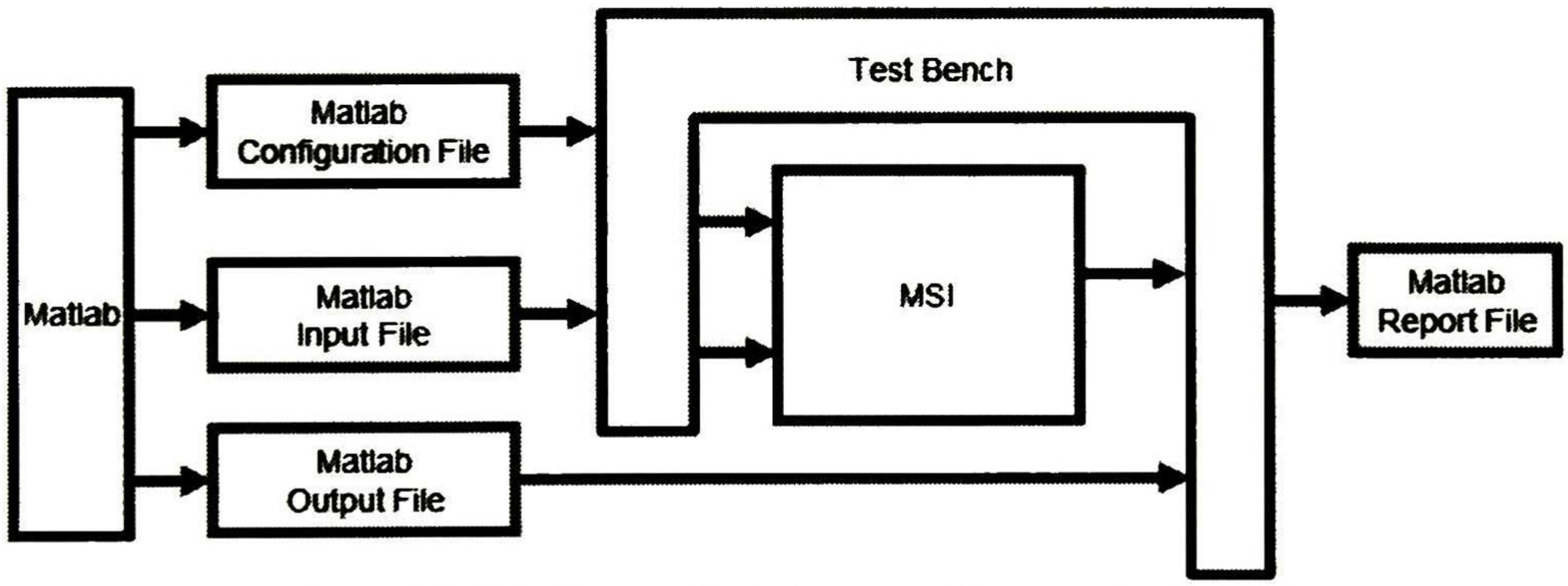


Figure 4-44 Testing methodology architecture for the MSI

A screen shot taken from the Questasim simulator Figure 4-45, shows the simulation results of the input process and output process for the 802.11a configuration 64QAM.

The Figure 4-45 shows the interleaving process in two separated process, data input process and data output process. The input process starts with logic "1" on the input_start signal and the data input starts to enter to the MSI, after all the data input is written the output is ready to go on, then the data output process starts.

The zoom in has the OUTPUT_TO_COMPARE, numErrors & exp_OUTA signals. The OUTPUT_TO_COMPARE signal is the generated output by the

MSI that goes to the test bench, the exp_OUTA is the expected output data generated by Matlab and saved on the Matlab output file. The numErrors signal is an error count signal of the test bench and counts every mismatch between the output of the MSI and the expected output generated by Matlab. As we can see on the zoom in of the Figure 4-45 the output and the expected output are the same, so the error count keeps the value of 0, this indicate that there is no error on the bits and that the simulation behaves has Matlab predicted.

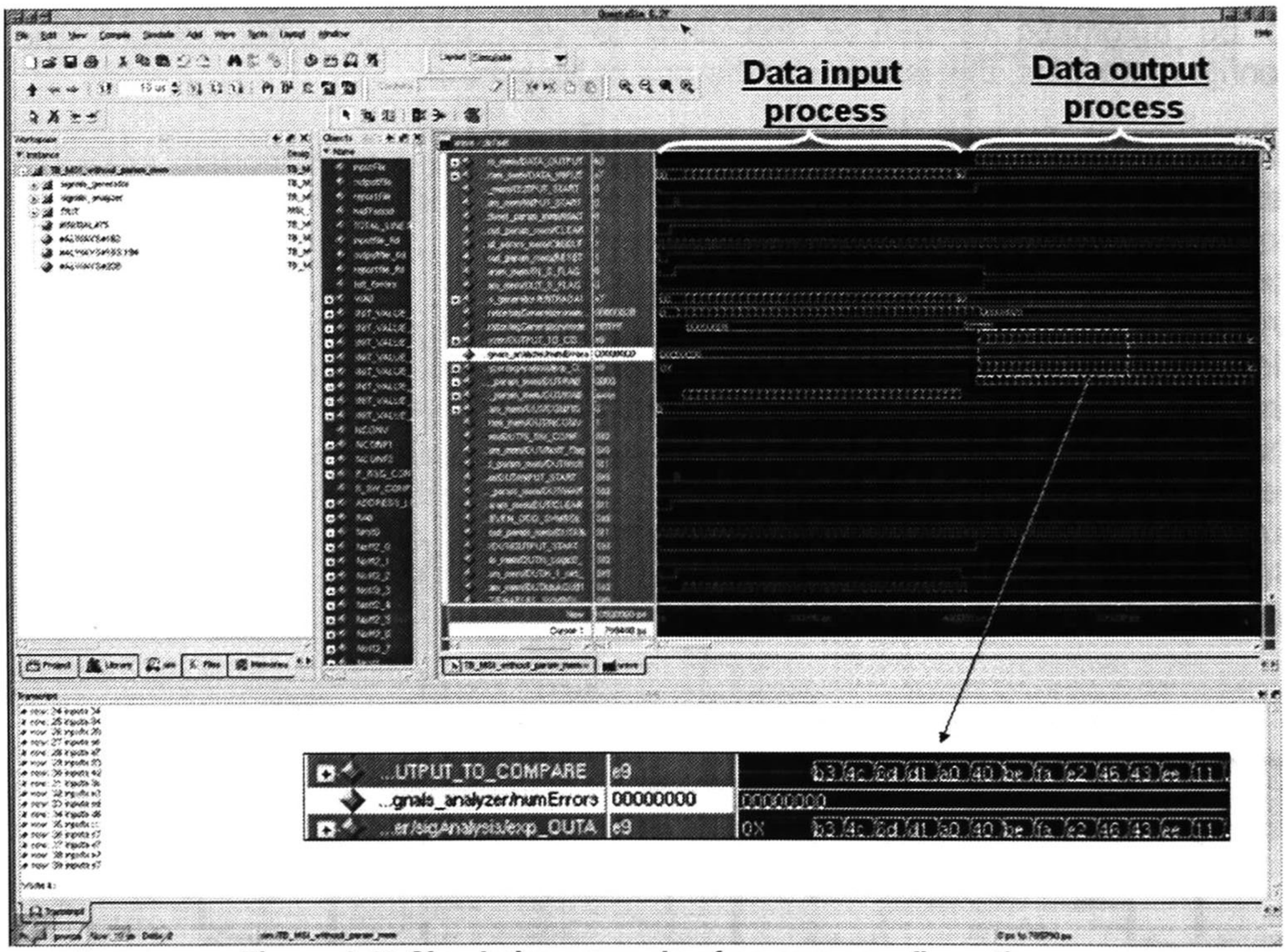


Figure 4-45 Simulation example of no error configuration

The testing methodology was designed to fully review the MSI architecture and to extract all the configuration parameters for the different interleaver configurations. The test was applied to every one of these configurations, the test plan is shown in Table 4-27.

#test	ID	Description	result
1	11a01	802.11a for BPSK test	ok
2	11a02	802.11a for QPSK test	ok
3	11a03	802.11a for 16QAM test	ok
4	11a04	802.11a for 64QAM test	ok
5	16e 01	802.16e for QPSK (NCBPS=96)test	ok
6	16e 02	802.16e for QPSK (NCBPS=192)test	ok
7	16e 03	802.16e for QPSK (NCBPS=288)test	ok
8	16e 04	802.16e for QPSK (NCBPS=384)test	ok
9	16e 05	802.16e for QPSK (NCBPS=480)test	ok
10	16e 06	802.16e for QPSK (NCBPS=576)test	ok
11	16e 07	802.16e for 16QAM (NCBPS=384)test	ok
12	16e 08	802.16e for 16QAM (NCBPS=576)test	ok

13	16e 09	802.16e for 64QAM (NCBPS=288)test	ok
14	16e 10	802.16e for 64QAM (NCBPS=576)test	ok
15	11n01	802.11n for BPSK (20Mhz iss=0)	ok
16	11n02	802.11n for BPSK (20Mhz iss=1)	ok
17	11n03	802.11n for BPSK (20Mhz iss=3)	ok
18	11n04	802.11n for BPSK (20Mhz iss=4)	ok
19	11n05	802.11n for QPSK (20Mhz iss=0)	ok
20	11n06	802.11n for QPSK (20Mhz iss=1)	ok
21	11n07	802.11n for QPSK (20Mhz iss=3)	ok
22	11n08	802.11n for QPSK (20Mhz iss=4)	ok
23	11n09	802.11n for 16QAM (20Mhz iss=0)	ok
24	11n10	802.11n for 16QAM (20Mhz iss=1)	ok
25	11n11	802.11n for 16QAM (20Mhz iss=3)	ok
26	11n12	802.11n for 16QAM (20Mhz iss=4)	ok
27	11n13	802.11n for 64QAM (20Mhz iss=0)	ok
28	11n14	802.11n for 64QAM (20Mhz iss=1)	ok
29	11n15	802.11n for 64QAM (20Mhz iss=3)	ok
30	11n16	802.11n for 64QAM (20Mhz iss=4)	ok
31	11n17	802.11n for BPSK (40Mhz iss=0)	ok
32	11n18	802.11n for BPSK (40Mhz iss=1)	ok
33	11n19	802.11n for BPSK (40Mhz iss=3)	ok
34	11n20	802.11n for BPSK (40Mhz iss=4)	ok
35	11n21	802.11n for QPSK (40Mhz iss=0)	ok
36	11n22	802.11n for QPSK (40Mhz iss=1)	ok
37	11n23	802.11n for QPSK (40Mhz iss=3)	ok
38	11n24	802.11n for QPSK (40Mhz iss=4)	ok
39	11n25	802.11n for 16QAM (40Mhz iss=0)	ok
40	11n26	802.11n for 16QAM (40Mhz iss=1)	ok
41	11n27	802.11n for 16QAM (40Mhz iss=3)	ok
42	11n28	802.11n for 16QAM (40Mhz iss=4)	ok
43	11n29	802.11n for 64QAM (40Mhz iss=0)	ok
44	11n30	802.11n for 64QAM (40Mhz iss=1)	ok
45	11n31	802.11n for 64QAM (40Mhz iss=3)	ok
46	11n32	802.11n for 64QAM (40Mhz iss=4)	ok
47	conv01	Convolutional	ok
48	conv02	Convolutional	ok
49	conv03	Convolutional	ok
50	symb01	DVB Symbol 2k	ok
51	symb02	DVB Symbol 2k	ok
52	symb03	DVB Symbol 2k	ok
53	symb04	DVB Symbol 8k	ok
54	symb05	DVB Symbol 8k	ok
55	symb06	DVB Symbol 8k	ok
56	bit01	bit interleaver 2	ok
57	bit02	bit interleaver 4	ok
58	bit03	bit interleaver 6	ok

Table 4-27 Test plan

4.5 Configuration Modes

The configuration parameters configure the MSI to implement any of the interleavers/deinterleavers used in the 802.11a, 802.11n, 802.16e and DVB standards.

To go to configuration mode the CONF_MODE signal needs to go to 1, and when the configuration parameters are ready to read the CONF_WRITE signal shall go to 1, this tells the MSI to start writing the incoming configuration parameters from the CONFIGURATION_BUS into the internal Param Memory, This configuration process is illustrated in Figure 4-46, when all the configuration parameters are written, the CONF_MODE and CONF_WRITE signals shall go to 0.

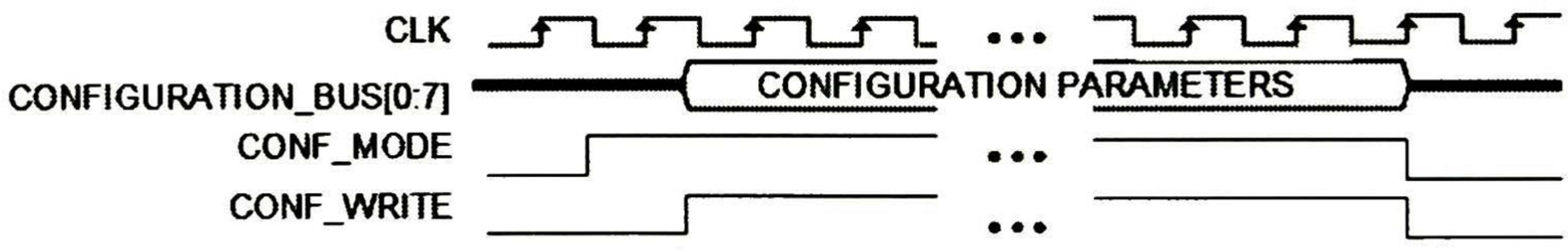


Figure 4-46 Example how to configure MSI for Configuration process

The parameters to configure the MSI shall be organized by groups of 8 bits, starting from the LSB of the CLK_BLOCK, and finishing with the MSB of the S_SW_CONF1, following the order that the Table 4-28 defines. An illustration of this process is shown in Figure 4-47.

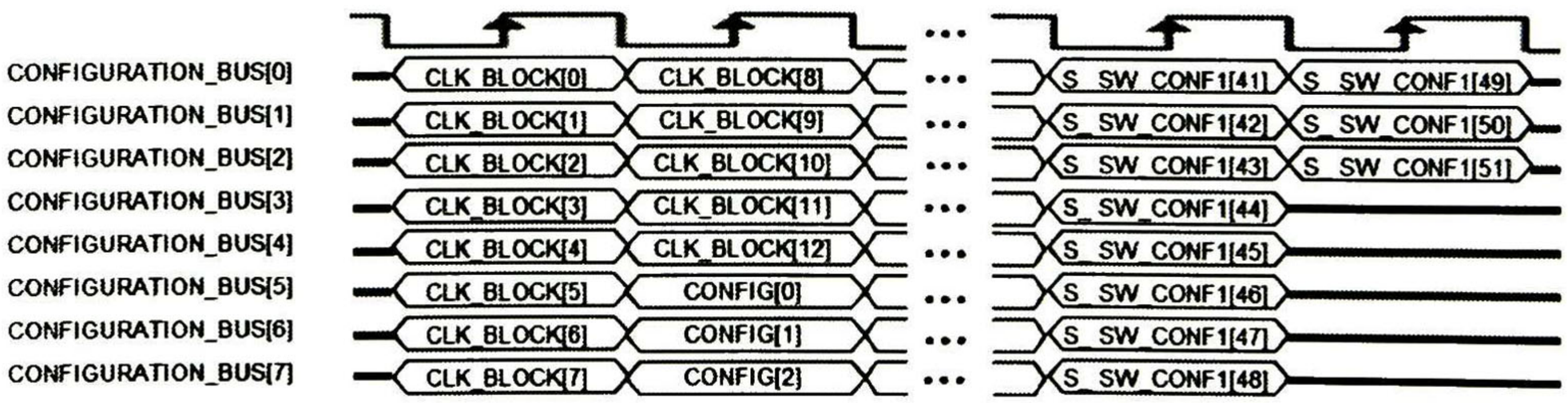


Figure 4-47 Configuration parameters on the CONFIGURATION BUS

A list of the configuration parameters that are needed to configure the MSI and his description is in the Table 4-28. These parameters need to be downloaded in the provided order.

BUS	PARAMETER	Size (bits)	DESCRIPTION
PARAM_MEM_CONF	CLK_BLOCK	13	Number of CLK cycles to implement the process the data
	CONFIG	3	Configuration selection signal to configure the Control Decoder
	EVEN/ODD_SYMBOL	1	Used in symbol Interleaving, when 1 the OFDM is even else is ODD
8AGU1_PARAM	NCONV	1	Enable of extra increment on the overflow
	INITIAL VALUE(0)	13	Initial value of the AGU1(0)
	INITIAL VALUE(1)	10	Initial value of the AGU1(1)
	INITIAL VALUE(2)	10	Initial value of the AGU1(2)
	INITIAL VALUE(3)	10	Initial value of the AGU1(3)
	INITIAL VALUE(4)	10	Initial value of the AGU1(4)
	INITIAL VALUE(5)	10	Initial value of the AGU1(5)
	INITIAL VALUE(6)	10	Initial value of the AGU1(6)
	INITIAL VALUE(7)	10	Initial value of the AGU1(7)
	NCONF1	10	Value of the increment in the Adder
	NCONF2	13	Modulus of the Adder
8AGU2_PARAM	Nrot(0)	13	Initial value of counter Nconf2 in AGU2(0)
	Nrot(1)	10	Initial value of counter Nconf2 in AGU2(1)
	Nrot(2)	10	Initial value of counter Nconf2 in AGU2(2)
	Nrot(3)	10	Initial value of counter Nconf2 in AGU2(3)
	Nrot(4)	10	Initial value of counter Nconf2 in AGU2(4)
	Nrot(5)	10	Initial value of counter Nconf2 in AGU2(5)
	Nrot(6)	10	Initial value of counter Nconf2 in AGU2(6)
	Nrot(7)	10	Initial value of counter Nconf2 in AGU2(7)
	Noff2(0)	6	Initial value of counter Nconf3 in AGU2(0)
	Noff2(1)	6	Initial value of counter Nconf3 in AGU2(1)
	Noff2(2)	6	Initial value of counter Nconf3 in AGU2(2)
	Noff2(3)	6	Initial value of counter Nconf3 in AGU2(3)
	Noff2(4)	6	Initial value of counter Nconf3 in AGU2(4)
	Noff2(5)	6	Initial value of counter Nconf3 in AGU2(5)
	Noff2(6)	6	Initial value of counter Nconf3 in AGU2(6)
	Noff2(7)	6	Initial value of counter Nconf3 in AGU2(7)
	NCONF2	13	Modulus of the NCONF2 counter
	NCONF3	6	Modulus of the NCONF3counter
	ConfigTable	6*2	Values of the selector of the mux
	NOFF_Flag	1	Enables the shift of the configuration table on the overflow
	IncTable	10*6*2	Values of the tables of increments Inc_Table0 Inc_Table1
	INC8	1	Enable for increment of 8, if not enabled the increment is 1
	OVER_ADD_8	8	Initial increment in the Overflow of the counter Nconf3
MEMORY_PARAM	MEM_SELECT	3	Enables the memory configuration Selected
	PARAM1	8	Value for the ADU
SYMBOL_PARAM	P-RSG_CONF	12	Interconnection vector
	ADDRESS_LIMIT	12	Address limit in Symbol Interleaver
	S_ SW_CONF0	4*13	Interconnection vector for S_SW0
	S_ SW_CONF1	4*13	Interconnection vector for S_SW1
	Total of configuration bits	659	

Table 4-28 Configuration parameter list and description for MSI

4.5.1 Configuration for 802.11a

The configurations for the 802.11a standard have to load the next parameters on the control manager. The bytes need to be loaded as the list in Table 4-29.

BUS	PARAMETER	CONFIGURATION(dec value)			
		BPSK	QPSK	16QAM	64QAM
PARAM_MEM_CONF	CLK_BLOCK	48/8=6	96/8=12	192/8=24	288/8=36
	CONFIG	0	0	0	0
	EVEN/ ODD_SYMBOL	0	0	0	0
8AGU1_PARAM	NCONV	0	0	0	0
	INITIAL VALUE(0)	0	0	0	0
	INITIAL VALUE(1)	3	6	12	18
	INITIAL VALUE(2)	6	12	24	36
	INITIAL VALUE(3)	9	18	36	54
	INITIAL VALUE(4)	12	24	48	72
	INITIAL VALUE(5)	15	30	60	90
	INITIAL VALUE(6)	18	36	72	108
	INITIAL VALUE(7)	21	42	84	126
	NCONF1	3*8 = 24	6*8 = 48	12*8 = 96	18*8 = 144
	NCONF2	48	96	192	288
8AGU2_PARAM	Nrot(0)	0	0	0	0
	Nrot(1)	1	1	1	1
	Nrot(2)	2	2	2	2
	Nrot(3)	3	3	3	3
	Nrot(4)	4	4	4	4
	Nrot(5)	5	5	5	5
	Nrot(6)	6	6	6	6
	Nrot(7)	7	7	7	7
	Noff2(0)	0	0	0	0
	Noff2(1)	0	0	1	1
	Noff2(2)	0	0	2	2
	Noff2(3)	0	0	3	3
	Noff2(4)	0	0	4	4
	Noff2(5)	0	0	5	5
	Noff2(6)	0	0	6	6
	Noff2(7)	0	0	7	7
	NCONF2	48	96	192	288
	NCONF3	8	8	12	18
	ConfigTable	0;0;0;0;0;0	0;0;0;0;0;0	2;3;2;3;2;3	0;2;3;0;2;3
	NOFF_Flag	0	0	0	0
	IncTable	[0;0;0;0;0;0] [0;0;0;0;0;0]	[0;0;0;0;0;0] [0;0;0;0;0;0]	[-1; 1; -1 ; 1 ; -1;1] [0;0;0;0;0;0]	[-1; -1; 2; 1;-1;2] [-2; 1; 1; -2; 1; 1]
	INC8	1	1	1	1
	OVER_ADD	0	0	0	0
MEMORY_PARAM	MEM_SELECT	2	2	2	2
	PARAM1	24	48	48	144
SYMBOL_PARAM	P-RSG_CONF	X	X	X	X
	ADDRESS_LIMIT	X	X	X	X
	S_ SW_CONF0	X	X	X	X
	S_ SW_CONF1	X	X	X	X

Table 4-29 Configuration parameters for 802.11a

4.5.2 Configuration for 802.11n

This standard has configurations for 20MHz and 40Mhz, and the combination of iss from 0 to 3. These configurations are presented in 8 tables listed bellow.

BUS	PARAMETER	CONFIGURATION(dec value) 20MHz iss=0			
		BPSK	QPSK	16QAM	64QAM
PARAM_MEM_CONF	CLK_BLOCK	52/8=7	104/8=13	208/8=26	312/8=39
	CONFIG	0	0	0	0
	EVEN/ODD_SYMBOL	0	0	0	0
8AGU1_PARAM	NCONV	0	0	0	0
	INITIAL VALUE(0)	0	0	0	0
	INITIAL VALUE(1)	4	8	16	24
	INITIAL VALUE(2)	8	16	32	48
	INITIAL VALUE(3)	12	24	48	72
	INITIAL VALUE(4)	16	32	64	96
	INITIAL VALUE(5)	20	40	80	120
	INITIAL VALUE(6)	24	48	96	144
	INITIAL VALUE(7)	28	56	112	168
	NCONF1	4*8 = 32	8*8 = 64	16*8 = 128	24*8 = 192
	NCONF2	52	104	208	312
8AGU2_PARAM	Nrot(0)	0	0	0	0
	Nrot(1)	1	1	1	1
	Nrot(2)	2	2	2	2
	Nrot(3)	3	3	3	3
	Nrot(4)	4	4	4	4
	Nrot(5)	5	5	5	5
	Nrot(6)	6	6	6	6
	Nrot(7)	7	7	7	7
	Noff2(0)	0	0	0	0
	Noff2(1)	0	0	1	1
	Noff2(2)	0	0	2	2
	Noff2(3)	0	0	3	3
	Noff2(4)	0	0	4	4
	Noff2(5)	0	0	5	5
	Noff2(6)	0	0	6	6
	Noff2(7)	0	0	7	7
	NCONF2	52	104	208	312
	NCONF3	8	8	16	24
	ConfigTable	0;0;0;0;0;0	0;0;0;0;0;0	2;3;2;3;2;3	0;2;3;0;2;3
	NOFF_Flag	0	0	0	0
	IncTable	[0;0;0;0;0;0] [0;0;0;0;0;0]	[0;0;0;0;0;0] [0;0;0;0;0;0]	[-1; 1; -1 ; 1 ; -1;1] [0;0;0;0;0;0]	[2; -1; -1; 2; -1;-1] [1; 1; -2; 1; 1;-2]
	INC8	1	1	1	1
	OVER_ADD				
MEMORY_PARAM	MEM_SELECT	2	2	2	2
	PARAM1	16	16	16	48
SYMBOL_PARAM	P-RSG_CONF	X	X	X	X
	ADDRESS_LIMIT	X	X	X	X
	S_ SW_CONF0	X	X	X	X
	S_ SW_CONF1	X	X	X	X

Table 4-30 Configuration parameters for 802.11n 20MHz iss=0

BUS	PARAMETER	CONFIGURATION(dec value) 20MHz iss=1			
		-BPSK	-QPSK	16QAM	64QAM
PARAM_MEM_CONF	CLK_BLOCK	52/8=7	104/8=13	208/8=26	312/8=39
	CONFIG	0	0	0	0
	EVEN/ ODD_SYMBOL	0	0	0	0
8AGU1_PARAM	NCONV	0	0	0	0
	INITIAL VALUE(0)	0	0	0	0
	INITIAL VALUE(1)	4	8	16	24
	INITIAL VALUE(2)	8	16	32	48
	INITIAL VALUE(3)	12	24	48	72
	INITIAL VALUE(4)	16	32	64	96
	INITIAL VALUE(5)	20	40	80	120
	INITIAL VALUE(6)	24	48	96	144
	INITIAL VALUE(7)	28	56	112	168
	NCONF1	4*8 = 32	8*8 = 64	16*8 = 128	24*8 = 192
	NCONF2	52	104	208	312
8AGU2_PARAM	Nrot(0)	0	0	88	132
	Nrot(1)	1	1	89	133
	Nrot(2)	2	2	90	134
	Nrot(3)	3	3	91	135
	Nrot(4)	4	4	92	136
	Nrot(5)	5	5	93	137
	Nrot(6)	6	6	94	138
	Nrot(7)	7	7	95	139
	Noff2(0)	0	0	24	12
	Noff2(1)	0	0	25	13
	Noff2(2)	0	0	26	14
	Noff2(3)	0	0	27	15
	Noff2(4)	0	0	28	16
	Noff2(5)	0	0	29	17
	Noff2(6)	0	0	30	18
	Noff2(7)	0	0	31	19
	NCONF2	52	104	208	312
	NCONF3	8	8	16	24
	ConfigTable	0;0;0;0;0;0	0;0;0;0;0;0	3;2;3;2;3;2	3;0;2;3;0;2
	NOFF_Flag	0	0	0	0
	IncTable	[0;0;0;0;0;0] [0;0;0;0;0;0]	[0;0;0;0;0;0] [0;0;0;0;0;0]	[-1; 1; -1 ; 1 ; -1;1] [0;0;0;0;0;0]	[2; -1; -1; 2; -1;-1] [1; 1; -2; 1; 1;-2]
	INC8	1	1	1	1
	OVER_ADD				
MEMORY_PARAM	MEM_SELECT	2	2	2	2
	PARAM1	16	16	16	48
SYMBOL_PARAM	P-RSG_CONF	X	X	X	X
	ADDRESS_LIMIT	X	X	X	X
	S_ SW_CONF0	X	X	X	X
	S_ SW_CONF1	X	X	X	X

Table 4-31 Configuration parameters for 802.11n 20MHz iss=1

BUS	PARAMETER	CONFIGURATION(dec value) 20MHz iss=2			
		-BPSK	-QPSK	16QAM	64QAM
PARAM_MEM_CONF	CLK_BLOCK	52/8=7	104/8=13	208/8=26	312/8=39
	CONFIG	0	0	0	0
	EVEN/ ODD_SYMBOL	0	0	0	0
8AGU1_PARAM	NCONV	0	0	0	0
	INITIAL VALUE(0)	0	0	0	0
	INITIAL VALUE(1)	4	8	16	24
	INITIAL VALUE(2)	8	16	32	48
	INITIAL VALUE(3)	12	24	48	72
	INITIAL VALUE(4)	16	32	64	96
	INITIAL VALUE(5)	20	40	80	120
	INITIAL VALUE(6)	24	48	96	144
	INITIAL VALUE(7)	28	56	112	168
	NCONF1	4*8 = 32	8*8 = 64	16*8 = 128	24*8 = 192
	NCONF2	52	104	208	312
8AGU2_PARAM	Nrot(0)	0	0	44	66
	Nrot(1)	1	1	45	67
	Nrot(2)	2	2	46	68
	Nrot(3)	3	3	47	69
	Nrot(4)	4	4	48	70
	Nrot(5)	5	5	49	71
	Nrot(6)	6	6	50	72
	Nrot(7)	7	7	51	73
	Noff2(0)	0	0	12	18
	Noff2(1)	0	0	13	19
	Noff2(2)	0	0	14	20
	Noff2(3)	0	0	15	21
	Noff2(4)	0	0	16	22
	Noff2(5)	0	0	17	23
	Noff2(6)	0	0	18	24
	Noff2(7)	0	0	19	25
	NCONF2	52	104	208	312
	NCONF3	8	8	16	24
	ConfigTable	0;0;0;0;0;0	0;0;0;0;0;0	2;3;2;3;2;3	3;0;2;3;0;2
	NOFF_Flag	0	0	0	0
	IncTable	[0;0;0;0;0;0] [0;0;0;0;0;0]	[0;0;0;0;0;0] [0;0;0;0;0;0]	[-1; 1; -1 ; 1 ; -1;1] [0;0;0;0;0;0]	[2; -1; -1; 2; -1;-1] [1; 1; -2; 1; 1;-2]
	INC8	1	1	1	1
	OVER_ADD				
MEMORY_PARAM	MEM_SELECT	2	2	2	2
	PARAM1	16	16	16	48
SYMBOL_PARAM	P-RSG_CONF	X	X	X	X
	ADDRESS_LIMIT	X	X	X	X
	S_ SW_CONF0	X	X	X	X
	S_ SW_CONF1	X	X	X	X

Table 4-32 Configuration parameters for 802.11n 20MHz iss=2

BUS	PARAMETER	CONFIGURATION(dec value) 20MHz iss=3			
		-BPSK	-QPSK	16QAM	64QAM
PARAM_MEM_CONF	CLK_BLOCK	52/8=7	104/8=13	208/8=26	312/8=39
	CONFIG	0	0	0	0
	EVEN/ ODD_SYMBOL	0	0	0	0
8AGU1_PARAM	NCONV	0	0	0	0
	INITIAL VALUE(0)	0	0	0	0
	INITIAL VALUE(1)	4	8	16	24
	INITIAL VALUE(2)	8	16	32	48
	INITIAL VALUE(3)	12	24	48	72
	INITIAL VALUE(4)	16	32	64	96
	INITIAL VALUE(5)	20	40	80	120
	INITIAL VALUE(6)	24	48	96	144
	INITIAL VALUE(7)	28	56	112	168
	NCONF1	4*8 = 32	8*8 = 64	16*8 = 128	24*8 = 192
	NCONF2	52	104	208	312
8AGU2_PARAM	Nrot(0)	0	0	132	198
	Nrot(1)	1	1	133	199
	Nrot(2)	2	2	134	200
	Nrot(3)	3	3	135	201
	Nrot(4)	4	4	136	202
	Nrot(5)	5	5	137	203
	Nrot(6)	6	6	138	204
	Nrot(7)	7	7	139	205
	Noff2(0)	0	0	4	6
	Noff2(1)	0	0	5	7
	Noff2(2)	0	0	6	8
	Noff2(3)	0	0	7	9
	Noff2(4)	0	0	8	10
	Noff2(5)	0	0	9	11
	Noff2(6)	0	0	10	12
	Noff2(7)	0	0	11	7
	NCONF2	52	104	208	312
	NCONF3	8	8	16	24
	ConfigTable	0;0;0;0;0;0	0;0;0;0;0;0	2;3;2;3;2;3	3;0;2;3;0;2
	NOFF_Flag	0	0	0	0
	IncTable	[0;0;0;0;0;0] [0;0;0;0;0;0]	[0;0;0;0;0;0] [0;0;0;0;0;0]	[-1; 1; -1 ; 1 ; -1;1] [0;0;0;0;0;0]	[2; -1; -1; 2; -1;-1] [1; 1; -2; 1; 1;-2]
	INC8	1	1	1	1
MEMORY_PARAM	MEM_SELECT	2	2	2	2
	PARAM1	16	16	16	48
SYMBOL_PARAM	P-RSG_CONF	X	X	X	X
	ADDRESS_LIMIT	X	X	X	X
	S_ SW_CONF0	X	X	X	X
	S_ SW_CONF1	X	X	X	X

Table 4-33 Configuration parameters for 802.11n 20MHz iss=3

BUS	PARAMETER	CONFIGURATION(dec value) 40MHz iss=0			
		-BPSK	-QPSK	16QAM	64QAM
PARAM_MEM_CONF	CLK_BLOCK	108/8=14	216/8=27	432/8=54	648/8=81
	CONFIG	0	0	0	0
	EVEN/ ODD_SYMBOL	0	0	0	0
8AGU1_PARAM	NCONV	0	0	0	0
	INITIAL VALUE(0)	0	0	0	0
	INITIAL VALUE(1)	6	12	24	36
	INITIAL VALUE(2)	12	24	48	72
	INITIAL VALUE(3)	18	36	72	108
	INITIAL VALUE(4)	24	48	96	144
	INITIAL VALUE(5)	30	60	120	180
	INITIAL VALUE(6)	36	72	144	216
	INITIAL VALUE(7)	42	84	168	252
	NCONF1	6*8 = 48	12*8 = 96	24*8 = 192	36*8 = 288
	NCONF2	108	216	432	648
8AGU2_PARAM	Nrot(0)	0	0	0	0
	Nrot(1)	1	1	1	1
	Nrot(2)	2	2	2	2
	Nrot(3)	3	3	3	3
	Nrot(4)	4	4	4	4
	Nrot(5)	5	5	5	5
	Nrot(6)	6	6	6	6
	Nrot(7)	7	7	7	7
	Noff2(0)	0	0	0	0
	Noff2(1)	0	0	1	1
	Noff2(2)	0	0	2	2
	Noff2(3)	0	0	3	3
	Noff2(4)	0	0	4	4
	Noff2(5)	0	0	5	5
	Noff2(6)	0	0	6	6
	Noff2(7)	0	0	7	7
	NCONF2	108	216	432	648
	NCONF3	8	8	24	36
	ConfigTable	0;0;0;0;0;0	0;0;0;0;0;0	2;3;2;3;2;3	3;0;2;3;0;2
	NOFF_Flag	0	0	0	0
	IncTable	[0;0;0;0;0;0] [0;0;0;0;0;0]	[0;0;0;0;0;0] [0;0;0;0;0;0]	[-1; 1; -1 ; 1 ; -1;1] [0;0;0;0;0;0]	[2; -1; -1; 2; -1;-1] [1; 1; -2; 1; 1;-2]
	INC8	1	1	1	1
MEMORY_PARAM	MEM_SELECT	2	2	2	2
	PARAM1	16	16	48	144
SYMBOL_PARAM	P-RSG_CONF	X	X	X	X
	ADDRESS_LIMIT	X	X	X	X
	S_ SW_CONF0	X	X	X	X
	S_ SW_CONF1	X	X	X	X

Table 4-34 Configuration parameters for 802.11n 40MHz iss=0

BUS	PARAMETER	CONFIGURATION(dec value) 40MHz iss=1			
		-BPSK	-QPSK	16QAM	64QAM
PARAM_MEM_CONF	CLK_BLOCK	108/8=14	216/8=27	432/8=54	648/8=81
	CONFIG	0	0	0	0
	EVEN/ ODD_SYMBOL	0	0	0	0
8AGU1_PARAM	NCONV	0	0	0	0
	INITIAL VALUE(0)	0	0	0	0
	INITIAL VALUE(1)	6	12	24	36
	INITIAL VALUE(2)	12	24	48	72
	INITIAL VALUE(3)	18	36	72	108
	INITIAL VALUE(4)	24	48	96	144
	INITIAL VALUE(5)	30	60	120	180
	INITIAL VALUE(6)	36	72	144	216
	INITIAL VALUE(7)	42	84	168	252
	NCONF1	6*8 = 48	12*8 = 96	24*8 = 192	36*8 = 288
	NCONF2	108	216	432	648
8AGU2_PARAM	Nrot(0)	0	0	232	348
	Nrot(1)	1	1	233	349
	Nrot(2)	2	2	234	350
	Nrot(3)	3	3	235	351
	Nrot(4)	4	4	236	352
	Nrot(5)	5	5	237	353
	Nrot(6)	6	6	238	354
	Nrot(7)	7	7	239	355
	Noff2(0)	0	0	16	24
	Noff2(1)	0	0	17	25
	Noff2(2)	0	0	18	26
	Noff2(3)	0	0	19	27
	Noff2(4)	0	0	20	28
	Noff2(5)	0	0	21	29
	Noff2(6)	0	0	22	30
	Noff2(7)	0	0	23	31
	NCONF2	108	216	432	648
	NCONF3	8	8	24	36
	ConfigTable	0;0;0;0;0;0	0;0;0;0;0;0	3;2;3;2;3;2	3;0;2;3;0;2
	NOFF_Flag	1	1	1	1
	IncTable	[0;0;0;0;0;0] [0;0;0;0;0;0]	[0;0;0;0;0;0] [0;0;0;0;0;0]	[-1; 1; -1 ; 1 ; -1;1] [0;0;0;0;0;0]	[2; -1; -1; 2; -1;-1] [1; 1; -2; 1; 1;-2]
	INC8	1	1	1	1
MEMORY_PARAM	MEM_SELECT	2	2	2	2
	PARAM1	16	16	48	144
SYMBOL_PARAM	P-RSG_CONF	X	X	X	X
	ADDRESS_LIMIT	X	X	X	X
	S_ SW_CONF0	X	X	X	X
	S_ SW_CONF1	X	X	X	X

Table 4-35 Configuration parameters for 802.11n 40MHz iss=1

BUS	PARAMETER	CONFIGURATION(dec value) 40MHz iss=2			
		-BPSK	-QPSK	16QAM	64QAM
PARAM_MEM_CONF	CLK_BLOCK	108/8=14	216/8=27	432/8=54	648/8=81
	CONFIG	0	0	0	0
	EVEN/ ODD_SYMBOL	0	0	0	0
8AGU1_PARAM	NCONV	0	0	0	0
	INITIAL VALUE(0)	0	0	0	0
	INITIAL VALUE(1)	6	12	24	36
	INITIAL VALUE(2)	12	24	48	72
	INITIAL VALUE(3)	18	36	72	108
	INITIAL VALUE(4)	24	48	96	144
	INITIAL VALUE(5)	30	60	120	180
	INITIAL VALUE(6)	36	72	144	216
	INITIAL VALUE(7)	42	84	168	252
	NCONF1	6*8 = 48	12*8 = 96	24*8 = 192	36*8 = 288
	NCONF2	108	216	432	648
8AGU2_PARAM	Nrot(0)	0	0	116	174
	Nrot(1)	1	1	117	175
	Nrot(2)	2	2	118	176
	Nrot(3)	3	3	119	177
	Nrot(4)	4	4	120	178
	Nrot(5)	5	5	121	179
	Nrot(6)	6	6	122	180
	Nrot(7)	7	7	123	181
	Noff2(0)	0	0	20	30
	Noff2(1)	0	0	21	31
	Noff2(2)	0	0	22	32
	Noff2(3)	0	0	23	33
	Noff2(4)	0	0	24	34
	Noff2(5)	0	0	25	35
	Noff2(6)	0	0	26	36
	Noff2(7)	0	0	27	37
	NCONF2	108	216	432	648
	NCONF3	8	8	24	36
	ConfigTable	0;0;0;0;0;0	0;0;0;0;0;0	2;3;2;3;2;3	2;3;0;2;3;0
	NOFF_Flag	0	0	0	0
	IncTable	[0;0;0;0;0;0] [0;0;0;0;0;0]	[0;0;0;0;0;0] [0;0;0;0;0;0]	[-1; 1; -1 ; 1 ; -1;1] [0;0;0;0;0;0]	[2; -1; -1; 2; -1;-1] [1; 1; -2; 1; 1;-2]
	INC8	1	1	1	1
MEMORY_PARAM	MEM_SELECT	2	2	2	2
	PARAM1	16	16	48	144
SYMBOL_PARAM	P-RSG_CONF	X	X	X	X
	ADDRESS_LIMIT	X	X	X	X
	S_ SW_CONF0	X	X	X	X
	S_ SW_CONF1	X	X	X	X

Table 4-36 Configuration parameters for 802.11n 40MHz iss=2

BUS	PARAMETER	CONFIGURATION(dec value) 40MHz iss=3			
		-BPSK	-QPSK	16QAM	64QAM
PARAM_MEM_CONF	CLK_BLOCK	108/8=14	216/8=27	432/8=54	648/8=81
	CONFIG	0	0	0	0
	EVEN/ ODD_SYMBOL	0	0	0	0
8AGU1_PARAM	NCONV	0	0	0	0
	INITIAL VALUE(0)	0	0	0	0
	INITIAL VALUE(1)	6	12	24	36
	INITIAL VALUE(2)	12	24	48	72
	INITIAL VALUE(3)	18	36	72	108
	INITIAL VALUE(4)	24	48	96	144
	INITIAL VALUE(5)	30	60	120	180
	INITIAL VALUE(6)	36	72	144	216
	INITIAL VALUE(7)	42	84	168	252
	NCONF1	6*8 = 48	12*8 = 96	24*8 = 192	36*8 = 288
	NCONF2	108	216	432	648
8AGU2_PARAM	Nrot(0)	0	0	348	522
	Nrot(1)	1	1	349	523
	Nrot(2)	2	2	350	524
	Nrot(3)	3	3	351	525
	Nrot(4)	4	4	352	526
	Nrot(5)	5	5	353	527
	Nrot(6)	6	6	354	528
	Nrot(7)	7	7	355	529
	Noff2(0)	0	0	12	18
	Noff2(1)	0	0	13	19
	Noff2(2)	0	0	14	20
	Noff2(3)	0	0	15	21
	Noff2(4)	0	0	16	22
	Noff2(5)	0	0	17	23
	Noff2(6)	0	0	18	24
	Noff2(7)	0	0	19	25
	NCONF2	108	216	432	648
	NCONF3	8	8	24	36
	ConfigTable	0;0;0;0;0;0	0;0;0;0;0;0	2;3;2;3;2;3	3;0;2;3;0;2
	NOFF_Flag	0	0	0	0
	IncTable	[0;0;0;0;0;0] [0;0;0;0;0;0]	[0;0;0;0;0;0] [0;0;0;0;0;0]	[-1; 1; -1 ; 1 ; -1;1] [0;0;0;0;0;0]	[2; -1; -1; 2; -1;-1] [1; 1; -2; 1; 1;-2]
	INC8	1	1	1	1
MEMORY_PARAM	MEM_SELECT	2	2	2	2
	PARAM1	16	16	48	144
SYMBOL_PARAM	P-RSG_CONF	X	X	X	X
	ADDRESS_LIMIT	X	X	X	X
	S_ SW_CONF0	X	X	X	X
	S_ SW_CONF1	X	X	X	X

Table 4-37 Configuration parameters for 802.11n 40MHz iss=3

4.5.3 Configuration for 802.16e

This standard has 10 different configurations, and there are different configurations for each modulation. 6 configurations for QPSK, 2 for 16QAM & 2 for 64QAM each of these are differentiated by the N_{CBPS} these configurations are presented in two tables.

BUS	PARAMETER	CONFIGURATION(dec value) 802.16e -BPSK					
		N_{CBPS}					
		96	192	288	384	480	576
PARAM_MEM_CONF	CLK_BLOCK	96/8= 12	192/8= 24	288/8= 36	384/8= 48	480/8= 60	576/8= 72
	CONFIG	0	0	0	0	0	0
	EVEN/ ODD_SYMBOL	0	0	0	0	0	0
8AGU1_PARAM	NCONV	0	0	0	0	0	0
	INITIAL VALUE(0)	0	0	0	0	0	0
	INITIAL VALUE(1)	6	12	18	24	30	36
	INITIAL VALUE(2)	12	24	36	48	60	72
	INITIAL VALUE(3)	18	36	54	72	90	108
	INITIAL VALUE(4)	24	48	72	96	120	144
	INITIAL VALUE(5)	30	60	90	120	150	180
	INITIAL VALUE(6)	36	72	108	144	180	216
	INITIAL VALUE(7)	42	84	126	168	210	252
	NCONF1	6*8= 48	12*8 = 96	18*8 = 144	24*8 = 192	30*8 = 240	36*8 = 288
	NCONF2	96	192	288	384	480	576
8AGU2_PARAM	Nrot(0)	0	0	0	0	0	0
	Nrot(1)	1	1	1	1	1	1
	Nrot(2)	2	2	2	2	2	2
	Nrot(3)	3	3	3	3	3	3
	Nrot(4)	4	4	4	4	4	4
	Nrot(5)	5	5	5	5	5	5
	Nrot(6)	6	6	6	6	6	6
	Nrot(7)	7	7	7	7	7	7
	Noff2(0)	0	0	0	0	0	0
	Noff2(1)	0	0	0	0	0	0
	Noff2(2)	0	0	0	0	0	0
	Noff2(3)	0	0	0	0	0	0
	Noff2(4)	0	0	0	0	0	0
	Noff2(5)	0	0	0	0	0	0
	Noff2(6)	0	0	0	0	0	0
	Noff2(7)	0	0	0	0	0	0
	NCONF2	96	192	288	384	480	576
	NCONF3	8	8	8	8	8	8
	ConfigTable	0;0;0;0 ;0;0	0;0;0;0 ;0;0	0;0;0;0 ;0;0	0;0;0;0 ;0;0	0;0;0;0 ;0;0	0;0;0;0 ;0;0
	NOFF_Flag	0	0	0	0	0	0
	IncTable [0;0;0; 0;0;0] [0;0;0; 0;0;0]	[0;0;0; 0;0;0] [0;0;0; 0;0;0]	[0;0;0; 0;0;0] [0;0;0; 0;0;0]	[0;0;0; 0;0;0] [0;0;0; 0;0;0]	[0;0;0; 0;0;0] [0;0;0; 0;0;0]	[0;0;0; 0;0;0] [0;0;0; 0;0;0]	[0;0;0; 0;0;0] [0;0;0; 0;0;0]
	INC8	1	1	1	1	1	1
MEMORY_PARAM	MEM_SELECT	2	2	2	2	2	2
	PARAM1						
SYMBOL_PARAM	P-RSG_CONF	X	X	X	X	X	X
	ADDRESS_LIMIT	X	X	X	X	X	X
	S_ SW_CONF0	X	X	X	X	X	X
	S_ SW_CONF1	X	X	X	X	X	X

Table 4-38 Configuration parameters for 802.16e BPSK

BUS	PARAMETER	CONFIGURATION(dec value) 802.16e			
		16QAM		64QAM	
		N_{CBPS}=384	N_{CBPS}=576	N_{CBPS}=288	N_{CBPS}=576
PARAM_MEM_CONF	CLK_BLOCK	384/8= 48	576/8=72	288/8=36	576/8=72
	CONFIG	0	0	0	0
	EVEN/ ODD_SYMBOL	0	0	0	0
8AGU1_PARAM	NCONV	0	0	0	0
	INITIAL VALUE(0)	0	0	0	0
	INITIAL VALUE(1)	24	36	18	36
	INITIAL VALUE(2)	48	72	36	72
	INITIAL VALUE(3)	72	108	54	108
	INITIAL VALUE(4)	96	144	72	144
	INITIAL VALUE(5)	120	180	90	180
	INITIAL VALUE(6)	144	216	108	216
	INITIAL VALUE(7)	168	252	126	252
	NCONF1	24*8= 192	36*8 = 288	18*8 = 144	36 * 8 = 288
	NCONF2	384	576	288	576
8AGU2_PARAM	Nrot(0)	0	0	0	0
	Nrot(1)	1	1	1	1
	Nrot(2)	2	2	2	2
	Nrot(3)	3	3	3	3
	Nrot(4)	4	4	4	4
	Nrot(5)	5	5	5	5
	Nrot(6)	6	6	6	6
	Nrot(7)	7	7	7	7
	Noff2(0)	0	0	0	0
	Noff2(1)	0	0	0	0
	Noff2(2)	0	0	0	0
	Noff2(3)	0	0	0	0
	Noff2(4)	0	0	0	0
	Noff2(5)	0	0	0	0
	Noff2(6)	0	0	0	0
	Noff2(7)	0	0	0	0
	NCONF2	384	576	288	576
	NCONF3	8	8	8	8
	ConfigTable	0;0;0;0;0;0	0;0;0;0;0;0	0;0;0;0;0;0	0;0;0;0;0;0
	NOFF_Flag	0	0	0	0
	IncTable	[0;0;0;0;0;0] [0;0;0;0;0;0]	[0;0;0;0;0;0] [0;0;0;0;0;0]	[0;0;0;0;0;0] [0;0;0;0;0;0]	[0;0;0;0;0;0] [0;0;0;0;0;0]
	INC8	1	1	1	1
MEMORY_PARAM	MEM_SELECT	2	2	2	2
	PARAM1				
SYMBOL_PARAM	P-RSG_CONF	X	X	X	X
	ADDRESS_LIMIT	X	X	X	X
	S_ SW_CONF0	X	X	X	X
	S_ SW_CONF1	X	X	X	X

Table 4-39 Configuration parameters for 802.16e 16QAM & 64QAM

4.5.4 Configuration for DVB

The DVB standard has 3 interleaver kinds, classified in two parts outer interleaver and inner interleaver.

4.5.4.1 Outer Interleaver

4.5.4.1.1 Convolutional Interleaver

The convolutional interleaver has only one configuration mode and the parameters to implement it are shown in Table 4-40.

The convolutional interleaver is not a block interleaver like the other interleavers in viewed in this document, the convolutional interleaver is a continuous interleaver, so it can manage as a variable size of bits per interleaving block, this size is configured as the CLK_BLOCK value.

BUS	PARAMETER	CONFIGURATION(dec value)
PARAM_MEM_CONF	CLK_BLOCK	Can be any value between 1 to 12
	CONFIG	3
	EVEN/ ODD_SYMBOL	0
8AGU1_PARAM	NCONV	0
	INITIAL VALUE(0)	0
	INITIAL VALUE(1)	X
	INITIAL VALUE(2)	X
	INITIAL VALUE(3)	X
	INITIAL VALUE(4)	X
	INITIAL VALUE(5)	X
	INITIAL VALUE(6)	X
	INITIAL VALUE(7)	X
	NCONF1	205
	NCONF2	2448
8AGU2_PARAM	Nrot(0)	0
	Nrot(1)	X
	Nrot(2)	X
	Nrot(3)	X
	Nrot(4)	X
	Nrot(5)	X
	Nrot(6)	X
	Nrot(7)	X
	Noff2(0)	0
	Noff2(1)	X
	Noff2(2)	X
	Noff2(3)	X
	Noff2(4)	X
	Noff2(5)	X
	Noff2(6)	X
	Noff2(7)	X
	NCONF2	2448
	NCONF3	8
	ConfigTable	0;0;0;0;0;0
	NOFF_Flag	0
	IncTable	[0;0;0;0;0;0; 0;0;0;0;0;0] [0;0;0;0;0;0; 0;0;0;0;0;0]

	INC8	0
MEMORY_PARAM	MEM_SELECT	1
	PARAM1	X
SYMBOL_PARAM	P-RSG_CONF	X
	ADDRESS_LIMIT	X
	S_ SW_CONF0	X
	S_ SW_CONF1	X

Table 4-40 Configuration parameters for convolutional interleaver

4.5.4.2 Inner Interleaver

4.5.4.2.1 Compact Bit Interleaver

Compact bit interleaver is a junction of the bit interleavers described in the DVB standard. One single bit interleaver interleaves 126 bits. In the structure of the compact bit Interleaver we can interleave 2, 4, or 6 bit interleavers at the same time. The configuration parameters are listed in Table 4-41.

BUS	PARAMETER	CONFIGURATION(dec value)		
		2 bit interleaver	4 bit interleaver	6 bit interleaver
PARAM_MEM_CONF	CLK_BLOCK	126*2/8=32	126*4/8=63	126*6/6=126
	CONFIG	1	1	1
	EVEN/ ODD_SYMBOL	0	0	0
8AGU1_PARAM	NCONV	0	0	0
	INITIAL VALUE(0)	0	0	0
	INITIAL VALUE(1)	150	150	150
	INITIAL VALUE(2)	1	300	300
	INITIAL VALUE(3)	151	450	450
	INITIAL VALUE(4)	2	1	600
	INITIAL VALUE(5)	152	151	750
	INITIAL VALUE(6)	3	301	X
	INITIAL VALUE(7)	153	451	X
	NCONF1	1	1	1
	NCONF2	1024	1024	1024
8AGU2_PARAM	Nrot(0)	0	0	0
	Nrot(1)	0	0	0
	Nrot(2)	1	0	0
	Nrot(3)	1	0	0
	Nrot(4)	2	1	0
	Nrot(5)	2	1	0
	Nrot(6)	3	1	X
	Nrot(7)	3	1	X
	Noff2(0)	0	0	0
	Noff2(1)	1	1	1
	Noff2(2)	0	2	2
	Noff2(3)	1	3	3
	Noff2(4)	0	0	4
	Noff2(5)	1	1	5
	Noff2(6)	0	2	X
	Noff2(7)	1	3	X
	NCONF2	126	126	126
	NCONF3	4	2	1
	ConfigTable	0;0;0;0;0;0	0;0;0;0;0;0	0;0;0;0;0;0
	NOFF_Flag	0	0	0
	IncTable	[0;150;X;X;X;X;X;X;X;X;X;X]	[0;150;300;450;X;X;X;X;X;X]	[0;150;300;450;600;750;X;X;X;X;X;X]

		[X;X;X;X;X;X;X X;X;X;X;X;X]	[X;X;X;X;X;X;X X;X;X;X;X;X]	[X;X;X;X;X;X;X X;X;X;X;X;X]
	INC8	1	1	1
MEMORY_PARAM	MEM_SELECT	2	2	2
	PARAM1	24	48	48
SYMBOL_PARAM	P-RSG_CONF	X	X	X
	ADDRESS_LIMIT	X	X	X
	S_ SW_CONF0	X	X	X
	S_ SW_CONF1	X	X	X

Table 4-41 Configuration parameters for compact bit interleaver

4.5.4.2.2 Symbol Interleaver

Symbol interleaver has 2 configuration modes 2K & 8K, and also can manage 3 configurations of word size, 2, 4 or 6 bits.

The symbol interleaver implements the EVEN\ODD signal which is the selector of OFDM symbol defined in the DVB standard. The EVEN\ODD signal switches the rolls of the AGU´s as is shown in the Table 4-42.

EVEN\ODD value	AGU1 function	AGU2 function
0	Write Address	Read Address
1	Read Address	Write Address

Table 4-42 Values and effects of EVEN\ODD signal

For even or odd symbol the configuration is only changed by the parameter EVEN/ODD_SYMBOL, if EVEN/ODD_SYMBOL=1 then the OFDM frame is even if not the OFDM frame is odd.

The configuration parameters for all the configurations of the symbol interleaver are shown in Table 4-43.

BUS	PARAMETER	CONFIGURATION(dec value)					
		2K MODE			8K MODE		
		2	4	6	2	4	6
PARAM_MEM_CONF	CLK_BLOCK	1512	1512	1512	6048	6048	6048
	CONFIG	2	2	2	2	2	2
	EVEN/ODD_SYMBOL	1 or 0	1 or 0	1 or 0	1 or 0	1 or 0	1 or 0
8AGU1_PARAM	NCONV	X	X	X	X	X	X
	INITIAL VALUE(0)	X	X	X	X	X	X
	INITIAL VALUE(1)	X	X	X	X	X	X
	INITIAL VALUE(2)	X	X	X	X	X	X
	INITIAL VALUE(3)	X	X	X	X	X	X
	INITIAL VALUE(4)	X	X	X	X	X	X
	INITIAL VALUE(5)	X	X	X	X	X	X
	INITIAL VALUE(6)	X	X	X	X	X	X
	INITIAL VALUE(7)	X	X	X	X	X	X
	NCONF1	X	X	X	X	X	X
	NCONF2	X	X	X	X	X	X
8AGU2_PARAM	Nrot(0)	0	0	0	0	0	0
	Nrot(1)	X	X	X	X	X	X
	Nrot(2)	X	X	X	X	X	X
	Nrot(3)	X	X	X	X	X	X

	Nrot(4)	X	X	X	X	X	X
	Nrot(5)	X	X	X	X	X	X
	Nrot(6)	X	X	X	X	X	X
	Nrot(7)	X	X	X	X	X	X
	Noff2(0)	0	X	X	X	X	X
	Noff2(1)	X	X	X	X	X	X
	Noff2(2)	X	X	X	X	X	X
	Noff2(3)	X	X	X	X	X	X
	Noff2(4)	X	X	X	X	X	X
	Noff2(5)	X	X	X	X	X	X
	Noff2(6)	X	X	X	X	X	X
	Noff2(7)	X	X	X	X	X	X
	NCONF2	1512	1512	1512	6048	6048	6048
	NCONF3	8	8	8	8	8	8
	ConfigTable	0;0;0;0;0;0	0;0;0;0;0;0	0;0;0;0;0;0	0;0;0;0;0;0	0;0;0;0;0;0	0;0;0;0;0;0
	NOFF_Flag	0	0	0	0	0	0
	IncTable	[0;0;0;0;0;0] [0;0;0;0;0;0]	[0;0;0;0;0;0] [0;0;0;0;0;0]	[0;0;0;0;0;0] [0;0;0;0;0;0]	[0;0;0;0;0;0] [0;0;0;0;0;0]	[0;0;0;0;0;0] [0;0;0;0;0;0]	[0;0;0;0;0;0] [0;0;0;0;0;0]
	INC8	0	0	0	0	0	0
MEMORY_PARAM	MEM_SELECT	4	5	6	4	5	6
	PARAM1	X	X	X	X	X	X
SYMBOL_PARAM	P-RSG_CONF	36	36	36	83	83	83
	ADDRESS_LIMIT	1512	1512	1512	6048	6048	6048
	S_ SW_CONF0	[0,7,.5,1,8,2,6,9,3,4,X,X]	[0,7,.5,1,8,2,6,9,3,4,X,X]	[0,7,.5,1,8,2,6,9,3,4,X,X]	[5,11,3,0,10,8,6,9,2,4,1,7,]	[5,11,3,0,10,8,6,9,2,4,1,7,]	[5,11,3,0,10,8,6,9,2,4,1,7,]
	S_ SW_CONF1	[0,7,.5,1,8,2,6,9,3,4,X,X]	[0,7,.5,1,8,2,6,9,3,4,X,X]	[0,7,.5,1,8,2,6,9,3,4,X,X]	[5,11,3,0,10,8,6,9,2,4,1,7,]	[5,11,3,0,10,8,6,9,2,4,1,7,]	[5,11,3,0,10,8,6,9,2,4,1,7,]

Table 4-43 Configuration parameters for symbol interleaver

5 Results

The results between different interleaver implementations are shown in this chapter. To make a comparison against the proposed architecture (MSI) the next interleaver implementations where selected:

- LUT
- Special matrix
- Crossbar

Three points were selected are shown to make a comparison:

- Reconfigurability
- Integration area
- Power consumption

All the architectures were programmed in Verilog, simulated in Modelsim and Questasim, synthesized into a technology of 65nm using Design Complier. Test vectors were created by Matlab to stimulate the architectures and verify the correct work of the design, Power theater tool was used to obtain the power consumption values of each implementation.

The power values shown in power consumption represent the average instantaneous power consumption of the architectures. The average takes since the first bit to interleaving enters to the DUT, till the last of the interleaved bits is at the output.

5.1 Reconfigurability

The reconfigurability was analyzed in terms of the capability of the implementations to map the different interleavers stated in the 802.11a, 802.11n, 802.16e and DVB standards.

The Table 5-1 shows the interleavers that are mapped by the 4 different interleaver implementations. Shows 2 general implementations LUT and the proposed MSI, special matrix and crossbar can not implement all the standards. Special matrix can only implement the 802.11a and 802.16e standards. The crossbar cannot implement the convolutional interleaver.

Interleaver Implementations	Interleavers Mapped					
	802.11a	802.11n	802.16e	Convolutional	Symbol	Compact Bit
MSI	✓	✓	✓	✓	✓	✓
LUT	✓	✓	✓	✓	✓	✓
Special Matrix	✓	X	✓	X	X	X
Crossbar	✓	✓	✓	X	✓	✓

Table 5-1 Interleavers mapped by the different interleaver implementations

5.2 Size

The interleaver architectures were mapped into a 65nm technology, with the intention to obtain the area of integration, the results obtained by the Design Compiler tool are shown in Table 5-2 and Table 5-3. The results in Table 5-2 show the total area of integration for different interleaver implementations. This shows that the biggest implementation is the crossbar. This result is because implementation of big crossbars, require an enormous amount of transistors, the number of transistors increases exponentially with the number of inputs. The special matrix implementation has the least area but it can't make all the configurations only the small configurations.

Interleaver Implementations	Combinational Area (μm^2)	Non combinational Area (μm^2)	Total Area (μm^2)
MSI	317,206	780,673	1,097,043
LUT	488,797	1,612,497	2,099,084
Special Matrix	35,298	55,341	90,639
Crossbar	20,614,986	16,302,951	36,917,937

Table 5-2 Total integration area of every architecture

A more realistic measurement of this result is shown in Table 5-3, this table describes the implementations in terms of the area that is active to implement every standard. This is because for different configurations a different amount of circuitry is used. The results in Table 5-3 show that the MSI is the best option on every case of configuration.

For the different configurations of the MSI the principal difference is the memory that is enabled to do the interleaving, for the 802.11a, 802.11n, 802.16e, compact bit interleaver configurations only the 16 memories of 64 bits of the 802 memory are enabled, for the convolutional interleaver the convolutional memory of 2048x8 bits is enabled and finally for the symbol interleaver the 4 symbol memory of 8192x2 bits are enabled.

Interleaver	Total Area (μm^2)	Integration area active on Interleavers Mapped					
		802.11a (μm^2)	802.11n (μm^2)	802.16e (μm^2)	Compact Bit (μm^2)	Convolutional (μm^2)	Symbol (μm^2)
MSI	1,097,043	68,016	68,016	68,016	68,016	217,214	872,149
LUT	2,099,084	2,099,084	2,099,084	2,099,084	2,099,084	2,099,084	2,099,084
Special Matrix	90,639	90,639	-	90,639	-	-	-
Crossbar	36,917,937	585,999	585,999	585,999	585,999	-	36,917,937

Table 5-3 Total integration active area for different interleaver configurations

The following tree figures (Figure 5-1, Figure 5-2, Figure 5-3) are an explanation of the integration active area on the different interleaver configurations. The black blocks represent the blocks that are disabled for the selected configuration.

Configurations:

- 802.11a ,802.11n, 802.16e, compact bit interleaver (Figure 5-1)
- Convolutional interleaver (Figure 5-2)
- Symbol interleaver (Figure 5-3)

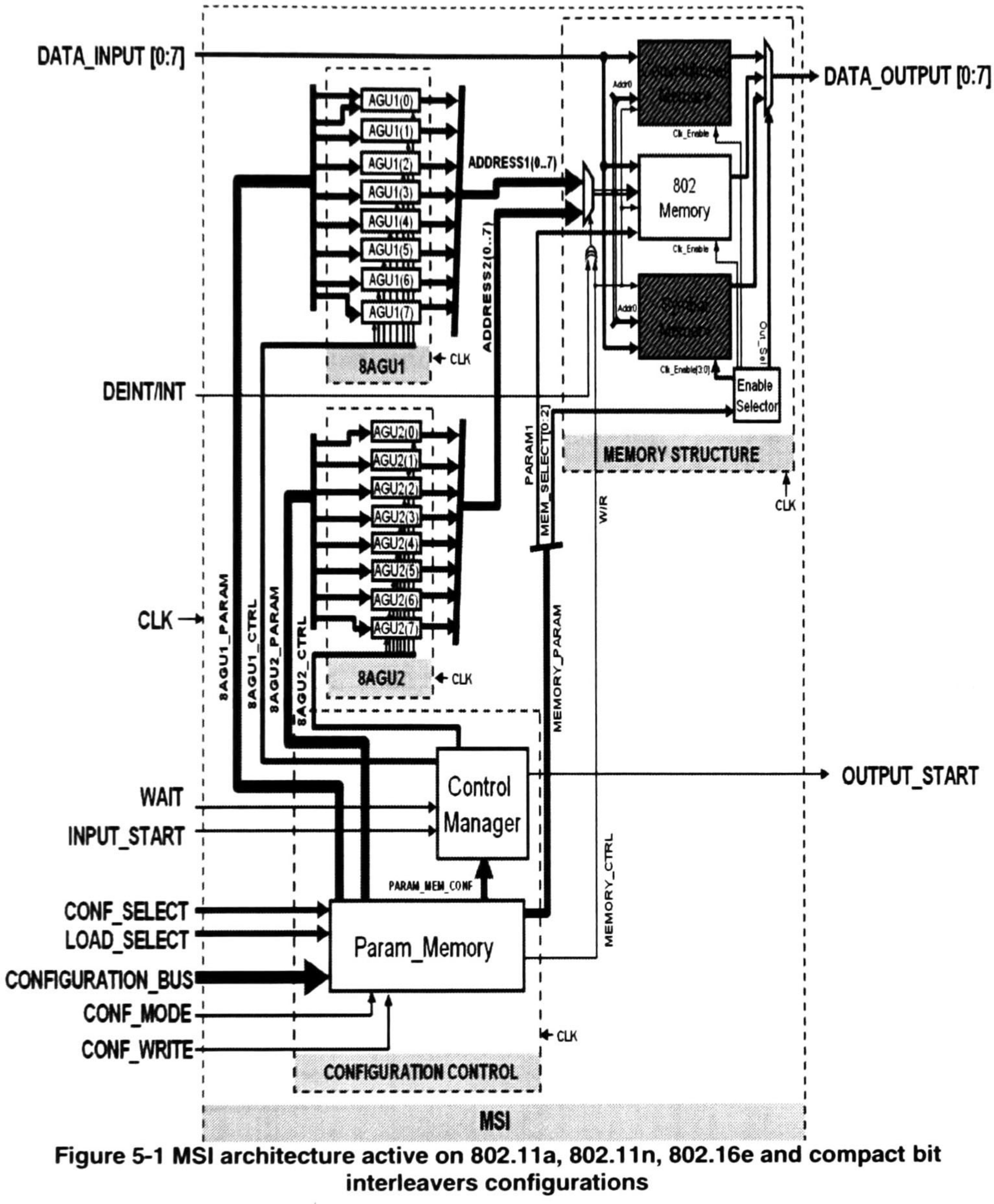


Figure 5-1 MSI architecture active on 802.11a, 802.11n, 802.16e and compact bit interleavers configurations

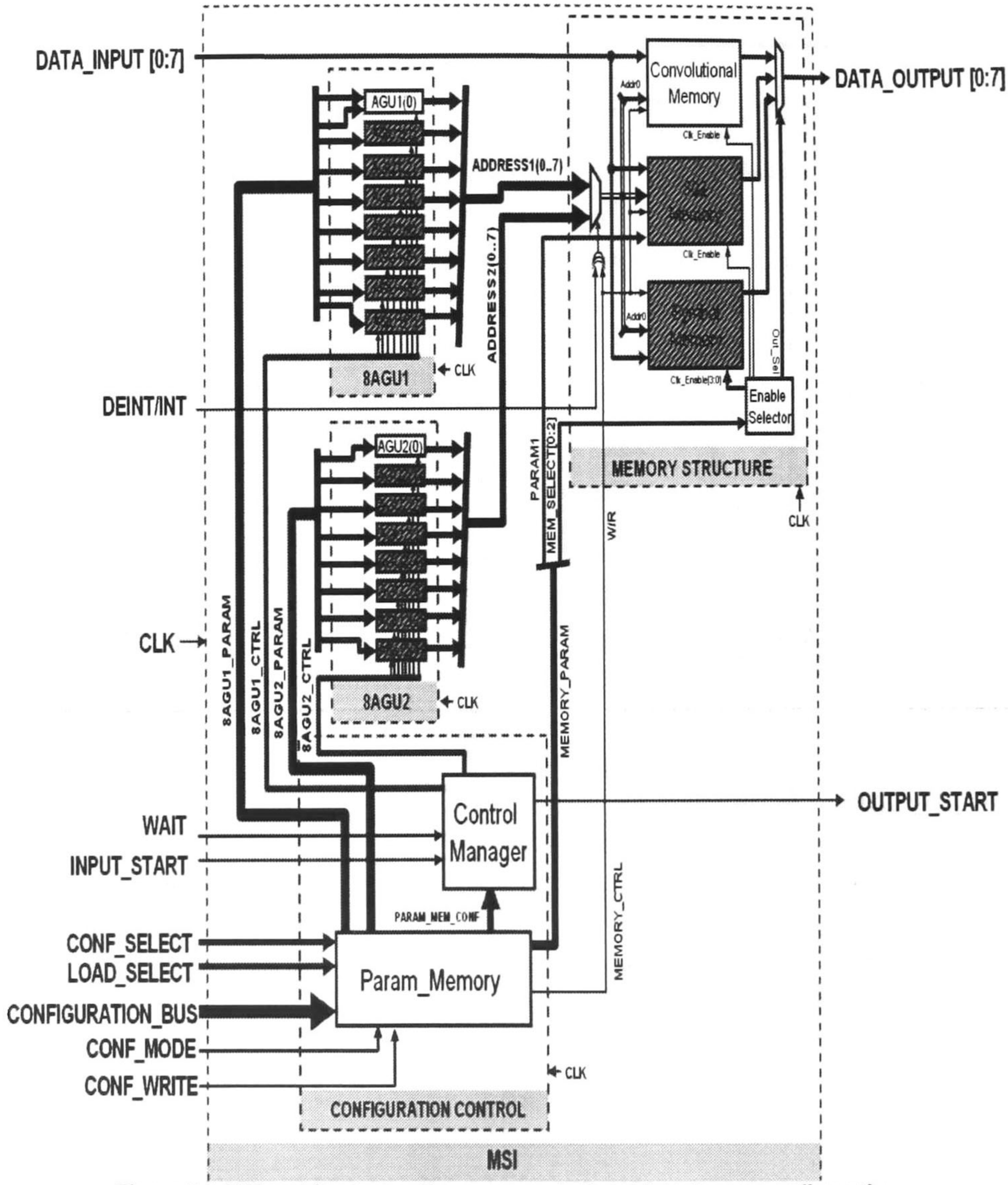


Figure 5-2 MSI architecture active on convolutional interleaver configuration

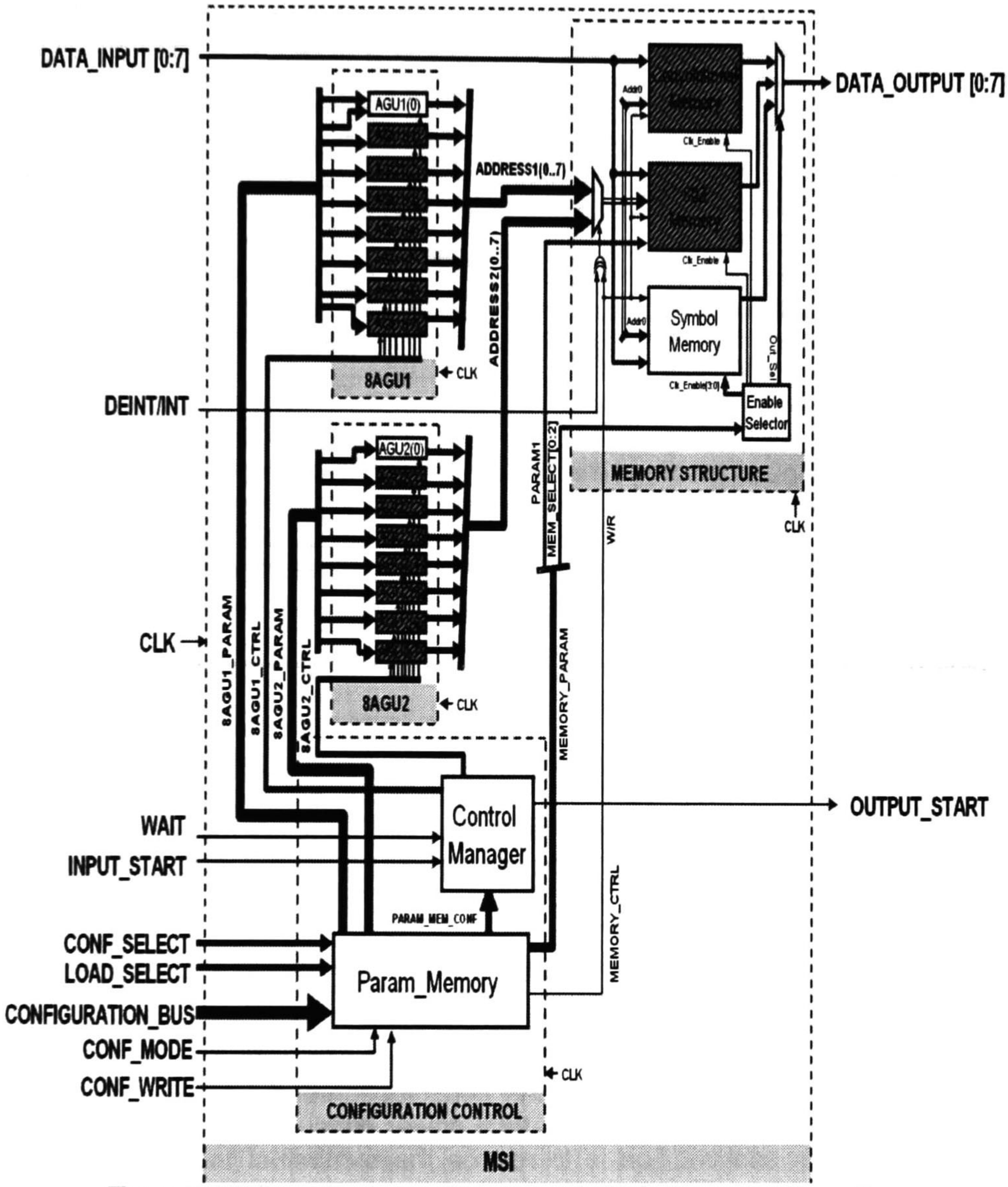


Figure 5-3 MSI architecture active on symbol interleaver configuration

5.3 Power Consumption

The power analysis is made trough the use of several tools, first after we have the tested designs in Verilog code, this code is mapped to a technology of 65nm, using the Design Compiler tool, this result is compiled in Questasim to review that the design works well for the different interleaver configurations. The generation of the signals activity file is done, this file is generated when we run the test bench and has all the information of the commutation of all the nodes inside the design, this information is one of the inputs of the Power Theater tool.

Power Theater tool reads the architecture file, signals activity file for the selected configuration and power consumption specifications for the library cells (the worst case library was selected) to generate the power consumption results for the selected configuration.

The results are presented in comparative table of power efficiency between the 4 architectures applied to some selected interleaver configurations.

The results are expected to be similar between the configurations 802.11a, 802.11n, 802.16e, and compact bit interleavers, because these use the same active area, as is described at chapter 0 Size.

The power consumption of the architecture is divided in two major parts in static power and in dynamic power.

The static power represents the consumption of the structure when no commutation of the internal signals is done, and it is because to the leakage current of the transistors, this power can tell us about the number of transistors that are used to implement the architecture, the bigger the number of transistors the bigger static power consumption.

The dynamic power represents the power that is consumed in the transitions of the logic states of the signals, when a signal goes from “0” to “1” or “1” to “0” a current is generated by the transistors to change the state, there is also a peak of current called glitch that is included. The more the states change in the architecture the more the dynamic power consumption of the architecture. This power is the bigger contributor to the total power consumption so one way to reduce the power consumption is to reduce the number of transitions in the nodes.

The power consumption results (Table 5-4) are expected to be of the same order for the 802.11a, 802.11n, 802.16e and compact bit interleaver because they use the same active area, and the results for the Convolutional and Symbol interleaver, are expected to be more power demanding.

Interleaver Configuration	Interleaver Architecture	Power consumption		
		Static	Dynamic	Total
802.11a	MSI	250µW	19.4mW	19.6mW
	LUT	471µW	324mW	324mW
	Special Matrix	21.5uW	17.3mW	17.4mW
	Crossbar	123.3µW	5076mW	5076mW
802.11a	MSI	250µW	19.4mW	19.6mW
	LUT	469W	326mW	326mW
	Special Matrix	NA	NA	NA
	Crossbar	123.3µW	5076mW	5076mW
802.11a	MSI	248µW	19.3mW	19.6mW
	LUT	471µW	326mW	326mW
	Special Matrix	21.5uW	17.3mW	17.4mW
	Crossbar	123.3µW	5076mW	5076mW
Compact bit interleaver	MSI	248µW	12.3mW	12.5mW
	LUT	471µW	326mW	326mW
	Special Matrix	NA	NA	NA
	Crossbar	123.3µW	5076mW	5076mW
Convolutional	MSI	254µW	42.5mW	42.8mW
	LUT	471µW	326mW	326mW
	Special Matrix	NA	NA	NA
	Crossbar	NA	NA	NA
Symbol	MSI	244µW	122mW	123mW
	LUT	471µW	326mW	326mW
	Special Matrix	NA	NA	NA
	Crossbar	7200µW	319,788mW	319,788mW

Table 5-4 Power consumption results for all the interleaver configurations

The Figure 5-4 shows the results of Table 5-4 in a graphical way, this graph shows the power in mW in logarithmic scale, versus the interleaver configurations, and with different colors the architecture evaluated. We can see that the crossbar has the bigger power consumption, this is expected because in this architecture exist bigger crossbars that commutate at the same time generating a big amount of dynamic power. The less consuming architecture overall is the MSI this has only power compared with the Special Matrix architecture on the 802.11a and 802.16e configurations.

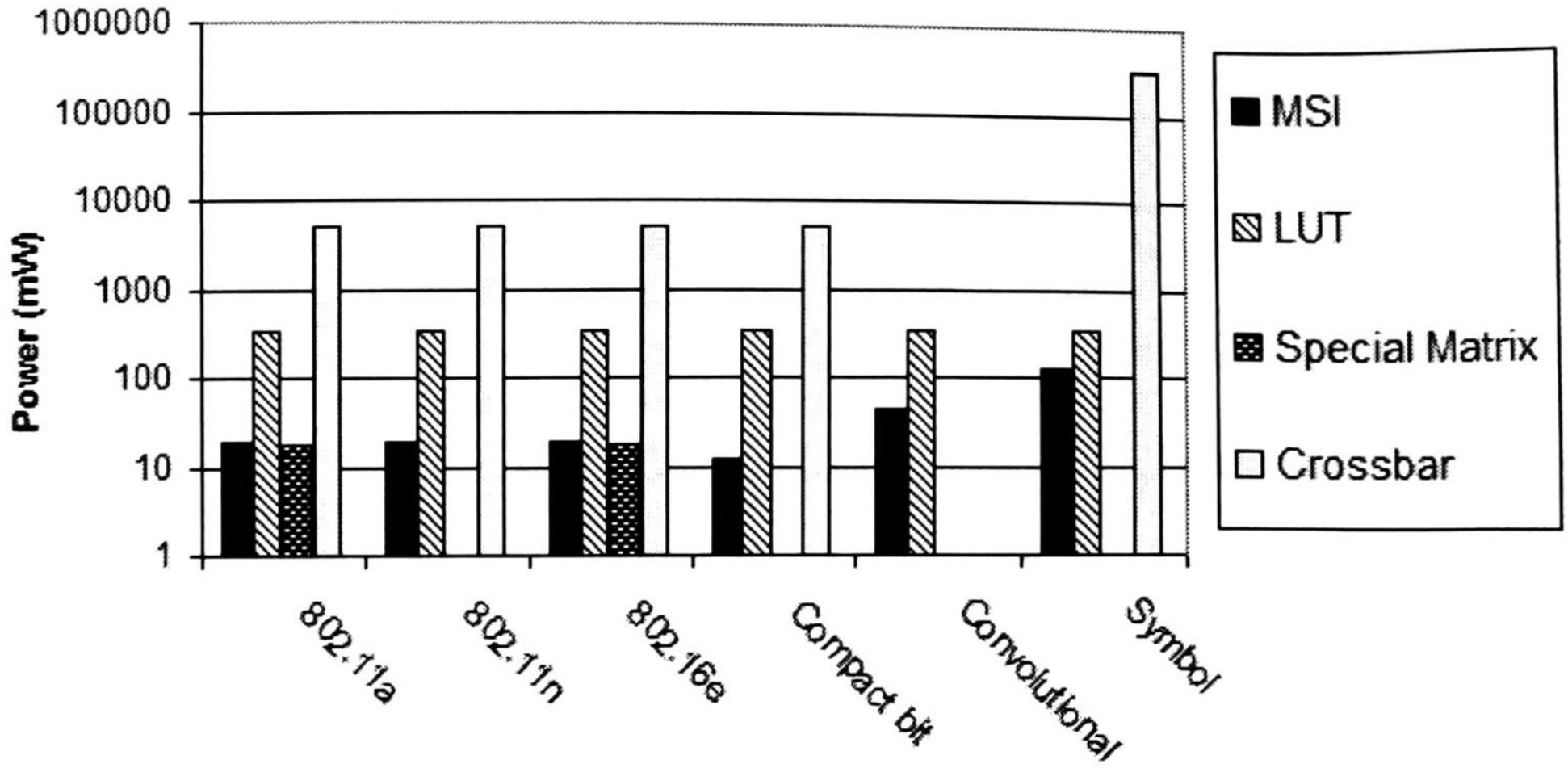


Figure 5-4 Comparative total power consumption

A screen shot of the Power Theater tool used to generate the power consumption results, is shown in the Figure 5-5, at the center of the figure the architecture hierarchy of the MSI, this block organization is generated by the synthesis tool (Design Compiler), in the window on the left the result of a test with the 802.11n configuration 64QAM.

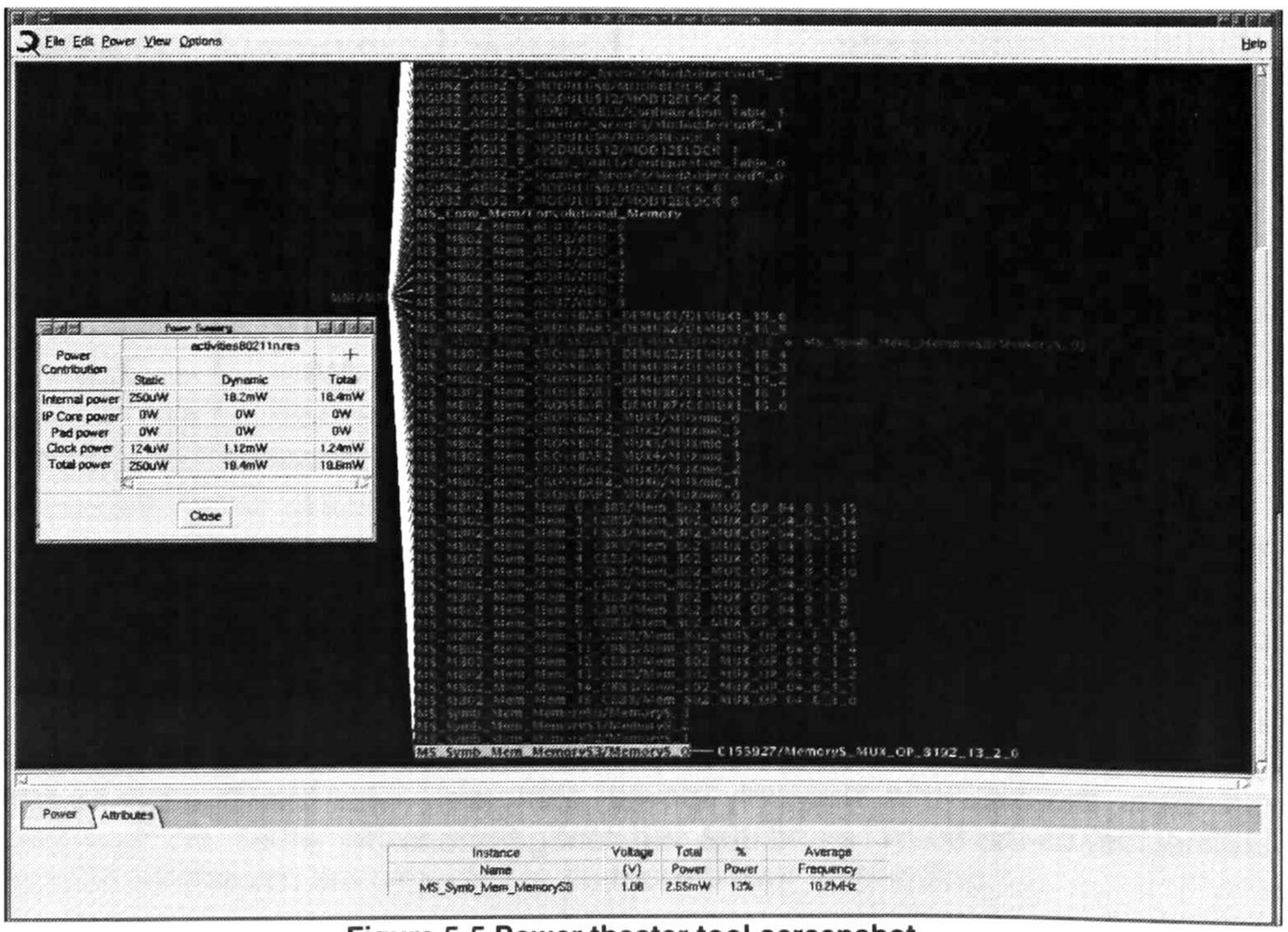


Figure 5-5 Power theater tool screenshot

6 Conclusions and Future Work

6.1 Conclusions

In this thesis the problem of reconfigurable interleaver implementations was addressed and solved. Particularly, in this thesis I show a solution in terms of a new architecture design. The proposed architecture was validated trough a testing methodology and the fulfillment of its test cases. Furthermore, the design was synthesized in 65nm; this allowed me to obtain an area and power consumption analysis. In order to compare my proposal with other approaches, the former design and analysis was carried out with three architectures, whose are the most representatives of the common approaches to implement interleavers. The comparison results show that the proposed approach achieves the best performance in terms of both area and power consumption, considering the fact that the proposed solution is reconfigurable.

As the proposed interest of this thesis was to create a multi-standard interleaver that can handle the interleavers at the 802.11a, 802.11n, 802.16e and DVB standards. As it was commented, these objectives were reached in this thesis, and thus, we can summarize the obtained performance of the design in the following lines:

1. The proposed interleaver architecture MSI can handle the 802.11a, 802.11n, 802.16e and DVB interleavers.
2. The interleaver was designed to work for one selected standard each time, but it can be reconfigured on the fly, to work as a different interleaver for a different standard at the second time, and so on.
3. The reconfigurablility results for the 802.11a, 802.11n, 802.16e, and DVB standards are summarized in the Table 5-1 Interleavers mapped by the different interleaver implementations.
4. The area results from synthesize in a 65nm technology are summarized in the Table 5-3 Total integration active area for different interleaver configurations.
5. The power consumption results are summarized in the Table 5-4 Power consumption results for all the interleaver configurations.

Furthermore, as reported in section 5.2, the area results shows that the MSI architecture is the less active area consuming for all the different architecture implementations.

The latter performance was also repeated in power consumption, and thus the results shown at chapter 5.3 reveal that the MSI is the less power consuming architecture of all the implemented architectures.

The previous results were based on three principal ideas developed with my team work in this thesis:

- Find a way to convert a convolutional interleaver into a kind of block interleaver. This was used to map the convolutional interleaver into a block interleaver inside the proposed MSI.

- Generate a simplified way to control the STS structure with the parallel optimized AGU's using the ADU's, reducing the number of AGU used to control the structure from 8 AGU and 16 CGU to only 8 AGU.

- Find a way to represent some of the interleaver definitions, as a simplified (linear) formula with some deterministic perturbations.

I want also to mention that I have accomplished several learning goals throughout the development of this thesis, i.e. this work has allowed me to get involved in the process of digital design. I also have learned how to generate technical design specifications, how to program RTL at Verilog, how to do logical synthesis to a 65nm silicon technology, how to create a validation methodology of a design, and how to do a power analysis of a design.

6.2 Future Work

Some topics can be defined to continue the work of this thesis:

- In this work, we have developed a transformation of the formula describing the interleaving process, this approach allowed us to reduce the interleave implementation when compared with the direct approach. However, this work was developed independently for each of the analyzed interleavers, and it is not clear if this approach will be adequate for all the formulas that can be encountered in describing an interleaver process. Due to the reduction in complexity achieved with this approach, it is worthy to perform further research in this topic.

- A way to convert a convolutional interleaver into a kind of block interleaver was developed for the specific case of the convolutional interleaver of the DVB standard, this approach give us the chance to implement the convolutional approach into the block interleaver architecture, allowing reuse of logic, reduction of the power consumption because the conversion of the shift registers into memories, and also a reduction of the logic of the AGU of this architecture. By this facts further research of this conversion and the generalization of this method to more convolutional architectures is an interesting point to work.

- Increasing speed in the digital systems makes necessary the development of faster designs. The MSI synthesized in 65nm works at a frequency of 100MHz, and is desirable to increase the frequency of this design. This can be done by implementing the pipeline technique to the MSI architecture.

References

[1] J. L. Ramsey, "Realization of optimum interleavers." *IEEE Trans. Inform. Theory*, vol. IT-16, pp. 338–345, May 1970.

[2] H. Schmit and D. E. Thomas, "Address Generation for Memories Containing Multiple Arrays," in *Proc. ICCAD*, Nov. 1995, pp. 510–514.

[3] Miranda, M.; Catthoor, F.; Janssen, M.; De Man, H. "ADOPT: Efficient Hardware Address Generation in Distributed Memory Architectures" in . Procc System Synthesis, 9th International Symposium on. Nov. 1996. pp. 20 – 25.

[4] Grant, D.M. Denyer, P.B. "Address Generation for Array Access Based on Modulus m Counters", EDAC. Proc. of the European Conference on Design Automation. Feb 1991. pp 118 – 123.

[5] Kusakabe, S. Nagai, T. Inenaga, K. Amamiya, M. "Address Generation of Dataflow Fine-Grain Parallel Data-Structures on a Distributed-Memory Computer" Proceedings of the 1996 Conference on Parallel Architectures and Compilation Techniques, Oct. 1996. pp 139 – 143.

[6] Rainer Leupers, Peter Marwedel "Algorithms for Address Assignment in DSP Code Generation" Computer-Aided Design University of Dortmund, Department of Computer Science 12, 44221 Dortmund, Germany. IEEE. Nov 1996. pp 109 – 112.

[7] Miguel Miranda, Martin Kaspar, Francky Catthoor, Hugo De Man. "Architectural Exploration and Optimization for Counter Based Hardware Address Generation". Proc. European Design and Test Conference. ED&TC 97. March 1997 pp 293 – 298.

[8] Ji Y. Kim and Myung H. Sunwoo. "Design of Address Generation Unit for Audio DSOP", Proceedings of 2004 International Symposium on Intelligent Signal Processing and Communication Systems, 2004. Nov. 2004 Page(s):616 – 619.

[9] Hulina, P.T.; Coraor, L.D.; Kurian, L. "Design and VLSI Implementation of an Address Generation Coprocessor". Computers and Digital Techniques, IEE Proceedings. Volume 142, Issue 2, March 1995 Page(s):145 – 151.

[10] D.M. Grant, "Optimization of Address Generator Hardware", European Design and Test Conference, 1994. EDAC, The European Conference on Design Automation. ETC European Test Conference. EUROASIC, The European Event in ASIC Design, Proceedings. March 1994 Page(s):325 – 329.

[11] Kjetil E. Vistnes and Oddvar. "Reconfigurable Address Generators for Stream-Based Computation Implemented on FPGAs". Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International. 4-8 April 2005. Page 4.

[12] D. Grant, P.B. Denyer, I. Finlay. "Synthesis of Address Generators", Computer-Aided Design, 1989. ICCAD-89. Digest of Technical Papers. IEEE International Conference on. Nov. 1989. Page(s):116 – 119.

[13] Vincent C. Gaudet, Rolland J. Gaudet, and P. Glenn Gulak "Programmable Interleaver Design for Analog Iterative Decoders" Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on Volume 49, Issue 7, July 2002 Page(s):457 – 464.

[14] Eric Tell, Dake Liu; A Hardware Architecture for a Multi Mode Block Interleaver; International Conference on Circuits and Systems for Communications (ICCSC), Moscow, Russia, June 2004.

[15] Libero Dinoi and Sergio Benedetto. "Variable-Size Interleaver Design for Parallel Turbo Decoder Architectures", Communications, IEEE Transactions on. Volume 53, Issue 11, Nov. 2005 Page(s):1833 – 1840.

[16] Ji Won Jung, In Ki Lee, Duk Gun Choi and Jin Hee Jeong. "Design and Architecture of Low-Latency High-Speed Turbo Decoders". ETRI Journal, vol.27, no.5, Oct. 2005, pp.525-532.

[17] Rostislav (Reuven) Dobkin, Michael Peleg. "Parallel Interleaver Design and VLSI Architecture for Low-Latency MAP Turbo Decoders" IEEE transactions on very large scale integration (VLSI) systems, vol. 13, no. 4, April 2005 Page(s):427 – 438.

[18] Gordian Prescher, Tobias Gemmeke. "A Parametrizable Low-Power High-Throughput Turbo-Decoder" Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005. (ICASSP '05). Volume 5, 18-23 March 2005 Page(s): v/25 - v/28 Vol. 5.

[19] D.C. Opferman and N.T. Tsao-Wu, "On a Class of Rearrangeable Switching Networks, Part I: Control Algorithm," Bell System Technical J. vol. 50, pp. 1579-1600, 1971.
[20] V.E. Benes, "Optimal Rearrangeable Multistage Connecting Networks", Bell Sys. Tech. Journal. 43 (1964) 1641-1656.
[21] Rajgopal Kannan. "The KR-Benes Network: A Control-Optimal Rearrangeable Permutation Network" Computers, IEEE Transactions on. Volume 54, Issue 5, May 2005 Page(s):534 - 544
[22] Michael J. Marcus. "Space-Time Equivalents in Connecting Networks". Massachusetts Institute of Technology. In Proc. 1970 Int. Conf. Commun., pp. (35-25)-(35-31).
[23] Michael J. Marcus. "The Theory of Connecting Networks and their Complexity: a Review" Proceedings of the IEEE, vol. 65, No. 9 September 1977. Page(s):1263 – 1271.
[24] Manuel E. Guzmán, Arturo Veloz, "Implementation of Interleavers", Oral Presentation, 2007.
[25] IEEE 802.11a Standard LAN/MAN Wireless LANS. http://grouper.ieee.org/groups/802/11/
[26] IEEE 802.11n Standard LAN/MAN Wireless LANS. http://grouper.ieee.org/groups/802/11/
[27] IEEE 802.16e Standard LAN/MAN Broadband Wireless LANS. http://grouper.ieee.org/groups/802/16/
[28] DVB Digital Video Broadcasting standard. http://www.dvb.org/

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL I.P.N.
UNIDAD GUADALAJARA

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó la tesis

Diseño e Implementación de un Interleaver Multi-Estándar
para los Estándares 802.11a, 802.11n, 802.16e y DVB

de (la) C. Carlos Rogelio Sánchez Ortiz El dia 14 de Marzo de 2008.

Dr. Ramón Parra Michel
Investigador CINVESTAV 2B
CINVESTAV Unidad Guadalajara

Dr. José Raúl Loo Yau
Investigador CINVESTAV 2B
CINVESTAV

Dr. Mario Angel Siller González
Pico
Investigador CINVESTAV 2A
CINVESTAV Unidad Guadalajara

CINVESTAV.
BIBLIOTECA CENTRAL
SSIT000006331