INSTITUTO FEDERAL DE SANTA CATARINA

MÁRIO ALLAN LEHMKUHL DE ABREU

Análise e validação do funcionamento da recepção das mensagens PTPv2 pelo bloco de *hardware* HA1588

São José - SC

Julho/2023

ANÁLISE E VALIDAÇÃO DO FUNCIONAMENTO DA RECEPÇÃO DAS MENSAGENS PTPV2 PELO BLOCO DE *HARDWARE* HA1588

Monografia apresentada ao Curso de Engenharia de Telecomunicações do campus São José do Instituto Federal de Santa Catarina para a obtenção do diploma de Engenheiro de Telecomunicações.

Orientador: Prof. Roberto de Matos, Dr.

São José - SC

Julho/2023

Mário Allan Lehmkuhl de Abreu

Análise e validação do funcionamento da recepção das mensagens PTPv2 pelo bloco de *hardware* HA1588

Este trabalho foi julgado adequado para obtenção do título de Engenheiro de Telecomunicações, pelo Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, e aprovado na sua forma final pela comissão avaliadora abaixo indicada.

São José - SC, 27 de julho de 2023:

Prof. Roberto de Matos, Dr.
Orientador
Instituto Federal de Santa Catarina

Prof. Marcos Moecke, Dr.
Instituto Federal de Santa Catarina

Prof. Eraldo Silveira e Silva, Dr.
Instituto Federal de Santa Catarina

As raízes dos estudos são amargas,
mas seus frutos são doces.
Aristóteles

AGRADECIMENTOS

Primeiramente gostaria de agradecer a Deus pela vida e saúde que possuo. Depois aos meus pais, irmão, irmã e aos familiares pelo incentivo e suporte. A todos os amigos, colegas e professores que contribuíram durante todo o curso. Agradeço também ao meu orientador por aceitar me orientar, pela boa metodologia de pesquisa e por sempre incentivar pela busca de novos conhecimentos. Assim agradeço a todos, pois através de todas essas ações foi possível a realização desse TCC.

RESUMO

A sincronização de relógio é um problema clássico de sistemas distribuídos. Com a expansão de aplicações no campo de *Internet of Things* (IoT), cidades inteligentes, entre outras aplicações de uso intenso da rede *Ethernet* para sensoriamento e controle, o sincronismo preciso do relógio pode ter papel fundamental dependendo da aplicação. Cada cenário possui seus requisitos de sincronização de tempo e protocolos dedicados para suprir essa necessidade. No contexto da sincronização de circuitos digitais, vem se tornando popular o uso do padrão do *Institute of Electrical and Electronic Engineers* (IEEE) 1588-2008, conhecido também por IEEE 1588v2 ou *Precision Time Protocol version 2* (PTPv2), o qual utiliza a *Ethernet* e o modelo principal-secundário na sincronização dos relógios a partir de troca de mensagens padronizadas. Adicionando o suporte de *hardware* ao protocolo, consegue-se alcançar uma precisão de nanosegundos no sincronismo de dois relógios separados por grandes distâncias, sendo uma solução de ótimo custo-benefício pois reaproveita a rede *Ethernet*. Para atingir tal precisão é necessário que os pacotes do protocolo com as mensagens PTPv2 sejam analisados assim que entram na interface de rede do equipamento, tarefa que só pode ser executada por um bloco específico de *hardware* para evitar a latência intrínseca adicionada pelos *device drivers*, sistema operacional e outras camadas de *software*. Dessa forma, esse trabalho propõe o estudo e a análise do bloco de *hardware* de código aberto HA1588, que implementa em *Hardware Description Language* (HDL) o protocolo IEEE 1588-2008. O objetivo é compreender como ocorre a recepção e identificação das mensagens do padrão e o processo de sincronização dos relógios entre dois sistemas. Assim, o bloco foi simulado por software e sintetizado em um kit de desenvolvimento em *Field Programable Gate Array* (FPGA) validando o seu funcionamento para posterior uso em projetos que envolvam a necessidade de sincronização dos relógios.

Palavras-chave: IEEE-1588-2008. Recepção e identificação de mensagens PTPv2. Bloco de *hardware* HA1588. Linguagem de descrição de hardware. FPGA.

ABSTRACT

Clock synchronization is a classic issue in distributed systems. With the applications proliferation in the Internet of Things (IoT) domain, smart cities, among other uses that heavily rely on Ethernet networks for sensing and control, accurate clock synchronization can play a pivotal role depending on the application. Each scenario has time synchronization requirements and dedicated protocols to meet this need. In the context of digital circuit synchronization, the use of the Institute of Electrical and Electronics Engineers (IEEE) 1588-2008 standard has been becoming popular, also known as IEEE 1588v2 or Precision Time Protocol version 2 (PTPv2), which utilizes Ethernet and the master-slave model in clock synchronization through standardized message exchange. By incorporating hardware support into the protocol, it becomes possible to achieve nanosecond-level precision in synchronizing two clocks separated by long distances, offering an excellent cost-effective solution as it reuses the Ethernet network. To achieve such accuracy, a specific block of hardware must analyze the protocol package with PTPv2 messages as soon as they enter the network interface of the equipment, a task that can only be performed by a specific block of hardware to avoid the intrinsic latency added by the device drivers, operational system and other software layers. Thus, this work proposes the study and analysis of the HA1588 open-source hardware block, which implements the IEEE 1588-2008 protocol in Hardware Description Language (HDL). The goal is to comprehend how the reception and identification of standard messages occur, as well as the process of clock synchronization between two systems. As a result, the block was software simulated and synthesized on a Field-Programmable Gate Array (FPGA) development kit, validating its operation for subsequent use in projects that require clock synchronization.

Keywords: IEEE-1588-2008. Reception and identification of PTPv2 messages. Block of hardware HA1588. Hardware description language. FPGA.

LISTA DE ILUSTRAÇÕES

Figura 1 – Relógios dos dispositivos de rede A e B com frequências diferentes. . . 17
Figura 2 – Relógios dos dispositivos de rede A e B com frequências iguais. 18
Figura 3 – Relógios dos dispositivos A e B com frequências iguais e fases diferentes. 18
Figura 4 – Relógios dos dispositivos A e B com frequências e fases iguais. 19
Figura 5 – Uso do *delay request-response mechanism* entre o principal e secundário. 23
Figura 6 – Mensagem PTPv2 dentro de UDP sobre IPv4 e sobre Ethernet. 24
Figura 7 – Formato básico de mensagem PTPv2. 24
Figura 8 – Formato do cabeçalho PTPv2. 25
Figura 9 – Diagrama de blocos da implementação da arquitetura do bloco HA1588. 27
Figura 10 – Interfaces GMII e MII. 27
Figura 11 – Diagrama de blocos da simulação de *hardware* do bloco HA1588. 30
Figura 12 – Bloco HA1588 detalhado. 31
Figura 13 – Componentes do bloco HA1588 utilizados no teste de identificação das mensagens PTPv2. 33
Figura 14 – Wireshark analisando a mensagem *sync*. 34
Figura 15 – Simulação do bloco HA1588 identificando a mensagem sync. 35
Figura 16 – Simulação do bloco HA1588 identificando a mensagem follow_up. . . . 36
Figura 17 – Web-server de exemplo do kit DE2-115. 37
Figura 18 – Detalhes da interface MII do módulo *Ethernet driver* do projeto web-service. 38
Figura 19 – Ligações do bloco a interface MII do Ethernet driver e no kit DE2-115. 39
Figura 20 – Cenário prático com computador como relógio principal enviando as mensagens PTPv2 ao FPGA como relógio secundário. 41
Figura 21 – Cenário prático implementado com computador como relógio principal conectado ao FPGA como relógio secundário 42
Figura 22 – Configurando o envio da mensagem *sync* no programa PackETH. . . . 43
Figura 23 – Configurando o envio da mensagem *follow_up* no programa PackETH. 44
Figura 24 – Análise da mensagem *sync* pelo wireshark na camada de acesso a rede. 45
Figura 25 – Análise da mensagem *sync* pelo wireshark na camada de internet. . . . 46
Figura 26 – Análise da mensagem *sync* pelo wireshark na camada de transporte. . . 47
Figura 27 – Análise da mensagem *sync* pelo wireshark na camada de aplicação. . . 48
Figura 28 – Análise da mensagem *follow_up* pelo wireshark na camada de aplicação. 48
Figura 29 – Kit FPGA como relógio secundário identificando a mensagem *sync*. . . 49
Figura 30 – Kit FPGA como relógio secundário identificando a mensagem *follow_up*. 50
Figura 31 – Comando *ping* enviado à interface de rede da placa DE2-115. 51

LISTA DE QUADROS

Quadro 1 – MessageId em hexadecimal das mensagens PTPv2. 22

LISTA DE ABREVIATURAS E SIGLAS

BMC *Best Master Clock.*

CAT *Category.*

CPU *Central Processing Unit.*

ERB Estações Rádio Base.

FPGA *Field Programable Gate Array.*

GMII *Gigabit Media Independent Interface.*

GPS *Global Positioning System.*

HA1588 *Hardware Assisted IEEE 1588 IP Core.*

HDL *Hardware Description Language.*

IEEE *Institute of Electrical and Electronic Engineers.*

IoT *Internet of Things.*

IP *Intelectual Propriety.*

IP' *Internet Protocol.*

IPv4 *Internet Protocol version 4.*

IPv6 *Internet Protocol version 6.*

LCD *Liquid Crystal Display.*

LED *Light Emitting Diode.*

MAC *Media Access Control.*

MII *Media Independent Interface.*

MPLS *Multiprotocol Label Switching.*

NTP *Network Time Protocol.*

P2P *Peer-To-Peer.*

Pcap *Packet Capture.*

PD *Path Delay.*

PPS *Pulse Per Second.*

PTP *Precision Time Protocol.*

PTPv1 *Precision Time Protocol version 1.*

PTPv2 *Precision Time Protocol version 2.*

RGS Registrador de Gerenciamento do Sistema.

Rt *Residency time.*

RTC *Real-Time Clock.*

RX *Receptor.*

TC *Transparent Clock.*

TCP *Transmission Control Protocol.*

TSU *Time Stamping Unit.*

TX *Transmissor.*

UDP *User Datagram Protocol.*

VLAN *Virtual Local Area Network.*

SUMÁRIO

1 INTRODUÇÃO . . . 14
1.1 Objetivo geral . . . 15
1.2 Objetivos específicos . . . 16
1.3 Organização do texto . . . 16

2 FUNDAMENTAÇÃO TEÓRICA . . . 17
2.1 Relógio de um equipamento de rede . . . 17
2.1.1 Sincronização do relógio . . . 17
2.2 Protocolo IEEE 1588v2 . . . 19
2.2.1 Mensagens PTPv2 . . . 20
2.2.2 Mecanismos de medição de atraso de propagação para a sincronização . . . 22
2.2.3 Transporte da mensagem PTPv2 . . . 23
2.2.4 Formatos de mensagem PTPv2 . . . 24

3 ESTUDO DO BLOCO HA1588 . . . 26
3.1 Interfaces GMII/MII . . . 27
3.2 Principais características da arquitetura dos módulos . . . 28
3.2.1 RTC . . . 28
3.2.2 TSU . . . 29
3.3 Simulação do *hardware* . . . 30
3.3.1 Detalhamento . . . 31
3.3.2 Configuração mínima para simulação . . . 32
3.3.3 Execução da simulação . . . 33

4 INTEGRAÇÃO DO BLOCO HA1588 . . . 37
4.1 Exemplo web-server do kit DE2-115 . . . 37
4.2 Integração do bloco HA1588 ao projeto web-server . . . 39

5 TESTES E RESULTADOS . . . 41
5.1 Configuração e envio das mensagens pelo relógio principal . . . 42
5.2 Análise das mensagens com o programa wireshark . . . 45
5.3 Identificação das mensagens pelo relógio secundário . . . 49
5.3.1 Mensagem de evento sync . . . 49
5.3.2 Mensagem geral follow_up . . . 50
5.3.3 Comando ping . . . 51

6 CONCLUSÕES E TRABALHOS FUTUROS . . . 52

REFERÊNCIAS 53

APÊNDICES 55

APÊNDICE A – CÓDIGO FONTE 56

APÊNDICE B – DETALHAMENTO DOS DIAGRAMAS DE BLOCO DO IP CORE HA1588 57

1 INTRODUÇÃO

O conceito de relógio é visto, pelo ser humano, como um equipamento que marca o tempo transcorrido entre dois eventos, por exemplo, início e fim do dia, partida e chegada, etc. Porém, para equipamentos digitais, o relógio é visto como um sinal periódico de tensão que oscila entre o valor 0, ausência de tensão, e 1, tensão máxima estabelecida pela tecnologia utilizada. Em teoria, um relógio sempre levará a mesma quantidade de ciclos para representar um segundo, porém na prática os relógios não são perfeitos e suas precisões variam de acordo com os padrões de fabricação e temperatura. Dessa forma, mesmo que dois equipamentos sejam sincronizados assim que ligados, seus relógios começam a perder o sincronismo de frequência ou fase com o tempo. Portanto, para garantir a sincronização dos relógios, um protocolo de sincronização tem o objetivo de sincronizar a frequência e fase a partir de um relógio de referência (NEWS, 2014), (CISCO, 2021).

Em um sistema de telecomunicação ou aplicações de monitoramento, como as utilizadas em sistemas de transmissão de energia, a sincronização do relógio entre os nodos da rede é um requisito essencial. Por exemplo, nas redes 4G e 5G as Estações Rádio Base (ERB) são distribuídas por vários pontos de uma região com quilômetros de distância umas das outras, e se baseiam em um relógio principal de alta precisão para sincronização de todo o sistema com uma variação máxima de 1 μs. A opção utilizada para a sincronização é o uso do *Global Positioning System* (GPS) em cada ERB, pois traz a vantagem de não haver necessidade de conexão entre os nós da rede para sincronização e o custo é diluído por todo sistema de celulares atendidos. Entretanto, em situações em que não há escala, o custo pode ser proibitivo, necessitando de outras formas de sincronização.

Uma alternativa é embutir nos equipamentos de rede *Ethernet* a capacidade de transmitir sinais de sincronismo de relógio juntamente com os dados, aproveitando assim a infraestrutura de rede entre os nodos e tornando o custo dessa implementação mais econômica. A qualidade de sincronização do relógio pode não ser igual ao do GPS dependendo dos fatores como o congestionamento na rede e do protocolo de sincronização de tempo utilizado. Contudo, esse método tem uma melhor relação custo-benefício, uma vez que aproveita o meio de comunicação já existente e utiliza o protocolo adequado ao tipo do cenário em que se quer sincronizar os relógios.

Dentre os protocolos de sincronização que fazem o uso da *ethernet*, os mais comuns são o *Network Time Protocol* (NTP) e o PTPv2. Entretanto, cada protocolo é utilizado em determinado cenário de acordo com as características de sincronismo exigidas pela aplicação. Por exemplo, em redes comerciais e domésticas, em que não é necessário alta

precisão no sincronismo, a precisão de milissegundos fornecidos pelo NTP é aceitável. Já em aplicações em que o requisito de precisão no sincronismo é de nanosegundos, o PTPv2 é o recomendável (SAMBERG, 2017). Entretanto, para alcançar esse nível de precisão, é necessário que os pacotes com as mensagens do protocolo sejam analisados assim que entram na interface de rede do equipamento (QI; WANG; YANG, 2011). Com o suporte de *hardware*, a leitura ocorre entre a camada física e a subcamada *Media Access Control* (MAC) da pilha de comunicação de rede. Isso permite reduzir atrasos causados pelos *drivers* e do próprio sistema operacional.

Uma das formas de adicionar o suporte de *hardware* para alcançar a precisão máxima sugerida pelo padrão IEEE 1588-2008, é utilizando plataformas de lógicas programáveis como os sistemas compostos por FPGA. Isso permite adicionar um bloco de *hardware*, chamados de *Intelectual Propriety* (IP) *core*, que implemente a recepção das mensagens e sincronização do relógio. Existem opções de IP *cores* com toda a documentação e suporte para integração no sistema final, mas, em geral, tem um custo proibitivo para projetos acadêmicos (DIGIKEY, 2018). Por outro lado, existem IP *cores* de código aberto e sem custo para uso, mas com informações limitadas e sem nenhum suporte, necessitando de um estudo aprofundado para entender a implementação e certificar o funcionamento do mesmo.

Dessa forma, este trabalho propõe um cenário para análise e estudo do IP *core* de código aberto, *Hardware Assisted IEEE 1588 IP Core* (HA1588)[1]. Ele foi implementado utilizando o *kit* de desenvolvimento para FPGA DE2-115 e o objetivo foi compreender como ocorre o funcionamento na prática da leitura e identificação das mensagens PTPv2 entre a camada física e a subcamada MAC. Além de ratificar o funcionamento do bloco na recepção de tais mensagens.

1.1 Objetivo geral

O objetivo do presente trabalho é a criação de um cenário mínimo que certifique o funcionamento do IP *core*, com relação à recepção de mensagens definidas no protocolo. Ele é implementado no kit de desenvolvimento FPGA DE2-115, fazendo o papel de nodo secundário e com o bloco integrado; e um computador que emulará o principal, enviando pacotes com as mensagens.

[1] https://opencores.org/projects/ha1588

1.2 Objetivos específicos

A partir do objetivo geral, os seguintes objetivos específicos foram definidos:

- Compreender o funcionamento básico do protocolo de sincronização de tempo PTPv2 baseado no padrão IEEE 1588-2008.
- Compreender a lógica de funcionamento do IP *core* de código aberto HA1588, criando diagramas de blocos detalhados de cada componente que compõem o projeto.
- Executar na prática um projeto que envolva o uso da interface de rede do kit FPGA DE2-115, permitindo a integração com o bloco HA1588.
- Modelar e implementar o cenário de teste, integrando o bloco HA1588, e executar as rotinas que verifiquem a recepção de mensagens PTPv2.

1.3 Organização do texto

O texto está organizado da seguinte forma: no Capítulo 2 é apresentado a fundamentação teórica sobre os principais conceitos da sincronização de relógio e do protocolo de sincronização de tempo PTPv2. No Capítulo 3 é exposto o estudo do bloco HA1588. No Capítulo 4 é exibido a integração do bloco HA1588. No Capítulo 5 é demonstrado os testes e resultados. Finalmente no Capítulo 6 são descritos as conclusões e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados os principais conceitos envolvidos no trabalho. Na primeira seção são descritos os conceitos de sincronização de relógios entre equipamentos de rede. Na segunda é feito um levantamento das informações relacionadas ao protocolo PTPv2.

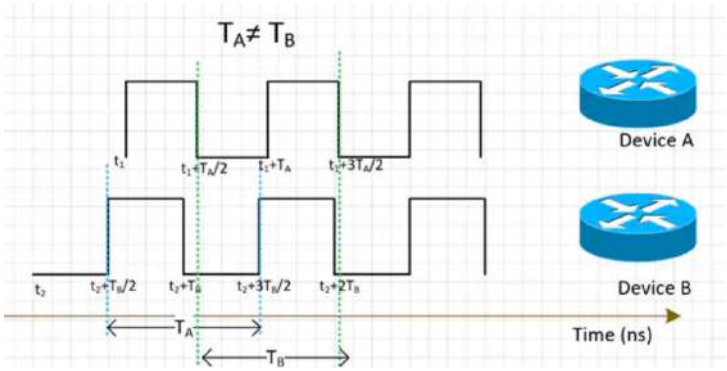
2.1 Relógio de um equipamento de rede

Para geração do sinal do relógio, os equipamentos de rede usam um oscilador de cristal com erro de $\pm 100ppm$ (partes por milhão), onde um relógio com a frequência de 250 MHz e 100 ppm terá uma faixa de frequência de 249,975 MHz a 250,025 MHz ao gerar o sinal do relógio. Assim, é permitido que o relógio de um determinado sistema tenha alguma variação desde que cumpra a exigência requisitada pela aplicação (CISCO, 2021).

2.1.1 Sincronização do relógio

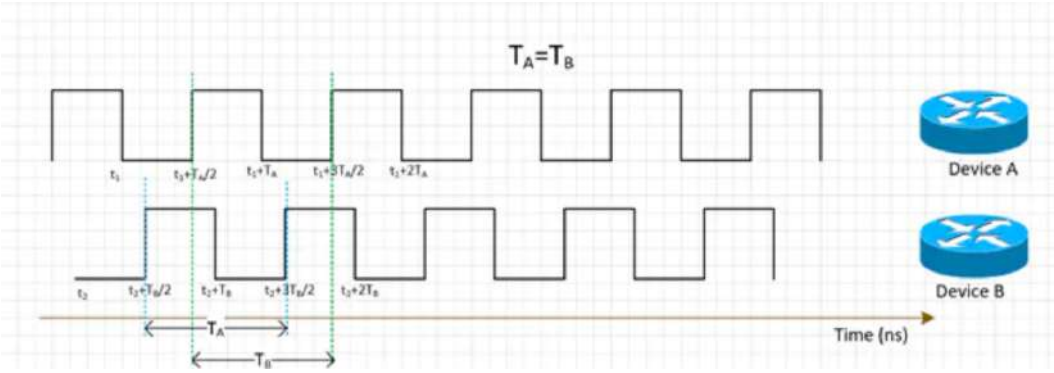
Sincronização do relógio é a ação de alinhar a frequência e a fase a partir de um relógio de referência. A primeira etapa é a sincronização de frequência que consiste em igualar a frequência de um relógio com a frequência de um relógio principal. Para exemplificar, nas Figura 1 e Figura 2 são apresentados dois dispositivos de rede, A e B, estão fora de sincronismo de frequência, e depois com elas sincronizadas respectivamente. (CISCO, 2021).

Figura 1 – Relógios dos dispositivos de rede A e B com frequências diferentes.



FONTE: (CISCO, 2021).

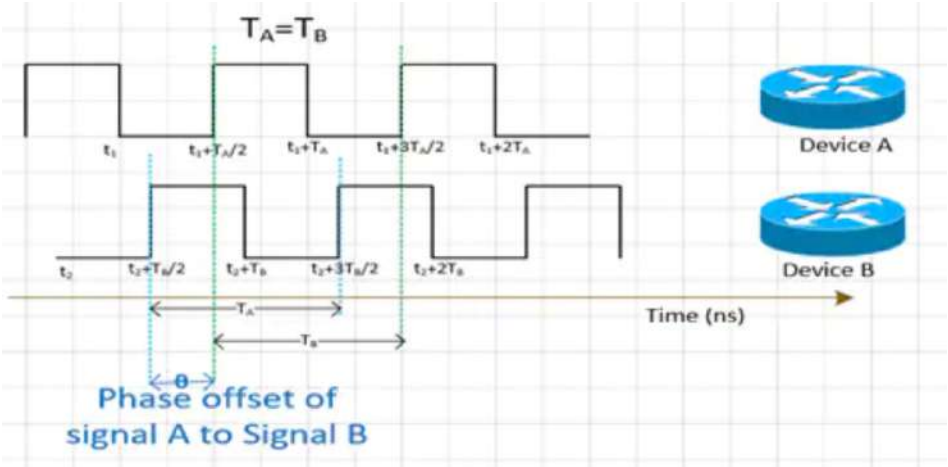
Figura 2 – Relógios dos dispositivos de rede A e B com frequências iguais.



FONTE: (CISCO, 2021).

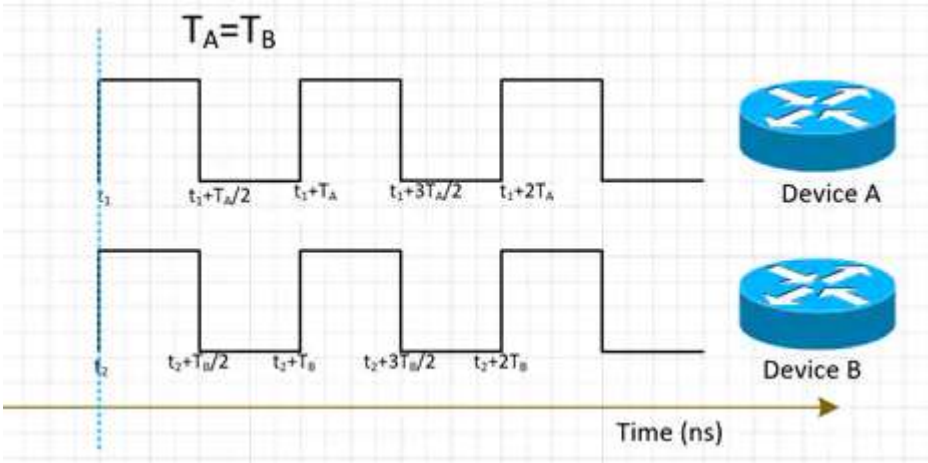
Entretanto, mesmo que as frequências de dois relógios sejam as mesmas, pode ocorre o atraso entre as bordas de subida e descida, chamado de deslocamento de fase. Esse deslocamento pode ser calculado pela frequência dos sinais e o tempo de atraso entre elas. Geralmente é descrito em termos de ângulo, em graus ou radianos, positivo ou negativo. A Figura 3 exibe um exemplo de deslocamento de fase em que os sinais de relógio dos dispositivos de rede A e B possuem as mesmas frequências, porém o período do sinal de relógio B não está alinhado com o relógio A. Assim, a sincronização de fase tem o objetivo de alinhar o início dos períodos dos sinais de relógios, do dispositivo a ser sincronizado com o dispositivo principal, como ilustrado na Figura 4 (SCIENCEAQ, 2018).

Figura 3 – Relógios dos dispositivos A e B com frequências iguais e fases diferentes.



FONTE: (CISCO, 2021).

Figura 4 – Relógios dos dispositivos A e B com frequências e fases iguais.



FONTE: (CISCO, 2021).

2.2 Protocolo IEEE 1588v2

O IEEE 1588 abreviado por *Precision Time Protocol* (PTP) foi definido pela IEEE em 2002 para sistemas de controle e medição utilizando a rede *ethernet*. Dessa forma, é chamado de IEEE 1588-2002, IEEE 1588 v1 ou *Precision Time Protocol version 1* (PTPv1) e aplica-se aos campos de automação industrial, testes e medições. Com o crescimento de redes de telecomunicações e energia, ocorreu um grande aumento na demanda por sincronização de tempo com precisão em nanossegundos. Sendo assim, pra suprir essa necessidade, o IEEE iniciou uma atualização em junho de 2006, culminando no padrão IEEE 1588-2008, também conhecido como IEEE 1588v2 ou PTPv2. A principal mudança foi a incorporação da lógica do PTP na camada física, portanto, agora o *hardware* intercepta as mensagens e faz o ajuste do relógio para a sincronização de todos os dispositivos do sistema (CISCO, 2021).

O modelo hierárquico do protocolo é do tipo principal-secundário para sincronização do tempo. O relógio principal determina o tempo de referência para todos os outros relógios. Eles se alinham a partir da troca de mensagens de temporização PTPv2 pela rede *ethernet* atingindo um valor de precisão do sincronismo em nanossegundos, caso todo o sistema tenha suporte a hardware. Com isso, as mensagens do protocolo são analisadas assim que entram na interface de rede do dispositivo, entre a camada física e a subcamada MAC da pilha de comunicação de rede, reduzindo atrasos causados pelos *drivers* e do próprio sistema operacional (HBM, 2019).

2.2.1 Mensagens PTPv2

Os relógios principal e secundário são mantidos sincronizados pela transmissão de *timestamp* (registro de data e hora) enviados nas mensagens do protocolo PTPv2, onde elas podem ser do tipo de eventos e gerais. As **mensagens de eventos** são mensagens temporizadas em que um *timestamp* preciso é gerado tanto na transmissão quanto no recebimento da mensagem. Abaixo são descritas as mensagens desse tipo:

- ***Sync***: Mensagem que inicia a sincronização do tempo entre o relógio principal e secundário. No PTPv1, por não ter suporte por *hardware*, duas mensagens são enviadas do principal ao secundário para informar o seu tempo, sendo conhecido por sincronização por duas etapas. Nesse cenário, a mensagem *sync* enviada ao secundário não transporta o tempo, apenas o registra no principal. A mensagem *follow_up* enviada em seguida é que transporta o tempo do principal. Já no PTPv2, por ter o suporte por hardware, o processo de sincronização é em uma etapa, de forma que o tempo é transportado e enviado do principal ao secundário na própria mensagem *sync*. Mesmo com essas diferenças, a sincronização de uma etapa é compatível com a de duas etapas.

- ***Delay_Req***: É uma solicitação do relógio secundário ao principal para ele retornar o tempo em que a mensagem *delay_req* foi recebida por ele usando a mensagem *delay_resp*. Essa mensagem possui data e hora no secundário e é usada para determinar o tempo de deslocamento entre o relógio secundário e principal.

- ***Pdelay_Req (Peer_delay_Req)***: Utilizada pelo *Transparent Clock* (TC) *Peer-To-Peer* (P2P) para calcular o tempo que as mensagens PTPv2 permanecem nesse tipo de relógio. (O TC não será comentado nesse trabalho).

- ***Pdelay_Resp (Peer_delay_Resp)***: Essa mensagem é enviada do relógio principal por exemplo, em resposta ao receber uma mensagem *pdelay_req* do TC.

As **mensagens gerais** são mensagens que não requerem *timestamp*, mas podem conter o *timestamp* da sua mensagem de evento associada. Abaixo são informados as mensagens desse tipo:

- ***Follow_Up***: Utilizado na sincronização do PTPv1 por não possuir suporte a *hardware*. Em sua mensagem enviada ao secundário transporta o tempo de saída da mensagem *sync* no relógio principal.

- ***Delay_Resp***: Em sua mensagem enviada ao secundário transporta o tempo de chegada da mensagem *delay_req* no relógio principal.

- ***Pdelay_Resp_Follow_Up***: É semelhante à mensagem *follow_up*, mas é gerada por um TC P2P. (O TC não será comentado nesse trabalho).

- ***Announce***: Utilizada para escolher o melhor relógio principal e gerar a topologia principal e secundário.

  – Quando um dispositivo é iniciado, o algoritmo *Best Master Clock* (BMC) é ativado. Este algoritmo é responsável por selecionar qual relógio servirá como a fonte de tempo para o dispositivo em questão. Para isso, exige-se que o dispositivo envie uma mensagem de *announce*, contendo informações sobre o relógio, como sua qualidade (dependente da precisão do oscilador e da fonte externa de tempo), sua prioridade e o domínio ao qual pertence. Com base nos valores das mensagens de *announce* recebidas por um relógio, suas portas podem assumir diferentes estados. Se não for encontrado nenhum relógio na região com uma referência de tempo superior, o relógio entrará no estado de principal. Se houver uma referência de tempo melhor disponível e o relógio não estiver sendo sincronizado, ele adotará o estado de secundário. Caso não seja possível se enquadrar em nenhum dos estados anteriores, o relógio assumirá o estado passivo. Para assegurar uma conexão constante entre os dispositivos e o melhor principal disponível, as mensagens de *announce* são periodicamente retransmitidas, mesmo quando não há alterações físicas na rede. Além de sua função na sincronização, as mensagens de *announce* também desempenham um papel na determinação da rota que um pacote deve seguir para alcançar seu destino. Isso resolve problemas de redundância e adaptação abrupta na topologia da rede (VICENTIN, 2020).

- ***Signaling***: Usado para outros fins na comunicação entre os relógios, como determinar a taxa de mensagens entre um relógio principal e seus secundários.

- ***Management***: Utilizado pelo gerenciador da rede para monitorar, configurar e manter um sistema PTPv2.

A identificação de cada mensagem enviada e recebida pela rede *ethernet* ocorre a partir do seu ***messageId***, onde no Quadro 1 são exibidos os valores em hexadecimal referente a cada mensagem PTPv2 (LTD, 2009).

Quadro 1 – MessageId em hexadecimal das mensagens PTPv2.

Mensagem PTPv2	**MessageId**
Sync	0x0
Delay_Req	0x1
Pdelay_Req	0x2
Pdelay_Resp	0x3
Reservado para futuras mensagens de evento	0x4 a 0x7
Follow_Up	0x8
Delay_Resp	0x9
Pdelay_Resp_Follow_Up	0xa
Announce	0xb
Signaling	0xc
Management	0xd

FONTE: Feito pelo autor.

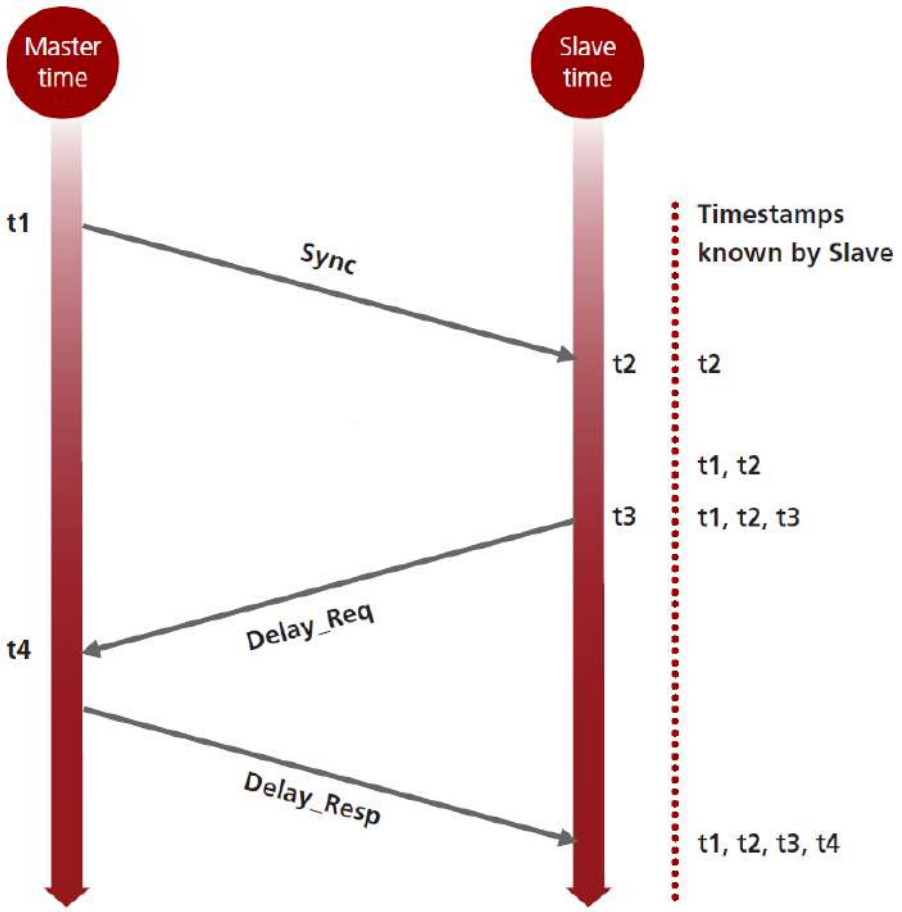
2.2.2 Mecanismos de medição de atraso de propagação para a sincronização

O PTPv2 possui dois mecanismos para calcular o atraso de propagação entre as portas dos relógios principal e secundário: o ***Delay Request-Response Mechanism***, que sincroniza o relógio secundário com o principal, e o ***Peer Delay Mechanism***, utilizado quando *switches* estão envolvidos. Neste trabalho, concentraremos a análise apenas no primeiro mecanismo.

O processo da troca de mensagens PTPv2 pelo ***delay request-response mechanism*** para calcular o atraso de propagação entre os relógios inicia com o principal enviando a mensagem ***sync*** para o secundário que marca o tempo em que ela foi enviado em ***t1***. O secundário recebe a mensagem ***sync*** e registra o tempo da chegada em ***t2***. O secundário envia uma mensagem ***delay_req*** ao principal e marca o tempo de envio em ***t3***. O principal recebe a mensagem ***delay_req*** e registra o tempo da chegada em ***t4***. O principal informa o tempo ***t4*** ao secundário enviando uma mensagem ***delay_resp***.

Todas essas etapas são ilustradas na Figura 5. Uma vez que o secundário conhece os tempos de ***t1, t2, t3 e t4***, ele pode calcular o *Path Delay* (PD) das mensagens. Contudo essa lógica não será abordada aqui pois o objetivo do trabalho é somente a recepção e identificação das mensagens de eventos e gerais.

Figura 5 – Uso do *delay request-response mechanism* entre o principal e secundário.

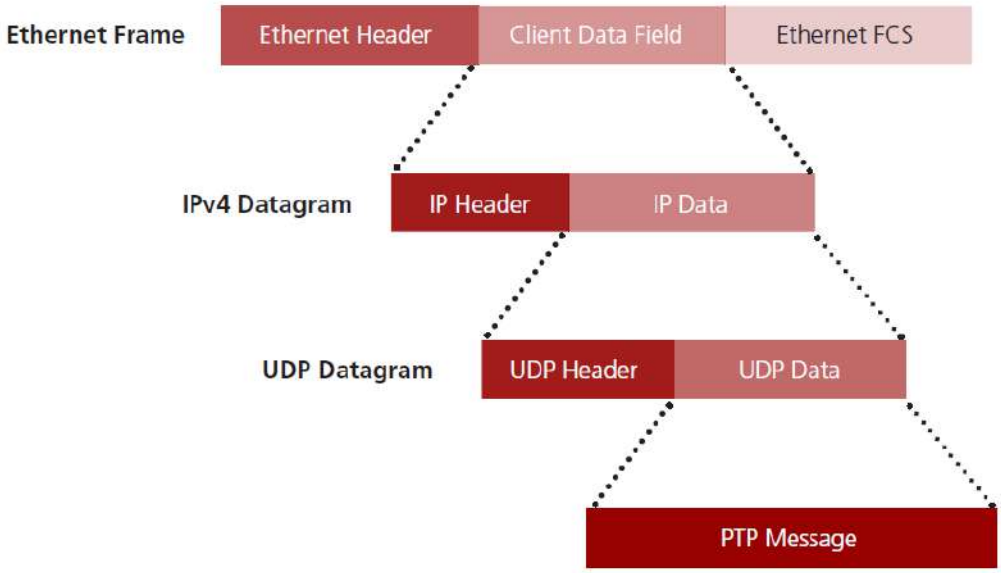


FONTE: (LTD, 2009).

2.2.3 Transporte da mensagem PTPv2

O transporte da mensagem PTPv2 entre os relógios principal e secundário acontece com a mensagem sendo encapsulada no campo dados do *User Datagram Protocol* (UDP). Em seguida é encapsulado no campo dados do datagrama *Internet Protocol version 4* (IPv4). Depois é novamente encapsulado no campo dados do quadro *Ethernet*. Por fim, o quadro é enviado pelo cabo de rede que interliga os relógios para a troca das informações. Na Figura 6, é apresentado esse processo de encapsulamento das mensagens PTPv2 (LTD, 2009).

Figura 6 – Mensagem PTPv2 dentro de UDP sobre IPv4 e sobre Ethernet.

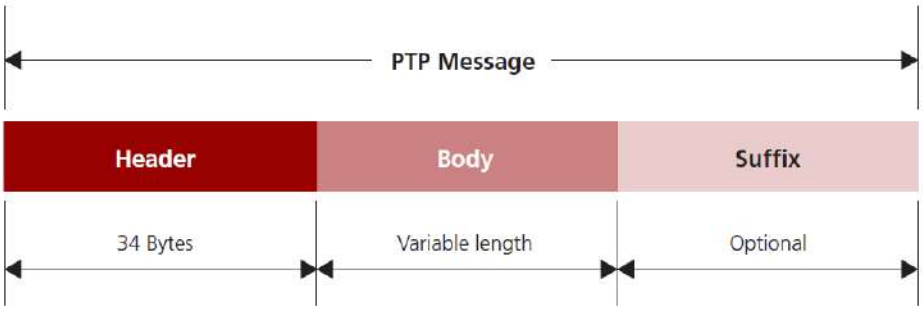


FONTE: (LTD, 2009).

2.2.4 Formatos de mensagem PTPv2

Todas as mensagens PTPv2 consistem em um *header* (cabeçalho), *body* (corpo) e sufixo opcional como exibido na Figura 7. Entretanto no trabalho o interesse é apenas no *header* (LTD, 2009).

Figura 7 – Formato básico de mensagem PTPv2.



FONTE: (LTD, 2009).

O ***header*** é comum em todas as mensagens PTPv2. Tem 34 bytes e seu formato é mostrado na Figura 8. Abaixo são exibidas informações sobre cada campo:

- **messageType**: Define qual tipo de mensagem está contida no corpo da mensagem, por exemplo "*sync*", "*follow_up*", "*delay_req*", "*delay_resp*", etc. (Esse campo terá mais ênfase no trabalho).

Figura 8 – Formato do cabeçalho PTPv2.

PTP Message Header Format									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
Reserved								1	5
Flags								2	6
correctionField								8	8
Reserved								4	16
sourcePortIdentity								10	20
sequenceID								2	30
controlField								1	32
logMessageInterval								1	33

FONTE: (LTD, 2009).

- **messageLength**: Define o comprimento total da mensagem PTPv2, ou seja, incluindo o *header*, *body* e qualquer sufixo.
- **domainNumber** identifica o domínio ao qual a mensagem PTPv2 pertence. Um domínio é um agrupamento lógico de relógios que sincronizam entre si usando o protocolo, mas que não são necessariamente sincronizados com relógios em outro domínio.
- **Flags**: Contém várias *flags* (Sinalizadores) para indicar o *status*.
- **correctionField**: Contém um valor de correção em nanossegundos para o *Residency time (Rt)* (tempo de residência) dentro de um TC (Relógio transparente) e também incluirá o PD (Atraso de caminho) para TC P2P.
- **sourcePortIdentity**: Identifica a porta de origem para esta mensagem.
- **sequenceID**: Contém (com algumas exceções) um número de sequência para tipos de mensagens individuais. (Esse campo terá mais ênfase no trabalho).
- **controlField**: É um campo histórico cujo valor depende do tipo de mensagem e está em conformidade com a versão 1 do padrão. O campo é semelhante ao campo ”messageType” mas com menos opções.
- **logMessageInterval**: É determinado pelo tipo da mensagem.

3 ESTUDO DO BLOCO HA1588

Para entender e testar o funcionamento do protocolo PTPv2 na prática, foi selecionado um IP *core*[1] de código aberto, chamado HA1588[2]. Ele é implementado em *verilog* e é composto pelos módulos *Real-Time Clock* (RTC), Registrador de Gerenciamento do Sistema (RGS) e dois *Time Stamping Unit* (TSU), um *Transmissor* (TX) e outro *Receptor* (RX) como ilustrado no diagrama de blocos da Figura 9. A seguir são descritas as funções de cada um desses blocos:

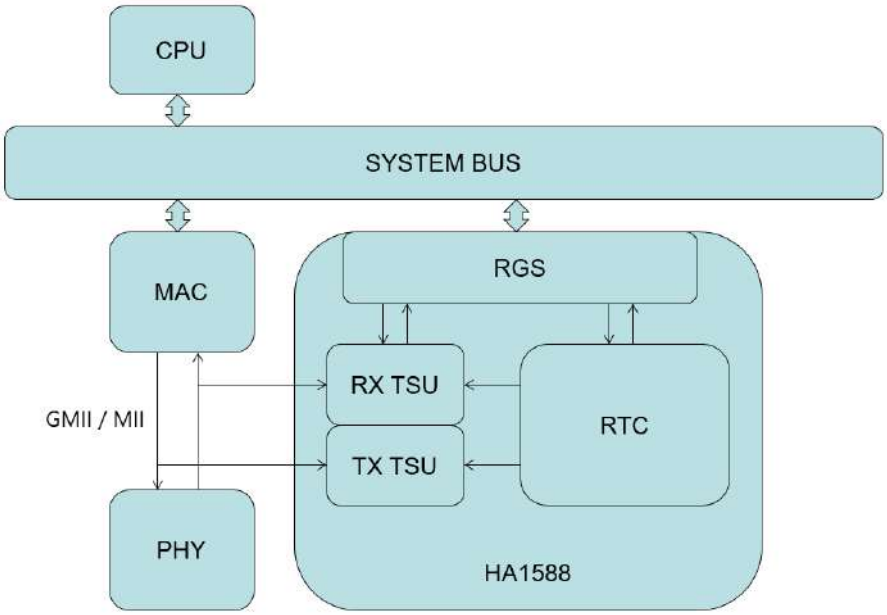
- **RTC**: Responsável pela geração do tempo do relógio no formato de segundos e nanossegundos.
- **TSU**: Responsável em identificar nos pacotes de rede recebidos e enviados pela interface de rede do dispositivo as mensagens de eventos, registrando o tempo (*timestamp*) fornecido pelo RTC no momento dessa ação.
- **RGS**: Gerencia as informações de configuração que a *Central Processing Unit* (CPU) envia pelo barramento do sistema para configurar os módulos RTC e TSU. Também repassa as informações processadas por esses módulos na saída do bloco HA1588 que está interligada ao barramento do sistema.

Para que o bloco envie e receba as informações dos pacotes de rede, ele é interligado entre a camada PHY e a subcamada MAC do dispositivo, onde intercepta as informações logo na entrada da interface de rede, o que garante uma precisão de nanossegundos na sincronização. O bloco é projetado para ser integrado em barramentos com padrão *Media Independent Interface* (MII) ou *Gigabit Media Independent Interface* (GMII), permitindo trabalhar respectivamente com velocidades de dados de 10/100 ou 1000 Mbp/s (OPENCORES, 2012).

[1] IP *core* - É um bloco de lógica ou dados usados na criação de um chip semicondutor. Na maioria das vezes pertence a uma determinada pessoa ou empresa (BUTLER, 2023).

[2] *Hardware Assisted* 1588 - https://opencores.org/projects/ha1588

Figura 9 – Diagrama de blocos da implementação da arquitetura do bloco HA1588.

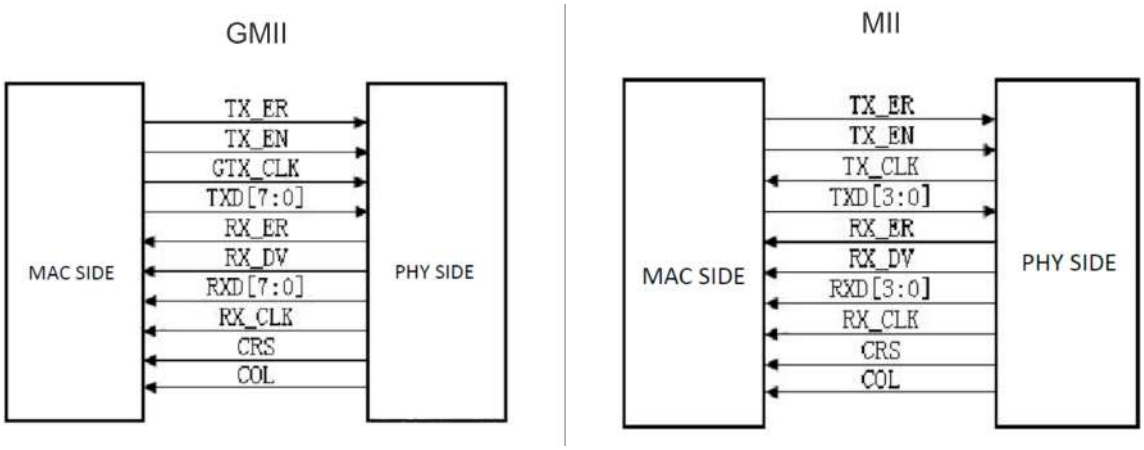


FONTE: (OPENCORES, 2012).

3.1 Interfaces GMII/MII

A GMII é uma interface padrão entre os chips que implementam a camada física (PHY) e a subcamada MAC. Ela permite que os dados sejam transferidos a velocidades de até 1 Gbp/s usando uma interface de dados de 8 *bits* com frequência de relógio de 125 *MHz*. Além disso, também é compatível com a especificação MII que permite a transferência de dados de 10 ou 100 Mbp/s, usando uma interface de dados de 4 *bits* com frequência do relógio de 2,5 e 25 *MHz* respectivamente. Os sinais dessas interfaces são ilustrados na Figura 10 e a função de cada sinal é comentada em sequência (GUYS, 2019).

Figura 10 – Interfaces GMII e MII.



FONTE: (GUYS, 2019).

- **TX**:
  - **GTX_CLK**: Sinal de *clock* para sinais *Gbp/s* TX (125 MHz - GMII).
  - **TX_CLK**: Sinal de *clock* para sinais 10/100 Mbp/s (MII).
  - **TXD**: Dados transmitidos de 8 ou 4 bits (GMII/MII).
  - **TX_EN**: Transmissor ativado.
  - **TX_ER**: Erro de transmissão (Informação corrompida).
- **RX**:
  - **RX_CLK**: Sinal de *clock* recebido (recuperado dos dados entrantes).
  - **RXD**: Dados recebidos de 8 ou 4 bits (GMII/MII).
  - **RX_DV**: Dados recebidos são válidos.
  - **RX_ER**: Dados recebidos possuem erros.
  - **COL**: Colisão Detectada (Só conexão *half-duplex*).
  - **CRS**: Detecção de Portadora (Só em conexão *half-duplex*).

3.2 Principais características da arquitetura dos módulos

Após ver os módulos que compõe o bloco HA1588 a seguir são descritos as principais características relevantes no trabalho sobre os módulos RTC e TSU.

3.2.1 RTC

O módulo RTC é o responsável por gerar o tempo e enviar ao TSU para ser usado no registro de tempo (*timestamp*) do envio e recepção das mensagens do tipo evento. Ele possui três saídas, sendo que duas informam o valor do tempo gerado. A primeira possui 48 *bits* informando o tempo dos segundos e a segunda contém 32 *bits* para indicar os nanossegundos. A última saída de 1 *bit* é do *Pulse Per Second* (PPS) para a medição da precisão do relógio, ou seja, toda vez que na saída dos nanossegundos o incremento do seu valor alcança o seu valor máximo, ela é zerada e a saída dos segundos é incrementada, nessa transição a saída PPS é definida com o valor 1 informando que houve o incremento dos segundos.

3.2.2 TSU

O módulo TSU é o responsável em identificar as mensagens de eventos PTPv2, registrando o tempo (*timestamp*) fornecido pelo RTC no momento dessa ação. Ele possui um monitor de interface GMII/MII com análise de pacotes PTPv2, suportando também as mensagens do protocolo na operação em duas etapas. Na leitura dos pacotes de rede que transportam as mensagens, aceita vários formatos de pacotes como o IEEE 802.3 com *tags Virtual Local Area Network* (VLAN), IPv4/UDP ou *Internet Protocol version 6* (IPv6)/UDP com ou sem *tags* VLAN e/ou rótulos *Multiprotocol Label Switching* (MPLS). Contém um submódulo nomeado de *PTP_Parse* que recebe os dados e é o responsável por analisar os pacotes, possuindo saída de 32 bits. A identificação das mensagens ocorre através da máscara *one-hot* configurável de 8 bits para marcar seletivamente o pacote de eventos PTPv2 com base no valor do tipo de mensagem onde:

- bit 0: *Sync*.
- bit 1: *Delay_Req*.
- bit 2: *Pdelay_Req*.
- bit 3: *Pdelay_Resp*.
- bits de 4 a 7: Reservado para futuros tipos de mensagem de evento PTPv2.

Após o processamento da identificação das mensagens de eventos ou gerais, o módulo gera o resultado em sua saída de 128 bits. Ela é composta pelo formato de informações onde os 16 bits mais significativos são de informações extras. Os próximos 80 bits contém o *timestamp* registrado nas mensagens identificadas do tipo evento [3]. Por fim nos 32 bits menos significativos da saída do TSU é constituída pelos dados de identidade do pacote PTPv2, sendo organizada da seguinte forma:

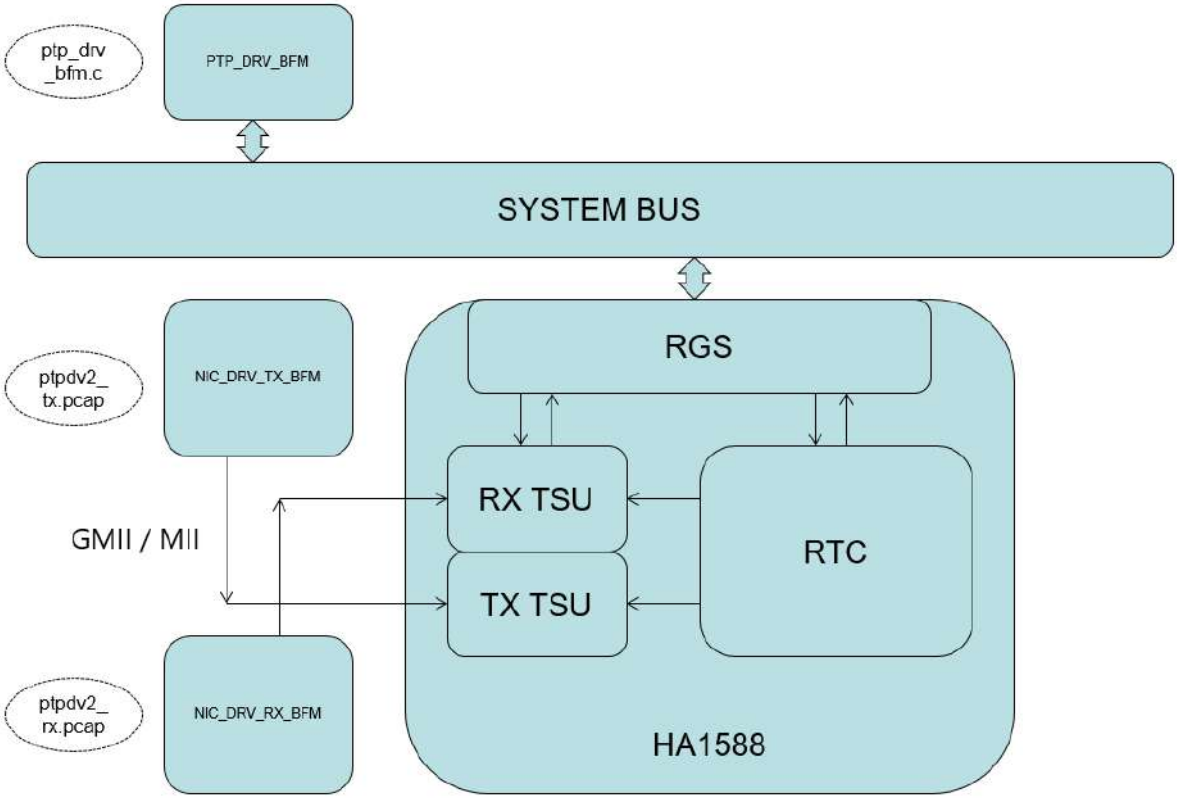
- 4 bits - *Message Id*.
- 12 bits - *Checksum*.
- 16 bits - *Sequence Id*.

[3] *Timestamp* de 80 bits - Os 48 bits mais significativos representam o valor dos segundos e os 32 bits menos significativos são os nanossegundos

3.3 Simulação do *hardware*

O bloco HA1588 é implementado em *verilog*, que é uma linguagem de descrição de *hardware* sintaticamente parecida com C, e permite descrever e simular projetos antes de implementar, garantindo flexibilidade e agilidade no processo de desenvolvimento. Além da implementação sintetizável, o projeto fornece um cenário de simulação que interagi com o bloco, permitindo assim, verificar o resultado processado das informações recebidas ao bloco, facilitando a compreensão do funcionamento da lógica do PTPv2. A CPU é simulada por um arquivo[4] em linguagem C e a camada física e a subcamada MAC são simuladas por dois arquivos[5] de extensão *.pcap* que contêm informações dos pacotes capturados de uma rede com mensagens PTPv2 de transmissão e recepção, onde esses arquivos podem ser criados por exemplo com o programa *wireshark*. A Figura 11 apresenta como fica o diagrama de blocos da simulação do *hardware* com o bloco HA1588.

Figura 11 – Diagrama de blocos da simulação de *hardware* do bloco HA1588.



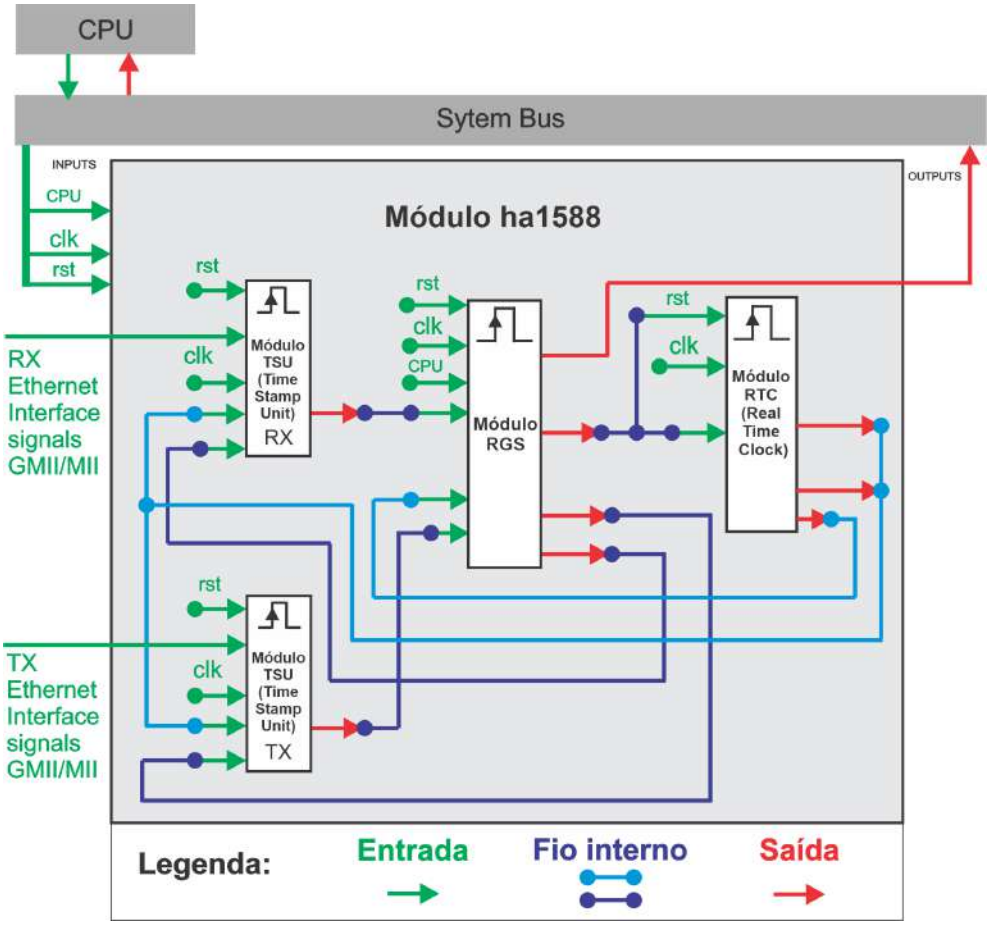
FONTE: (OPENCORES, 2012).

[4] ptp_drv_bfm.c
[5] ptpdv2_tx.pcap e ptpdv2_rx.pcap - (*Packet Capture* (pcap))

3.3.1 Detalhamento

Conhecendo as características da arquitetura e dos diagramas de blocos com o cenário prático e simulado, já é possível ter uma noção do funcionamento do bloco HA1588. Entretanto, para compreender os detalhes do seu funcionamento foi analisado a lógica dos arquivos *verilogs* (.v), C (.c) e captura de pacotes (.pcap). A partir desse estudo foram criados diagramas de blocos bem detalhados do bloco e seus módulos[6], mas para uma fácil compreensão, aqui é exibido um diagrama simplificado das ligações entre os módulos para explicar a sua lógica. Nele são descritos as entradas (em verde), como do *clock*, *reset*, CPU e as interfaces GMII/MII RX e TX. Os fios internos (em azul escuro e claro) que interligam os módulos e as saídas (em vermelho) dos módulos e do bloco com a informação processada, onde a saída do bloco é interligada ao barramento do sistema. Todos esses detalhes são exibidos na Figura 12, e com o fim do detalhamento foi compreendido o funcionamento do bloco HA1588 sendo explicado a seguir.

Figura 12 – Bloco HA1588 detalhado.



FONTE: Feito pelo Autor.

[6] Caso tenha interesse em ver todos os detalhes dos diagramas criados de cada módulo e a tabela verdade do RTC acesse o repositório https://drive.google.com/drive/folders/1Ywep7WPxm2VkIhavfVwKDjsmFyhpNJF0?usp=sharing

O resumo da lógica e funcionamento do bloco HA1588 inicia pela CPU enviando informações ao módulo RGS para configurar primeiro o RTC. Essas informações são recebidas a cada borda de subida do *clock* (*clk*), sendo enviada por exemplo a informação do valor do período que o RTC vai operar para gerar o tempo em segundos e nanossegundos. Então o RGS processa a informação e na próxima borda de subida do *clk*, o resultado é enviado (em azul escuro) ao módulo RTC. Ele recebe e processa as informações e na próxima borda de subida do *clk* o resultado do valor do relógio em segundos, nanossegundos e do PPS é enviado (em azul claro) as portas de entrada do módulo TSU. Dessa forma ao receber uma mensagem PTPv2 do tipo evento, é com esse tempo que é registrado o *timestamp* na mensagem. O RTC também envia (em azul claro) o resultado do tempo gerado ao RGS, no qual na próxima borda de subida do *clk* ele envia ao barramento do sistema.

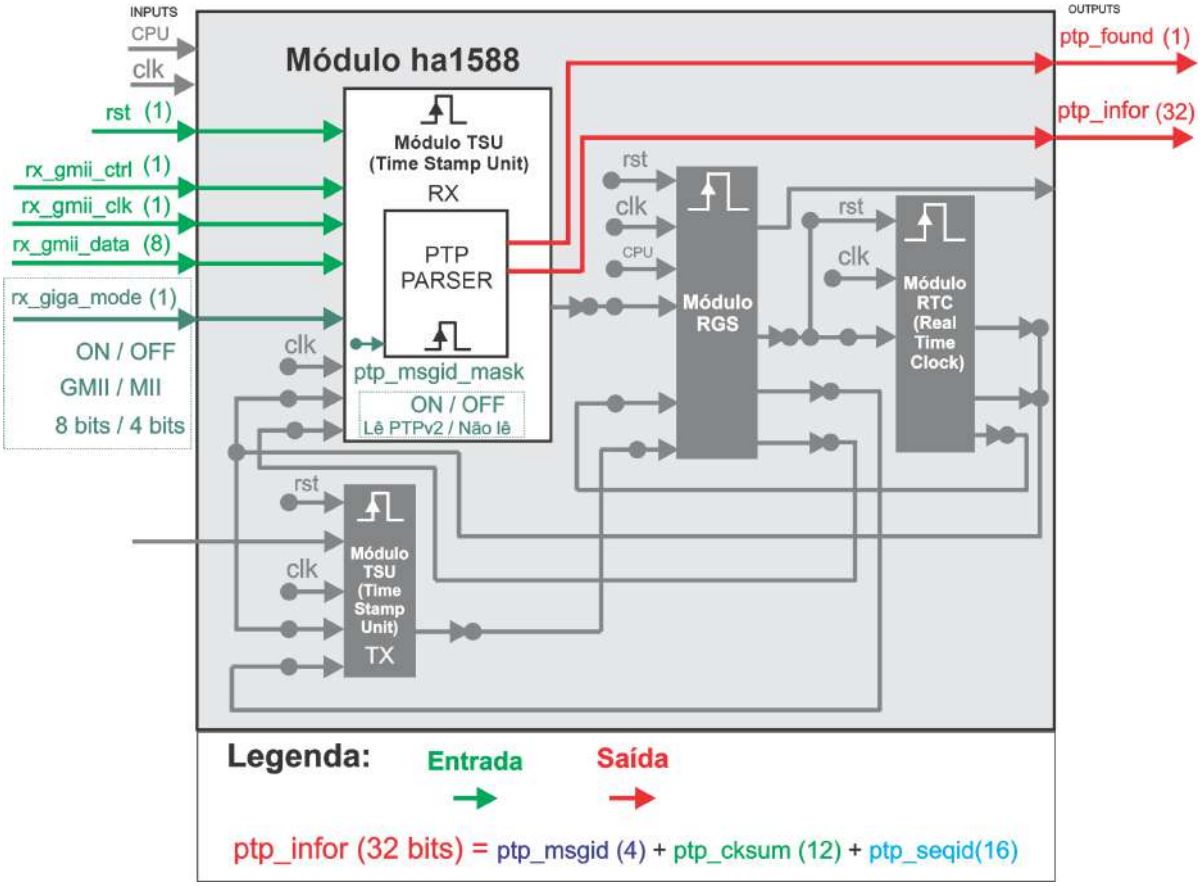
Após configurar o RTC, a etapa seguinte é a configuração do TSU pela CPU. Uma das informações enviadas é a configuração para ativar ou desativar a leitura das mensagens PTPv2 do tipo evento no submódulo *PTP_Parser* que compõe o TSU. O RGS processa as informações e na próxima borda de subida do *clk* o resultado é enviado (em azul escuro) aos módulos TSU RX e TX . Em conjunto as informações do relógio vindo do RTC (em azul claro) também já estão no TSU. Com isso o que falta para o bloco executar sua lógica é receber os pacotes de rede com as mensagens PTPv2 a partir interface de rede, isto é, através na camada física e a subcamada MAC utilizando a interface GMII/MII.

3.3.2 Configuração mínima para simulação

Após compreender o funcionamento do bloco HA1588, foi definido somente enviar ao *kit* de desenvolvimento FPGA DE2-115 a lógica da leitura e identificação das mensagens PTPv2. Pelo fato do grande número de informações do bloco que teria que ser adaptada no FPGA. Sendo assim, o foco no bloco HA1588 se fez nas saídas do submódulo *PTP_Parse* presente no módulo TSU. Elas são compostas pela saída ***ptp_found*** de 1 *bit* que identifica se a mensagem é do tipo evento, e a saída ***ptp_infor*** de 32 *bits* que mostra os dados de identidade da mensagem PTPv2. Ela é organizada em *message Id* nos 4 bits mais significativos, o *checksum* nos 12 bits seguintes e o *sequence Id* nos 16 bits menos significativos. Após essa definição a Figura 13 ilustra como ficou o bloco nesse novo cenário. Logo foi desativado os outros módulos focando apenas no módulo TSU RX. O módulo TSU TX também foi desativado pois o FPGA que faz o papel do relógio secundário, não envia informações ao computador que representa o relógio principal. Com isso o TSU TX não tem informações para ler e identificar. Sobre essa função do *PTP_Parse*, para que ele identifique as mensagens nesse cenário, a sua entrada *ptp_msgid_mask* de 8 bits é definida com o valor "0xFF", habilitando (ON) essa ação.

As entradas da CPU e *clk* também foram desativadas, ficando apenas a entrada

Figura 13 – Componentes do bloco HA1588 utilizados no teste de identificação das mensagens PTPv2.



FONTE: Feito pelo Autor.

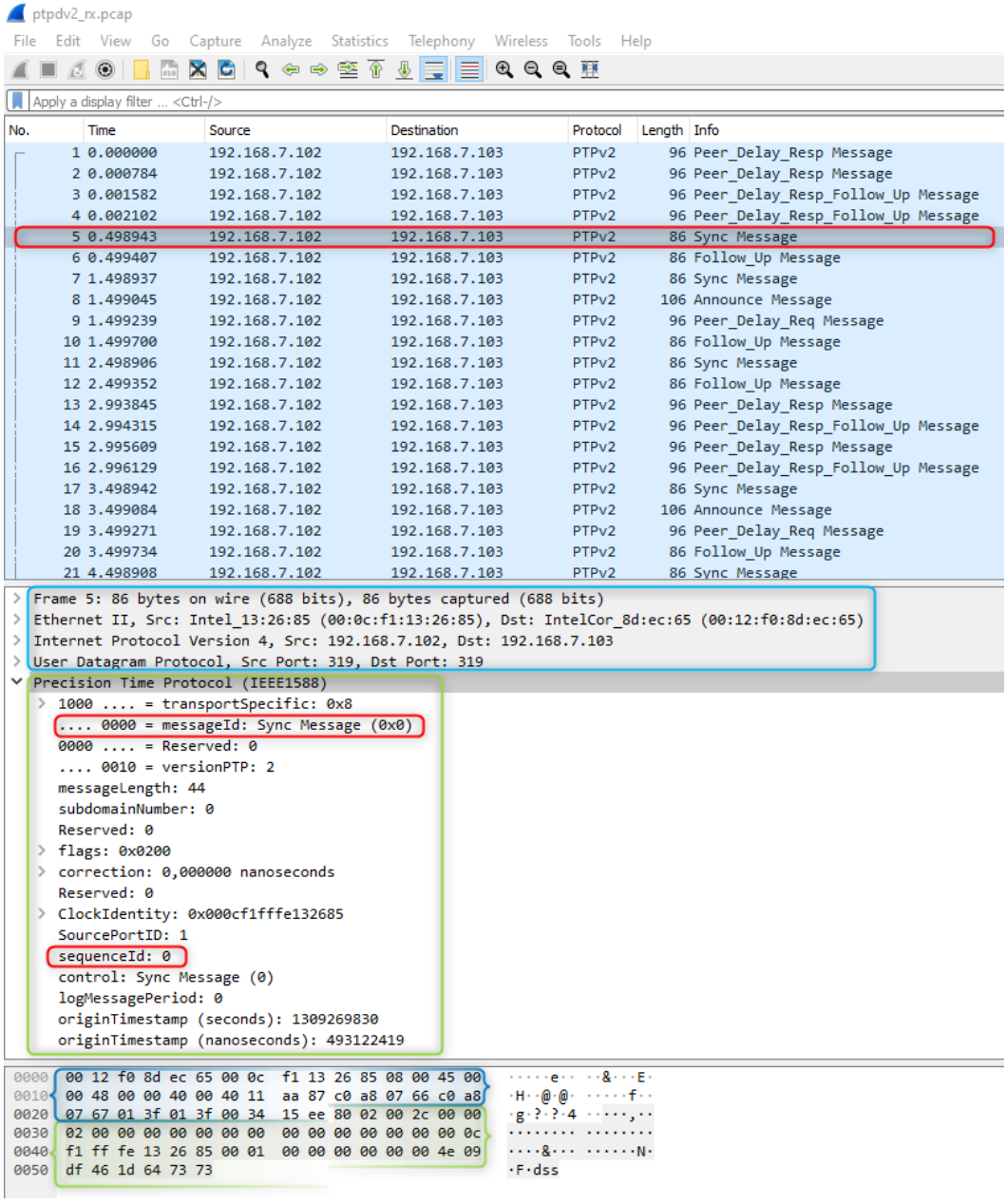
do ***rst*** de 1 bit. As entradas ***"rx_gmii_clk"*** de 1 bit, ***"rx_gmii_ctrl"*** de 1 bit e ***"rx_gmii_data"*** de 8 bits, ligadas a interface ***gmii*** que se comunicam com a interface de rede. E a entrada *rx_giga_mode* de 1 bit que escolhe qual interface vai ser utilizada. Sobre a entrada de *clk*, ela foi desativada pois não interagem na lógica da leitura e identificação das mensagens, pois quem faz esse papel é a entrada ***rx_gmii_clk*** de 1 bit vindo da interface GMII/MII.

3.3.3 Execução da simulação

Com o sistema configurado, o próximo passo foi realizar a simulação do novo cenário para verificar se os resultados processados estavam de acordo com a lógica implementada. Assim, para o bloco HA1588 receber as informações, o mesmo já possui um arquivo de simulação em *verilog* que executa a leitura do arquivo ptpdv2_rx.pcap, que contém as mensagens PTPv2 como visto na Figura 14. Para verificar a lógica do bloco foi executado por esse arquivo a leitura da mensagem *sync* como destacado. De todas as informações processadas pelo bloco dessa mensagem, os parâmetros que garantem que a

mensagem foi identificada corretamente são o *messageId* com o valor 0x0 e o *sequenceId* com o valor 0, onde essas informações também são ilustradas na Figura 14.

Figura 14 – Wireshark analisando a mensagem *sync*.

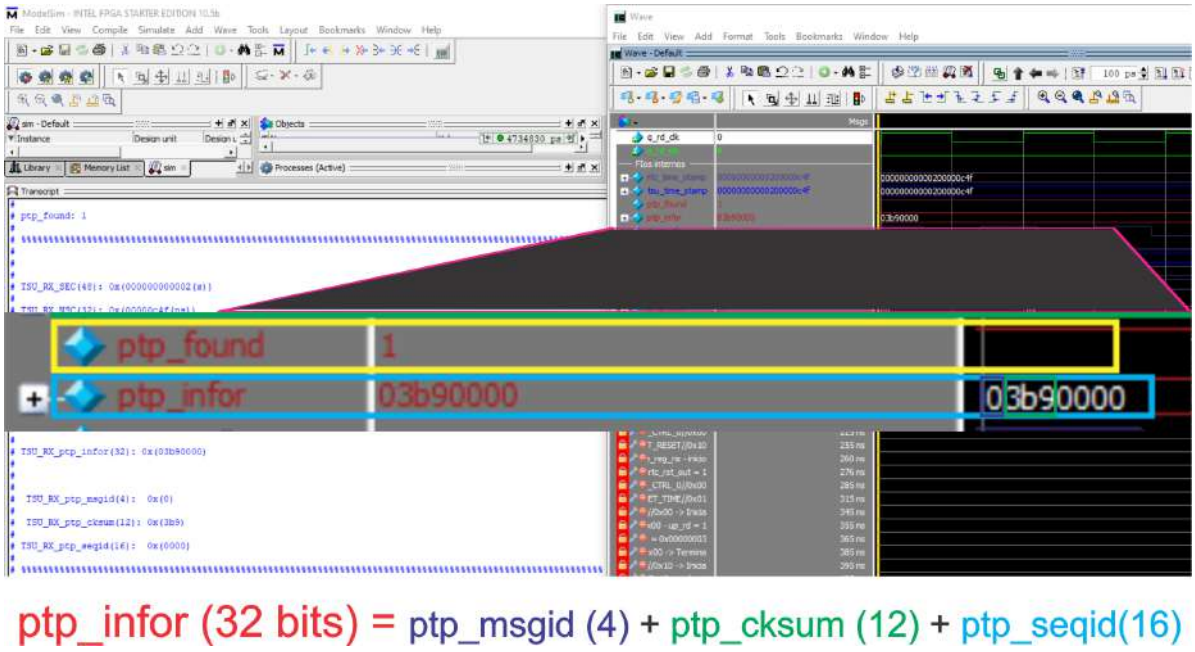


FONTE: Feito pelo Autor.

Para realizar a simulação do bloco se fez uso do programa *modelSim*, que permite ver as informações definidas nas entradas, nos fios internos e os resultados nas saídas. Dessa maneira se fez a simulação do bloco com a mensagem *sync* do tipo evento, onde com o *modelSim* é visto na Figura 15, os resultados das saídas ***ptp_found*** de 1 bit com o valor 1 (marcado em amarelo), que significa que a mensagem identificada é do tipo evento e a saída ***ptp_infor*** de 32 bits com o valor 0x03b90000 (marcado em azul

claro). Os bits dessa saída informam que nos 4 bits mais significativos que representam o valor do *ptp_msgid* (marcado em azul escuro), ou seja, o *messageId* da mensagem *sync* sendo o valor 0x0. Nos 12 bits seguintes é informado *ptp_cksum* (marcado em verde), isto é, o *checksum* da mensagem calculado pelo bloco com o valor 0x3b9. Por fim nos 16 bits menos significativos é descrito o *ptp_seqid* (marcado em azul claro) que equivale ao *sequenceId*, possuindo o valor 0x0000, sendo o valor 0 visto no *wireshark*.

Figura 15 – Simulação do bloco HA1588 identificando a mensagem sync.

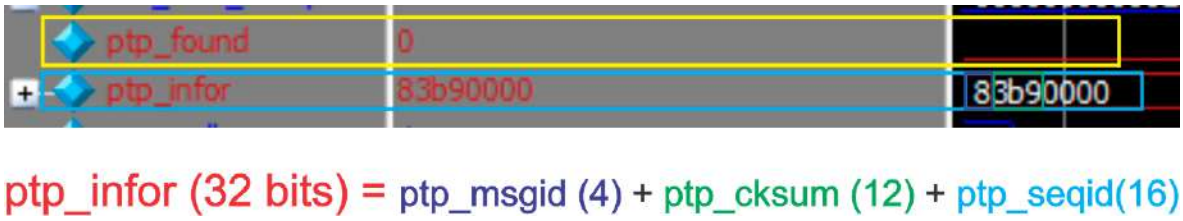


FONTE: Feito pelo Autor.

Na sequência foi realizado a simulação com a mensagem *follow_up* do tipo geral, possuindo os parâmetros *messageId* com o valor 0x8 e o *sequenceId* com o valor 0, sendo estas informações obtidas também do arquivo ptpdv2_rx.pcap. Dessa maneira se fez a simulação do bloco com a mensagem *follow_up* do tipo geral, onde é visto na Figura 16, os resultados das saídas ***ptp_found*** com o valor 0 (marcado em amarelo), identificando a mensagem como do tipo geral e a saída ***ptp_infor*** de 32 bits com o valor 0x83b90000 (marcado em azul claro).

Os bits dessa saída informam que os 4 bits mais significativos representam o valor do *ptp_msgid* (marcado em azul escuro), sendo o valor 0x8. Nos 12 bits seguintes é informado *ptp_cksum* (marcado em verde), com o valor 0x3b9. Por último nos 16 bits menos significativos é descrito o *ptp_seqid* (marcado em azul claro), possuindo o valor 0.

Figura 16 – Simulação do bloco HA1588 identificando a mensagem follow_up.



FONTE: Feito pelo Autor.

A simulação também foi executada com as outras mensagens do protocolo do tipo evento e geral, sendo que em todas elas após o processamento da lógica do bloco obtiveram sucesso nos resultados das saídas ***ptp_found*** e ***ptp_infor***. Devido a muitas imagens a se mostrar, se focou em apenas uma de cada tipo pois já valida o funcionamento do bloco HA1588.

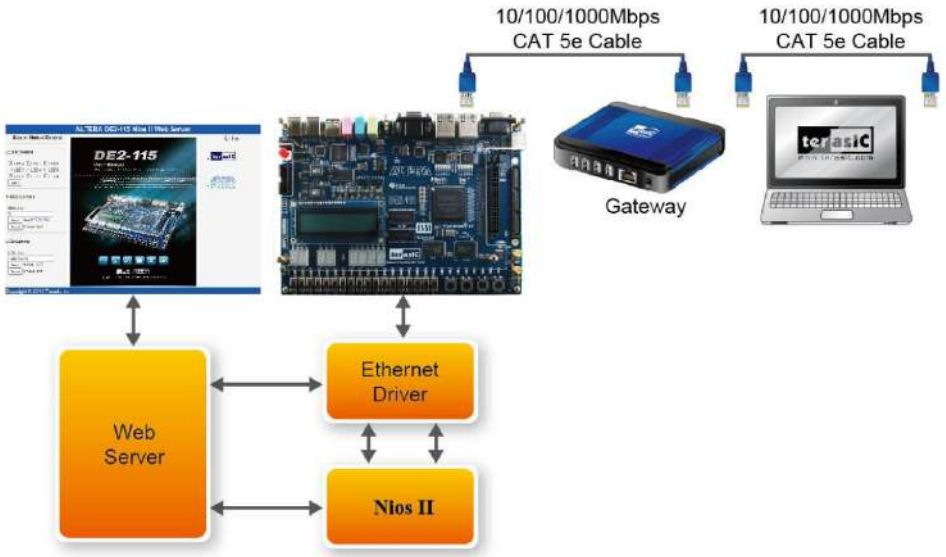
4 INTEGRAÇÃO DO BLOCO HA1588

Após entender o bloco, o próximo passo foi sintetizá-lo e gravá-lo na FPGA *EP4CE115F29C7*, que compõe o *kit* de desenvolvimento DE2-115. Entretanto, o bloco não possui a lógica para interagir com a interface de rede do FPGA, dessa forma, foi preciso integrar ele a um projeto com essa finalidade. A opção escolhida foi o projeto de exemplo de um *web-server* que acompanha a documentação do *kit* DE2-115.

4.1 Exemplo web-server do kit DE2-115

O *kit* desenvolvimento FPGA DE2-115 acompanha uma série de exemplos em *verilog* para ser executado nele. O objetivo é aprender sua lógica e testar as interfaces de entradas e saídas presente no *kit*, sendo que o exemplo escolhido aqui foi o do *web-server*. A Figura 17 exemplifica o resumo do seu funcionamento. O projeto é composto pelo bloco *ethernet driver* responsável por receber e enviar as informações da interface de rede. O bloco *Nios II* que tem a função de processar as informações vindas do bloco *ethernet driver* e *web-server*. Por último o bloco *web-server* que contém a lógica e a página *web*, onde o usuário interage com o *kit* a partir de um navegador web enviando comandos para a placa.

Figura 17 – Web-server de exemplo do kit DE2-115.

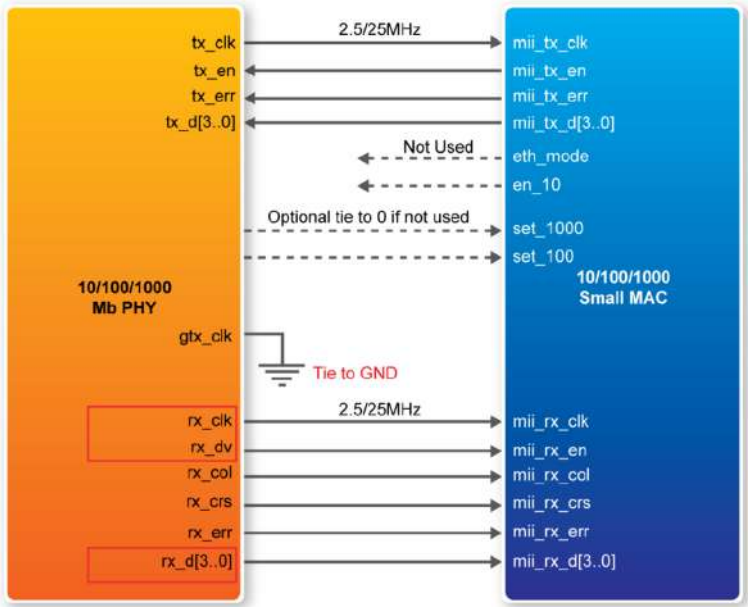


FONTE: (TERASIC, 2017).

Esses comandos interagem com os LEDs [1], *displays* de 7 segmentos e tela de LCD [2]. A interação com a placa é feita através de um computador conectado com um cabo de rede *ethernet* 10/100/1000 CAT [3] 5e (Ou superior) diretamente com ela ou ambos estando na mesma rede local.

Para enxergar com mais detalhes o bloco *ethernet driver*, a Figura 18 exibe como é o seu funcionamento. É visto as conexões de transmissão (TX) e recepção (RX) entre a camada *PHY* e a subcamada MAC da interface de rede, onde no *kit* DE2-115 é suportada apenas a interface MII. Desse modo se alcança velocidades de 10/100 Mbp/s com o envio de 4 *bits* de informação por vez a uma frequência de 2,5/25 MHz respectivamente. Na integração desse projeto com o bloco HA1588 não é necessário utilizar todas as conexões apresentadas, pois como foi definido, o foco é no bloco TSU (RX). Dessa modo as conexões utilizadas aqui são as ***"rx_clk"*** que possui o sinal do *clock* da interface MII, o ***"rx_dv"*** que verifica se os dados recebidos são válidos e o ***"rx_d[3:0]"*** que repassa 4 bits de informação por vez do quadro *Ethernet* com a mensagem PTPv2.

Figura 18 – Detalhes da interface MII do módulo *Ethernet driver* do projeto web-service.



FONTE: (TERASIC, 2017).

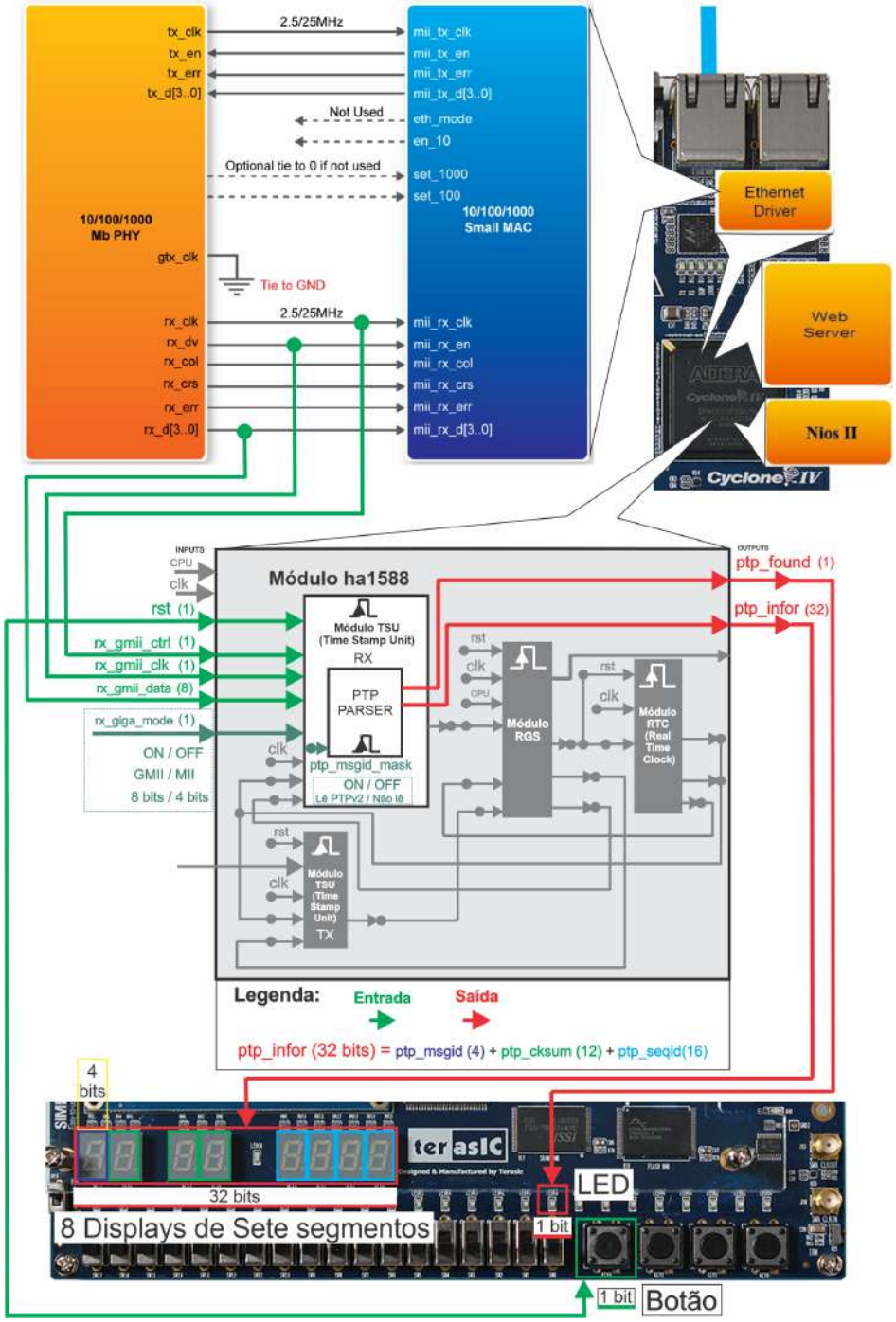
Para sintetizar o projeto na placa DE2-115, foram carregados pelos programas *quartus prime 18.1* o arqivo.sof com a lógica do circuito, e pelo o *nios II software build tools for eclipse - quartus prime 18.1*, o arquivos.elf com a lógica na linguagem C para a placa obter um endereço *Internet Protocol* (IP') e a interface *web*.

[1] *Light Emitting Diode* (LED)
[2] *Liquid Crystal Display* (LCD)
[3] *Category* (CAT)

4.2 Integração do bloco HA1588 ao projeto web-server

Entendendo o básico do funcionamento do projeto do *web-server*, o passo seguinte foi sua integração com o bloco HA1588 redefinido para a lógica do TSU RX. A Figura 19 ilustra como ficou as interligações das entradas e saídas do bloco ao *kit* de desenvolvimento FPGA DE2-115.

Figura 19 – Ligações do bloco a interface MII do Ethernet driver e no kit DE2-115.



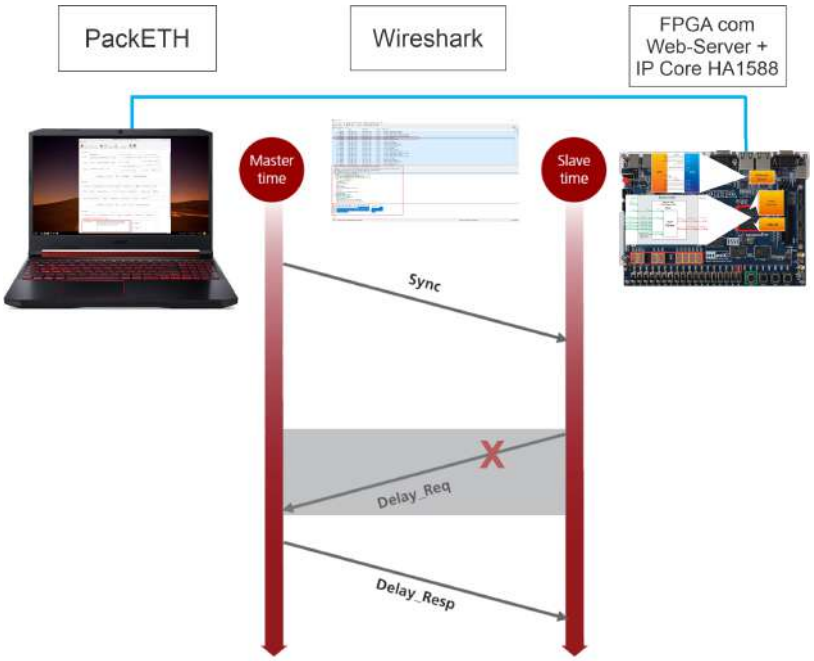
FONTE: Feito pelo Autor.

As entradas do bloco ***"rx_gmii_clk"*** de 1 bit, ***"rx_gmii_ctrl"*** de 1 bit e ***"rx_gmii_data"*** de 8 bits são interligadas respectivamente aos sinais ***"rx_clk"*** de 1 bit, ***"rx_dv"*** de 1 bit e ***"rx_d[3:0]"*** de 4 bits da interface MII presente no bloco *Ethernet driver* do projeto *web-server*. Essas informações são recebidas e enviadas pela interface de rede do *kit*, porém o foco se fez somente nas informações recebidas ao bloco HA1588. Como o kit DE2-115 suporta apenas a interface MII, a entrada ***"rx_giga_mode"*** de 1 bit é definido com o valor 0, para receber as informações em 4 bits por vez sendo a característica dessa interface. Por fim a entrada ***"rst"*** de 1 bit é interligada a um botão do *kit* para ser acionado definindo o valor 1 em sua entrada para que os valores das saídas se iniciem em zero. Com o retorno do valor do ***"rst"*** para 0 ocorre o inicio do processamento do FPGA com a lógica do bloco HA1588 integrado ao *web-server*. Com essa ação inicia-se a geração dos resultados nas saídas ***"ptp_found"*** de 1 bit que é interligada a um LED, sendo que se ele acender, o valor dessa saída é 1 significando que a mensagem recebida e identificada é do tipo evento. Se o LED não aceder, então o valor da saída é zero e a mensagem é do tipo geral. Já na saída ***"ptp_infor"*** de 32 bits, ela se comunica com os oito *dispalys* de 7 segmentos presente no *kit*, onde cada *display* representa 4 bits de informação e sendo possível exibir o resultado de 32 bits. Desses bits os 4 bits mais significativos representam o valor do *ptp_msgid* (marcado em azul escuro), os 12 bits seguintes o *ptp_cksum* (marcado em verde) e os últimos 16 bits menos significativos é o *ptp_seqid* (marcado em azul claro).

5 TESTES E RESULTADOS

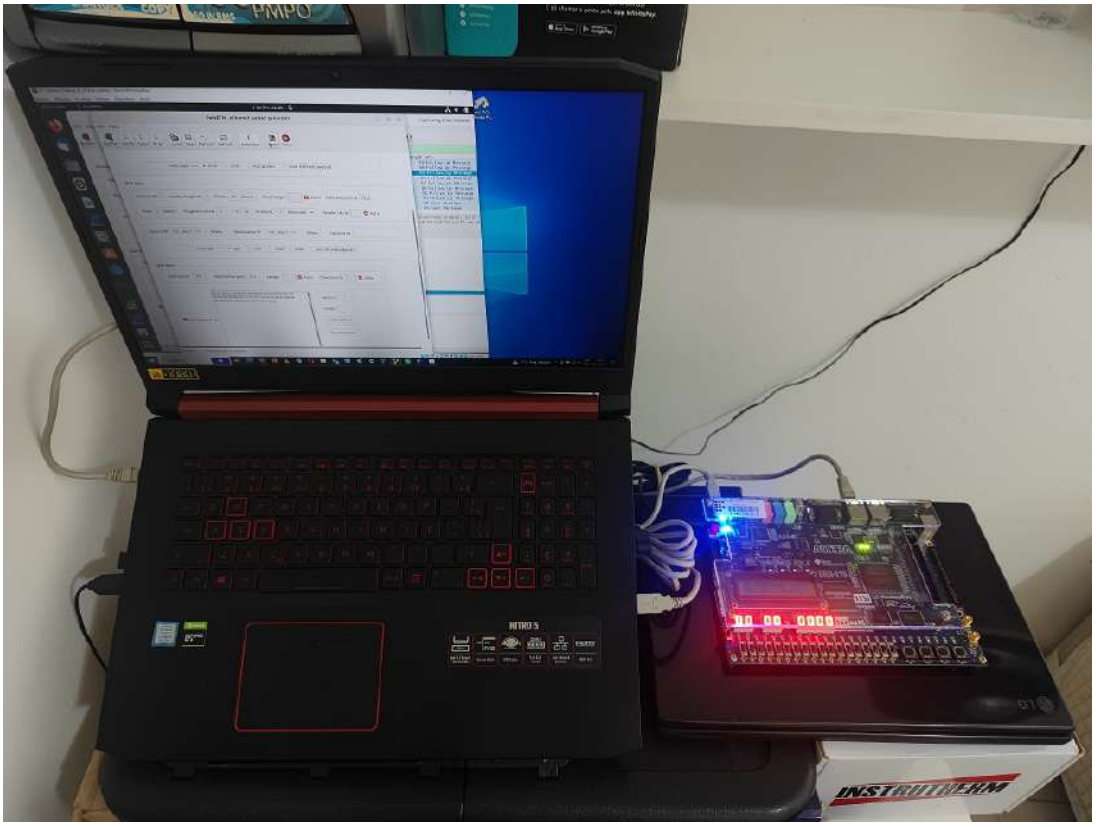
Com a integração do bloco implementada a próxima etapa foi a criação do cenário prático reproduzindo parcialmente o fluxo do protocolo na troca de mensagens para realizar a sincronização, pois aqui somente o relógio principal envia a mensagem ao secundário. Nele ocorre somente identificação das mensagens PTPv2 recebidas onde as do tipo evento são identificadas por um LED. A Figura 20 demonstra como ficou o cenário prático com o relógio *principal* representado por um computador com o programa *packETH*, que tem a função de enviar as mensagens ao relógio secundário representado pela placa FPGA DE2-115 sintetizada com o bloco HA1588 acoplado ao bloco *web-server*. O objetivo é apenas ler e identificar as mensagens PTPv2 recebidas, sendo assim, o secundário não envia nenhuma mensagem. Para se comunicarem ambos são conectados ponto a ponto por um cabo de rede *ethernet*. Por fim para analisar o correto envio das mensagens é utilizado o programa *wireshark* que captura as informações que trafegam nas interfaces de rede de ambos os dispositivos. Em sequência a Figura 21 exibe esse cenário prático implementado.

Figura 20 – Cenário prático com computador como relógio principal enviando as mensagens PTPv2 ao FPGA como relógio secundário.



FONTE: Feito pelo Autor.

Figura 21 – Cenário prático implementado com computador como relógio principal conectado ao FPGA como relógio secundário .



FONTE: Feito pelo Autor.

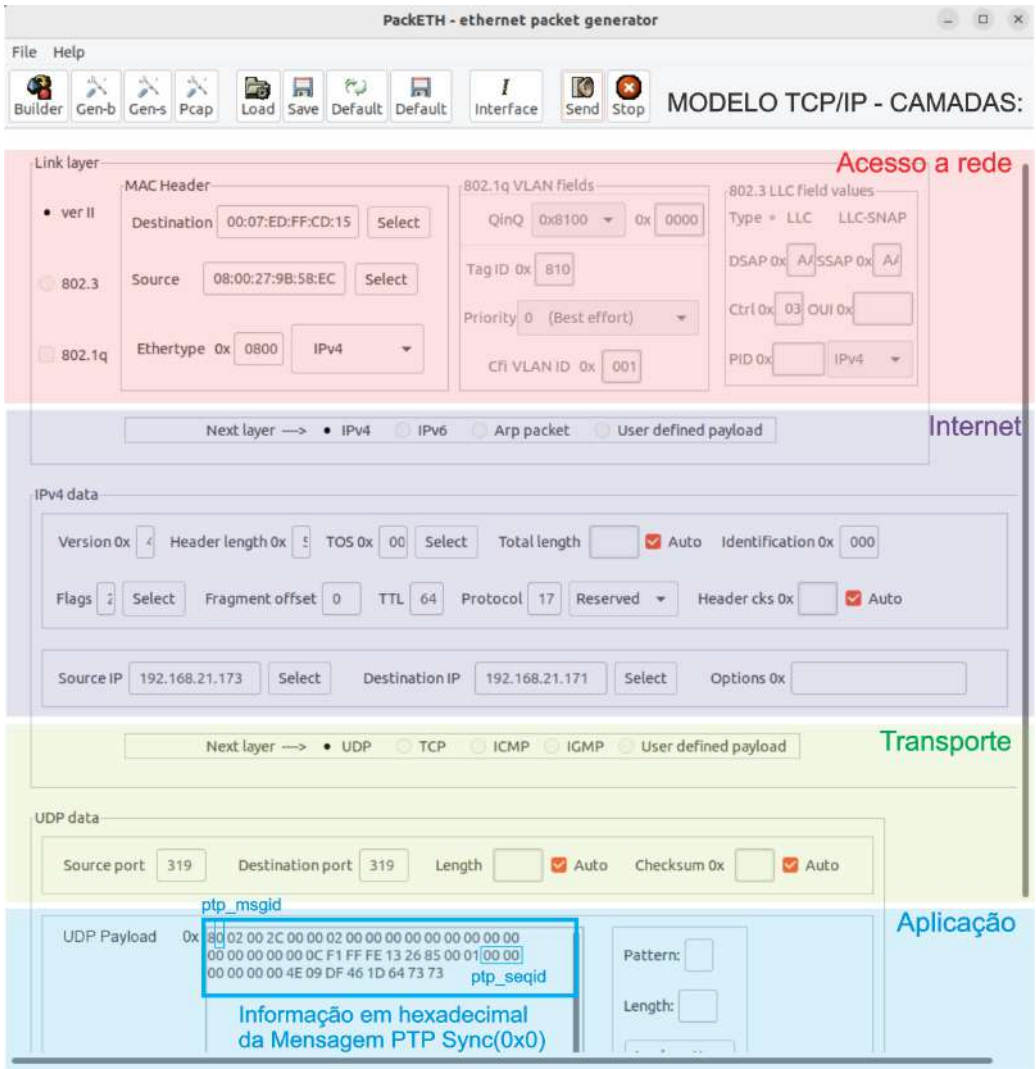
5.1 Configuração e envio das mensagens pelo relógio principal

O cenário prático começa com a configuração no computador do programa *packETH*, na montagem do quadro *ethernet* para o envio dos dados da mensagem PTPv2 a interface de rede do kit. Ele monta o quadro a partir do modelo dos protocolos de *internet Transmission Control Protocol* (TCP)/IP', seguindo a sequência das suas quatro camadas (acesso a rede, *internet*, transporte e aplicação) como ilustrado na Figura 22. A configuração do quadro se inicia na parte superior do programa, na opção "*Builder*", preenchendo as informações na camada de acesso a rede (em vermelho) onde é configurado o cabeçalho MAC, informando os endereços MAC de origem (08:00:27:9B:58:EC)[1] e destino (00:07:ED:FF:CD:15)[2] e qual o protocolo vai ser encapsulado ao quadro (*Ethertype*), sendo aqui o IPv4 (0x0800). Na sequência temos a camada de *internet* (em roxo) no qual é escolhido o tipo do formato do endereço IP', sendo aqui utilizado o formato IPv4 e são definidos os endereços IP's de origem (192.168.21.173)[1] e destino (192.168.21.171)[2]. Em seguida na camada de transporte (em verde) é escolhido o meio de transporte dos dados

[1] Computador -Relógio principal
[2] Placa DE2-115 - Relógio secundário

do tipo UDP e informado o valor das portas de origem[1] e destino[2] sendo ambas 319, que é o valor de porta definido para o envio e recebimento das mensagens do protocolo (PORTALCHAPECO, 2022). Por fim na camada de aplicação (em azul) é preenchido as informações em hexadecimal da mensagem PTPv2 no campo UDP *payload*, sendo nesse exemplo setado com a mensagem *sync* (0x0) do tipo evento[3]. No conjunto de bytes da mensagem foram destacados os bytes que representam a "*ptp_msgid*" e o "*ptp_seqId*" para facilitar a análise das informações pelo programa *wireshark*. Finalizado as configurações o próximo passo é ir na parte superior do programa, na opção superior "*Interfaces*" e escolher a interface de rede de envio. Por último na opção "*Send*" envia-se o pacote *ethernet* pela rede até a interface de rede do *kit* DE2-115.

Figura 22 – Configurando o envio da mensagem *sync* no programa PackETH.

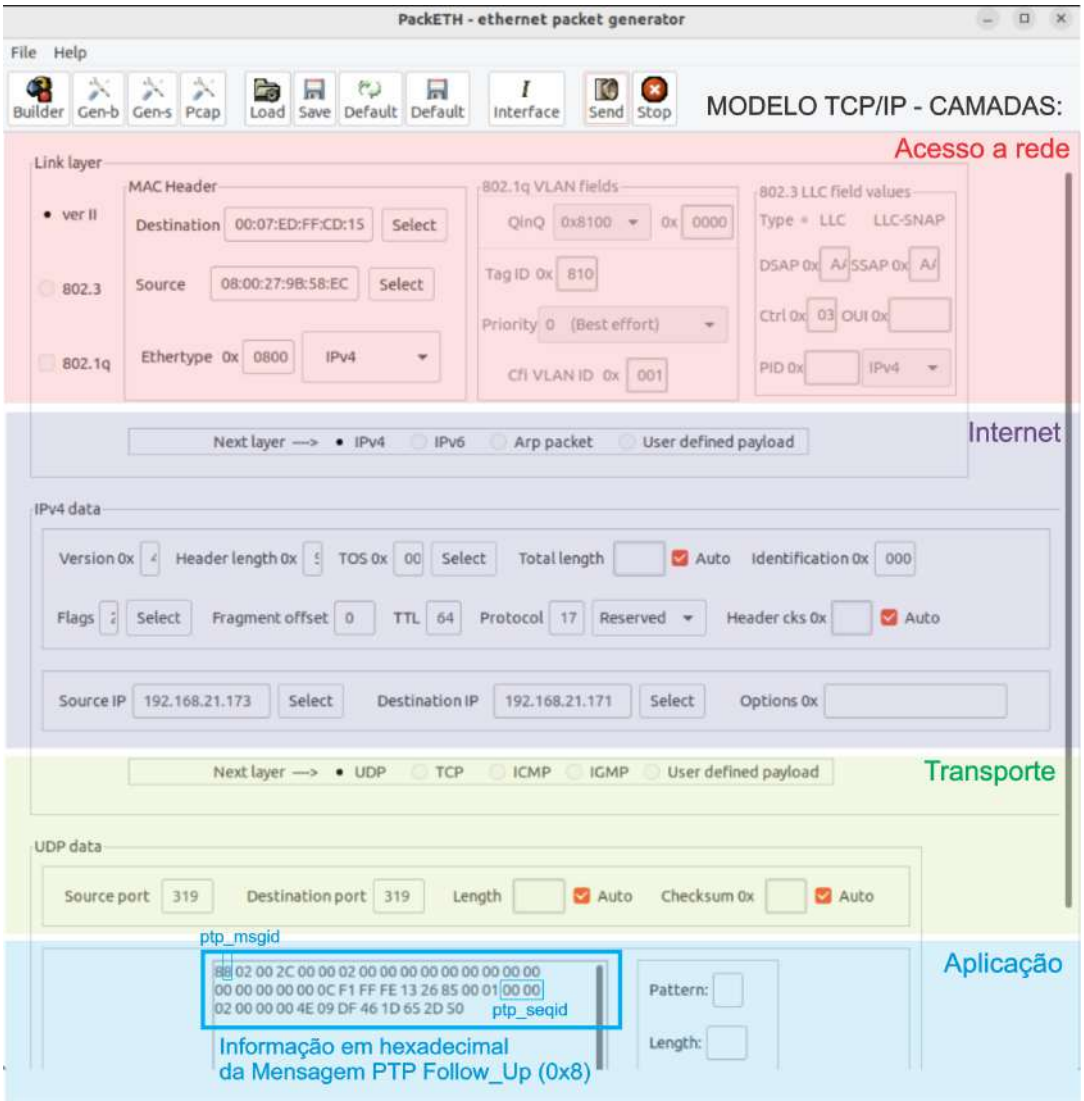


FONTE: Feito pelo Autor.

[3] As informações em hexadecimal dessa mensagem foram obtidas pelo programa *wiresahrk* através do aquivo.pcap utilizado na simulação.

Em sequência foi executado o envio da mensagem PTPv2 *follow_up* do tipo geral para verificar a identificação dessa mensagem na placa DE2-115. A Figura 23 ilustra o programa *packETH* configurado para enviar o pacote de rede com essa mensagem contendo as mesmas informações do exemplo da mensagem *sync*, exceto as informações hexadecimais da camada da aplicação.

Figura 23 – Configurando o envio da mensagem *follow_up* no programa PackETH.

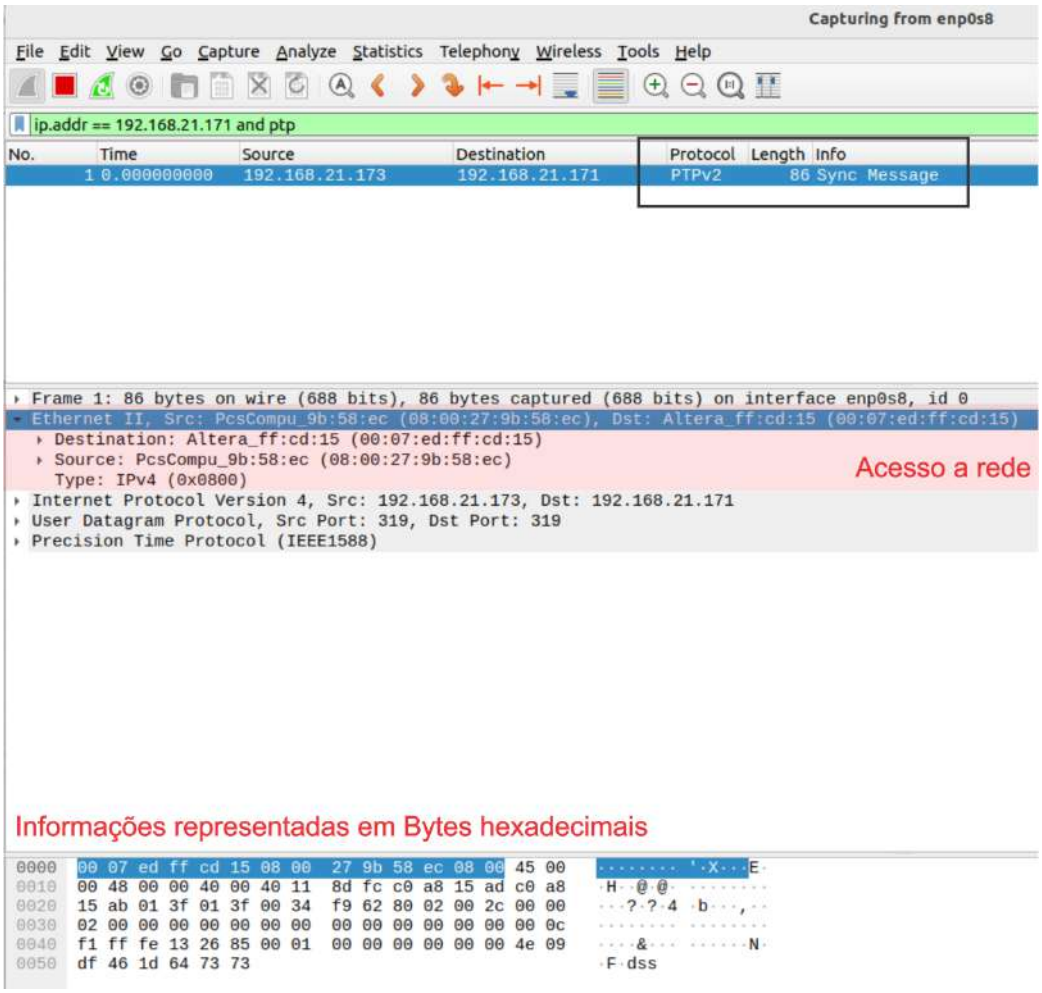


FONTE: Feito pelo Autor.

5.2 Análise das mensagens com o programa wireshark

Com o envio do pacote *Ethernet* pela interface de rede do computador a interface do kit DE2-115 já é possível capturar e analisar as informações pelo *wireshark*. Nele a análise das informações são organizadas pelo padrão de camadas do modelo TCP/IP'. A seguir são exibidas as informações da captura do pacote de rede com a mensagem PTPv2 *sync* do tipo evento pelo *wireshark*. Começando pela Figura 24 é visto os endereços IP's de origem (192.168.21.173) e destino (192.168.21.171) e a mensagem *sync* capturada. Mais abaixo é visto as informações que representam a camada de acesso a rede (em vermelho) e que foram configuradas no programa *packEth*, sendo elas os endereços MAC de origem (08:00:27:9B:58:EC) e destino (00:07:ED:FF:CD:15) e o (*Ethertype*) sendo o IPv4 (0x0800). Por último é demonstrado como as informações são representadas em hexadecimal.

Figura 24 – Análise da mensagem *sync* pelo wireshark na camada de acesso a rede.

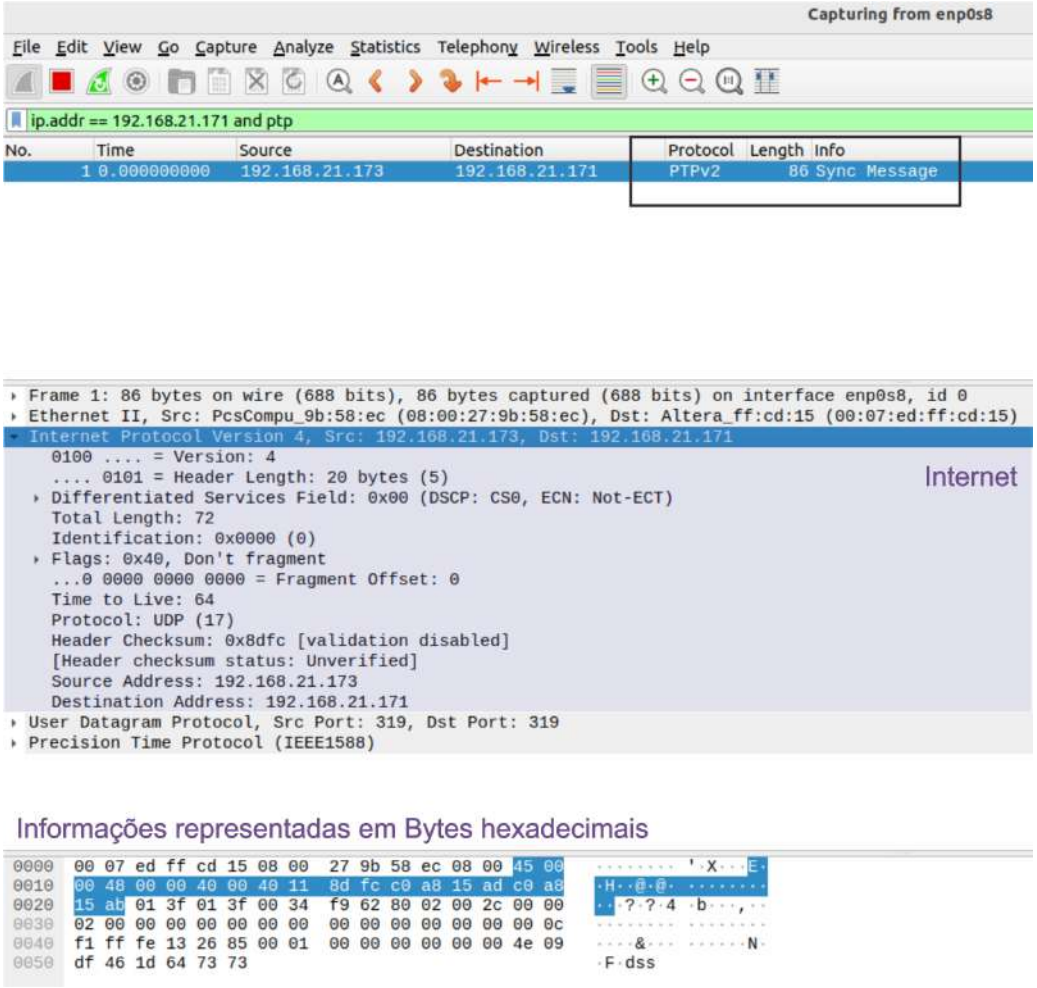


FONTE: Feito pelo Autor.

As próximas informações da mensagem *sync* analisadas pelo *wiresrak* são relacionadas a camada *internet* (em roxo), onde algumas informações inseridas no programa

packEth como os endereços IP's de origem (192.168.21.173) e destino (192.168.21.171) que estão presentes junto com outras informações que o programa configurou automaticamente. Por fim é mostrado em destaque esse conjunto de dados em hexadecimal. Todas essas descrições são observadas na Figura 25.

Figura 25 – Análise da mensagem *sync* pelo wireshark na camada de internet.

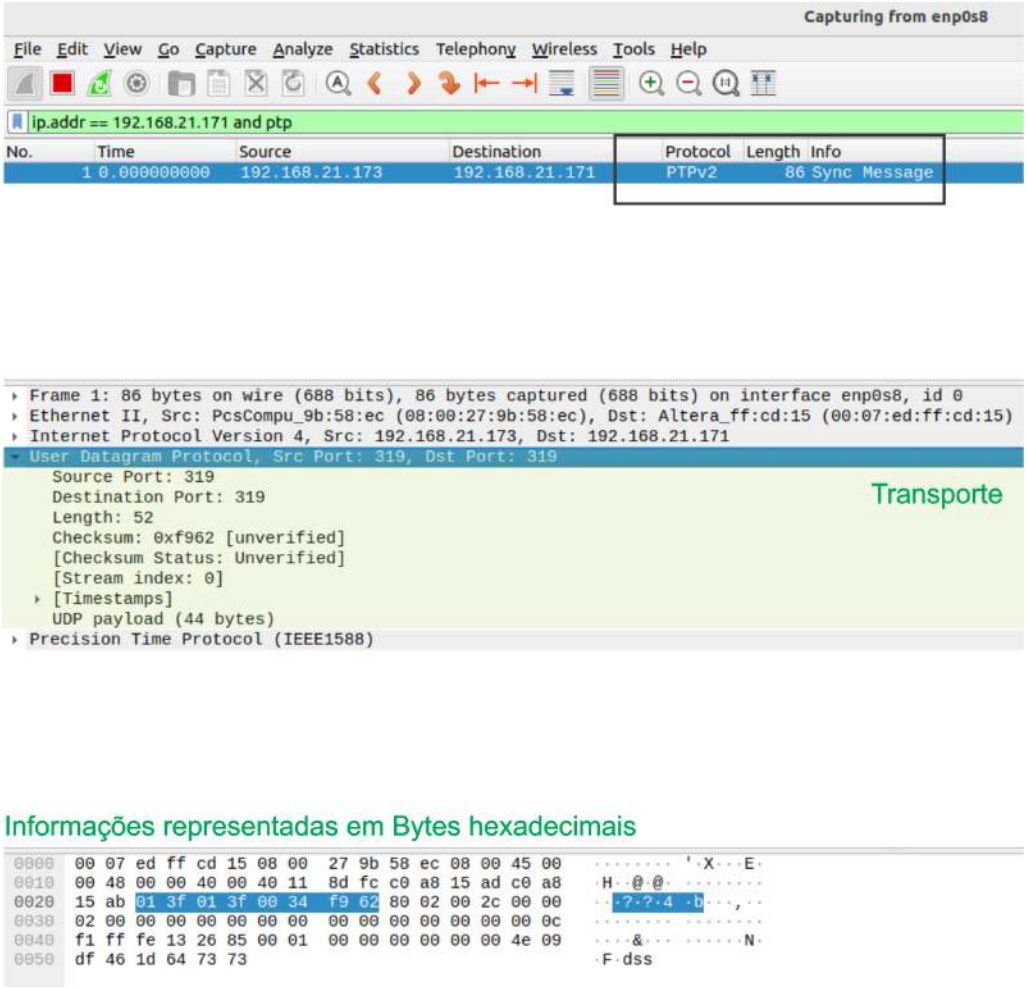


FONTE: Feito pelo Autor.

Seguindo adiante é a vez das informações da camada de transporte (em verde) serem analisadas pelo o *wireshark* como é ilustrado na Figura 26. Nela vimos as informações que foram inseridas a partir do *packEth* das portas de origem e destino com o valor 319 para envio e recebimento das mensagens do protocolo entre o computador e o a placa DE2-115, além de outras informações que o próprio programa insere automaticamente. Para terminar são apresentados ao fim da imagem os valores em hexadecimal desses dados.

Para finalizar a última análise do *wireshark* sobre o pacote *Ethernet* capturado com a mensagem *sync* é sobre os dados da camada de aplicação (em azul). Nela vimos as informações dos parâmetros que compõe o cabeçalho do protocolo PTPv2. Porém

Figura 26 – Análise da mensagem *sync* pelo wireshark na camada de transporte.

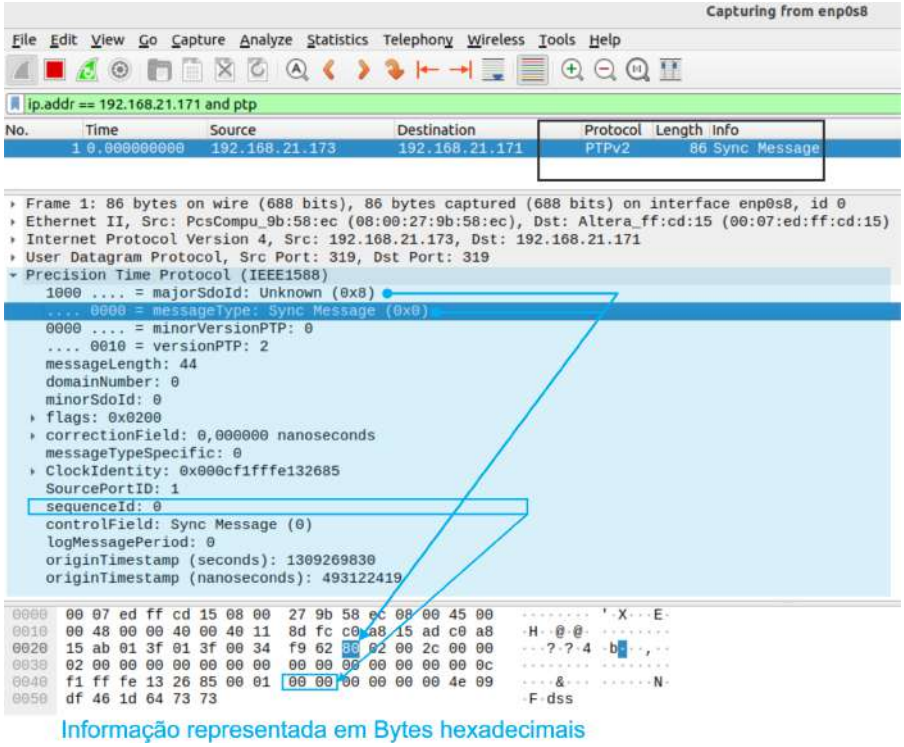


FONTE: Feito pelo Autor.

como já destacado os parâmetros relevantes aqui são "*messageType*" com o valor "0x0" que representa o "*messageId*" da mensagem *sync* e o "*sequenceId*" com o valor "0" que indica um número de sequência para as mensagens do protocolo, sendo essas informações exemplificadas na Figura 27. Essas informações estão em destaque pois fazem parte do resultado da saída "*ptp_found*" após o processamento do bloco HA1588.

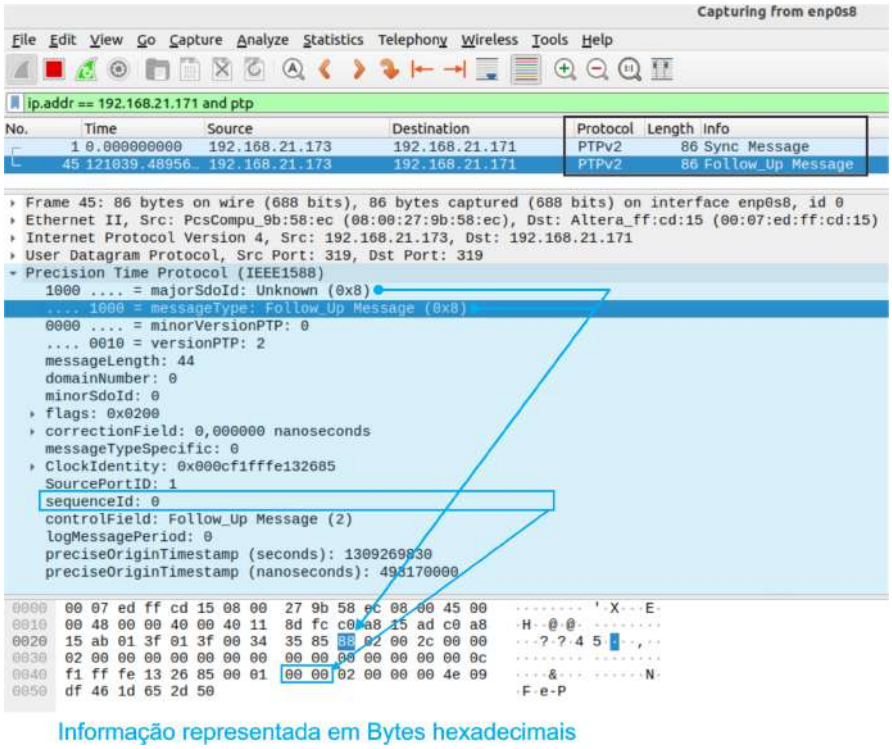
Em seguida foi analisado a mensagem *follow_up* exibindo o "*messageType*" com o valor "0x8" que representa o "*messageId*" da mensagem e o "*sequenceId*" com o valor "0", sendo essas informações exemplificadas na Figura 28.

Figura 27 – Análise da mensagem *sync* pelo wireshark na camada de aplicação.



FONTE: Feito pelo Autor.

Figura 28 – Análise da mensagem *follow_up* pelo wireshark na camada de aplicação.



FONTE: Feito pelo Autor.

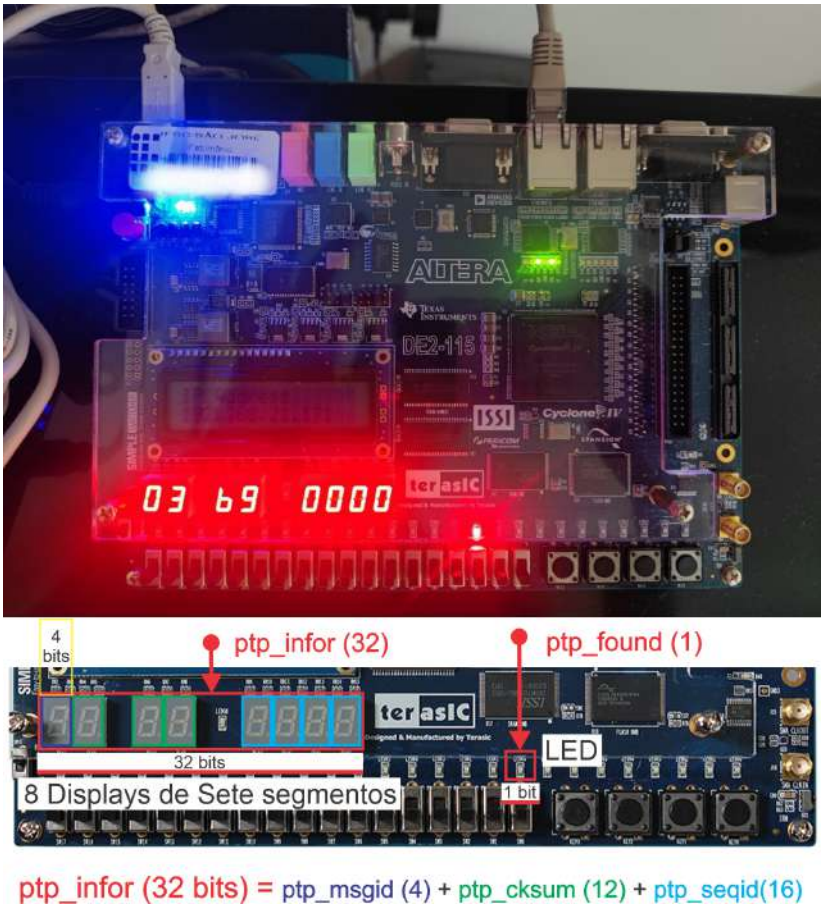
5.3 Identificação das mensagens pelo relógio secundário

A etapa final do cenário é o recebimento do quadro *Ethernet* com a mensagem PTPv2 enviado pelo programa *packETH* na interface de rede da placa DE2-115 representando o relógio secundário. Após o processamento das informações pelo bloco HA1588, os resultados das saídas ***ptp_found*** de 1 bit e ***ptp_infor*** de 32 bits são exibidos respectivamente por um LED e pelos oito *dispalys* de 7 segmentos da placa DE2-115.

5.3.1 Mensagem de evento sync

O resultado do precessamento da mensagem *sync* é exemplificado na Figura 29. É visto os mesmos resultados demonstrados na simulação pelo programa *modelSim* no envio da mensagem ***sync*** ao bloco. Sendo que o LED estando acesso, significa que o valor da saída ***ptp_found*** é 1, informando que uma mensagem PTPv2 do tipo evento foi identificada. Já nos oito *dispalys* de 7 segmentos (4 *bits* por *display*) é exibido o valor da saída ***ptp_infor*** onde os 4 *bits* mais significativos (em azul forte) representam a informação do ***ptp_msgid*** da mensagem recebida, sendo ***sync*** = 0x0.

Figura 29 – Kit FPGA como relógio secundário identificando a mensagem *sync*.



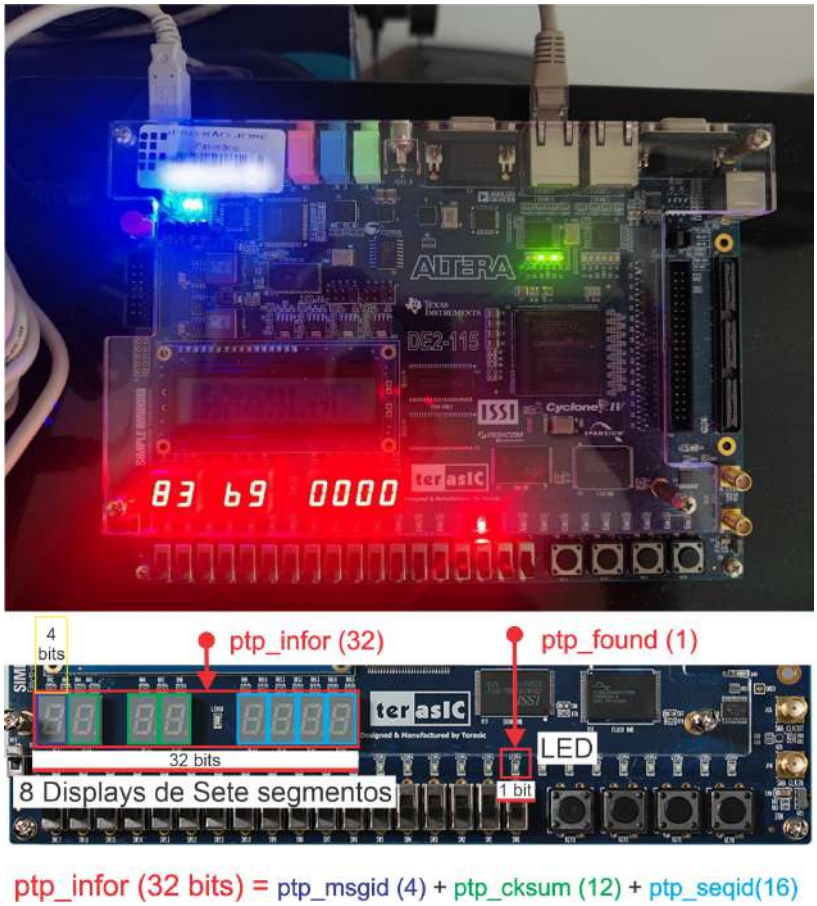
FONTE: Feito pelo Autor.

Nos próximos 12 *bits* seguintes é informado o valor do cálculo do *checksum* da mensagem, em ***ptp_cksum*** (em verde) no qual esse exemplo obteve o valor 0x3b9. Por último nos 16 *bits* menos significativos é exibido o valor do ***ptp_seqid*** (em azul claro) com o valor 0x0000, pois o valor do *sequenceId* dessa mensagem é "0". Dessa forma foi verificado a partir do teste no cenário prático que o envio da mensagem ***sync*** do relógio principal ao secundário, o mesmo obteve o resultado nas saídas *ptp_found* = 1 e *ptp_infor* = 0x03B9000, sendo o mesmo resultado obtido na simulação com o programa *modelsim*. Também foi feito o teste prático com as outras mensagens de evento, onde todas apresentaram a identificação correta nas saída do bloco. Assim, se constatou que na prática a lógica do bloco HA1588 processada pelo FPGA também funcionou corretamente.

5.3.2 Mensagem geral follow_up

O resultado do processamento da mensagem *follow_up* é exibido na Figura 30. Os oito *dispalys* de 7 segmentos exibem o valor da saída ***ptp_infor*** de 32 *bits*, e como já mencionado, os 4 *bits* mais significativos representam a informação do ***ptp_msgid*** da mensagem recebida, sendo ***follow_up* = 0x8**.

Figura 30 – Kit FPGA como relógio secundário identificando a mensagem *follow_up*.



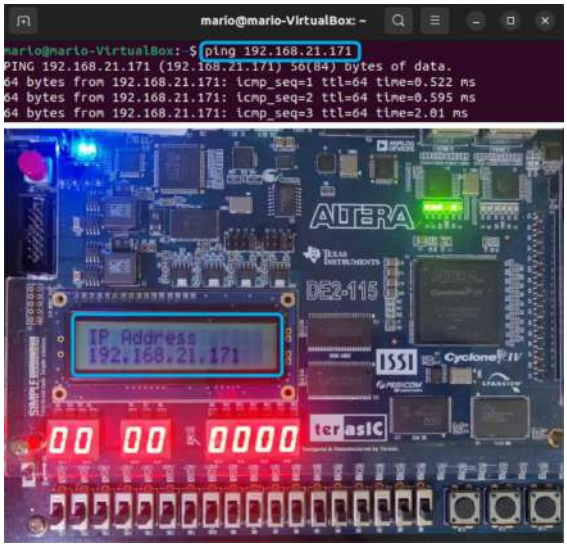
FONTE: Feito pelo Autor.

Nos próximos 12 *bits* seguintes é informado o ***ptp_cksum*** possuindo também o valor de 0x3b9. Por fim nos 16 *bits* menos significativos é exibido o valor do ***ptp_seqid***, também com o valor de 0x0000. Assim, nessa saída as informações da mensagem *follow_up* enviadas do relógio principal ao secundário foram informadas corretamente. Entretanto, o valor da saída ***ptp_found*** não foi o esperado, ficando diferente do obtido na simulação, onde o valor deveria ser zero pois a mensagem é do tipo geral. Dessa forma o LED não era pra acender, identificando incorretamente a mensagem geral follow_up como se fosse uma mensagem PTPv2 de evento. Também foi feito o teste prático com as outras mensagens gerais e o resultado foi que em todas, a saída ***ptp_found*** também obteve o valor 1, o que fez o LED acender. Verificado no código do bloco HA1588 se alguma alteração foi feita em relação a simulação gerando esse resultado, porém estava tudo igual.

5.3.3 Comando ping

Para verificar a correta implementação do código se executou o comando *ping* ao IP' da interface de rede da placa DE2-115. Como não foi enviado nenhuma mensagem PTPv2, o resultado das saídas foi o esperado, sendo ***ptp_infor* = 0x00000000** e o ***ptp_found* = 0** como ilustrado na Figura 31. Sendo assim, o LED não acendeu confirmando que a lógica do bloco está correta. Uma análise mais detalhada para uma possível correção foi feita, mas sem sucesso. Pelo pouco tempo para a data de entrega, foi decidido deixar a solução desse problema como trabalho futuro.

Figura 31 – Comando *ping* enviado à interface de rede da placa DE2-115.



FONTE: Feito pelo Autor.

O link[4] contém vídeos e imagens dos resultados obtidos nos cenários práticos executados com a mensagem *sync*, *follow_up* e o comando *ping*.

[4] https://drive.google.com/drive/folders/1c_U8CnPHIdyHQRWqU81yfD-YTvhjTQLV?usp=sharing

6 CONCLUSÕES E TRABALHOS FUTUROS

Durante a elaboração deste trabalho foram estudadas as características básicas de sincronização entre relógios de dispositivos eletrônicos e os protocolos disponíveis para tal. Mais especificamente, foi necessário compreender o funcionamento do protocolo de sincronização de tempo proposto pelo padrão IEEE 1588-2008 e como esse funcionamento foi mapeado na implementação da lógica do bloco de *hardware* HA1588.

Nesse estudo, foram criados diagramas detalhados do IP *core* HA1588 e seus submódulos – RGS, RTC e TSU – bem como foram executadas simulações para compreensão da lógica de funcionamento do bloco. Isso permitiu mapear dentro do TSU um componente de interesse na identificação das mensagens do protocolo, chamado de PTP *parser*, o qual foi isolado e teve suas saídas analisadas para que fosse possível identificar o tipo de mensagem recebida pela rede.

A etapa seguinte foi sintetizar o bloco HA1588 no *kit* de desenvolvimento FPGA DE2-115 para analisá-lo na prática. Para isso, foi selecionado o projeto exemplo *webserver*, que acompanha o *kit* de desenvolvimento e faz o uso da interface *ethernet*. Depois de verificar o correto funcionamento do exemplo, o bloco foi integrado na interface MII externalizando as saídas do PTP *parser* para demonstrar a identificação das mensagens.

O passo final foi implementar um cenário prático com o relógio principal enviando as mensagens PTPv2 ao relógio secundário. Para emular o relógio principal, foi utilizado o programa *packETH* e arquivos de dados contendo as mensagens PTPv2 de interesse. Para analisar o correto envio do pacote de rede foi utilizado o programa *wireshark* que captura e analisa as informações recebidas e enviadas da interface de rede.

Com o cenário funcional, os testes foram executados e conseguiu-se verificar que o bloco HA1588 identificou corretamente as mensagens do tipo *sync* enviadas do relógio principal ao secundário. Além disso, foram feitos mais testes práticos com as outras mensagens de evento e todas apresentaram a identificação correta nas saída do bloco. Entretanto, a mensagem geral $follow_up$ foi identificada como mensagem de evento, contrariando o esperado e o resultado da simulação.

Apesar dessa inconsciência, o cenário montado foi concluído e está funcional para servir como base para outros trabalhos. Sendo assim, como trabalho futuro sugere-se a verificação de outros submódulos do HA1588 e o uso em um cenário completo de sincronização do relógio entre dois sistemas.

REFERÊNCIAS

BUTLER, S. *O que é Núcleo de Propriedade Intelectual (Núcleo de IP)? Gerenciando o IP do Semicondutor*. 2023. Disponível em: https://www.perforce.com/blog/mdx/what-is-ip-core#what. Acesso em: 29 Jun 2023. 26

CISCO. *Conceitos básicos de PTP e SyncE com a configuração do Cisco IOS XR*. 2021. Disponível em: https://www.cisco.com/c/pt_br/support/docs/ios-nx-os-software/ios-xr-software/217579-configure-ptp-and-synce-basics-with-cisc.html. Acesso em: 25 jun 2022. 14, 17, 18, 19

DIGIKEY. *IP-25GEUMACPHYF*. 2018. Disponível em: https://www.digikey.com/en/products/detail/intel/IP-25GEUMACPHYF/9687188. Acesso em: 17 Ago 2023. 15

GUYS, S. *interface types (MII, GMII, RGMII) advantages and disadvantages – Ethernet Experiment – FPGA Board for Beginner – Experiment 14*. 2019. Disponível em: https://fraserinnovations.com/fpga-tutor/fpga-beginner-tutorial-ethernet-experiment-fpga-board-for-beginner-experiment-11/. Acesso em: 31 Jan 2023. 27

HBM. *Protocolo PTP na Aquisição de Dados e Ensaios*. 2019. Disponível em: https://www.hbm.com/pt/5143/protocolo-ptp-na-aquisicao-de-dados-e-ensaios/. Acesso em: 17 nov 2019. 19

LTD, C. S. *Implementing IEEE 1588v2 for use in the mobile backhaul %*. 2009. Disponível em: https://opencores.org/websvn/filedetails?repname=ha1588&path=. Acesso em: 06 jul 2022. 21, 23, 24, 25

NEWS, R. W. *IEEE 1588v2A Look at Time and Frequency Synchronization*. 2014. Disponível em: https://www.rcrwireless.com/20140513/wireless/ieee-1588v2. Acesso em: 15 out 2019. 14

OPENCORES. *Núcleo IP IEEE 1588 assistido por hardware*. 2012. Disponível em: https://opencores.org/projects/ha1588. Acesso em: 20 nov 2019. 26, 27, 30

PORTALCHAPECO. *Lista de Portas TCP/UDP*. 2022. Disponível em: https://www.portalchapeco.com.br/jackson/portas.htm. Acesso em: 07 Mar 2023. 43

QI, M.; WANG, X.; YANG, Z. Design and implementation of ieee1588 time synchronization messages timestamping based on fpga. In: . [s.n.], 2011. p. 1566 – 1570. Clock Synchronization;Design and implementations;Hardware circuits;Hardware solutions;IEEE1588;Media Independent Interface (MII);Precise time protocols;Time synchronization;. Disponível em: http://dx.doi.org/10.1109/DRPT.2011.5994146. 15

SAMBERG, L. *IEEE-1588 v2 (PTP)*. 2017. Disponível em: https://wiki.mef.net/pages/viewpage.action?pageId=29230774. Acesso em: 15 out 2019. 15

SCIENCEAQ. *Como calcular o deslocamento de fase*. 2018. Disponível em: http://pt.scienceaq.com/Math/100410963.html. Acesso em: 29 jun 2022. 18

TERASIC. *DE2-115 User Manual*. 2017. 10-11 p. Disponível em: http://www.terasic.com.tw/attachment/archive/502/DE2_115_User_manual.pdf. Acesso em: 01 Fev 2023. 37, 38

VICENTIN, R. de S. *IMPLEMENTAÇÃO DO PROTOCOLO DE TEMPO PRECISO (PTP) EM UMA MÁQUINA LINUX*. 2020. Disponível em: https://repositorio.ufsc.br/bitstream/handle/123456789/204604/TCC_Rafael_S_Vicentin.pdf?sequence=2&isAllowed=y. Acesso em: 17 Ago 2023. 21

Apêndices

APÊNDICE A – CÓDIGO FONTE

O código fonte em *Verilog* do IP *Core* HA1588 acoplado ao projeto *Web-server* encontra-se no link do *github* https://github.com/marioallan/TCC_IFSC_Eng_Tele_10F_2020-2_2023-1. Ou envie um e-mail para **marioallan.la@gmail.com**.

APÊNDICE B – DETALHAMENTO DOS DIAGRAMAS DE BLOCO DO IP CORE HA1588

Os detalhes dos diagramas de bloco criados do IP *core* HA1588 estão presentes no link a seguir https://drive.google.com/drive/folders/1Ywep7WPxm2VkIhavfVwKDjsmFyhpNJF0?usp=sharing