Instituto Tecnológico y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática
Maestría en Diseño Electrónico

Low-Cost CAN Protocol Logic Analyzer

TRABAJO RECEPCIONAL que para obtener el **GRADO** de
MAESTRO EN DISEÑO ELECTRÓNICO
Presenta: **CESAR CARLOS ROBLES MARTINEZ**

Director **DR. JOSE LUIS PIZANO ESCALANTE**

Tlaquepaque, Jalisco. 25 de septiembre de 2021.

To my parents

Summary

Keeping an electronics system up to date represents a continuous challenge because technology is in constant motion. In this work, the understanding of the Field Programmable Gate Array (FPGA) technology is the central point. The most important feature of FPGA is allowing virtually the customization of any desired circuit; to perform a specific function through the usage of the hardware description Verilog language. Supported by its major benefit which is the real parallelism and high performance. The Control Area Network (CAN) protocol is a robust protocol with some strong advantages such: no master-slave scheme, differential communication, and high noise immunity, that is why this technology has a major application in the automotive field but is not limited to it. Is also used in aerospace, avionics fields, and more. Therefore, focusing on the avionics area the proposal of a low-cost CAN protocol logic analyzer is treated in this document, this analyzer aims to provide a reliable and cheaper alternative to software and tools available on the market that perform testing and debugging of Printed Circuit Boards (PCBs) or Systems, based on the CAN for communication. Besides, This tesis provides the proposal architecture and internal elements that integrates the low-cost CAN protocol logic analyzer and, generating hard saving related to PCB scrap due to the missing accurate tool to perform the CAN testing.

Contents

Maestría en Diseño Electrónico ..... **i**

Low-Cost CAN Protocol Logic Analyzer ..... **i**

1. Controller Area Network **3**

1.1. CAN TRANSCEIVER 3
1.2. CAN CONTROLLER 3
1.3. CAN FRAME 4
1.3.1 SOF (Start of Frame) 4
1.3.2 Arbitration field 5
1.3.3 Standard version 5
1.3.4 Extended version 5
1.3.5 Control field 6
1.3.6 Data field 6
1.3.7 CRC field 7
1.3.8 Acknowledge field 7
1.3.9 EOF (End of frame) 8
1.3.10 Bit stuffing 8
1.3.11 Error handling 9
1.4. CAN NETWORK 9

2. Verilog HDL proposal CAN analyzer **10**

2.1. MICROARCHITECTURE 10
2.2. DATAPATH 11
2.2.1 Input Register 12
2.2.2 ID check 12
2.2.3 Coupling register 13
2.2.4 Field Registers 13
2.2.1 Mux Data 14
2.2.2 Mux Field 14
2.2.3 Shift field register 14
2.3. CAN CONTROL 17
2.3.1 CAN System Enable 18
2.3.2 Stuff Detectors 19
2.3.3 EOF Detector 20
2.3.4 Main Control Unit 21
2.3.5 Counter Mux Field 21
2.3.6 Counter Field 22

Appendix **27**

A. COMPLEMENTARY INFORMATION FOR THE CAN ANALYZER 29
NIOSII PROCESSOR 29

Introduction

The CAN protocol has gained popularity in the automotive industry due to its excellent features, such as noise immunity, differential transmission, bus topology, and the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) which allows a bus without the need of a Master-Slave communication. The CAN protocol requires the physical and datalink layers from the Open System Interconnection (OSI) model to perform properly, and it also complies with the specifications (ISO11898-1, ISO11898-2) [1],[2]. As a result, CAN is an excellent alternative for avionics applications due to its sturdiness.

On the other hand, the Field Programmable Gate Array (FPGA) provides a robust platform for developing customized circuits and data parallelism, features that open a new possibility to implement the same designs into another FPGA because of its main advantage over other programmable technologies; The *Reconfigurability* that does not depend on a specific pinout configuration. Therefore, the hardware description using Verilog gives virtually unlimited possibilities to carry out any desired circuit using FPGA technology.

Keeping the previous as the central topics and moving on the engineering world, there will always be a natural need for the improvement of any system or application to increases its performance and make it safer to avoid issues, for this reason, the migration of a complete system from a conventional RS232 (Recommended Standard 232) protocol to a CAN represents an enormous challenge. So, in this context the company that made this transition faces many challenges because all the Printed Circuit Boards (PCBs) now require a complex and safer testing stage. However, based on previous experience, deploying the correct version and tested PCBs involves many new components that the laboratory staff is not acquainted with. Therefore, this new technology represents many challenges with the system's delivery time to the final user.

This work proposes a solution to constant PCB failure cases. All the PCBs implemented at the company are designed to use CAN as standard communication. Besides, the testing methods are not the best and many times depend on the skills of the user. Uncountable situations can lead to PCB failures, for example, a mismatch between CAN Low (CAN-L) and CAN High (CAN-H) connections, bad welding on testing harnesses, non-standard connectors, and using the wrong

components for the test. Although an analyzer from a specific vendor is involved in the testing process and considering that all the suites available in the industry represent a high cost, buying a full license for a system that uses only a few features represents an obstacle because the company considers it an unjustified purchase. As a result, the testing stage is limited to the trial version which comes with many restrictions, such as a limited number of messages, run-time limitation, among others more. Also, the time to perform a diagnostic test and analysis is minimal. Thus, when a PCB fails, it is discarded automatically increasing the PCB scrap. Hence, using the FPGA technology to develop a CAN prototype is a cheap alternative to licensed software and CAN analyzer tools. Considering that the system already makes usage of the FPGA technology. This proposal also pretends to provide the capability to embed the CAN analyzer on the main control unit. As a result, there will not be a need for additional hardware to perform the analysis of the CAN bus which is a critical point on the field application.

The structure of this work is as follows: In Unit 1 the understanding of the CAN frame and the fields the composes it is the focus topic, Next, Unit 2 describes the proposed microarchitecture for the CAN analyzer using the hardware description *Verilog* that is separated in the *Control* and *Datapath* stages. Finally, the conclusions over the proposed solution.

I am thankful to my tutor Ph.D. Jose Luis Pizano Escalante for all the support and guidance all along with this work and the knowledge he has shared with me. Also, with my coordinator Ph.D. Omar Humberto Longoria Gandara.

1. Controller Area Network

The Controller Area Network (CAN) is a powerful communication protocol that follows the standard ISO11898 [1] and [2] which is based on differential signal transmission. CAN is characterized by the CSMA/CA and its high immunity to noise interferences due to the physical medium which is a twisted pair of cables. The CAN network management does not follow conventional master-slave communication. Then, the transmission data requires another type of method to achieve sending the frames over the bus without collisions or missing data.

The feature of the CAN that allows the transmission is that all the frames have access to the bus through a unique and specific identifier. Therefore, any module connected to the network can communicate with any other module but only if the ID is within the list. Besides, each node needs a control unit and a transceiver to get access to the bus. However, not having a master taking over the bus or modules requires using techniques to achieve the correct transmission and keep the network synchronized such as error detection and bit stuffing (that will be seen in Unit 2). So, CAN provides a powerful serial bus communication for applications that require robustness and feasibility.

1.1. CAN transceiver

The CAN transceiver is the representation of the physical layer from the OSI model, and it performs as a bridge between every module that is connected to the CAN network. It is necessary to mention that this device takes the differential voltage level values from 1.5V to 2.5V and 2.5V to 3.5V for CAN L and CAN H respectively (Fig. 1-1) and turns them into serial communication and vice versa. Notice that in CAN a recessive bit is = to 1 and a dominant bit is = 0.

1.2. CAN Controller

The CAN controller represents the datalink layer from the OSI model, for this reason, every module connected on the bus depends on a controller to performs the frames management. The

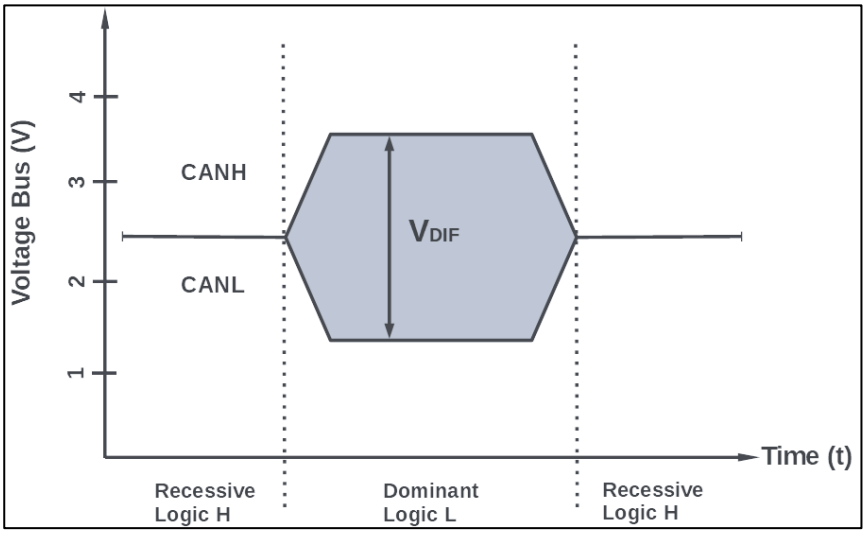

Fig. 1-1 Voltage level of the CAN, when the recessive level is = 1 and the dominant level is = 0

standard of CAN provides all the aspects for every type of frame which are: *data frame*, *remote frame*, and *error frame*. As mentioned earlier, this element oversees error detection and frame management using the bit stuffing and the arbitration field that allows selecting the message's priority. In CAN protocol the 0x00 ID has the highest importance over the rest so, the messages with the highest priority most have the lowest ID.

1.3. CAN Frame

Understanding the CAN frame is a primordial starting point in this first proposal, the analyzer only performs the detection of the *data frame* type in the standard version of the CAN protocol because the system uses only this type of frame. Fig. 1-2 describes the standard version of the CAN frame. Besides, the CAN frame includes the following fields and features described below.

1.3.1 SOF (Start of Frame)

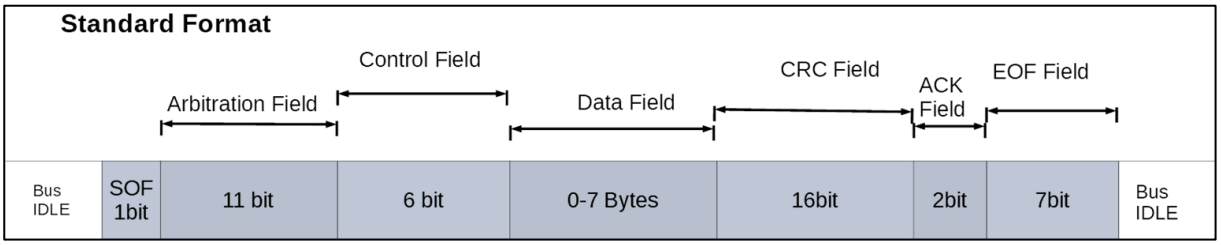

Fig. 1-2 CAN standard version frame

The CAN bus level in the idle state is the recessive level. This bit starts the transmission by asserting the bus at the dominant level and synchronizing all the modules in the network.

1.3.2 Arbitration field

A network without a master requires control to decide the frame transmission and, the arbitration field acts as a supervisor for the CAN network. Therefore, it manages the ID frames priority depending on the ID's number. Any module connected to the CAN network depends on this control to send or receive data, the lowest ID, the first in transmit. The arbitration field is different in the standard and the extended version of the CAN. So, both frames have a similar composition however, pay attention to the main differences between the identifiers of each one.

1.3.3 Standard version

In the CAN standard version Fig. 1-3, this field is 12 bit long, integrated with the Base ID (11 bits), and the RTR (1bit) bit.

1.3.4 Extended version

For the extended version, 29 bits integrate the field Fig. 1-4, using the next order, the Base ID, SRR (1bit), IDE (1bit), Extender ID (18 bits), and RTR (1bit).

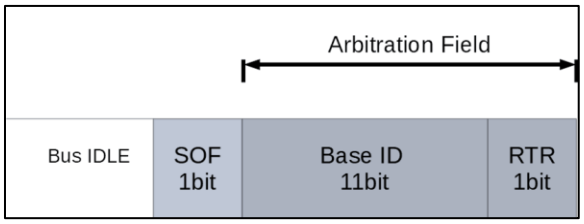

Fig. 1-3 Arbitration field in the standard version of the CAN, the RTR bit is always dominant level.

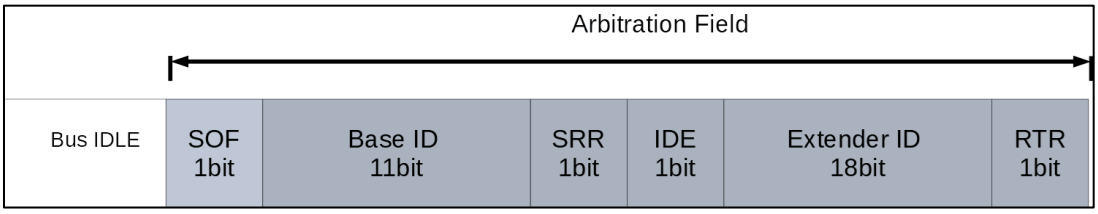

Fig. 1-4 Arbitration field in the extended version of the CAN, the SRR, IDE, and RTR bit are dominant levels.

1.3.5 **Control field**

It Consists of 6 bits that determine the number of bytes to be sent, the IDE bit, r0 bit, and, Data Length Code (DLC 4 bits), which integrates this field. Fig. 1-5 shows the possible combination of this field.

1.3.6 **Data field**

It is 64-bits long divided into 8 bytes, sending the lowest byte first and starts with the Most Significant Bit (MSB). Besides, byte 0 is sent first and byte 7 is the last. The Data field fits according to the DCL number provided from the Control field.

1.3.7 CRC field

This field is 15 bits in length plus a delimiter bit, is a checksum algorithm that uses a polynomial (1) to determine if the frame is correct as expected.

$$X^{15} + X^{14} + X^{10} + X^{8} + X^{7} + X^{4} + X^{3} + 1 \quad (1)$$

1.3.8 Acknowledge field

The acknowledge field includes the ACK bit and the Delimiter bit. The transmitter sends the ACK bit as dominant, then waits for a response from any other modules that will overwrite it as recessive. If the transmitter does not receive a response, then the frame is dropped.

DLC_3	DLC_2	DLC_1	DLC_0	Data Field							
0	0	0	0								
0	0	0	1	Byte 0							
0	0	1	0	Byte 0	Byte 1						
0	0	1	1	Byte 0	Byte 1	Byte 2					
0	1	0	0	Byte 0	Byte 1	Byte 2	Byte 3				
0	1	0	1	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4			
0	1	1	0	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5		
0	1	1	1	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	
1	0	0	0	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7

Fig. 1-5 The picture shows the DCL bits that belong to the *Control field*. Also, the *Data field* length distribution depending on the DLC number.

1.3.9 EOF (End of frame)

The end of the frame consists of seven consecutive recessive bits that end the frame transmission.

1.3.10 Bit stuffing

The CAN protocol makes usage of the bit stuff technique Fig. 1-6 that consists of the bit polarity detection. For the CAN controller receiver, if it detects five consecutively bits of the same value, this will remove the next bit. On another side, in the case of the transmitter, it will add an extra bit. As a result, it allows keeping the synchronization of all connected modules. Moreover, these stuff bits only apply from the *Arbitration* field to the *Data* field.

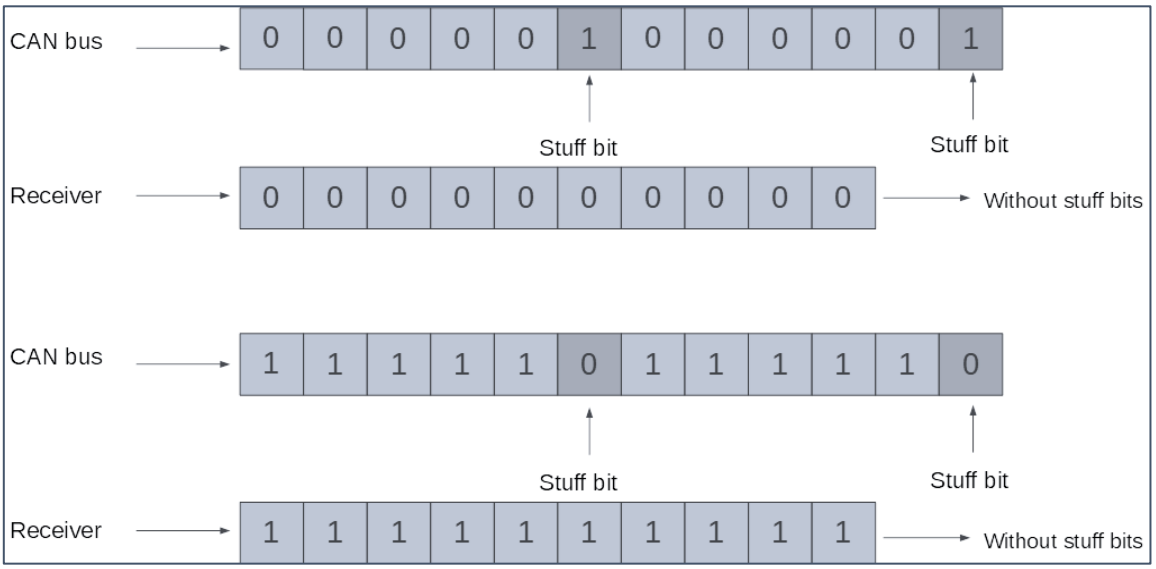

Fig. 1-6 Bit stuffing technique. The two possible combinations, after 5 bits of the same value, one opposite bit value is added. In the case of the receiver, it removes these stuff bits.

1.3.11 Error handling

This CAN analyzer only performs one of the error detections that follows the CAN protocol standards. The bit stuffing error detection is essential while receiving a frame from the CAN bus because it is mandatory to subtract the stuffing bits from the frame to achieve the correct separation of each field that composes the CAN frame. The next versions of this CAN analyzer will perform the bit, CRC and, Acknowledge errors.

1.4. CAN Network

The CAN protocol demands at least two nodes to establish the communication. Moreover, to follow the specification of the CAN protocol, adding a transceiver MCP2551 [3] is required per node due to the CAN bus differential voltage levels. As well, a 120Ω resistor as a terminal connection at the beginning and end of the CAN bus is needed to comply with the protocol characteristic transmission line impedance Fig. 1-7.

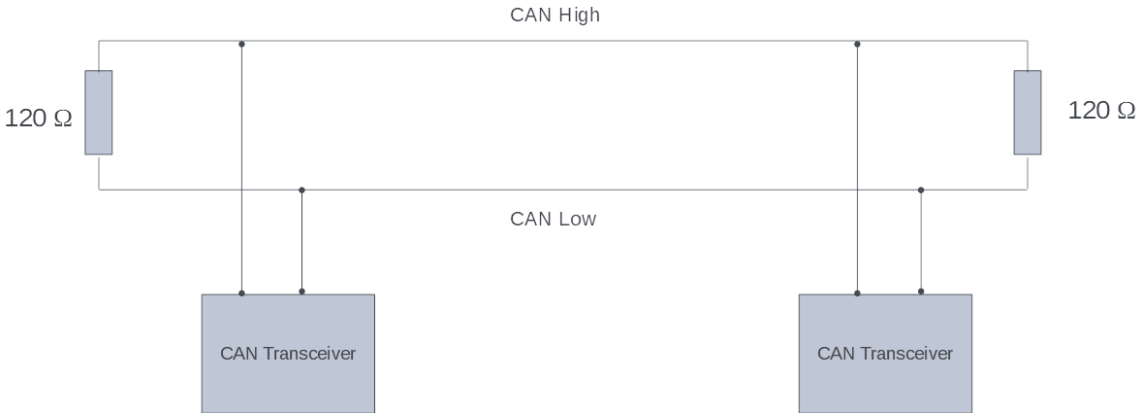

Fig. 1-7 CAN bus connection using the 120-termination resistance.

2. Verilog HDL proposal CAN analyzer

The Verilog HDL and the FPGA technology provide together a powerful and flexible tool for the development and customized design circuits for any purpose. Also, Verilog provides real parallelism and high performance due to its reconfigurability. One of the advantages of *Verilog* is the scalability what enables the possibility to reuse modules created previously for new projects. In this context, this CAN analyzer is the first approach to a more complex and robust analyzer. Furthermore, the same design created on the same platform can fit into another FPGA of the same family if its size allows it. As a result, the hardware description is the base for the CAN analyzer.

The following 2.1 section, presents the CAN analyzer's microarchitecture which explains the proposal *Control* and *Datapath* hardware stages.

2.1. Microarchitecture

The microarchitecture for the CAN analyzer implementation consists of two main parts the *CAN Control*, and the *Datapath* Fig. 2-1. Both elements work together to capture the frame and deliver the information to the NiosII processor (*see appendix A*). For this project, the selected development boar is the DE2-115 that integrates a Cyclone IV FPGA from Altera [4]. Section 2.3 is about all the modules that compose the *Datapath* and how the data is captured and driven to the NiosII processor [5]. Then in section 2.4 the CAN control is the main topic, which describes how to control the *Datapath* and its behavior to coordinate the transmission process.

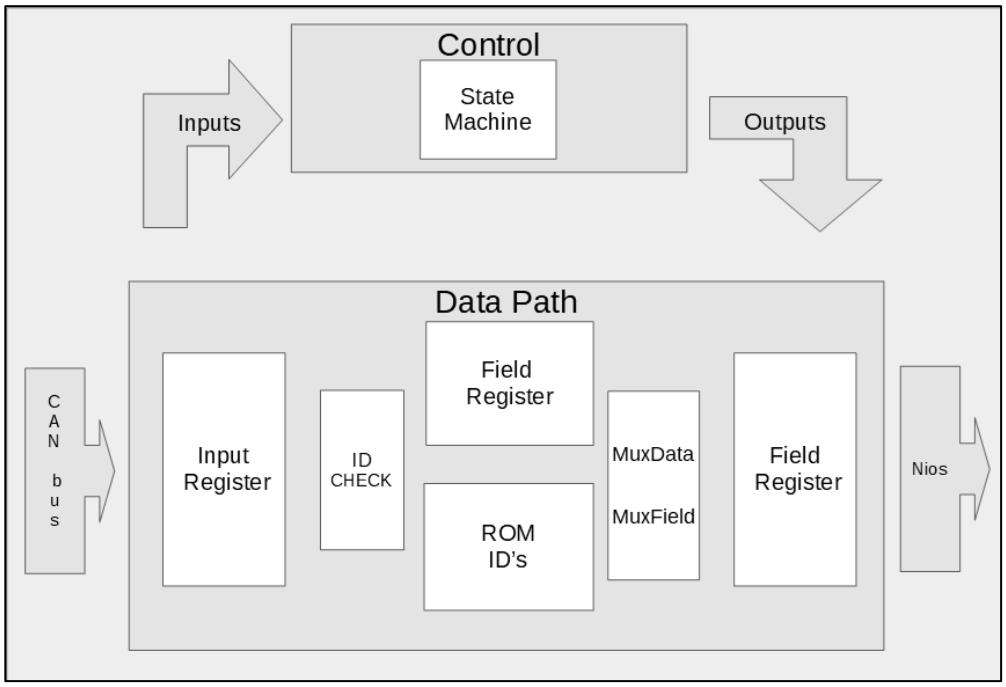

Fig. 2-1 Microarchitecture of the CAN analyzer

2.2. Datapath

The *Datapath* Fig. *2-1* is the part of the core in charge of managing data frames. First, the *Input Register* captures the CAN frame from the bus. Once this process is over, it needs to compare the captured ID with the ID bank, and if both match, the flow continues otherwise the core drops the frame. Next, the *DataPath* takes the CAN frame and then distributes it in its main fields, which are, *Arbitration*, *Control*, *Data*, *CRC*, and *Acknowledge*. At this point, two multiplexers to feed the *Output Register* are used, the first *MuxData* separates eight bytes from the *Data Field* into singular bytes from 0 to 7, and the second which is *MuxField* delivers the data of each field included the output of the *MuxData*. However, it is mandatory to adjust the length of the fields because the *CRC* size field is 16 bits long. Lastly, sending the data to the NiosII depends on the *Output Register* that sends each field one at a time.

2.2.1 Input Register

The *Input Register* is a dedicated shift register and the element that allows the CAN core to capture the frames. In other words, it performs as a buffer while catching the CAN frame. Also, here is where the bit stuffing and error detection takes place. Fig. 2-2 shows the three signals that control the register; *bufferShift*, which allows shifting the register one bit at a time; *bufferLoad*, which loads the data till the whole frame is captured; and *bufferStuff*, which skips the stuffing bits in the frame.

2.2.2 ID check

While transmitting the frames it is a must to know whether the message is within the network *ID bank* or not. If the CAN frame is not recognized the core will drop the frame. Hence, the element that takes over this function is the *ID check* module that compares the bank ID list stored into the CAN core and compares it with the received CAN frame Fig. 2-3. If both matches, then the frame goes to the next *Datapath* stage that is the *Field Registers.* The signal that enables

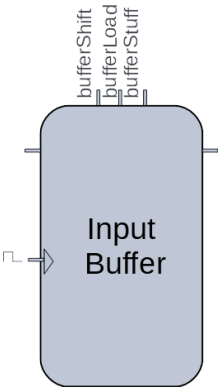

Fig. 2-2 Input Register count with three control signals, *bufferShift* to capture each frame bit, *bufferLoad* that storages the whole frame once captured and, *bufferStuff* for bit stuffing detection.

this register is *couplingReg* and must be a 1 logic. Besides, the *Control* unit provides this activation, and the feedback response is provided by the comparator through the *checkID* signal.

2.2.3 Coupling register

This register is only to match the data timing and keeps the synchronization between signals. Otherwise, there will be missing data while transmitting the fields to the registers to store the data.

2.2.4 Field Registers

Once the CAN core captured the frame, the next step is to stock the corresponding information to each specific field, *Arbitration*, *Control*, *Data*, *CRC*, and *Acknowledge* **Error!**

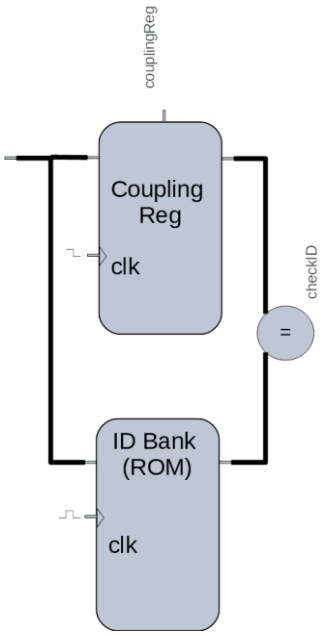

Fig. 2-3 *ID Bank* and *Coupling Register*, both connected to the ID comparator to verify if the frame is within the CAN network.

Reference source not found.. This step is because the NiosII will receive the data serially, but before it must pass through two multiplexers that manage the time transmission the *MuxData* and the *MuxField*.

2.2.1 Mux Data

Due to the flexibility of the *Data Field* that fits depending on the number of bytes provided by the *Control Field*, the implementation of a multiplexer to separate each byte helps to determine which data sends to the next stage Fig. 2-4. *Besides*, controlling the selector will provide a buffer while sending one byte at a time. The *Mux Data* size is 3x8 and, its output is directly connected to another multiplexer input to manages the data flow.

2.2.2 Mux Field

As the *Mux Data* separates de *Data field*, a new multiplexer *Mux Field* (Fig. 2-5) takes over similarly but in this case with each field that composes the frame. Moreover, keeping in mind that each field has a different length, and to achieve symmetry for easy data management, all fields need to fit the CRC length that is 16 bits. So, the multiplexer is a 3x8 as the *Mux Data.*

2.2.3 Shift field register

The final module of the *Datapath* is a shift register in charge of delivering the information to the NiosII processor, each field is sent from the *Mux field* depending on the selector's value. On another side, the activation of this register is provided by the *Counter Mux Field* which is part of the *Control* stage that is treated in section 2.3.

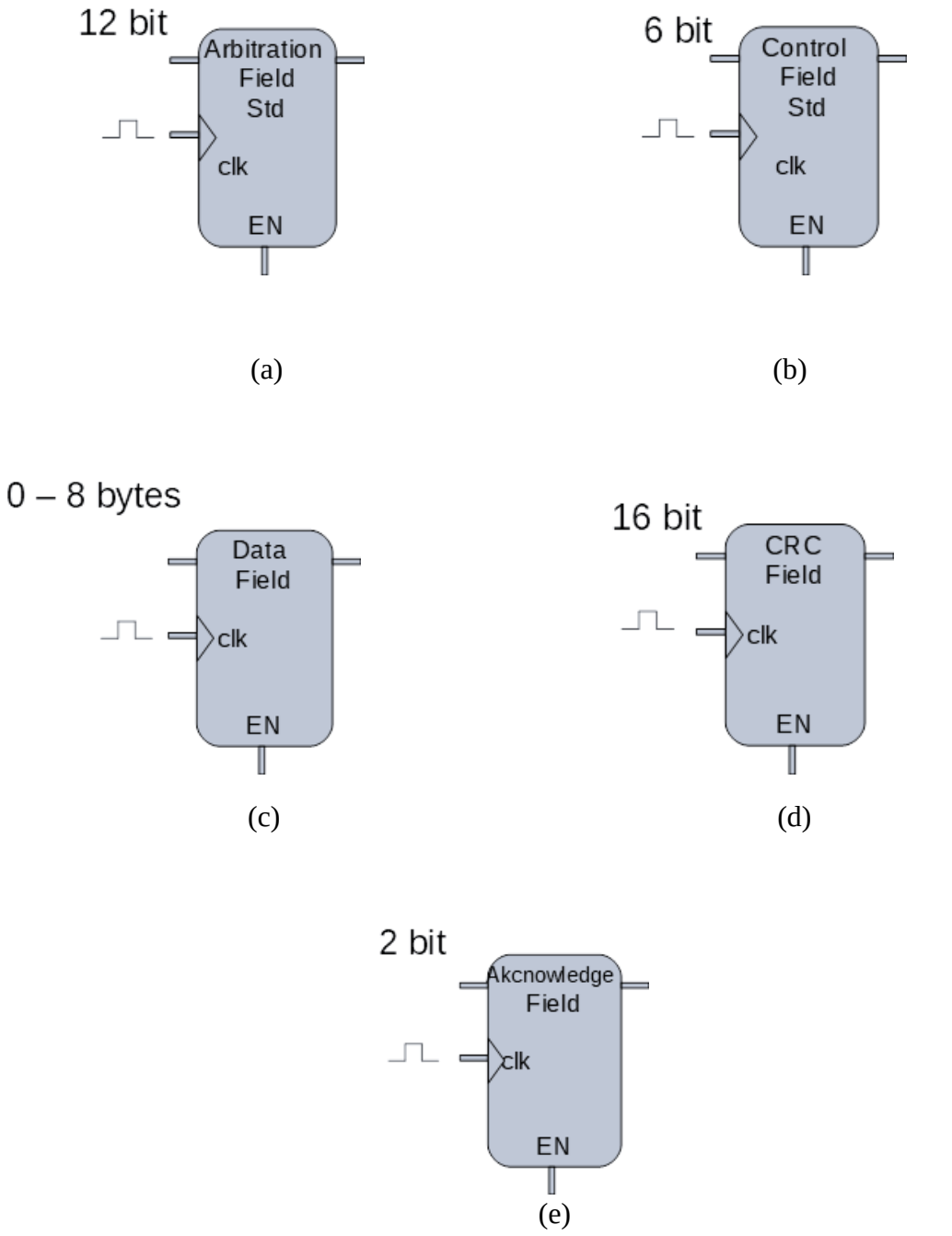

Fig. 2.4 Registers to storage the fields (a)Arbitration field register; CAN standard version (b) Control field register; CAN standard version (c)Data field register; CRC field register (e); Acknowledge field register.

Verilog HDL proposal CAN analyzer

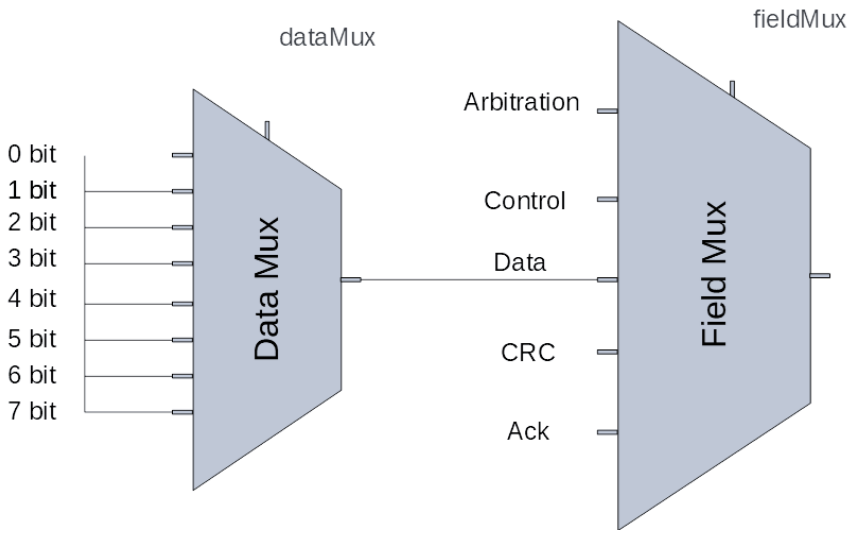

Fig. 2-4 *Mux Data* and Field Mux interconnection

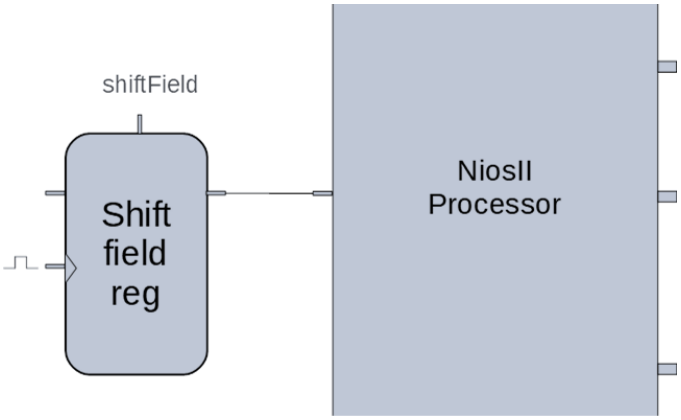

Fig. 2-5 *Field Register* connected to the NiosII input to deliver the data in serial form.

2.3. CAN Control

The *CAN Control* core facilitates the data flow through the *Datapath* by controlling the activations of each part and, it consists of five state machines and two counters. The *CAN SystemEnable* machine is in charge to synchronize the input bits to the complete system. Two machines perform as a sequence detector to drop the stuff bits carried by the frame, one for recessive bits *One Stuff Detector* and, one more for dominant bits *Zero Stuff Detector.* The fourth machine is in charge to determine when to stop capturing the frame based on the *EOF* bit chain. Finally, the last machine is the main control unit for the whole *DataPath.*

On the other side, the two counters work together to send the information to the NiosII through the *Shift Field Register*. The *Counter Mux Field* switches the selector *Mux Field* value when the *Shift Field Register* has finished sending a field to the NiosII processor. At the same time, the *Counter Field* provides the shifting activation to the *Shift Field Register* allowing in this way sending the complete field serially.

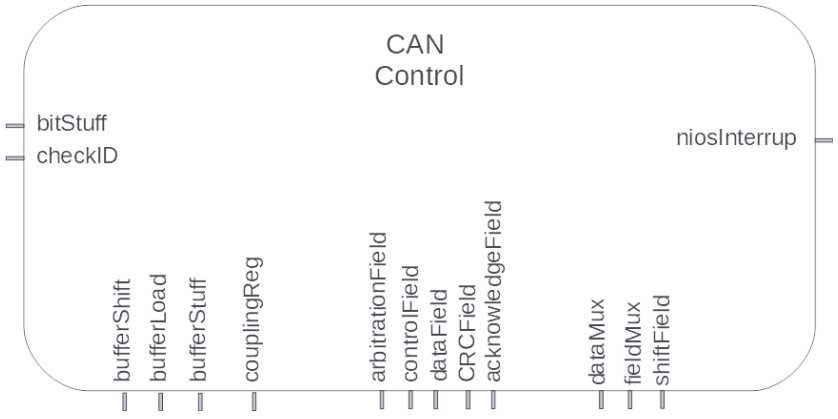

Fig. 2-6 CAN *Control Unit* and its signals to control the *Datapath.*

Verilog HDL proposal CAN analyzer

2.3.1 CAN System Enable

The synchronization between the *Transmitter* TX and, the *Receiver* RX modules is a must requirement for successful serial transmission. In other words, making that the RX module capable of capturing the data from the bus is the CAN analyzer starting point. Whit this in mind, the CAN analyzer runs 7 times (*which is an arbitrary value*) faster than the CAN bus standard baud rate, providing correct timing while capturing the frame (*modifying the "CAN System Enable module" allows to run different baud rates*).

It consists of a state machine that detects not only the changing from 0 (dominant bit) to 1 (recessive bit) and vice-versa but bits of the same polarity Fig. 2-7. Besides, this machine allows feeding the *Stuff Detectors* and driving the *Input Register* control through the main machine. Also,

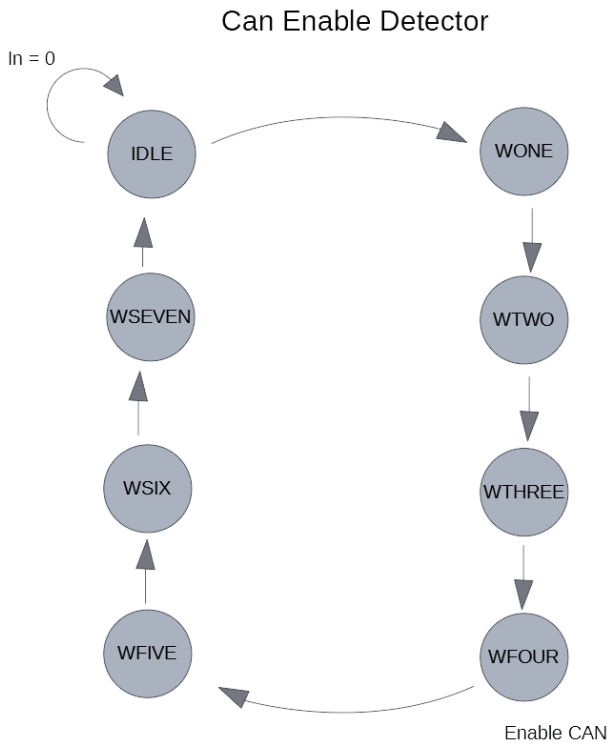

Fig. 2-7 CAN enable the state machine to synchronize the CAN bus and the CAN analyzer.

this module follows a technique that consists of capturing the bit in the middle of its duty cycle. Thus, avoiding the rising and falling edges that may produce metastability (*see appendix A*) to the system Fig. 2-8.

2.3.2 Stuff Detectors

To follow the CAN standard and according to the stuffing technique. There are two elements in charge of stuff bit detection, one for the dominants (0) and one more for the recessive bits (1). Two dedicated stuff bit detectors perform this action which is based on state machines as the *CAN system enable* Fig. 2-9. Each detector oversees the bus to detect a bit changing looking for the characteristic bit pattern of six same bits either dominant or recessive. As soon as any detector reads a stuff bit, flag *stuff enable* is raised which allows the main control unit entry into the STUFF state.

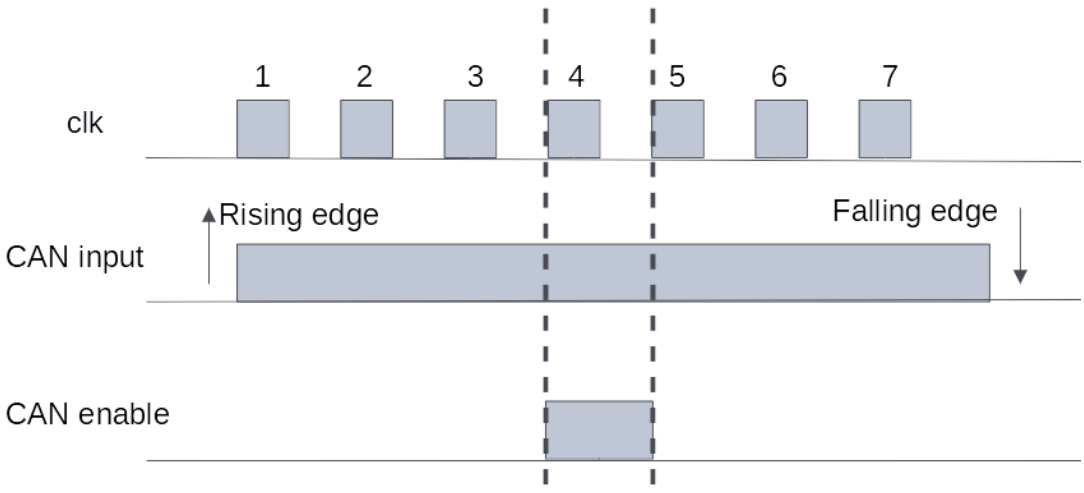

Fig. 2-8 The *CAN enable* module enables the bit frame capture in the middle of each duty cycle.

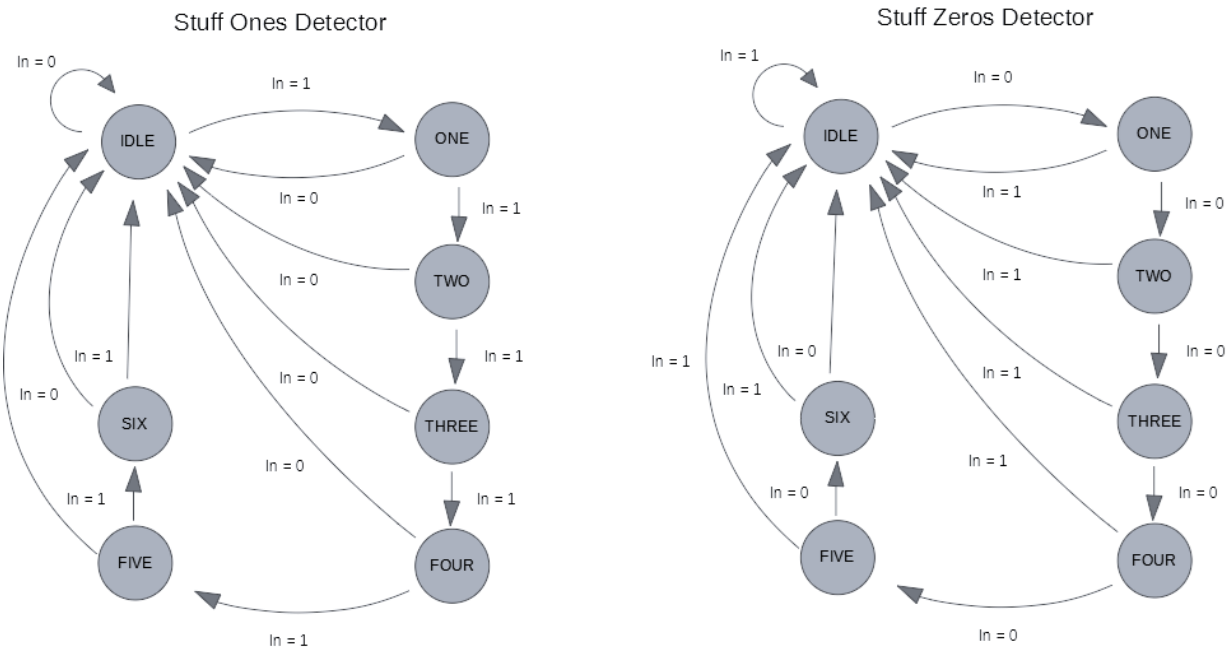

Fig. 2-9 Stuff bit detector state machines, on the left side the state machine corresponds to the recessive bits and, on the right side the dominant bits machine.

2.3.3 **EOF Detector**

The EOF field is the CAN frame's element which helps to determine when to stop catching the frame. Remember that this field is integrated by seven recessive bits, and, in this case, a new detector is required to read this field. Important to realize, this module works separately from the previous *Stuff detectors* because the EOF field is longer than the stuffing patterns. Nonetheless, the set of these three detectors provide the correct operation of the main control unit.

This module counts the consecutive number of ones (Fig. *2-10*) and if this reaches the seven consecutive recessive bits. The *stop flag* is activated, driving the state machine goes into the IDLE state.

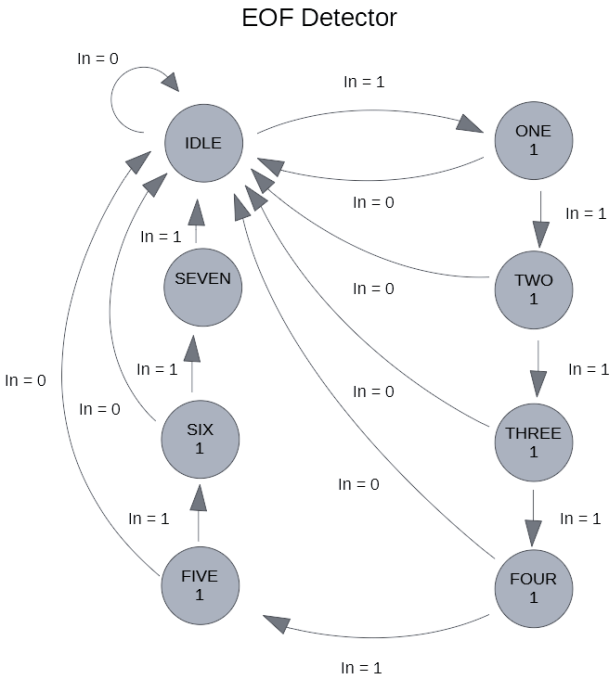

Fig. 2-10 EOF detector

2.3.4 **Main Control Unit**

In combination with the modules described previously the *Main Control Unit* is the module that controls the *Data Path* and provides timing for the data transmission. Fig. 2-11 shows the arrangement and the connections among the state machines that in combination allow the CAN hardware stage to capture the CAN frame.

2.3.5 **Counter Mux Field**

The *Mux field* demands a selector control to determine when to send the next CAN field and avoid losing data. For this reason, a counter helps to change the multiplexer's selector value. This action is through a flag that comes from the *Counter Field* that enables changing the *Counter*

Mux Field value. This module counts from 0 to 4 and when it reaches the maximum value it goes to 0 waiting for the CAN frame.

2.3.6 Counter Field

Due to the CAN information provided from the *Mux Field* is in parallel last step is needed to serialize it. Hence, this counter manages the *Shift Field Register* shifting by counting the 16 bits that compose each field. Keeping that in mind, the maximum value of this counter is from 0 to 15. While the counter is active, it enables the *Field Register* conceding shifting the information to the NiosII input.

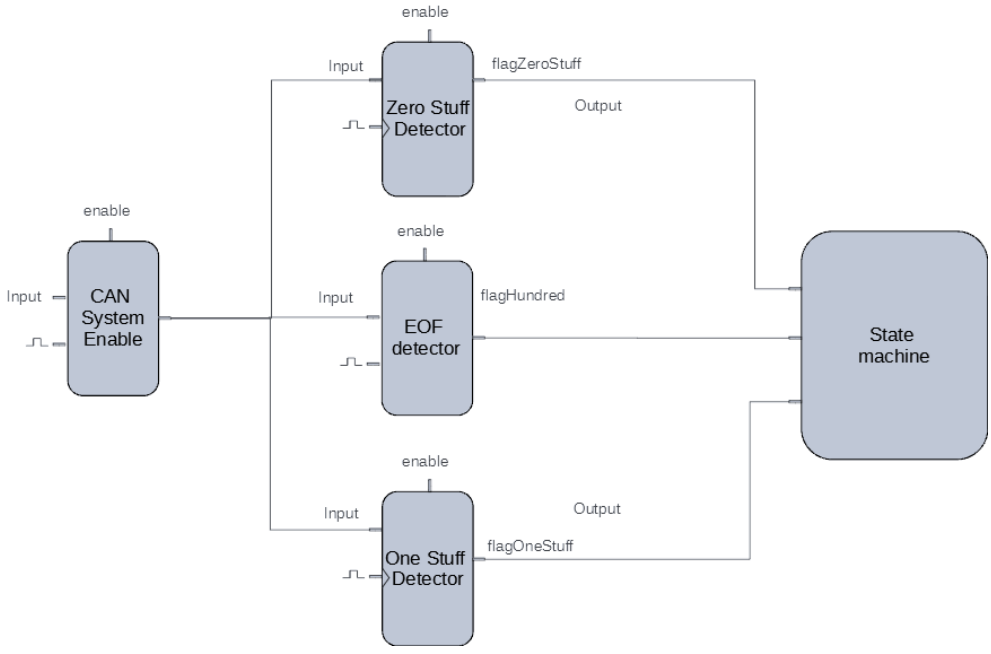

Fig. 2-11. Arrangement of the five state machines that allow capturing the frame from the CAN bus.

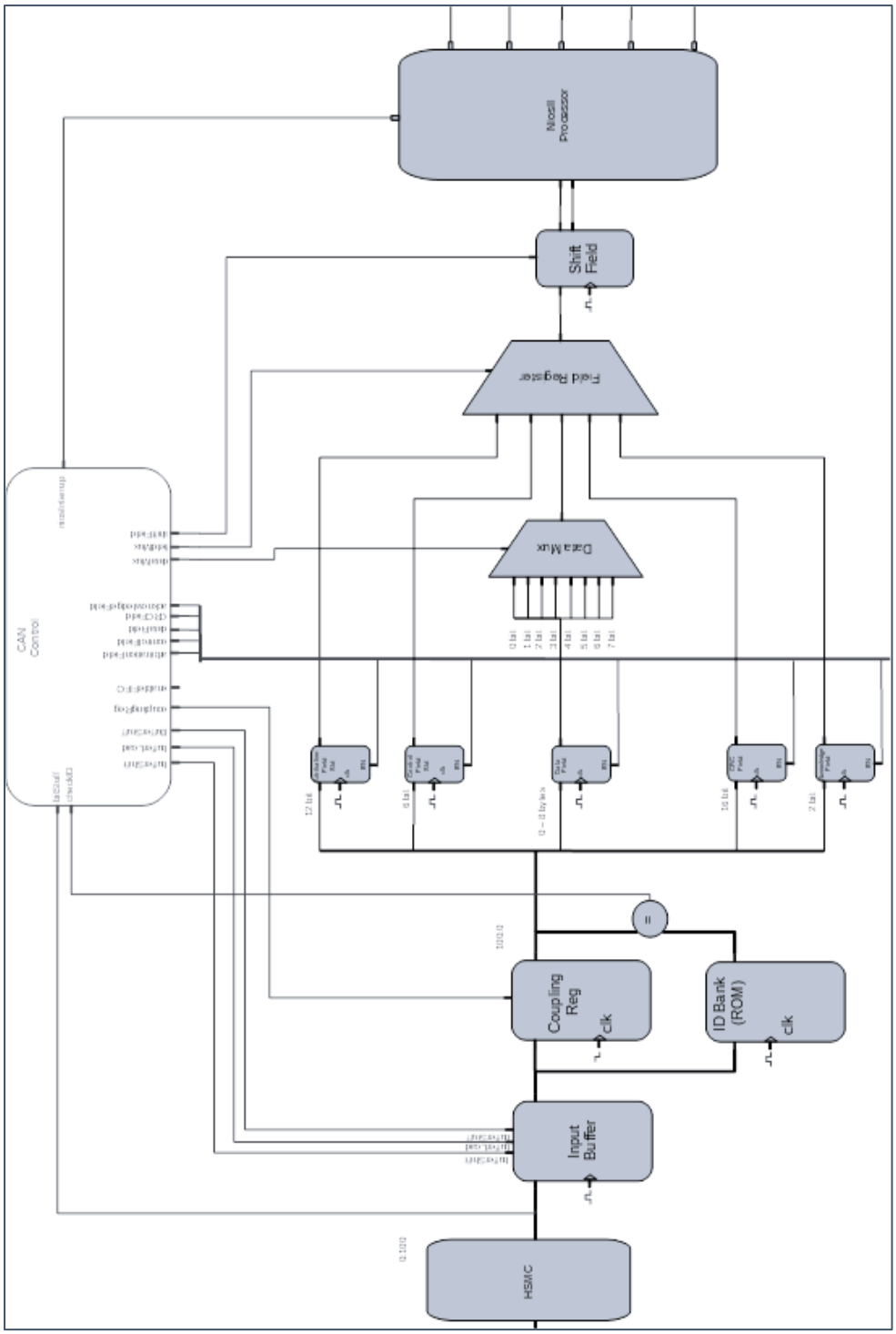

Fig. 2-12 CAN core microarchitecture.

Conclusions

This proposal CAN analyzer as an alternative brings significant benefits such: The FPGA allows the possibility to move the same design into other platforms based on the same technology with minimal requirements or, even embedded the analyzer in a system that is based on FPGA; Also, it is easy to add new features due to *Verilog's* flexibility. As a result, the CAN analysis without the need of a commercial CAN suite from a third-party supplier is a cheaper alternative to licensed software and analyzers available on the market; avoiding increases the PCBs scrap by misjudgment while testing stage due to missing or limitations regarding tools or licenses to make use of the needed features to test the CAN. Therefore, the CAN core streamlines the testing process by reducing the time consumption on each board by providing no dependency on additional hardware or software to perform the analysis which represents a high improvement for delivering the PCB on time for its posterior integration.

Since the CAN analyzer is the first proposal, the next new versions pretend to integrate the development of the functions such as analyzing the frame in the CAN extended version and the TX module to achieve the frame manipulation, in addition to sending diagnostic frames to the modules integrated within the CAN network.

Appendix

A. COMPLEMENTARY INFORMATION FOR THE CAN ANALYZER

NiosII processor

The NiosII[5] is an embedded processor for FPGAs from Intel that allows flexible customization for embedded design. The NiosII makes use of the C programming language which enhances its manipulation and facilitates processing data obtained from the Hardware stage. Also, comes with many Intellectual Properties (IPs) which contain useful features for peripheral management. Thus, the NiosII processor is an excellent complement to the hardware core. Fig. *13* shows the 2 versions of this processor:

- NiosII/e: This is the basic configuration of this processor and its free
- NiosII/f: The performance version optimized which requires the purchase to make use of it

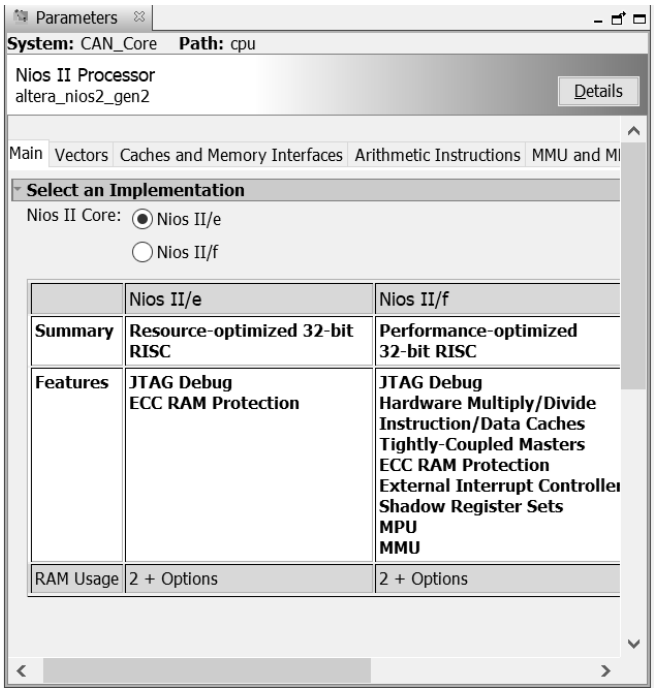

Fig. 13 NiosII processor.

Bibliography

[1] *Road vehicles-Controller area network (CAN)*, 11898-1, 2015(E).

[2] *Road vehicles-Controller area network (CAN)*, 11898-2, 2016(E).

[3] *MCP2551 High-Speed CAN Transceiver*, Microchip Technology Inc., USA, 2007.

[4] *DE2-115 World Leading FPGA Based Products and Design Services*, Terasic Inc, East Dist, Hsinchu City. Taiwan,2013.

[5] *Nios II Class Software Developer's Handbook*, Altera, San Jose, CA, USA, 2015.

Index

A

Acknowledge field, 7
Arbitration field, 5

C

CAN Control, 17
CAN controller, 4
CAN transceiver, 3
Control field, 6
CRC field, 7

D

Data field, 6
Datapath, 11

E

Extended version, 5

N

NiosII, 29

S

SOF, 4
Standard version, 5