HSR: Ethernet Redundancy Protocol for Critical Applications

Guilherme Bessa
Instituto Superior Técnico, Universidade de Lisboa
Lisboa, Portugal
Email: guilherme.bessa@tecnico.ulisboa.pt

Abstract—The control of the electrical substations depends greatly on the reliability of the measured signals and on trustworthy data transfer. Electrical substations and other critical applications require networks resistant to single point failures that provide reduced recovery time after a failure. To provide this type of robustness, the IEC 61850 standard selected the protocols HSR and PRP as redundancy protocols for industrial Ethernet networks in electrical substations. These protocols establish rules of communication, what leads to the obligation of fulfilling a specific functioning of the nodes in HSR or PRP networks.

The implementation of the network nodes requires a link redundancy entity (LRE) that deals with the redundancy issues, making it transparent to the application. For the HSR protocol this implementation must be done in hardware to accomplish the timing requirements of the network. One of the most important requirement of the LRE operation is the ability to make a quick identification and rejection of the redundant traffic. To do so in an efficient way, the non-redundant traffic must be registered in memory for the identification of the redundant traffic, which requires an adequate data structure and search algorithm.

This dissertation proposes an architecture for an FPGA implementation of the HSR protocol. A Verilog description and several behaviour verification tests were written. The implementation of the architecture was simulated and the intended behaviour of the proposed solution was confirmed.

Keywords: Redundancy Protocols; HSR; FPGA; Ethernet; Verilog; Electrical Substations

I. INTRODUCTION

Modern critical industrial applications often require fault-tolerant high availability networks. Industrial Ethernet networks use redundancy protocols to assure that kind of robustness. The best suited protocols for Ethernet critical networks are the HSR (High-availability Seamless Redundancy) and PRP (Parallel Redundancy Protocol) defined in the IEC 62439-3 standard. In a network using the HSR protocol, every node has the switching capability to decide where to send the received packets and must be also responsible to identify and reject redundant data. The aim of the dissertation was to implement in hardware (in a programmable device as FPGA) the entity responsible for the switching and redundancy tasks of the HSR nodes. To accomplish this goal by proposing an architecture and creating an hardware description (in Verilog), the required functionalities have to be studied and listed. A simulation model was also written to test the behaviour of the device.

One of the main obstacles is to ensure rapid data packets classification as redundant or non-redundant. This classification is the result of the comparison between the packet being received and the already received packets, which makes essential the use of an adequate comparison algorithm and an efficient data structure.

This work proposal arose from a project candidature led by Efacec named SCALE - Scalable Centralized Grid Protection, Automation and Control, that was recently approved in February 2021.

II. HSR PROTOCOL

An HSR network is an Ethernet network following HSR protocol and must be composed solely by HSR adapted nodes with two ports capable of sending and receiving data. The nodes are identified by their MAC address and the common HSR node is the DANH (Doubly Attached Node for HSR). This way, the network can be displayed as a ring with two opposite data paths (figure 2). The ring shaped network is the typical HSR network, although other topology variants can be used. Because HSR is an Ethernet protocol, the data packet unit is named frame, and frames can be unicast (with one destination node) or multicast (all nodes are destinations).

Each node can insert frames in the ring, by sending a pair of two equal frames to both ports, making them travel through the opposite paths to find the destination(s) node(s). If a failure occurs in one of the paths, the frame of the other path will still be able to reach its destination. This is how the resistance to single failures is achieved. The destination nodes shall accept the first received frame of the pair and reject the second (the duplicate).

A. Switching decision

When a node receives a frame, one of the following decisions is selected based on the destination and source addresses of the frame - the frame is:

1) Rejected - if the node is the source node of the frame, or if the frame is considered redundant.
2) Sent to the Host - if the frame is unicast and the node is its destination (Dedicated unicast).
3) Forwarded - if the frame is unicast and the node is not its destination (Non-dedicated unicast).
4) Forwarded and sent to the Host - if the frame is multicast and the node is not its source (Multicast).

The switching decision is made during the reception of the frame. If possible, the forwarding operation must start right after the switching decision is available, which means forwarding the frame while it is still being received (cut-through mode).

B. HSR frames

The HSR frames (figure 1) are not standard Ethernet frames, they are only suited for HSR networks because they contain a dedicated HSR tag. The HSR tag is composed by three
fields: sequence number, payload size and path identifier. The sequence number is a frame identifier field that is incremented over time by the source nodes.

C. Life cycle of frames

The insertion and removing of frames in the ring are detailed next.

- **Multicast frames** - Eliminated by the source node: The frame pair is sent by the source node. Every destination node sends the received frames to the Host and also forwards them. Both frames travel the whole ring in opposite directions and are removed by the source node (figure 2).

- **Unicast frames** - Eliminated by the destination node: The frame pair is sent by the source node and has only one destination. The non-destination nodes forward the frames until they reach the destination node where they are removed from the ring. The first frame to arrive is sent to the Host of the node and the second one is simply rejected. If for some reason one of the frames pass through the destination node it will travel the ring until reaching the source node where it is eliminated (figure 3).

- **Circular frames**: In the case of a multicast frame missing the source node or a unicast frame missing both source and destination nodes the frames would be forever forwarded in the ring. These frames are considered circular frames and the nodes must be able to reject them. The difference between duplicated and circular frames is the port from which they were received. The duplicated frame is received in the opposite port of the original frame reception, while the circular is received from the same port as the original.

- **Invalid frames**: Invalid frames are detected by the nodes if any type of erroneous situation occurs. These frames are rejected if possible but sometimes errors are only detected in an advanced stage of the reception, possibly during a cut-through forwarding. In this case, the invalid frame will travel the whole ring until it is eliminated by the source node.

III. STATE OF THE ART

This section aggregates the studies on three distinct areas.

A. Electrical Substations

Communication protocols in electrical substations are defined by the IEC 61850 standard. The electrical substations automation system (SAS) includes a group of protection, control and communication devices, and is usually divided in three levels of devices: level 2 (Station Level), level 1 (Bay/Unit Level) e o level 0 (Process Level) [9]. Between these levels there are the station data bus and the process data bus, that usually are Ethernet networks. These buses have very strict constraints. The choice of the redundancy protocol is done according to the tolerance time the system has before any critical consequence, and for the process and station bus the tolerance times are defined in IEC 61850-5 [5]. According to [8] the majority of the station bus requirements are supported by RSTP and MRP redundancy protocols. On the other hand these protocols can not support the process bus requirements, what makes necessary the use of zero recovery time and zero frame loss protocols: HSR or PRP. The use of these protocols allows interconnection of devices from different manufacturers [6].

B. Improving HSR

The main disadvantage of HSR networks is the great amount of redundant frames. Some works try to reduce the redundant traffic without affecting the features of the protocol.
1) Quick Removing [2]: Quick Removing suggests removing multicast frames from the ring before they return to the source node. The nodes should reject a multicast frame when they already received an identical frame, because this means that every other node has already received one of the frames. The reduction of multicast traffic depends on the number of nodes and tends to 50% with infinite nodes.

2) VRing [2]: Same authors suggest another improvement named VRing (Virtual Ring) which has similarities with the VLAN (Virtual LAN). This idea is intended for non-ring HSR networks, namely the mesh topologies. VRing virtually isolates a group of nodes creating a virtual HSR ring. It is useful when these nodes have a continuous and intense traffic exchange. Once again, the traffic reduction is correlated with the number of nodes in the network. This improvement is focused on topology and path management and not on the functioning of the node, so it was not applied in this dissertation. Additionally, in a mesh topology most of the nodes are not DANHs.

3) Port Locking [3]: Port Locking is intended for networks composed by several rings connected by QuadBoxes (special node for ring interconnections) that function as ports. The goal of Port Locking is to block the crossing of frames to rings where they do not have destination nodes.

4) DVP (Dual Virtual Paths) [10]: The goal is to create virtual paths between the network nodes so that unicast frames are directly sent from source node to destination node. A DVP tag is added to the frames to distinguish DVP traffic from normal HSR traffic. This approach is intended for complex networks using QuadBoxes. The QuadBoxes should forward DVP frames directly to the destination, so DVP frames do not need to be duplicated.

Of these four ideas, only Quick Removing was considered on the elaboration of the device architecture presented in this work. The other three ideas were not used because they focus on the topology of the network and path management and not on the node behaviour.

C. FPGA Implementation

As already discussed, the redundancy identification algorithm and its associated data structure should be well adapted to the necessities of the network. Research on this topic was done and a theoretical analysis was found in [4]. The author analyses three algorithms:

- Circular buffer;
- Hash table with open addressing and aging tag;
- Hashing combined with circular buffer;

An environment with several nodes was created in software (C++) and the three algorithms were tested by simulation. To evaluate the performance of the algorithms the concept of frame rejection ratio was introduced. The ratio is 100% if the number of rejected frames is equal to the number of accepted frames, which means that from every pair, one frame was accepted and other was rejected. The greater the number of nodes, the worst the rejection ratio because it is harder for the nodes to keep the record of every frame. The author concluded that the hashing combined with the circular buffer is the best suited algorithm because it has good performance results and is more appropriate for an hardware implementation. The chosen algorithm produced a simulated 100% rejection ratio for ring networks with less than 64 nodes. This algorithm was adopted in this work and the architecture presented in the next chapter was also based on the architecture proposed by the author. The details of the algorithm are explained in section IV-D.

IV. Architecture

A. Input and output interface

The device should be implemented in FPGA and should be connected to an Ethernet network. The connection to the network is made through two ports, A and B, capable of sending and receiving frames. From the perspective of the device, the node connections can be considered to be made with two lines A and B, respective to the ports. The device is also connected to the Host through another port. The 3 ports can be interpreted as 6 ports if they are divided in sending and receiving ports (TX_A, TX_B, RX_Host) and receiving ports (RX_A, RX_B, TX_Host). The communication between the device and the Ethernet PHYs is made with a GMII (Gigabit Media-Independent Interface) interface which supports an Ethernet of 1 Gbit/s. GMII was also considered for the communication with the Host.

B. Reception and processing behaviour

1) Receiving phase: Over time, a stream of data is constantly received from lines A and B, being processed separately but in a similar way. Frames are recognized by their preambles. After a frame is recognized, it starts to be stored in a reception buffer where the frame stays until a switching decision is made. The frame fields are analyzed in the processing blocks and the decisions are achieved after some time (figure 4).

![Figure 4. Circuit diagram.](image)

The processing block uses the destination address (DA) and the source address (SA) fields to evaluate how to treat the receiving frames according to the subsection II-A. The device can understand if the node is the destination or the source by comparing the SA and DA with the MAC address of the node. Multicast frames are identified by a specific DA value. The received frame shall be rejected and eliminated if the node is the source node, but also if there are errors or if the frame is identified as redundant (duplicated or circular). This identification will be explained later.
2) Sending phase: After the switching decision is taken, if the frame is to be sent, whether it should go to the Host or to the network, it is always first sent to one or two outgoing buffers. Each sending port has an outgoing buffer where the frames that will be sent in that port are stored. These buffers coordinate the sending of frames, one at a time, through the sending port. If, for example, a frame received without errors on line A is multicast and was not identified as duplicate or circular, the switching decision is to send it to line B and to the Host. Then, the frame is sent to the outgoing buffer of port B and to the outgoing buffer of the Host port. If it was a non-dedicated unicast frame, it would just be sent to the outgoing buffer of port B, and if it was a dedicated frame it would be sent only to the Host buffer. The transmission from these buffers out to the exterior of the device is done through the interface output signals. The forwarded frames should be sent in cut-through mode whenever possible, which implies that as soon as the frame begins to be loaded in the outgoing buffer, it also begins to be sent to the output interface. When, on the contrary, the frame is sent without cut-through mode (store-forward mode) the frame is fully loaded in the output buffers before being transmitted. The relationship between the various transmissions and the cut-through mode is defined as follows:

- Forwarding a frame to the opposite line is usually done in cut-through mode, risking an error detection during the rest of the reception. In case of error detection after the start of transmission, an error sequence is used (more details in section IV-E2).
- Sending frames to the Host is always done without cut-through mode, to avoid sending frames with errors, since the delay does not affect the network traffic.
- Sending frames from the Host to the network lines does not use cut-through because the transmission priority is always given to the forwarding frames. A frame from the Host is only sent when there are no frames to forward. If a frame is being received on the opposite line, the device should wait until it is known whether that frame will be forwarded or not, before sending any Host frames.

C. Host frames

The Host relationship with the redundancy and switching device of the node is not defined in the standards and will depend on the real case. In this work, it was defined that the Host frames are received using a GMII interface, similarly to how frames are received from the network lines. The Host frames are sent to the outgoing buffers of lines A and B simultaneously. The only operation performed on these frames is the CRC error check. In this work it is also assumed that the Host always sends and receives the frames already with the insertion of the HSR tag.

D. Duplicates and circular frames identification

To identify the redundant frames (circulars and duplicates), each node has to store a piece of information from each received frame, using one or more memory tables (the rejection tables). Frames are uniquely identifiable by the set of fields destination address (DA), source address (SA) and sequence number (Seq), that must be stored. From the point of view of the node, a redundant frame is a frame whose set of parameters “DA SA Seq” is already registered in a memory table. In order to register the Ethernet port through which frames are received, the nodes independently store, in different tables, frames received from line A and line B. When the reception of a frame starts, the tables are searched, and if the frame is found, then it is rejected. If the frame is not found in the tables, the reception and the processing can progress. Non-redundant frames received without errors are registered in the memory.

Each line has three rejection tables: one for circular frames, one for dedicated unicast duplicates and another for multicast duplicates. The tables for duplicates can reduce the entry size from “DA SA Seq” (14 bytes) to “SA Seq” (8 bytes) because dedicated unicast duplicates all share the same DA, as well the multicast duplicates. The switching and redundancy device element of each node will then have six tables to register the incoming frames: three tables associated with line A and three tables for line B, two of them for duplicates (dedicated unicasts and multicasts) and one for circular frames (all kinds).

The process of identifying duplicates and circular frames begins after obtaining the DA, SA and Seq frame fields of the received frame. Depending if the frame is multicast or unicast, the duplicates table for multicast/unicast frames of the opposite line of reception is searched, and also the circular frames table of the reception line is searched. If an error is detected during the identification, the process is aborted. The search in the tables is done by comparing the set of identification fields (DA, SA, Seq or SA, Seq) with the entries in the tables. The frame is rejected if there is a correspondence between the contents of the tables and the identification fields. If the frame is not registered in the tables, then it is sent according to its switching decision. If the receiving frame is not found, it must be written in the reception line’s table for duplicates and table for circulars. The result of the identification is achieved during the reception of the frame, but the tables are updated only after the frame is completely received without errors.

The information in the tables is constantly updated due to the arrival of new frames. To implement this system it is necessary an adequate data structure and an efficient search algorithm. As mentioned in section III-C, several structures were studied in [4] and the best suited one was adopted in this work. The author proposed an hash table combined with circular buffers.

In an hash table there is an hash function that maps keys into addresses of entries in the table, or buckets. The structure proposed by the author consists of assigning to each bucket a circular buffer with a certain number of entries named bins. The number of buckets was defined as $m$, and the number of $bins$ in each bucket is defined as $max\_bin$. In a writing process, for example, a bucket is mapped and the writing takes place in one of the $bins$ belonging to that bucket. The key used and the written content are the identifying fields “DA SA Seq” or “SA Seq”. The $bin$ to write is always the $bin$ that comes after the last written $bin$ in the bucket. This way, if all the $bins$ are occupied, the overwritten $bin$ is always the one with the oldest content. To keep track of where to write and read, there is a write pointer and a read pointer for each bucket. The buckets are circular buffers because the write and read pointers increment sequentially, pointing to consecutive $bins$ inside the bucket, and returning to the first $bin$ after reaching the last one.
In the search process, after the key maps the bucket, the key is compared with the content of the bins until an equal entry is found or until every bin was read. In the worst case (frame not found), the number of reads and comparisons is \( \text{max}_\text{bin} \).

Figures 5 and 6 illustrate the proposed structure and the concepts of bucket and bin. Figure 5 shows the mapping of the frame fields in a bucket address using the hash function. In figure 6, it should be noted that the buckets are separated by dashed lines. In fact the buckets are abstract structures achieved by the circular pointers algorithm.

In [4] the author analyzed several solutions for the implementation of the tables structure. Those solutions were simulated separately in software. In each test, a ring network was simulated in which all nodes used the same tested structure. The algorithm and the structure presented above were considered the best suited, due to good results and less hardware implementation complexity, when compared to other solutions. According to the results of the simulations, this algorithm provides 100% duplicate rejection ratio in rings that do not exceed 64 nodes. The simulation parameters \( m \) and \( \text{max}_\text{bin} \) were adopted for this work. Table I shows the parameters and the key/entry’s content for each type of table.

<table>
<thead>
<tr>
<th>Table</th>
<th>Key and content</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duplicates (unicast)</td>
<td>SA SEQ</td>
<td>( m=64 ), ( \text{max}_\text{bin} = 4 )</td>
</tr>
<tr>
<td>Duplicates (multicast)</td>
<td>SA SEQ</td>
<td>( m=32 ), ( \text{max}_\text{bin} = 4 )</td>
</tr>
<tr>
<td>Circular frames</td>
<td>DA SA SEQ</td>
<td>( m=64 ), ( \text{max}_\text{bin} = 4 )</td>
</tr>
</tbody>
</table>

Figure 5. Structure of the table and its access. A key \( k \) composed by the frame fields is used to map the bucket 31, through the hash function \( H(k) \) [4].

**E. Other features**

1) **Comparison window**: When two equal frames (from the same pair) arrive in a very short time interval, it may be the case that the identification process of the first frame is not yet completed when the second frame arrives. Therefore, there is still no record of the first frame in the tables, leading to the non-rejection of the duplicate. To circumvent this problem, a comparison window is used. The comparison window works by establishing a time window in the processing of each line that starts when the “DA SA Seq” key is obtained and ends when a new key is obtained. During this window, if a frame with the same key arrives on the opposite line, that frame is rejected without the need to search in the tables.

2) **Error sequence**: If an error occurs during the reception of a frame that is already being forwarded in cut-through mode, the source node of the frame is marked by the receiving node in a NodesTable, so that cut-through is not performed in the next frame received from that source. The processing blocks of each line have a NodesTable, which includes an entry for every network node reporting if the most recent frame received from that node had an error. After the complete reception of each frame, the receiving node updates the error/no error record in the NodesTable. Additionally, when an error occurs during cut-through, an error sequence is sent, indicating to the following nodes that this frame has already been detected as invalid. From that moment on, the frame will always be forwarded together with the error sequence, until it is rejected by the source node. This way, only the node that identified the error will not perform cut-through on the next frame from the same source, reducing the effect on the traffic flow. However, before registering the error in the NodesTable, it is necessary to wait for a while and try to detect a possible error sequence that has already been added by another node. If the sequence is detected, the NodesTable is not updated.

A frame received when the error bit for its source node is set will not be sent in cut-through mode, being first totally stored in the outgoing buffer and, if this time there is no error, it is sent and the table is again updated to clear the error indication, so that the next frame is sent in cut-through mode, if possible. This functionality is relevant in case of a defective node in the network, rejecting its frames in the adjacent nodes, as it prevents the contamination of the network traffic with invalid frames.

---

**V. IMPLEMENTATION**

**A. Introduction**

After defining all the specifications and features, a hardware description was written in Verilog language. The description was synthesized for FPGA and a testbench was written to simulate the circuit and test its behaviour. In all these stages, the Vivado Design Suite program provided by Xillinx was used. In the following sections, the various hardware approaches to the fundamental parts of the circuit will be detailed. The most used resources, besides conventional logic, were:

- **FIFO** tables: a data organizing method that implements the “First In First Out” concept, in which the read data is always the oldest entered in memory. Data written is organized in an output queue. Whenever a writing occurs,
the content goes to the end of the queue. Following the same logic, whenever a reading occurs, the data at the front of the queue leaves the table. FIFO memories are used in the circuit to implement the buffers, since the intended operation is exactly as described above.

- **DPRAM** (Dual-Port Random Access Memory): a RAM memory that allows simultaneous reading and writing operations. All tables in the device, including FIFOs, are DPRAMs.
- **FSM** (Finite State Machine): a computation model in which a certain finite number of states and the transition conditions between states which depend on the inputs of the machine are defined. State machines were used in this work to detect sequences (such as the preamble of the received frames) or to define various operation modes for a single block.

### B. Circuit

1) **Scheme**: The circuit was divided into blocks that communicate with each other, each block having a specific function. Some more complex blocks contain sub-blocks. Figure 7 shows a block diagram of the entire circuit.

The diagram represents the circuit behaviour. Frames received in lines A and B are received at the reception ports and are processed and evaluated by the RMR and RPB blocks of the respective input line. Lines A and B behave in a symmetrical way. The data output of the RMR blocks are connected to the input of the forwarding outgoing FIFOs of the opposite lines, as well as to the input of the outgoing FIFOs to the Host (RX FIFOs). The Host frames are sent directly to the outgoing sending FIFOs. The transmission of frames from outgoing FIFOs through the sending ports is controlled by the sending control blocks. For the sake of clarity, the NodesTables are not present in the diagram.

2) **Interface signals**: The circuit input and output signals correspond to the signals defined by the GMII interface. Table II describes the reception (circuit inputs) and sending/transmission (circuit outputs) signals of the GMII interface [1]. The data signals have 8 bits, which means that 8 bits (1 byte) can be received/sent on every clock cycle.

Figure 8 shows the top level circuit input and output signals. Each port has identical sending and receiving clocks, that is, only three distinct clocks were used, \( rx_{-}clk_A \), \( rx_{-}clk_B \) and \( host_{-}clk \).

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Receiving Signals</strong></td>
<td></td>
</tr>
<tr>
<td>RXCLK</td>
<td>Reception clock</td>
</tr>
<tr>
<td>RXD[7:0]</td>
<td>Received data</td>
</tr>
<tr>
<td>RXDV</td>
<td>Indicates valid data</td>
</tr>
<tr>
<td>RXER</td>
<td>Indicates errors</td>
</tr>
<tr>
<td><strong>Transmission Signals</strong></td>
<td></td>
</tr>
<tr>
<td>GTXCLK</td>
<td>Transmission clock</td>
</tr>
<tr>
<td>TXD[7:0]</td>
<td>Transmitted data</td>
</tr>
<tr>
<td>TXDV</td>
<td>Enables transmission</td>
</tr>
<tr>
<td>TXER</td>
<td>Indicates transmission errors</td>
</tr>
</tbody>
</table>

![Figure 8. Input and output signals of the device grouped by ports.](image)

3) **RMR (Receiving Main Routine)**: This is the block that first uses the signals received by the GMII interface. In this block (figure 9), frames are identified through the preamble, and then stored in the RMR FIFO (circuit receiving FIFO, equivalent to the receiving buffer, explained in chapter 4) and errors are constantly being checked. Depending on the decision signals generated by the RPB block, the frames are either sent from the RMR block or rejected. In the event of an error during a cut-through transmission, the RMR is responsible for attaching the error sequence to the frame. The RMR functions are distributed over several sub-blocks. All RMR sub-blocks of line A and B use the reception clock (GMII) of the respective line, in order to process data byte by byte synchronously. The RMR sub-blocks are detailed next:

- **MII Processing**: Detects the preamble and activates a start signal.
- **Valid Register**: Indicates valid frame processing.
- **CRC Check**: Performs the cyclic redundancy check.
- **Error Assertion**: Management of error signals.
- **RMR_FIFO**
- **FIFO Write/Read Control**: Controls FIFO operations.
- **Error Sequence Detector and Appender**

4) **RPB (Receiving Processing Block)**: After the preamble is identified by the RMR block, the start signal is also sent to the RPB block, allowing it to acquire the frame fields “DA SA Seq” that make up the access keys to the rejection tables. This block makes two decisions in two independent processes, using the acquired key:
1) Switching decision - Decides the destination of the frame that is being received according to the destination and source addresses (the switch decision signal carries this decision).

2) Redundancy identification - Searches the received frame in the duplicates and circular frames rejection tables and, if a match is detected, asserts an entry found signal. At the end of the search, the sending decision signal is sent to the RMR block, indicating that a decision has been made and the RMR_FIFO starts immediately sending the frame out of the block.

The switch decision signal indicates which FIFOs should receive the frame coming out of the RMR block. The frame is only received by the destination FIFOs only if the entry found signal is not activated. Otherwise, dummy reads (to clean up the RMR_FIFO) are performed. In summary, the most important output signals are:
- sending decision - when activated the RMR_FIFO begins to read, to send the frame out of the RMR block.
- switch decision - decides the destination of the frame.
- entry_found - if activated, it does not allow the writing of the outgoing FIFOs.

The RPB block uses the reception clock of the respective input line. A diagram of the RPB block is illustrated in figure 10. The RPB sub-blocks are detailed below, and afterwards the frame rejection tables addressing is explained.

- Receive Processing: Makes the switching decision, composes the keys and computes the mapped keys.
- Interface RP-DPRAM: The Interface RP-DPRAM sub-block implements the interface between the RPB block and the DPRAM frame rejection tables. It is responsible for controlling the write and read addresses and the permissions.

The size of the tables is max_bin × m (4 × m in this implementation). The buckets represent table zones, that is, groups of 4 entries. The mapped key represents the bucket address, because it does not address a specific entry, but a group of 4 entries whose addresses share the same combination of most significant bits.

To address a bin, two more bits are necessary to identify the position of the read/write pointers. These two bits are named the bin address. Therefore, the full addresses for each entry in the tables, are composed by (mapped key; bin address), in which the mapped key selects the bucket, while the bin address selects the bin within the bucket.

The read and write pointer positions of all buckets in all tables will keep varying over time, depending on the operations each bucket is subjected. This requires storing the position of the pointers that represent the two-bit addresses, identifying a bin out of four. The storage solution created was the creation of bin tables, auxiliary to the rejection tables, which store the addresses of the bins, i.e. the current position of the write and read pointers of each bucket. For each rejection table there is an auxiliary table of bins containing the bin addresses to access the rejection table. The auxiliary tables have the same number of buckets (m) as the correspondent rejection tables, but only one entry per bucket. Each entry contains only four bits: two bits corresponding to the position of the read bin and the other two bits corresponding to the position of the write bin. As the bucket structure is the same between the main and auxiliary tables, the access to their buckets is done in the same way. This means that the mapped key maps the buckets of both the rejection table and its auxiliary bin table. To obtain the full read/write address, the mapped key is used to obtain the value of the write and read pointers (bin addresses), and with these values complete the full write and read addresses of the rejection table.

Because the identification process involves searching and writing in different rejection tables, the bin tables reflect that: both the read bin and the write bin addresses are obtained from the same table of bins, although they are used in different rejection tables.

This means that the bin tables are not really associated with the corresponding rejection tables, but actually associated with the three possible operations of each row:
1) Reading from the unicasts table of line A/B and writing to the opposite unicasts table (B/A).
2) Reading from the multicasts table of line A/B and writing to the multicasts table (B/A).
3) Reading from the A/B circulars table and writing to the same circulars table (A/B).

The bin tables are auxiliary tables for these operations. Six tables are required because there are six operations available. Figure 11 shows the operations that a search process on line A can perform. If the search process took place on line B, the operations would be symmetrical. Each search process always does two of these three operations: one for circulars (3) and one for duplicates (1 or 2). The two operations always occur in parallel. In the figure, only three of the six bin tables are represented because only the search in line A is illustrated. The search process algorithm containing the two parallel operations (that involve reading and then writing) is explained below step by step.

**Search Process Algorithm:**

I. Using the mapped keys received by the Receive Processing block, the auxiliary bin tables for both operations are read.

II. For each operation, the obtained read and write bins are added to the mapped key, creating the read and write addresses for the rejection tables: (mapped key; read bin address) and (mapped key; write bin address). Read permission of the rejection tables is activated (read_enable signal).

III. For each operation, the result read is compared with the key. If any result is equal to the key, the reading process is interrupted and the entry_found signal is activated. If both results are different, the read addresses are changed by incrementing the bin addresses. The results read are compared again and the process repeats if the frame is not found.

a) If after the fourth and last read address the frame was still not found, the reading process is inter-

rupted, the entry_not_found signal is activated and the key is written in the write address of the writing tables of both operations. However, if the switching decision of the frame is to just forward it, the key should only be written in the circular frames table, because the destination address is not the node’s MAC address and therefore it is not a duplicate.

IV. At the end of the process the position of the pointers is always updated in the bin tables. For each of the two read operations:

- If the frame was found, the read bin address is incremented in relation to the position found. The write pointer is not changed.
- If the frame was not found, the write bin address is increased and the read bin address must return to its position before the reading operation.

5) Outgoing FIFOs and Control: The three control blocks are the RX_Control (port Rx_Host) and TX_Control A and B (ports Tx_A and Tx_B). Each one controls the transmission and reception of the outgoing FIFOs of the respective ports. It is relevant to note that these controllers work with the three different clocks: two clocks for the receiving routines and another clock for the sending routine. On the network ports, the Forwarding FIFOs receive the frames from the RMR of the opposite line, and the Sending FIFOs receive the frames from the Host. On the Host port, the frames from RMR A/B are sent to the RX_FIFO A/B. Outgoing FIFOs can contain multiple stored frames, so a mechanism is needed to ensure that each frame is sent separately. So, auxiliary FIFOs were created, one for each output FIFO. During reception, the control block counts the number of cycles the frame takes to be fully received. After a frame is fully stored in the main FIFO, an entry is written in the corresponding auxiliary FIFO, indicating the size of the stored frame and whether errors have occurred or not (figure 12). Before sending a frame, the control block consults this information by reading the auxiliary FIFO and, if there is no error indication, proceed with the transmission during the number of cycles registered. When there is an error indication, the FIFO must use the dummy read technique for that frame. This way, the frame is removed from the FIFO, but is not actually sent to the GMII interface. The controller also counts the number of frames in the controlled FIFOs. Therefore, the action of the control block is to manage the transmission of frames one at a time from the two controlled FIFOs through the associated port, respecting the forwarding priority and sending interval. It also controls the cut-through mode and the dummy reads for frames with errors. When using the cut-through mode, the auxiliary FIFOs are not used.

C. Synthesis and implementation

After completing the hardware description in Verilog language, the synthesis and implementation phases were followed, which are part of the project flow established by the Xilinx Vivado Design Suite software that was used in this work. The software allows several project flows for FPGA. The flow used was the traditional RTL (Register Transfer Level) to bitstream flow. An FPGA bitstream is a file that contains information for programming the FPGA. For synthesis and implementation, the target device was an FPGA from the
Artix-7 family with the reference xc7a15. This family, among the most recent, is one of the cheapest and is sufficient for the intended implementation. The RTL design (result of the Verilog description) is transformed into a lower abstraction netlist in the synthesis. Next, the implementation phase makes positioning decisions to implement the circuit on the target device, transforming the netlist into a project ready for the bitstream generation. The temporal analysis after implementation assesses whether the temporal behavior of the circuit meets the requirements [11]. In addition to the temporal analysis, the implementation provides an FPGA resources usage summary (figure 13).

D. Timing Analysis

The Static Timing Analysis (STA) analyzes the paths the signals take between the inputs and the outputs of the circuit, and verifies the propagation delay along each path. For each path, it checks whether or not there is a violation of the requirements. In Vivado Design Suite the result of this verification is a slack in seconds for each path, that represents the difference between the timing requirement and the actual path propagation time. A negative slack indicates that the path violates the timing requirement for a certain amount of time, while a positive slack indicates how much the timing requirements could decrease. When the system meets the requirements there are no negative slacks, and the path with the lowest slack is considered the critical path, that is, the path that is the closest to fail the timing requirements. The critical path can be used to calculate the maximum frequency (fmax) for operation of the circuit, given by equation 1, where T is the clock period and WNS (Worst Negative Slack) is the critical path slack. The timing analysis reported a WNS of 0.967 ns. The circuit clock frequency is 125 MHz, which corresponds to a period of 8 ns, so the maximum frequency is 142.19 MHz, higher than the target frequency of 125 MHz.

\[
f_{\text{max}} = \frac{1}{(T - \text{WNS})} = 142.19 \text{MHz}
\]  

VI. SIMULATION AND RESULTS

For testing the implementation, a simulation environment was created based on a testbench described in Verilog. The Xilinx Vivado Design Suite was used for the simulation. The testbench defines the variation of the input signals over time, while the response (output signals) are obtained and subsequently analyzed. Several different frame creation routines have been written in order to be able to carry out tests for specific features. The testbench considers the following assumptions for all tests:

- The tests simulate a node in an HSR ring network.
- Multicast frames are all broadcast frames (all nodes in the network are destinations).
- The frames must be sent through network ports with a time interval defined by the network rules. This interval must be greater than the Interpacket Gap of the Ethernet frames, which consists of a minimum of 12 octets of meaningless data after the frames are sent. An interval of 31 cycles (248 ns) was chosen.

The tests focused on Reception, Transmission, Redundant frames and Invalid frames, verifying nearly all features proposed. The tests are presented next:

1. Reception of unicast frames dedicated to the node: Different frames are received and correctly sent to the Host.
2. Reception of non-dedicated unicast frames: Different frames are received and correctly forwarded in cut-through mode.
3. Reception of multicast frames: Different received frames are correctly sent to the Host and forwarded in cut-through mode.
4. Transmission of Host frames: A frame of each type is received from the Host and sent through the A and B ports.
5. Forwarding priority test: Consists in adding to test 4, some frames being received in the A and B lines. If these network frames should be forwarded, the transmission of the Host frames does not start immediately. If the network frames should not be forwarded then the transmission of the Host frames starts immediately (exactly as test 4).
6. Duplicates rejection test: Two frames are received at port A and then after a while an identical set of frames is received at port B, what leads to the complete rejection of the second set. Only dedicated unicasts and multicasts are tested because non-dedicated unicasts are not stored in the rejection tables. This test is shown on figure 14 to give an example of the results of the tests.
7. Comparison window test: Repetition of test 6 but both sets of identical frames are received simultaneously. Again, the second set is rejected. This time each type of frames is tested because non-dedicated unicasts are rejected by the comparison window algorithm.
8. Circular frames rejection test: Several frames are received at port A and then after a while, a set of identical frames is received at port A, what leads to the complete rejection of the second set. Each type of frame is tested.

9. Invalid size: Oversize and undersize frames are received and correctly rejected.

10. $rx_{-}er$ error before forwarding starts: The GMII error signal $rx_{-}er$ is activated during an early stage of the reception of several frames. The frames are correctly rejected.

11. $rx_{-}er$ error after forwarding starts: The $rx_{-}er$ signal is activated during the reception of a frame that is already being forwarded in cut-through mode. The transmission stops and the error sequence is sent. The NodesTable is updated. Another frame with the same source as the first but without errors is then received at the same port and is not sent in cut-through mode. The NodesTable is again updated. One last frame is received at the same port as the others and is correctly sent in cut-through mode.

12. Error sequence: a frame with a error sequence appended is received. The device forwards the frame in cut-through mode. When the error sequence is detected the transmission stops and the error sequence is sent. The NodesTable is not updated because another node already detected the error. A frame with the same source as the first is then received at the same port and is correctly sent in cut-through mode.

![Figure 14](image-url). Input and output signals of test 6. The frames are identified by orange letters, ‘UD’ for the unicast frame dedicated to the node, and ‘M’ for the multicast frame. The frames are received first at port A and after a while the duplicates are received at port B, which are rejected. The accepted frames are sent according to their switching decision.

VII. CONCLUSION

A. Conclusions

This work presents the FPGA implementation of a device that includes the Link Redundancy Entity (LRE) for a DANH node to be used in an HSR ring network.

First, the set of functionalities and operating requirements of the node was listed, and solutions for rejecting redundant frames were also analyzed. The node must follow the communication rules of the HSR protocol and should forward the received frames as soon as possible (using the cut-through technique). It also has the ability to detect and process frames with errors. The rejection of redundant frames required an adequate data structure and algorithm. The chosen algorithm uses an hash table combined with circular buffers which was proposed by [4].

According to the requirements, a system architecture was detailed and a hardware description for the FPGA implementation was developed.

The circuit described in Verilog was simulated with a set of tests that confirmed the correct operation and the fulfillment of the requirements defined in the standard.

B. Future Work

Looking ahead, the work should be continued by conducting tests on FPGA. In a first phase, these tests may be carried out in a laboratory environment, using FPGA development boards with two Ethernet ports and some form of communication with a Host. For basic tests, the Host may be a processor embedded in the FPGA, with, for example, communication with the existing CPU on the PC. To ensure more rigorous implementation, it would also be important to acquire the IEC 62439-3 standard, which defines the HSR.

A feature to be implemented in the future is the ability to insert the HSR tag in the frames received from the Host, and to remove the HSR tag from the frames received from the network. It can be useful to increase the transparency of the protocol, however it increases the complexity of the hardware. If transparency is not an essential factor, this task is very simple to perform in software and therefore can be easily performed by the Host.

The approval of the SCALE project in February 2021 ensures that the work will continue, aiming to create a product ready to use in real applications.

REFERENCES


