Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Computação
Departamento de Engenharia Elétrica

Trabalho de Conclusão de Curso

Modelagem de Arritmias Cardíacas em Hardware

para Validação e Avaliação de Marca-passos

Autor:

Gabriel Villanova N. Magalhães

Campina Grande, Paraíba
Data: 2018

Modelagem de Arritmias Cardíacas em Hardware

para Validação e Avaliação de Marca-passos

Autor:
Gabriel Villanova Novaes Magalhães

Orientador:
Prof. D.Sc. Gutemberg Gonçalves dos Santos Júnior

Prof. D.Sc. Gutemberg Gonçalves dos Santos Júnior

Prof. D.Sc. Antonio Marcus Nogueira Lima

Componentes da banca

Trabalho de Conclusão de Curso (TCC) apresentado no curso de Engenheria Elétrica, como requisito parcial para obtenção do título de Bacharel em Engenheria Elétrica na Universidade Federal de Campina Grande (UFCG).

Campina Grande, Paraíba
Data: 2018

"Nós somos o que fazemos repetidamente. A excelência, portanto, não é um ato, mas um hábito." - Aristóteles.

Resumo: Em décadas passadas, mais de 600.000 dispositivos de estimulação cardíaca sofreram *recall*, 40% devido a problemas de software [8]. Isso mostra que desenvolver firmwares livres de *bugs* para esses sistemas ainda é um grande desafio, e que as metodologias usadas para validação não conseguem identificar cenários que levem esses equipamentos a falha. Por serem sistemas críticos, podendo causar danos à saúde dos pacientes ou até mesmo a morte, é de extrema importância desenvolver equipamentos com alto nível de confiabilidade. Isso incentiva, portanto, a formalização de uma metodologia de validação para esses dispositivos. Uma abordagem que se mostra pertinente, é a criação de uma plataforma que simule diferentes condições de batimentos cardíacos e que permita a conexão com marca-passos comerciais, de tal forma a conseguir estimular e detectar falhas nesses sistemas, possibilitando ainda o *debug* e o levantamento de uma cobertura de testes suficiente para validação de quaisquer desses dispositivos. Além disso, permite também a avaliação da conformidade desses dispositivos pelas Normas Brasileiras para equipamentos eletromédicos implantáveis, em especial, os marca-passos cardíacos. Ademais, a existência de alguns *frameworks* consagrados pela comunidade acadêmica para modelagem de batimentos cardíacos com e sem arritmias, à exemplo do Virtual Heart Model [2]-[3] desenvolvido pela University of Pennsylvania e disponibilizado de forma gratuita, mostra-se promissor a criação de uma plataforma com os requisitos estabelecidos.

Palavras-chave: Validação, Sistemas Cíber-Físicos, Dispositivos Médicos, Autômatos, Sistemas de Tempo Real.

Abstract: In past decades, more than 600,000 cardiac pacing devices have been recalled, 40 % due to software problems [8]. This shows that developing bugs-free firmwares for these systems is still a major challenge, and that the methodologies used for validation can not identify scenarios that lead to such failure. Because they are critical systems that can cause harm to patients' health or even death, it is extremely important to develop equipment with a high level of reliability. This encourages, therefore, the formalization of a validation methodology for these devices. One approach that is pertinent is the creation of a platform that simulates different heartbeat conditions and allows the connection with commercial pacemakers, so as to be able to stimulate and detect flaws in these systems, also allowing the textit debug and the collection of sufficient test coverage for validation of any of these devices. In addition, it also allows the evaluation of the conformity of these devices by the Brazilian Standards (NBR) for implantable electromedical equipment, especially cardiac pacemakers. In addition, the existence of some frameworks established by the academic community for modeling heartbeats with and without arrhythmias, such as the Virtual Heart Model [2]-[3] developed by the University of Pennsylvania, is available free of charge. promising the creation of a platform with the established requirements.

Keywords: Validation, Cyber-Physical Systems, Medical Devices, Automata, Real-Time Systems.

"Eu dedico esse trabalho primeiramente a Deus, pois sem Ele nada disso seria possível. Aos meus pais, George e Angela, que sempre acreditaram no meu potencial, possibilitando a realização do sonho de me tornar Engenheiro. A minha namorada Mirtys, por todos os incentivos e cuidados. E a todos os meus professores, em especial, o Prof. Gutemberg Júnior, pelos ensinamentos, incentivos e confiança no meu trabalho."

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

VHM	Virtual Heart Model
E/S	Entrada e Saída
HDL	Hardware Description Language
FPGA	Field Programmable Gate Array
ECG	Eletrocardiograma
SA	Sinoatrial
AV	Atrioventricular
PAC	Premature Atrial Contraction
PVC	Premature Ventricular Contraction
HoC	Heart-On-Chip
PVS	Pacemaker Verification System
PCI	Placa de circuito impresso

Sumário

1 Introdução

Um grande desafio em projetos de dispositivos médicos, especialmente em aparelhos implantáveis complexos que controlam e acionam órgãos em contextos imprevisíveis, é seu desenvolvimento livre de *bugs* de software. Durante a década de 1990 e 2000 mais de 600.000 marca-passos sofreram *recall*. Destes, 200.000 (ou 41%), foram devido a problemas de firmware, e esse número está aumentando. Atualmente, não existe uma metodologia formal ou plataforma experimental para validar o correto funcionamento do software nesses dispositivos médicos [1].

A partir dessa observação, a University of Pennsylvania desenvolveu um *framework* no MATLAB/Simulink® para modelagem de tempo real do funcionamento eletrofisiológico do coração, em seu bom e mal funcionamento (isto é, durante a arritmia). A plataforma chamada Virtual Heart Model (VHM) [2]-[3], facilita a construção de modelos de coração com autômatos temporizados, em que os parâmetros desses autômatos são retirados de análises clínicas de sinais de eletrocardiogramas. O VHM ainda possui uma interface de E/S (Entrada e Saída) que permite a conexão de modelos de marca-passos, proporcionando a avaliação do comportamento do coração virtual perante estímulos do marca-passo e comportamento do marca-passo perante estímulos do coração.

Uma grande vantagem desse *framework*, se deve ao fato dele ter sido desenvolvido no MATLAB/Simulink®, que permite, por meio das *toolboxes* HDL Coder™ e Simulink® Coder™, gerar modelos em HDL (*Hardware Description Language*) ou linguagem C, respectivamente. Isso possiblita que os modelos construídos podem ser computados em FPGAs ou Microcontroladores, e algumas vantagens podem ser conseguidas, como: 1) implementação de um modelo de coração em hardware de tempo-real; 2) possibilidade de conectar marca-passos reais as plataformas computacionais, por meio de uma interface A/D (Analógica/Digital); 3) permitir uma avaliação do comportamento do marca-passos reais em um coração virtual.

Devido a capacidade de criação de diversos modelos de coração no *framework* VHM no MATLAB/Simulink®, é possível desenvolver uma plataforma de avaliação e validação do funcionamento de diversos marca-passos comerciais sobre diferentes contextos de funcionamento do coração, permitindo, portanto, a identificação de falhas nos softwares desses sistemas. Além disso, pode-se integrar a plataforma a função de verificar a conformidade desses dispositivos sobre as Normas Brasileiras de Equipamentos Eletromédicos Implantáveis, em especial, a NBR ISO 14708-1, 14708-2, 14708-6 e NBR IEC 60601-2-31.

2 Objetivos

O objetivo desse trabalho foi estudar a metodologia, desenvolvida pela University of Pennsylvania, de modelagem matemática do sistema de condução elétrica do coração humano com arritmias cardíacas, e a partir dessa ferramenta construir uma plataforma de hardware capaz de validar marca-passos reais e verificar se estes estão em conformidade com as Normas Brasileiras de Equipamentos Eletromédicos Implantáveis. Dado esse objetivo, o trabalho fora dividido em objetivos específicos, listados a seguir em ordem cronológica de execução:

1. Estudo do funcionamento do sistema elétrico cardíaco;
2. Estudo sobre arritmias cardíacas;
3. Modelagem matemática do sistema elétrico do coração com e sem arritmias;
4. Simulação de modelos utilizando o *framework* VHM;
5. Implementação em hardware dos modelos;
6. Implementação de inteface A/D para marca-passos;
7. Estudo de Normas Brasileiras de Equipamentos Eletromédicos Implantáveis;
8. Construção de uma plataforma de avaliação de marca-passos.

3 Fundamentação Teórica

3.1 Sistema de condução elétrico do coração

Nesse seção, serão abordados os conhecimentos fundamentais do funcionamento do sistema de condução elétrico do coração, sendo eles: 1) as células básicas que compõe esse sistema; 2) geração e propagação de corrente elétrica; 3) os processos de despolarização e repolarização; 4) como a propagação de corrente no coração é refletida no ECG, ou seja, como se configura a diferença de potencial lido pelos eletrodos superficiais ao coração. Esse estudo, introdutório, dará base para o entendimento e desenvolvimento dos modelo matemáticos do sistema elétrico do coração contidos nesse relatório.

O coração pode ser visto como uma composição de três tipos de células, sendo elas: células de marca-passo, que em condições normais é a fonte de eletricidade do coração; as células de condução elétrica, ou seja, os caminhos da progragação elétrica no coração; células miorcárdicas, que faz a função mecânica de contração e relaxamento do coração. Essas células podem ser referenciadas como células cardíacas.

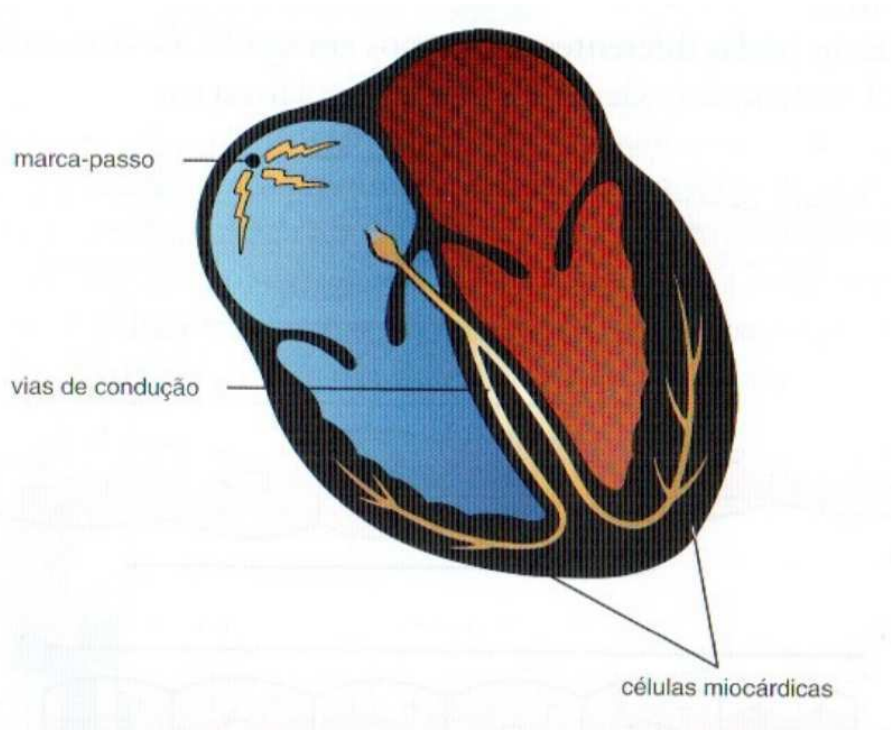


Figura 1: Esquema de coração e seus três componentes fundamentais (Fonte: [9]).

Para entender como cada uma das células cardíacas executam sua função, é preciso entender inicialmente a eletrofisiologia delas. Basicamente, em seu estado de repouso, essas células são eletricamente polarizadas, ou seja, o seu meio interno é carregado negativamente em relação ao seu meio externo por conta de uma desequilibrio iônico natural de íons de potássio, sódio, cloro e cálcio [9], como pode ser visualizado nas Figuras 2, 3.

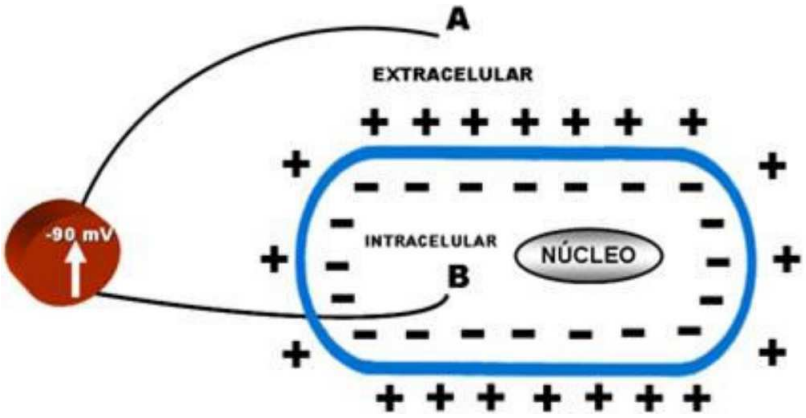


Figura 2: Diferença de potencial entre o meio intracelular e extracelular (Fonte: [10]).

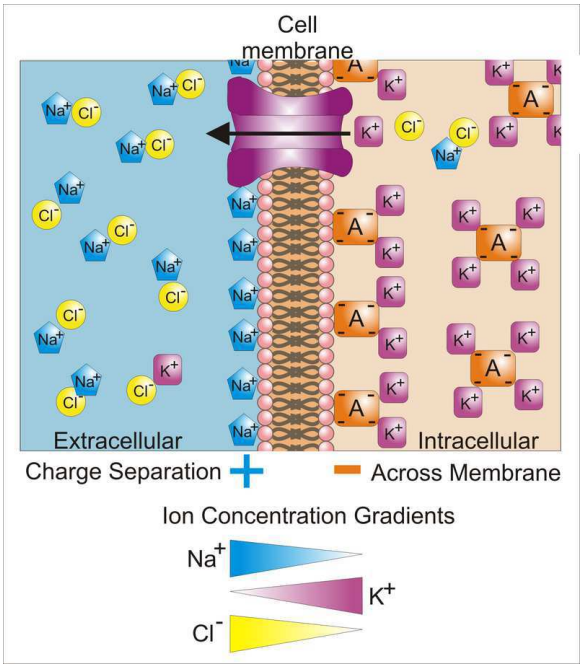


Figura 3: Desequilíbrio iônico entre um meio intra e extracelular (Fonte: [11]).

As células de marca-passo possuem uma capacidade de se despolarizar de forma espontânea [1]. A **despolarização** é o evento elétrico fundamental do coração e representa o processo de perda de elétrons nas células [9]. Quando a despolarização acontece nas células de marca-passo, faz com que as células vizinhas se despolarizem, fazendo com que se propague um sinal elétrico por toda a via de condução do coração. A frequência que ocorre a despolarização nessas células é variável e é em função das característica inatas da célula e dos estímulos neuro-hormonais externo, sendo normalmente algum valor entre 60 a 100 vezes por minuto. Em sua grande maioria, as células de marca-passo ficam localizadas no átrio direito do coração e esse agrupamento é chamado pelos cardiologistas de **nó sinoatrial** (SA) e fica localizado na parte superior do átrio direito.

[1] As demais células cardíacas podem se despolarizar, mas não de forma espontânea e sim de forma induzida.

Após a despolarização estar completa, as células cardíacas restauram a sua polaridade de repouso por meio de um processo chamado de **repolarização** [2]. Todas as ondas de um ECG são manifestações destes dois processos: despolarização e repolarização [9], ou seja, o funcionamento do coração é basicamente a execução infinita (ou pelo menos enquanto o coração estiver funcionando) desses dois processos, um seguido do outro.

Se registrassemos o sinal elétrico de tensão no ciclo de despolarização e repolarização em uma única célula, seria observado uma forma de onda similar a apresentada na Figura 4, chamada de pontencial de ação. Ela representa o ciclo completo de despolarização e repolarização em um célula. Quando uma célula de marca-passo gera um potencial de ação desse, as células vizinhas reproduzem tal comportamento, isto é, ocorre despolarização e repolarização nas demais células, causando um efeito de propagação no tempo do potencial de ação, como apresentado na Figura 5. É importante entender esse conceito para a modelagem matemática do coração feita posteriormente.

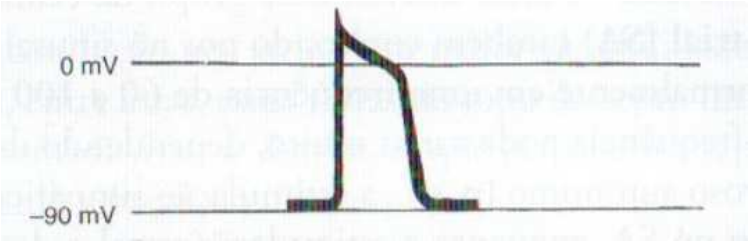


Figura 4: Potencial de ação genérico (Fonte: [9]).

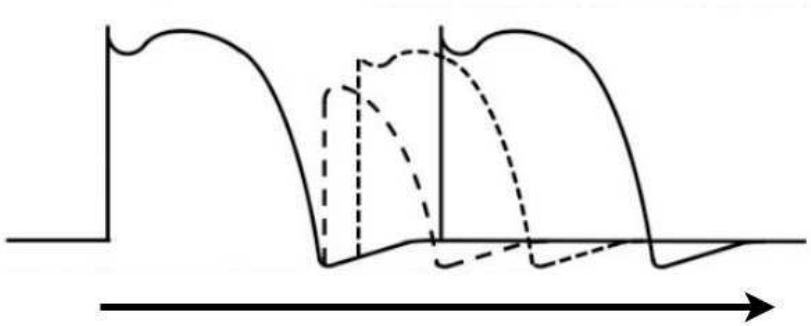


Figura 5: Propagação no tempo do potencial de ação nas células.

Em seguida tem-se as células de condução elétrica, as responsáveis pela condução de corrente elétrica pelo coração. Os caminhos que elas constroem são essenciais por ter a função de retardar a propagação do potencial de ação, necessário para que o fluxo sanguíneo ocorra de forma saúdavel, e transportar a corrente elétrica para regiões distantes do coração. A Figura 6, ilustra de forma simplificada como essas células estão espalhadas no coração.

[2]Esse processo é realizado pelas bombas transmembrana, que invertem o fluxo de íons [9].

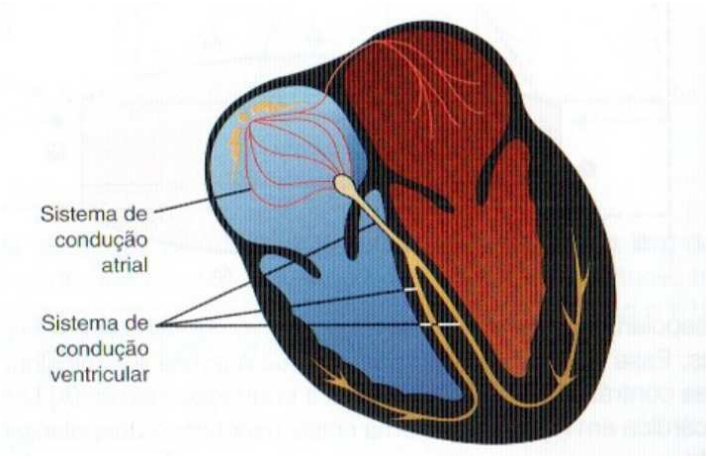


Figura 6: O sistema elétrico do coração (fonte: [9]).

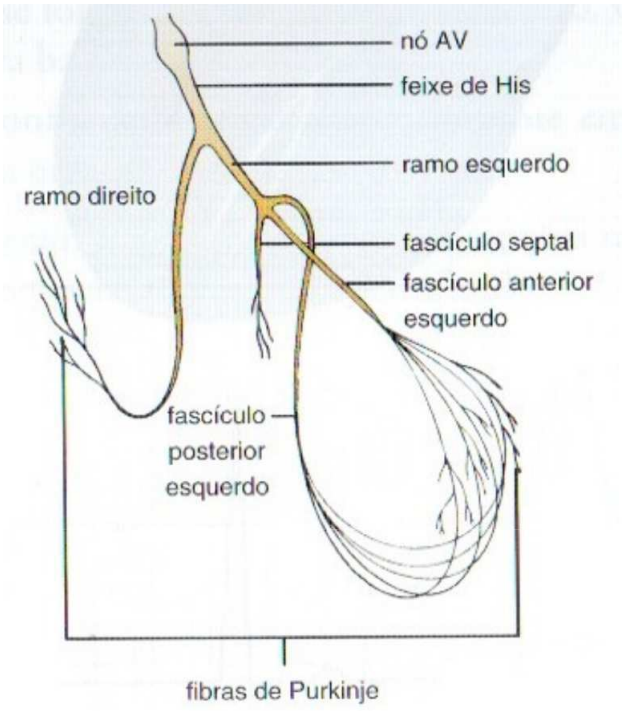


Figura 7: Circuito elétrico do coração (fonte: [9]).

Para entender a importância dessas células no coração, é preciso entender como um coração saúdavel executa o bombeamento sanguíneo. O sangue chega no coração pelos átrios e nesse momento inicial o ciclo de despolarização se inicial no nó SA. Isso faz com que aja propagação de corrente elétrica nos átrios, ocasionando uma contração e abertura do "portão elétrico" que separa os átrios dos ventrículos (devido a corrente elétrica). Quando o pontecial de ação chega ao **nó atrioventricular** (AV), um novo atraso de progação da corrente acontece, isso permite que os átrios termine de expulsar o sangue antes que os ventrículos se contraiam e expulsem o sanguem ali contido para foi do coração. Na Figura 7 pode-se observar o complexo circuito elétrico do coração e como os cardiologistas chamam cada ramo dele.

Por fim, existem as células miorcárdicas, elas constituem a grande maioria do tecido cardíaco, sendo as responsáveis pelo trabalho mecânico de contração e relaxação do coração. Quando uma onda de despolarização atinge a célula miorcárdica, o cálcio é liberado para dentro da célula e isso leva a contração [9].

As células miorcárdicas podem trasmitir uma corrente elétrica do mesmo modo que as células de condução elétrica, mas o fazem com menos eficiência. Assim, uma onda de despolarização, ao atingir as células miorcárdicas, irão se espalhar lentamente por todo o miorcárdio [3] [9].

O sinal elétrico de tensão que chega ao miocárdio, que é geralmente lido por eletrodos, são as formas de onda vistas nos ECGs, em outras palavras, os eletrodos usados para capturar o comportamento do coração não "enxerga" os potenciais de ação nas células, na verdade, eles capturam o efeito dos potenciais de ação no tecido muscular cardíaco, o miorcárdio. O ECG, portanto, reflete cada etapa do ciclo de funcionamento do coração, da despolariazação a repolarização, no miorcárdio. A Figura 8 ilustra a relação entre o potencial de ação na célula e como isso é refletido no ECG.

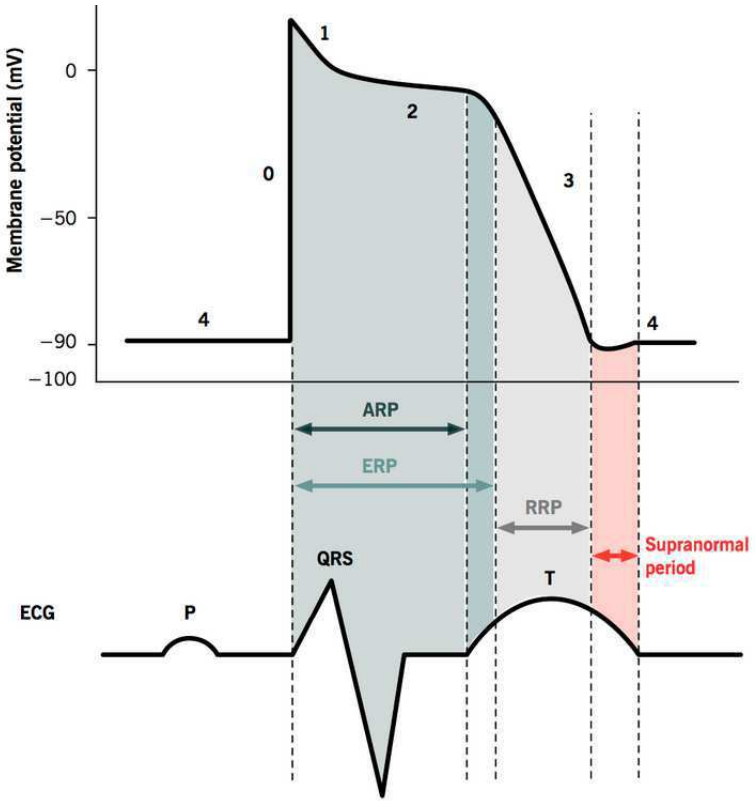


Figura 8: Sinal de ECG e relação com potencial de ação (fonte: [12]).

[3] O miocárdio é um músculo cardíaco, isto é, a própria parede do coração. A porção mais interna se chama endocárdio e a mais externa epicárdio. O miocárdio é composto por um tecido muscular especial, o músculo cardíaco, e tem como função básica ejetar o sangue que se encontra no interior do coração (fonte: Wikipédia).

Basicamente, o sinal de ECG é dividido em três regiões, P, QRS e T. A primeira, P, representa a despolarização gerada no nó SA, ou seja, nas células de marca-passo, e que se propaga para os dois átrios, direito e esquerda. A próxima, o complexo QRS, é na verdade um conjunto de sinais que pode variar esse nome de acordo com o formato da onda, mas em corações saúdaveis é do formato QRS, como apresentado na Figura 8, e representa a atividade cardíaca de despolarização nos ventrículos direito e esquerdo. Enfim, o sinal T representa a repolarização, restaurando a eletronegatividade do coração para que o ciclo se repita.

Na Figura 8, pode ser observado ainda as regiões ARP, ERP, RRP e Período supranormal, no sinal do ECG, elas representam alguns perídos do ECG com algumas propriedades importantes do coração, sendo elas as seguintes [12]:

- Período refratário absoluto (ARP): a célula é completamente impraticável para um novo estímulo;
- Período refratário efetivo (ERP): ARP + segmento curto da fase 3 durante o qual um estímulo pode fazer com que a célula despolarize minimamente, mas não resultará em um potencial de ação propagado (ou seja, as células vizinhas não despolarizam);
- Período refratário relativo (RRP): um estímulo maior que o normal despolariza a célula e causa um potencial de ação;
- Período supranormal: um período hiperexcitável durante o qual um estímulo mais fraco que o normal irá despolarizar as células e causar um potencial de ação. As células nessa fase são particularmente suscetíveis a arritmias quando expostas a estímulos inapropriadamente cronometrados, motivo pelo qual é necessário sincronizar o estímulo elétrico durante a cardioversão para evitar a indução de fibrilação ventricular.

Essas regiões, definem alguns comportamentos importantes que auxiliam na construção de um modelo matemático do sistema elétrico do coração, como será visto posteriormente. Na próxima seção, serão abordados algumas das principais arritmias, que também foim modeladas matematicamente.

3.2 Arritmias cardíacas

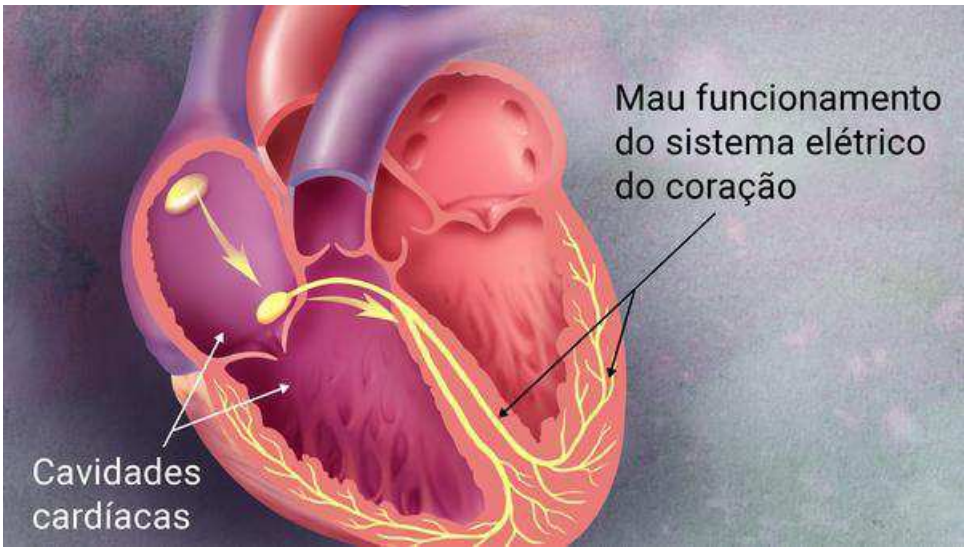


Figura 9: Desenho do coração (fonte: `https://g.co/kgs/taFzYz`.)

Como já foi visto na Seção 3.1, o ritmo cardíaco é estabelecido pela despolarização espontânea do nó SA, dessa forma toda alteração desse ritmo normal (também chamado de ritmo sinusal normal) gerado pelo nó SA é considerado uma arritmia. O termo arritmia se refere a qualquer distúrbio na frequência, na regularidade, no local de origem ou na condução do impulso elétrico cardíaco [9]. Existem diversos tipos de arritmias, porém, será revisado aqui apenas as que foim modeladas matematicamente, sendo elas: a taquicardia, bradicardia, contração atrial prematura e a contração ventricular prematura.

Considera-se que a frequência normal [4] do batimento cardíaco varia entre 60 a 100 batimento por minuto (bpm). Com base nesses valores, pode-se dizer quando um batimento está acima ou abaixo do normal. A taquicardia e bradicardia, designam exatamente isso, pois sigificam, respectivamente, um aumento e uma dimunuição no ritmo cardíaco normal.

A taquicardia, significa que há um aumento na frequência cardíaca, e ela pode se iniciar tanto nos ventrículos como nos átrios, ou seja, o processo de despolarização ocorre em uma frequência alta, em diferentes partes do coração. Devido a essa elevação do batimento cardíaco de forma desordenada, o coração não consegue bombear de forma eficiente o sangue para o resto do seu corpo, precisando de tratamento, sendo um deles o emprego de marca-passos artificiais.

[4] É claro que ao fazer um exercício físico o batimento cardíaco se elevará ou se estiver em repouso o batimento decairá, porém, no contexto da arritmia essa elevação ou decaimento ocorre quando o ritmo cardíaco deveria estar normal, entre 60 ou 100 bpm.

A bradicardia, por sua vez é classificado pela diminuição da frequência cardíaca, e é causada normalmente pelo mal funcionamento do nó SA, ou seja, as células marca-passo não executam o processo de despolarização numa frequência normal e sim, abaixo do normal, o que causa uma lenta propagação da corrente elétrica pelo coração. Para essa patologia um dos tratamentos também é o uso de marca-passos artificiais, que pode enviar sinais elétricos para o coração com intuito de restabelecer o ritmo normal.

Além dessas, existem a Contração Atrial Prematura, mais conhecida como PAC (do inglês, *Premature Atrial Contraction*) e a Contração Ventricular Prematura, mais conhecida como PVC (do inglês, *Premature Ventricular Contraction*). Basimente, como o próprio nome já diz, o PAC é caracterizado pelo processo de despolarização antecipada nas partes superiores do coração, atrio direito e esquerdo. Por outro lado, o PVC é caracterizado pelo aparecimento desse processo de forma antecipada nos ventrículos. Em ambas, o tratamento geralmente é feito por fármacos, sem necessidade, a priori, de marca-passos eletrônicos.

Na próxima seção será visto que a modelagem matemática do sistema de condução elétrica do coração pode ser desenvolvida conhecendo os intervalos de tempo da propagação dos potenciais de ação nas células do coração. Isso significa, que a partir de um modelo matemático genérico do coração pode-se derivar as arritmias cardíacas apenas alterando as variáveis de tempo.

3.3 Modelagem matemática do sistema elétrico do coração

A modelagem matemática desenvolvida e apresentada nessa seção é fruto de um projeto de pesquisa que foi iniciado pelo grupo mLab [5] da University of Pennsylvania (UPenn) com objetivo de desenvolver e validar projetos de marca-passos.

Uma das metodologias de validação já desenvolvidas é a modelagem matemática da condução elétrica do coração por autômatos temporizados, em que o modelo, por meio de uma interface, permite a interação entre um marca-passo e o coração virtual, proporcionando a validação e teste de caixa preta dos dispositivos implantáveis [1].

3.3.1 Modelo por nós

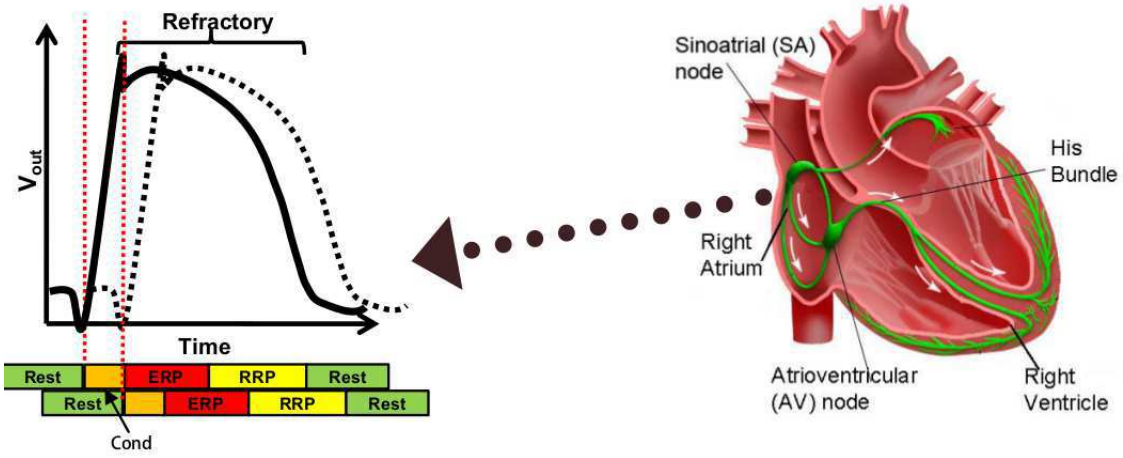


Figura 10: Propagação do potencial de ação e seus respectivos estados no tempo (fonte: [3].)

Um modelo matemático para o coração pode ser desenvolvido modelando as células individuais do coração, porém isso gera um alto esforço computacional e é irrelevante para testar marca-passos. Em vez disso, pode-se analisar as propriedades temporais do coração em um nível de abstração maior, dividindo cada parte do sistema elétrico do coração e o modelando por um autômato temporizado [1].

Como ja foi visto na Seção 3.1, cada célula passa pelo ciclo de despolarização e repolarização, e esse efeito gera uma curva chamada potencial de ação que é dividida em 4 estados, Rest, Cond, ERP e RRP, como pode ser observado na Figura 10. Esses estados são mantidos por um determinado período de tempo [6] de tal forma, que conhecendo esses tempos, pode-se criar uma rede de autômatos temporizados que se comunicam entre si, e com isso, respeitando as restrições de tempo, modelar o efeito de propagação de corrente elétrica do coração.

[5] Página oficial: `http://mlab-upenn.github.io/medcps/`
[6] Os intervalos de tempo foim retirados de experimentos laboratóriais, como mostra [1].

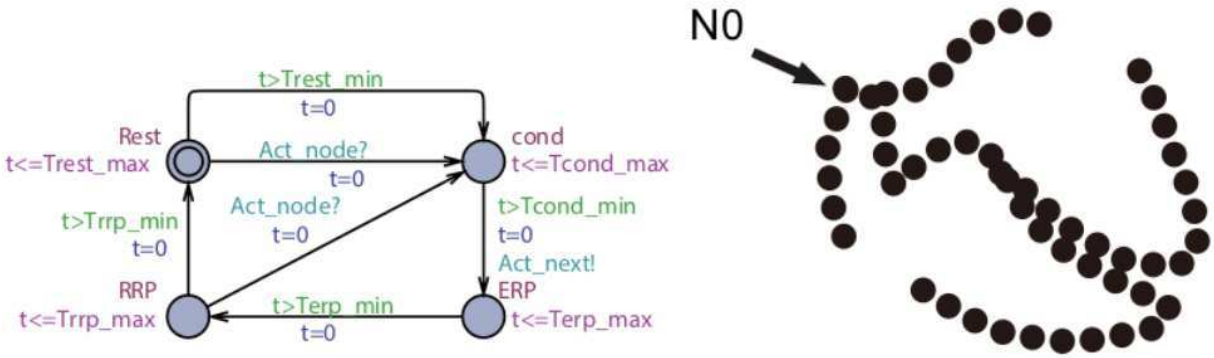


Figura 11: Modelo inicial dos tecidos do coração (fonte: [3].)

Entretanto, apenas o efeito de temporarização não é suficiente, pois é preciso que as "macro células" modeladas sejam capazes de se despolarizar foi do ciclo. Isso possibilita a modelagem de diversos tipos de arritmias e permite que estímulos de marca-passo atuem e controlem o modelo. Dessas observações, uma primeira síntese para cada autômato ou nó foi feito, conforme é apresentado na Figura 11.

Em resumo, todo tecido, representado por um ponto (ou nó) na Figura 11, se inicia no estado Rest (descanso) que pode se auto despolarizar ou ser despolarizado por um evento externo de entrada chamado *Act_ node?*, quando algum desses eventos acontecem o autômato alcança um novo estado, o Cond (condução). Nesse estado, a mesma lógica é atribuida, com a diferença que ele envia um evento externo, chamado *Act_ next!* para ativar a despolarização na próxima célula (propagando o sinal), isso acontence quando o tempo de duração do estado é atingido. O próximo estado é o ERP, sua característica é que ele não sofre processo de despolarização externa, pois representa o processo de repolarização, então quando o tempo de duração é atingido o autômato alcança o último estado, o RRP. Esse, representa o fim da repolarização e pode sofrer auto despolarização e despolarização externa pelo evento *Act_ node?* [1].

De posse das característica temporais de cada autômato de nó, representado por N1, N2, ..., Nn, o modelo do coração, ou a rede de autômatos, pode ser construído a partir da composição paralela deles, isto é:

$$H = N_1||N_2||...||N_n \tag{1}$$

O problemas desse modelo é que, para um coração real, o número de nós é muito grande e seus valores de parâmetros temporais são difíceis, talvez impossíveis, de determinar, o que torna a verificação de H inviável [3]. Além disso, por ser um sistema de tempo real, o processamento deve respeitar as restrições de tempo, e o resultado das composições paralelas dos autômatos faz crescer exponencialmente o número de estados do modelo, inviabilizando portanto o processamento em tempo real. Na Figura 12, por exemplo, é ilustrado o resultado da composição paralela entra dois nós, perceba que o número de estados cresce de 4 (autômato de nó) para 14 (composição).

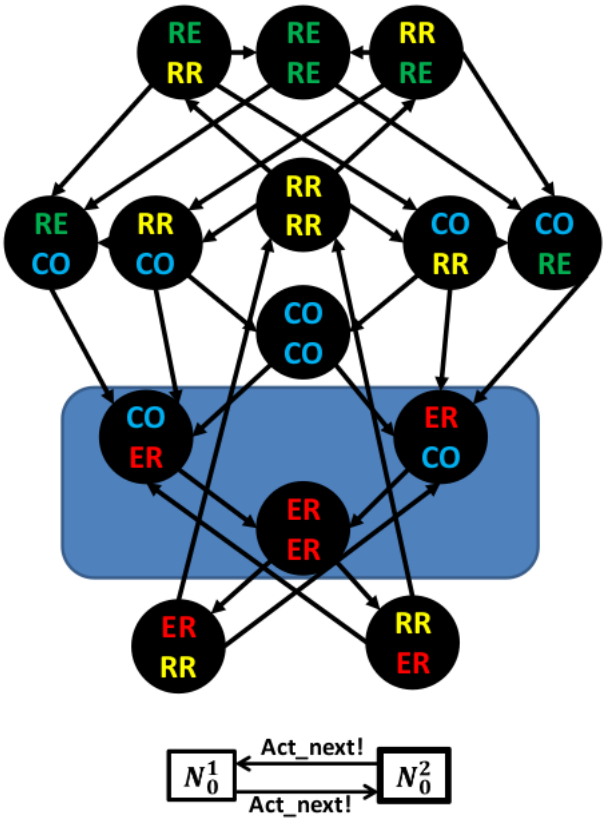


Figura 12: Composição paralela entre dois nós (fonte: [3].)

Dessa forma, pensou-se em uma nova forma de abstração, dividir o coração em autômatos de caminho e nó, essa abordagem será discutido na próxima seção.

3.3.2 Modelo de nós e caminhos

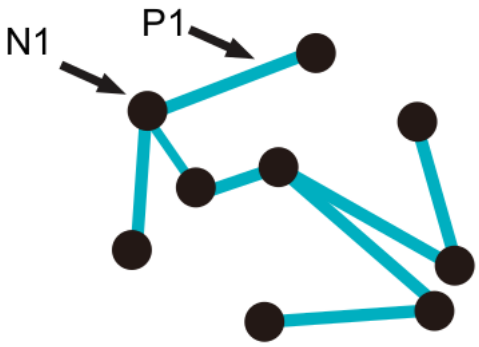


Figura 13: Modelo de coração com nós de extermidade e caminho entre eles (fonte: [3].)

Observou-se que o coração poderia ser modelado não somente com nós, mas com nós (N) e caminhos (P), conforme apresentado na Figura 13. Isso reduz significativamente a quantidade de estados do autômato resultantedo modelo, consequentemente o esforço computacional também diminui, proporcionando o processamento em tempo real, necessário para conexão de marca-passos reais ao modelo.

Além disso, o autômato de nó foi modificado, reduzindo seus estados de 4 para 3. Na verdade considerou-se o potencial de ação agora com apenas os estados: Rest, ERP e RRP, excluindo o estado Cond, como pode ser observado na Figura 14.

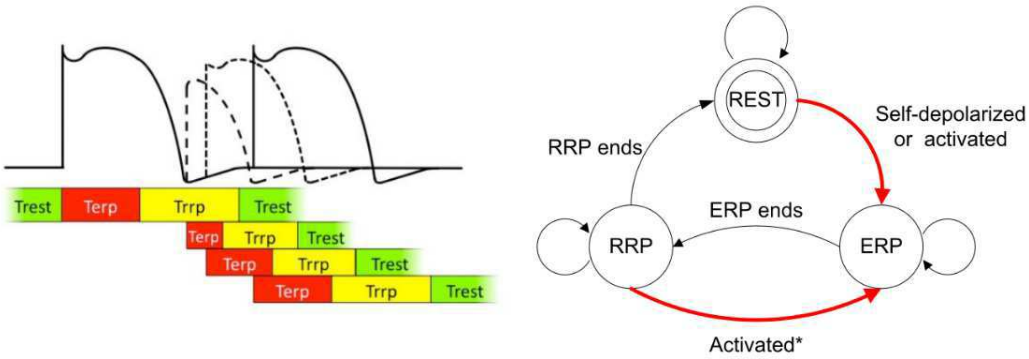


Figura 14: Estados do potencial de ação (fonte: [1].)

Os autômatos de nó começam no estado Rest, nesse estado um temporizador *count down* inicia a contagem até atigir o valor zero, quando isso ocorre auto despolarização, ou seja, o estado sofre uma transição e ativa [7] o autômato de caminho vizinho, porém, esse estado pode sofre um despolarização foi de hora, causada pelo evento externo *Act_Node!*, e isso causa transição para estado ERP de forma antecipada. No estado ERP, um temporizador *count down* também é iniciado, ao atingir zero causa transição, ele não pode ser ativado por eventos externo. Por fim, o estado RRP, executa o mesmo processo, após seu

[7]Nesse contexto, ativar um autômato significa dizer que esse autômato começara a computar sua lógica.

temporizador associado atingir o tempo zero, o autômato atinge o estado Rest, porém, pode sofrer ativação externa evento (*Act_Node!*) quando isso acontece o estado alcançado é o ERP e um novo sinal de ativação para o autômato de caminho é enviado. O ciclo se repete para cada autômato de nó e os valores de tempo utilizados e as equações são detalhadas em [1].

O autômato de caminho, ilustrado na Figura 15 inicia no estado Idle até que um dos nós aos quais ele se conecta os ative (eventos *Act_1* ou *Act_2*). O autômato inicia o temporizador de condução anterógrado ou retrógrado de acordo com o evento de ativação. Depois que o cronômetro anterógrado ou retrógrado expirar, o caminho ativará o nó na extremidade oposta (evento *Act_Node!*). Como esse nó ativa todos os caminhos aos quais ele está conectado, o caminho em que a ativação se originou deve ir para o estado Conflict para evitar o refluxo. Se um caminho já estiver em condução anterógrada (estado Ante) ou retrógrada (estado Retro) quando um segundo sinal de ativação entrar pela extremidade oposta, o caminho entrará no estado Double. O temporizador Ante e Retro conta até que os temporizadores correspondam ao mesmo local e o caminho vai para o estado Conflict [1].

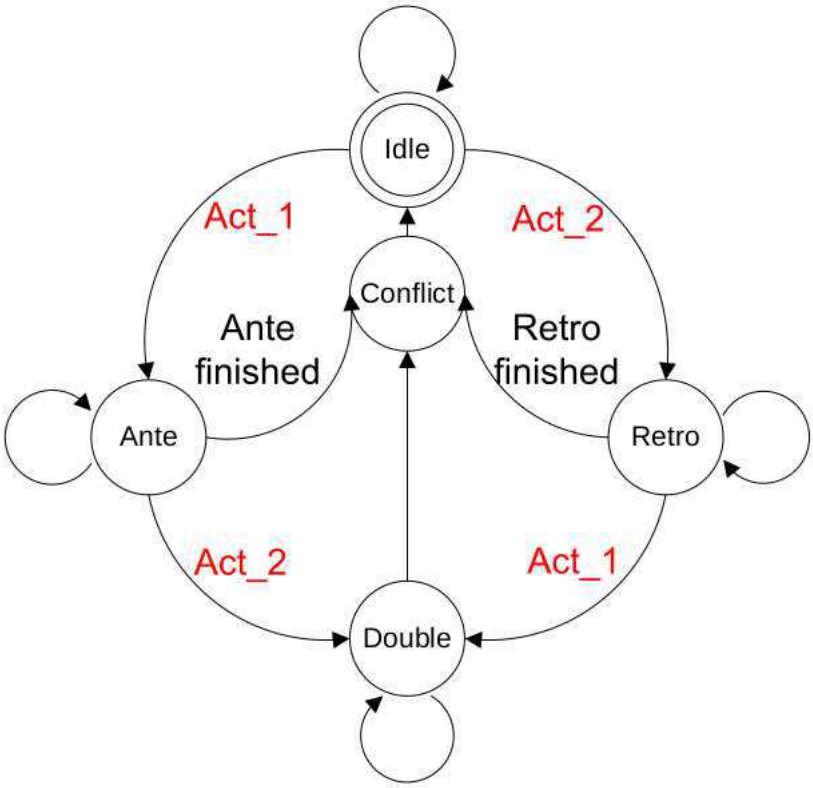


Figura 15: Autômato de caminho do modelo de coração (fonte: [1].)

Essa abordagem reduz a quantidade de estados do modelo do coração, por diminuir a quantidade de autômatos necessários para compor o modelo, isto é, a Equação 2 produz menos estados que a Equação 1. Na Figura 16 é possível observar que a redução de estados é significativa, tendo em vista que esse autômato representa um ramo de condução inteiro do coração. Para exemplificar, imagine um ramo com 9 autômatos de nó, o autômato resultante desse ramo seria calculado segundo a Equação 3, supondo 3 estados para cada nó, essa operação geraria aproximadamente $3^9 = 19683$ estados. Substituindo o intervalo por apenas um autômato de caminho a equação é simplificada para a Equação 4, onde aproximadamente $3 * 5 * 3 = 45$ estados seriam criados, considerando que o autômato de caminho contém 5 estados.

$$H = N_1||P_1||N_2||...||P_n||N_n \tag{2}$$

$$old = N_1^1||N_1^2||N_1^3||N_1^4||N_1^5||N_1^6||N_1^7||N_1^8||N_1^9 \tag{3}$$

$$new = N_1^1||P_1||N_1^9 \tag{4}$$

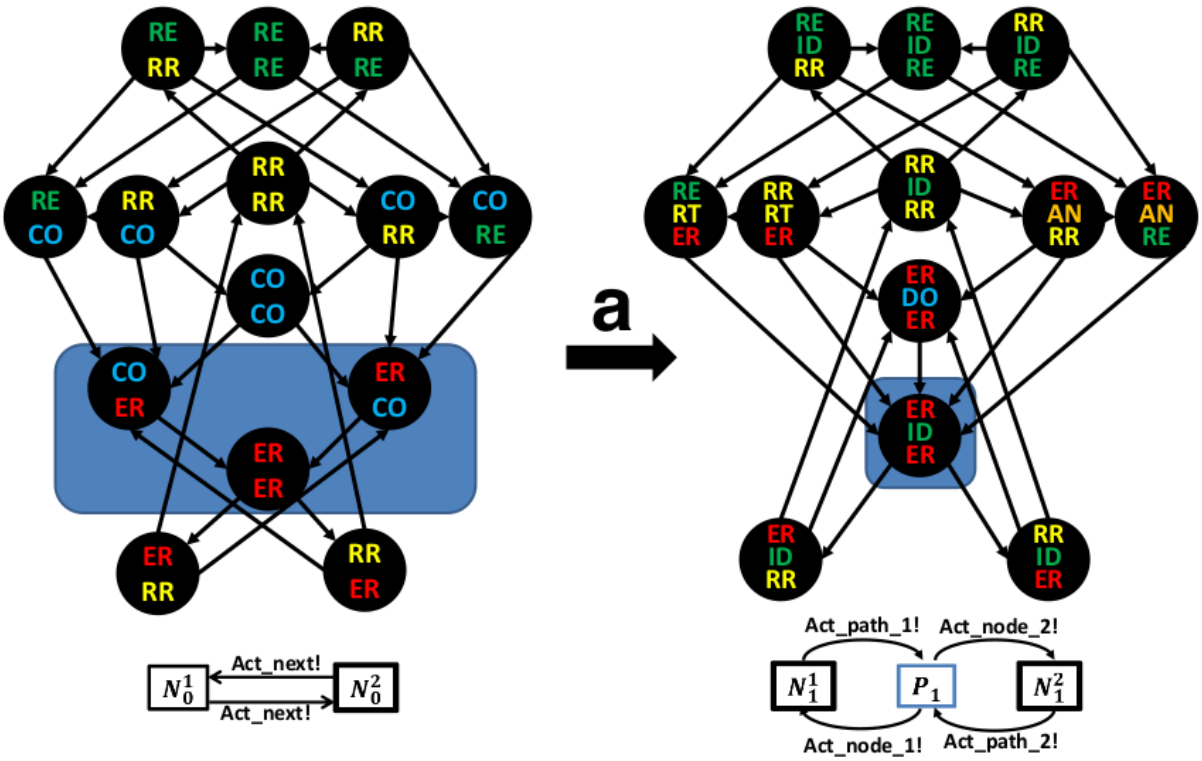


Figura 16: Redução de autômato entre dois nós (fonte: [1].)

Além da redução significativa de estados para proporcionar computação em tempo real, essa metodologia permite que se estime, atráves de experimentos clínicos, os valores temporais desses autômatos, o que seria difícil ou quase impossível considerando somente os nós. Ademais, outras abstrações foim realizadas e avaliadas, como pode ser observado em [3], mas o modelo escolhido foi esse, uma rede de condução entre nós e caminhos, por representar melhor o modelo de condução elétrica cardíaca. Nas próximas seções, será apresentado o *framework* VHM que implementa esse modelo no MATLAB/Simulink.

4 *Framework* Virtual Heart Model

Um ambiente de simulação foi desenvolvido em MATLAB e Simulink, permitindo que o usuário pudesse criar, modificar e salvar casos de teste, bem como simular sinais de eletrograma e fornecer a estimulação programada em tempo de execução. Os códigos estão disponíveis para download no endereço `http://mlab-upenn.github.io/medcps/downloads_link.html`.

A Figura 17 apresenta o modelo do coração no **simulador** VHM, que foi dividido em 33 nós e 34 caminhos, sendo as estrelas amarelas os nós e as linhas azuis os caminhos e todos eles implementando autômatos com diferentes temporizadores associados. Esses valores de tempo são ajustados pelo usuário por meio de uma matriz de valores. Essa matriz é o principal objeto para modelar qualquer tipo de comportamento de propagação do coração, isto é, conhecendo os valores de tempo para cada nó e caminho do coração para um determinado tipo de comportamento cardíaco basta atualizar a matriz com esses respectivos valores que o simulador reproduzirá o comportamento cardíaco.

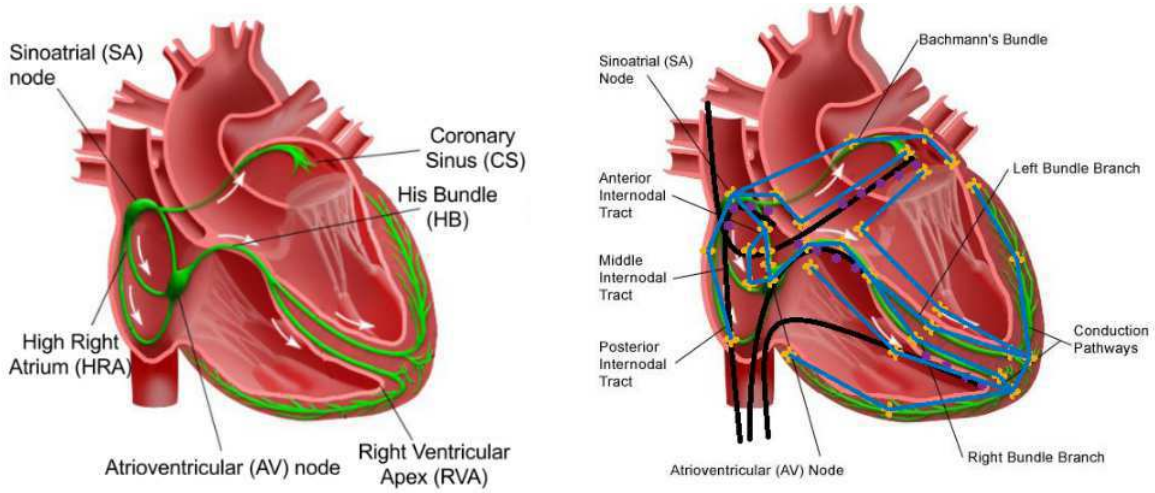


Figura 17: Divisão de nós e caminhos do modelo de coração do simulador VHM (fonte: [8]).

A plataforma gera dois tipos de sinais, o sinal de eletrograma funcional e um sinal formal (pulsos digitais). Os pulsos digitais são formados pelos autômatos de nó entre o Rest e ERP (equivalente a um pulso do relógio) e pelos autômatos de caminho na transição entre os estados Idle e Ante ou Idle e Retro. Esses eventos são monitorados pelo VHM que à partir deles geram eletrogramas sintéticos baseado em dados estocásticos dos sinais reais que tal nó ou caminho representa, sendo esse o eletrograma funcional. O pulso digital na verdade é prolongado durante o tempo em que o sinal real está acima de um limiar de tensão e esse sinal é chamado de sinal formal. A Figura 18 ilustra na coluna esquerda do desenho o sinal funcional e na coluna direita o sinal formal. No exemplo o sinal visto seria observado colocando-se dois eletrodos Uni_1 e Uni_2 em algum ramo do sistema de condução do coração.

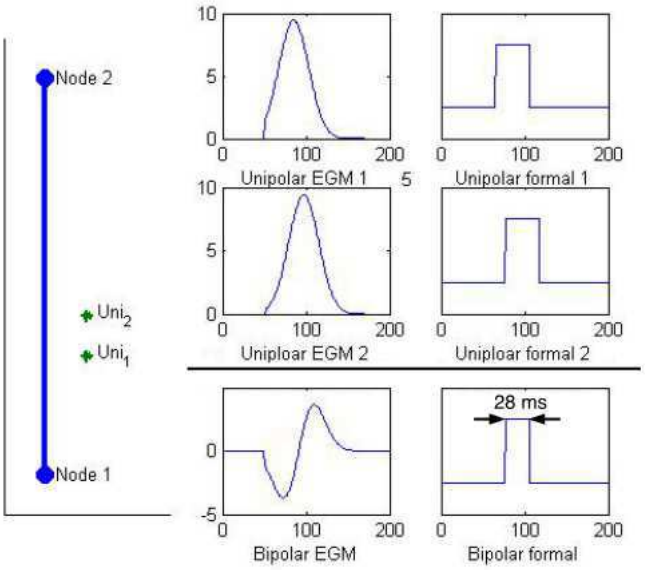


Figura 18: Exemplo de sinal elétrico gerado entre dois nós e caminho (fonte: [1]).

Segundo [1], a plataforma fornece duas interfaces, um sinal formal para software de dispositivos médicos e um eletrograma funcional para implementação de dispositivos reais (veja a Figura 19). Na verdade, a interface do eletrograma funcional não permite a conexão com dispostivivos reais, pois esses sinais "analógicos" não estão disponíveis no mundo real, apenas em simulação. Mas é possível implementar em alguma plataforma computacional (FPGA ou Microcontrolador, por exemplo) os sinais da interface interface formal (pulsos digitais) e construir uma interface A/D adicional para conectar projetos de marca-passos, como será visto posteriormente.

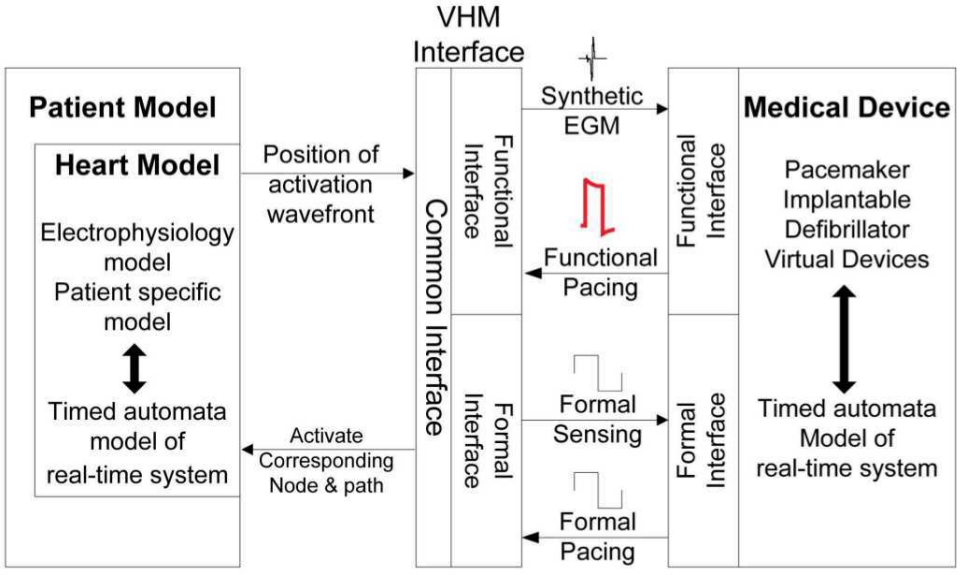


Figura 19: Estrutura da plataforma VHM (fonte: [1]).

O uso do simulador é feito executando o fonte VHM_main.m, que abrirá uma janela onde pode-se carregar algum modelo de patologia cardíaca, esse modelo é uma matriz de valores de tempo para cada autômato de nó e caminho, e é carregada usando uma variável do tipo .mat (contendo os valores) pelo ícone Load Model. O repositório do projeto oferece três patologias, ou três variável do tipo .mat, sendo elas: *Atrioventricular Nodal Reentry Tachycardia* (AVNRT), *Atrial Flutter* (AFlutter) e *Wenckebach-type A-V nodal response* (Wenckbach.mat). Carregado algum desses modelos através do Load Model, a simulação é iniciada clicando sobre o botão Run, e visualiza-se o eletrograma síntetico do caso de estudo clicando sobre o botão Show EGM. A interface do simulador VHM é apresentada na Figura 20, na tabela superior direita dessa Figura estão os autômatos de nó (veja nó SA, AV por exemplo) e logo abaixo os autômatos de caminho (SA_CT_a, CT, etc).

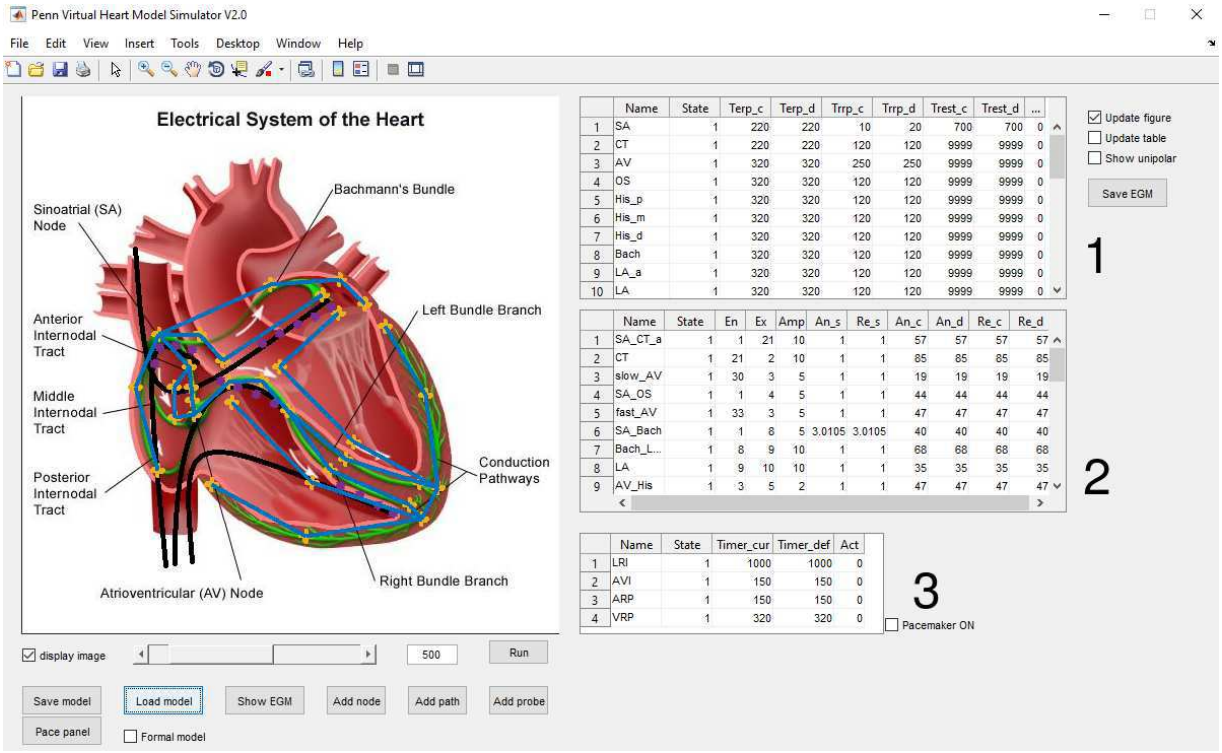


Figura 20: Tela inicial do simulador VHM

Inicialmente foi feito o estudo do simulador com o caso AVNRT, para isso carregou-se o modelo denominado EP_AVNRT. Isso foi feito para que se pudesse entender em mais detalhes a implementação real dos autômatos e seus parâmetros de tempo, para comparar teoria e implementação, como extrair/interpretar as informações de saída do simulador, o que é possível criar/derivar dele, como reproduzir tais resultados com o Simulink e como criar um hardware equivalente e reproduzir os mesmos resultados.

foi verificado em mais destalhes os ítens da tabela dos autômatos de nó e caminho da Figura 20, com isso foim criadas as Tabelas 1, descrevendo os ítens dos autômatos de nó e a Tabela 2 para os ítens dos autômatos de caminho.

Parâmetro	**Descrição**
Name	Nome do autômato de Nó.
State	Estado atual do autômato: REST (1), ERP (2) ou RRP (3) (veja Figura **??**).
Terp_c [ms]	Tempo ERP Atual (current). Essa variável decrementa o valor do tempo ERP Default até chegar a zero, quando isso aconte o autômato alcança estado RRP.
Terp_d [ms]	Tempo ERP Default. Essa variável recebe valor de tempo esperado do estado ERP, conseguido à partir de amostragem real desse sinal.
Trrp_c [ms]	Tempo RRP Atual (current). Essa variável decrementa o valor do tempo RRP Default até chegar a zero, quando isso aconte o autômato alcança estado REST.
Trrp_d [ms]	Tempo RRP Default. Essa variável recebe valor de tempo esperado do estado RRP, conseguido à partir de amostragem real desse sinal.
Trest_c [ms]	Tempo REST Atual (current). Essa variável decrementa o valor do tempo REST Default até chegar a zero, quando isso aconte o autômato alcança estado ERP (repetindo ciclo).
Trest_d [ms]	Tempo REST Default. Essa variável recebe valor de tempo esperado do estado REST, conseguido à partir de amostragem real desse sinal.
Act	Esse sinal serve para ativação de caminho conectado a ele, modelando o sistema de condução elétrica do coração.

Tabela 1: Descrição dos parâmetros do autômato de nó na versão sem modificação do Simulador VHM.

Parâmetro	**Descrição**
Name	Nome do autômato de Caminho.
State	Estado atual do autômato: Idle, Ante, Retro, Double e Conflit.
En	*Entry node index* ou índice de nó de entrada. Esse valor explicita qual nó (do 1 à 33) está conectado com extremidade de "entrada" do caminho. É constante.
Ex	*Exit node index* ou índice de saída de nó. Esse valor explicita qual nó (do 1 à 33) está conectado com extremidade de "saída" do caminho. É constante.
Amp [mV]	Fator de amplitude do sinal elétrico, DDP máxima de 10mV.
An_s [ms]	Forward Speed ou velocidade de avanço, modela velocidade da propagação do sinal elétrico.
Re_s [ms]	Backward Speed ou velocidade para trás, modela velocidade da propagação do sinal elétrico no sentido oposto ao fluxo de propagação normal. É igual ao An_s no modelo VHM.
An_c [ms]	Forward Timer Current, tempo de progagação do sinal, essa variável decrementa valor Forward Timer Default (An_d) até zero. Em zero, ocorre mudança de estado do autômato.
An_d [ms]	Forward Timer Default, tempo característico de propagação do sinal elétrico, extraído de dados clínicos.
Re_c [ms]	Backward Timer Current, tempo de progagação do sinal no sentido contrário à propagação normal, essa variável decrementa valor Backward Timer Default (Re_d) até zero. Em zero, ocorre mudança de estado do autômato.
Re_d [ms]	Backward Timer Default, tempo característico de propagação do sinal elétrico no sentido contrário à propagação normal, extraído de dados clínicos. É igual ao An_d.

Tabela 2: Descrição dos parâmetros do autômato de caminho na versão sem mod. do Simulador VHM.

Cada nó do modelo de coração recebe um nome que equivale a sua localização no coração, por exemplo, o nó SA é um autômato que modela o ponto (ou pequena região) onde localiza-se o nó Sinotrial (SA). Porém, a plataforma não ilustra onde estão localizados os demais nós e caminhos, precisando portanto de uma investigação nos códigos e nas simulações para identificar cada nó e caminho do modelo de coração representado na Figura 17, isso não foi realizado.

Ao executar a simulação, nessas condições, foi obtido como saída do ECG os sinais apresentados na Figura 21. Esse resultado pode ser interpretado, e é equivalente, ao monitoramento dos pulsos elétrico no sistema elétrico do coração em cada região dele: HRA, CS, HIS e RVA. A Figura 22 apresenta onde estão localizados no modelo do coração esses sinais.

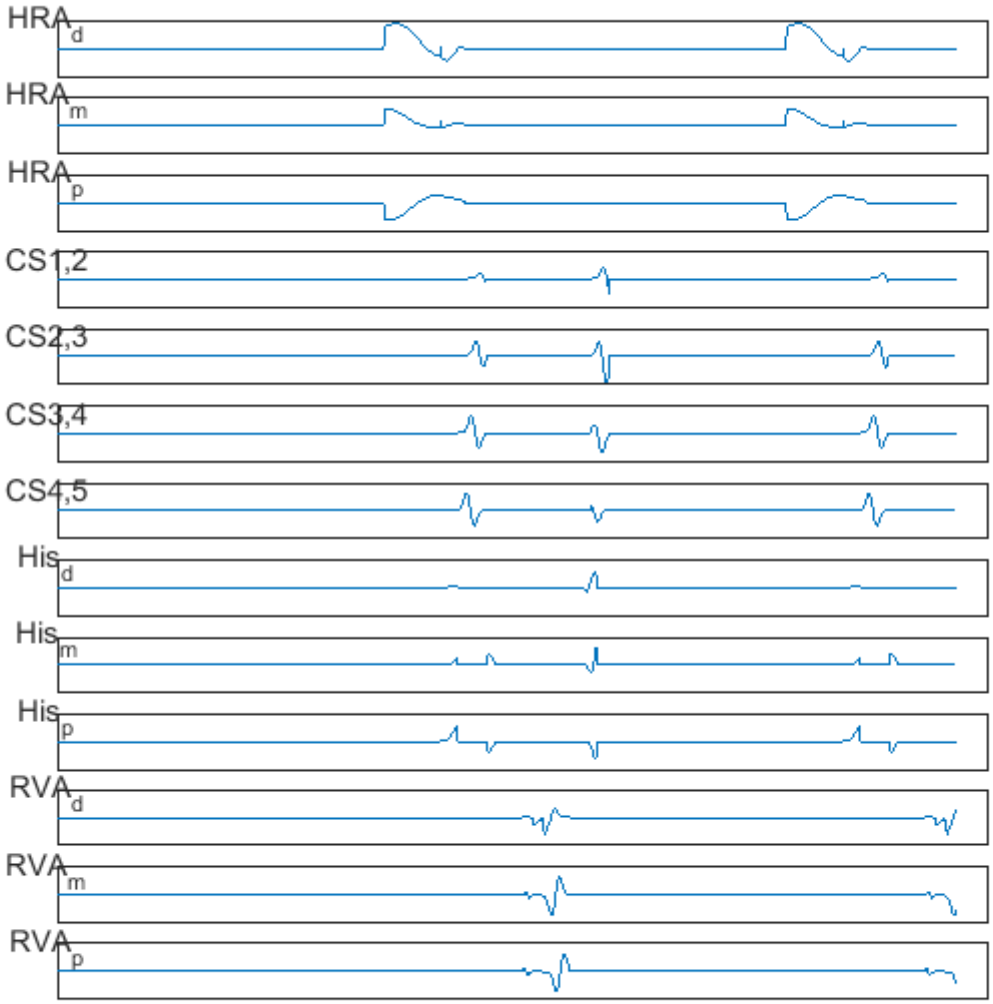


Figura 21: Amostra de simulação com versão VHM original e sem atuação de marca-passo.

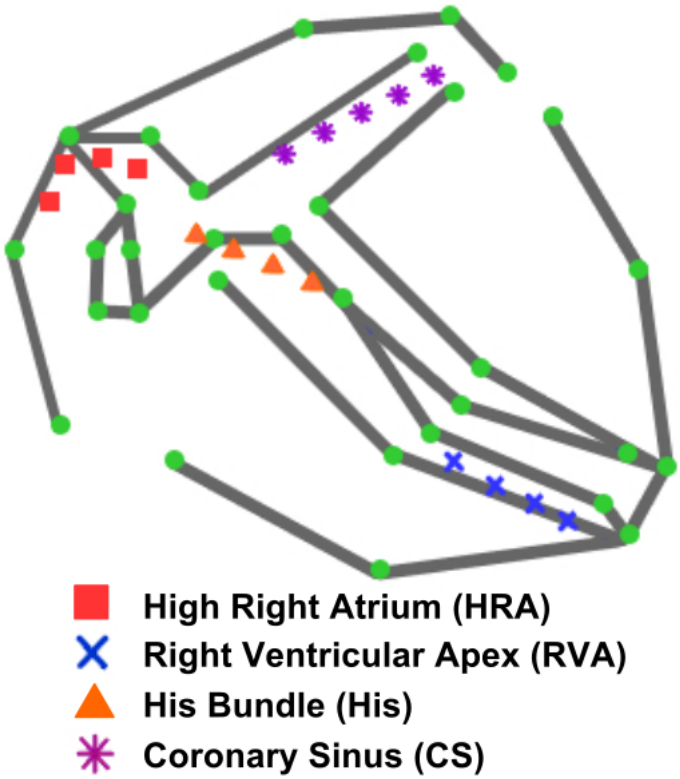


Figura 22: Amostra de simulação com versão VHM original e sem atuação de marca-passo (fonte: [1]).

Esse resultado pode ser modificado, alterando os parâmetros de tempo para modelar diferentes condições cardíacas. Além disso, é preciso investigar a localização dos autômatos no modelo para poder explorar de forma mais aprofundada o *framework*. Entretanto, escolheu-se adiar esse trabalho e reproduzir o projeto Heart-On-Chip (HoC) apresentado em [8], que ilustra como usar o simulador VHM para gerar modelos em códigos Verilog ou C do coração e implementá-los em alguma plataforma com capacidade de processa-los em tempo real, apresentados na próxima seção.

4.1 *Heart-On-Chip*: Estudo e desenvolvimento

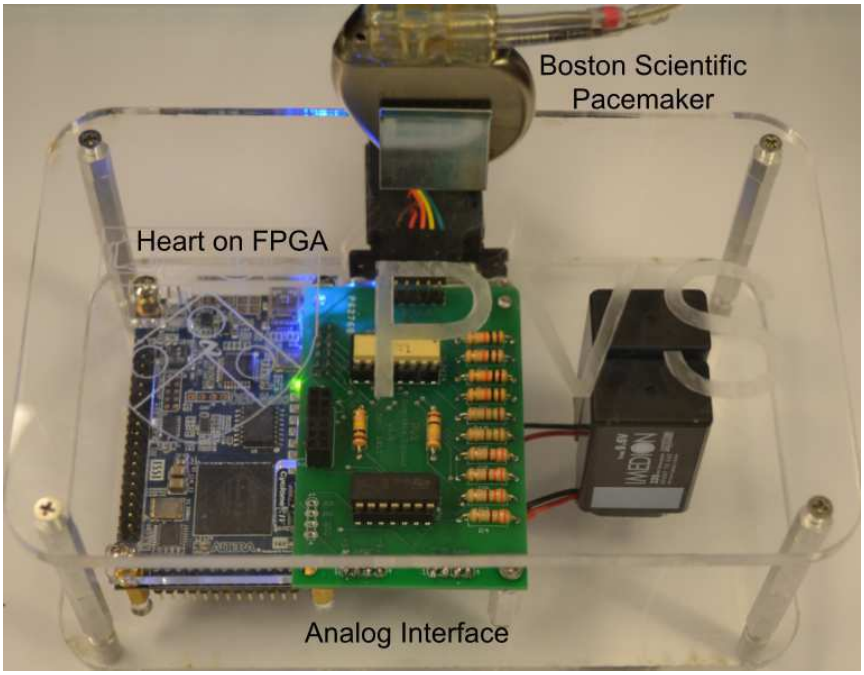


Figura 23: Plataforma HoC criada pela UPenn (fonte: [8].)

Antes de apresentar o HoC, é preciso saber que o VHM não constrói seus modelos apenas com *scripts* no MATLAB, existe também a implementação dos autômatos e o modelo VHM desenvolvidos no Simulink/**Stateflow** [8]. Isso significa que, ao integrar ao Simulink as *toolboxes* HDL Coder e Simulink Coder, que permitem, respectivamente, gerar modelos equivalentes em Verilog/VHDL e C/C++, é possível construir os modelos descritos em alto nível pelo Stateflow nessas linguagens e com isso, fazer com que esses códigos possam ser computados em plataformas como FPGAs e Microcontroladores. Então, o projeto HoC pode ser visualizado como a implementação de um modelo de coração construído no Simulink, traduzido em um modelo Verilog e computado em tempo real em uma FPGA para conectar marca-passos reais ao modelo.

O trabalho apresentado em [8] apresenta o fluxo de como usar o VHM para construir um modelo virtual de coração para ser processado em tempo real em uma plataforma de prototipação de hardware digital, sendo de interesse do presente trabalho reproduzir esses resultados para que à partir dele se possa construir uma plataforma computacional para validação e avaliação de caixa preta de marca-passos reais.

A plataforma desenvolvida pela UPenn é apresentada na Figura 23, perceba que existe uma interface analógica para condicionar os sinais digitais para analógicos e enviá-los para o marca-passo e vice-versa, condicionar os sinais analógicos para digital e enviá-los para FPGA, permitindo portanto a conexão de uma marca-passo real.

[8]Stateflow é uma ferramenta lógica de controle usada pada modelar **sistemas reativos** com autômatos

Na repositório oficial [9] encontra-se o projeto VHM Simulink Code, nele estão contidos todos os fontes que implementam o VHM no Simulink. Basicamente, tem-se os autômatos de nó e caminho construídos com o Stateflow, ilustrados pelas Figuras 24 e 25, que são encapsulados, replicados com tempos distintos e conectados para formar o modelo do coração. Os parâmetros de tempo desses autômatos, são carregados também com uma matriz de valores que implementam diversos comportamentos do coração. Toda essa malha é encapsulada e pode-se gerar o modelo equivalente em Verilog, apresentado adiante.

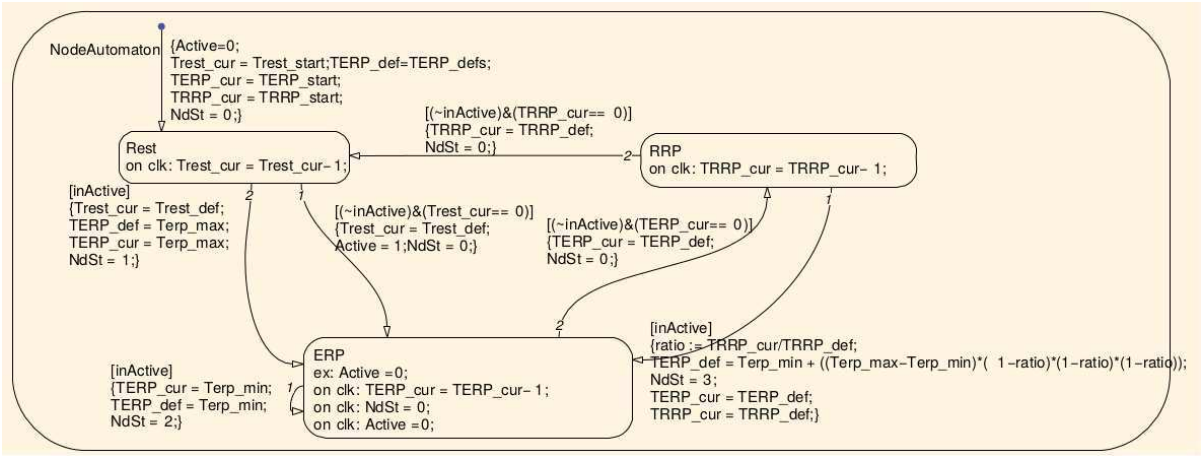


Figura 24: Autômato de nó desenvolvido no Simulink (fonte: [1].)

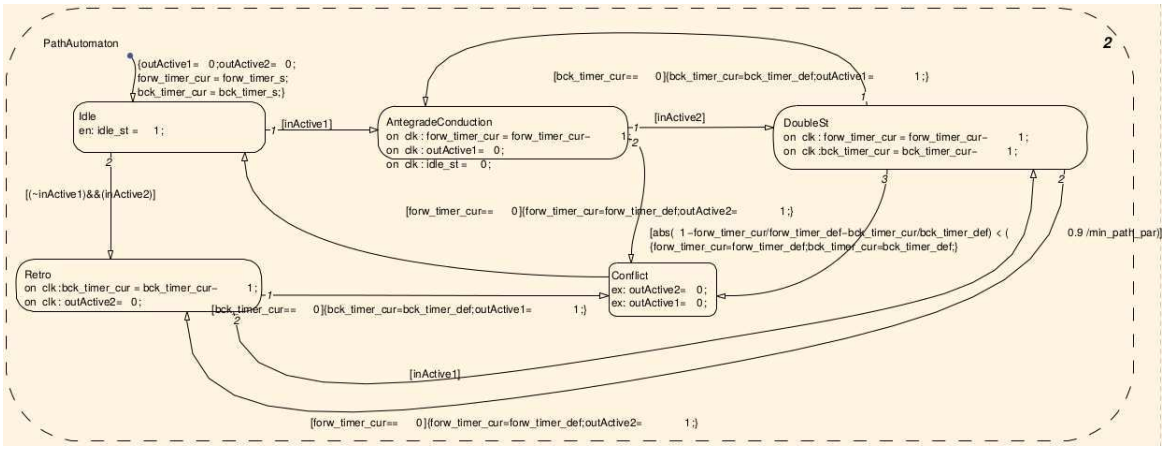


Figura 25: Autômato de caminho desenvolvido no Simulink (fonte: [1].)

[9]Repositório oficial: `http://mlab-upenn.github.io/medcps/downloads_link.html`

O repositório conta com um exemplo de implementação de modelo usando esses autômatos, porém, diferentemente da implementação no MATLAB que usa 33 autômatos de nó e 34 autômatos de caminho, o modelo exemplo usa apenas 7 nós e 7 caminhos, como apresentado na Figura 26. Não foi investigado o motivo de tal simplificação, devendo ser feito posteriormente.

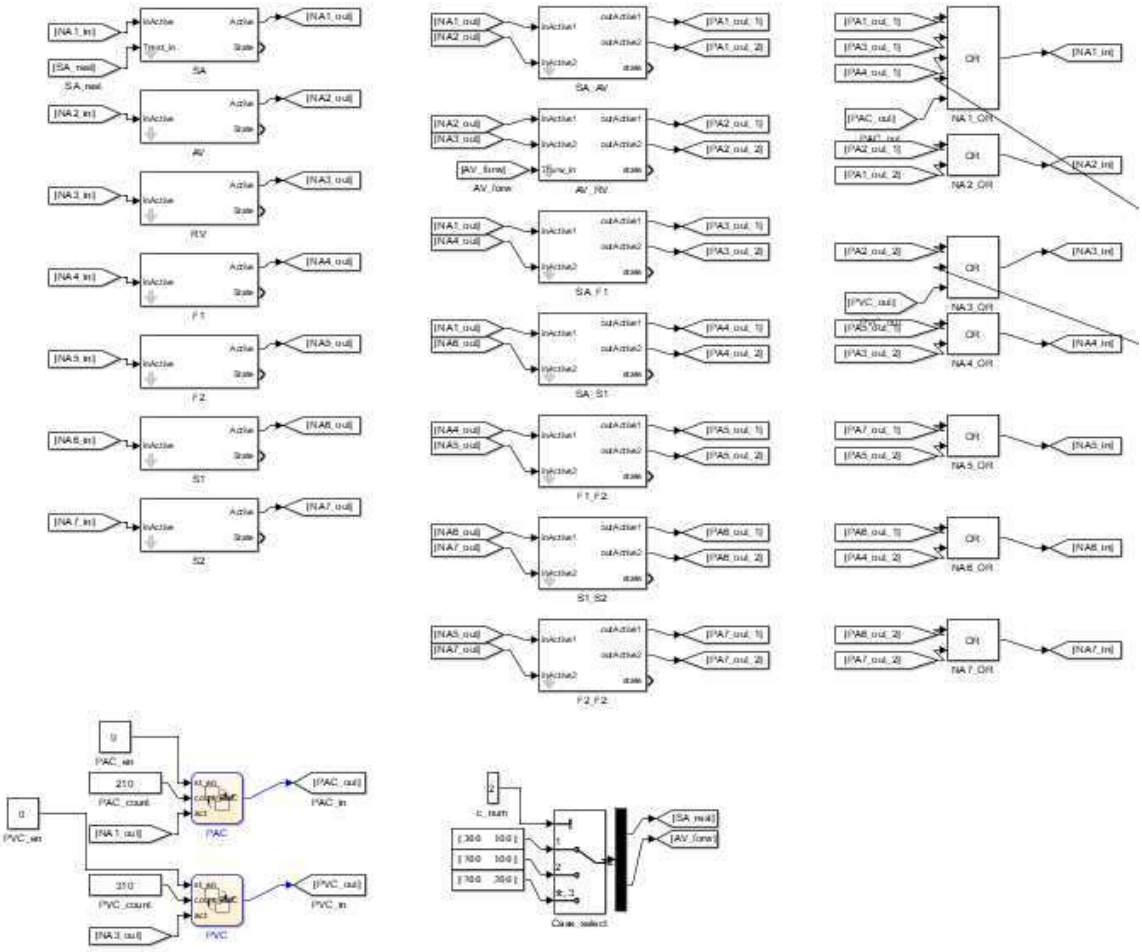


Figura 26: Replicação e conexão de vários autômatos de nó e caminho no Simulink.

O modelo construído possuí, além do comportamento normal do coração as opções de modificar os parâmetros de tempo dos autômatos para modificar o comportamento do coração em tempo de execução. As patologias cardíacas disponíveis são a bradicardia e taquicardia (veja Figura 27), além dessas é possível enviar pulsos pela interface externa para os átrios e/ou ventrículos graças aos autômatos PAC e PVC, como pode ser observado no canto inferior esquerdo da Figura 26.

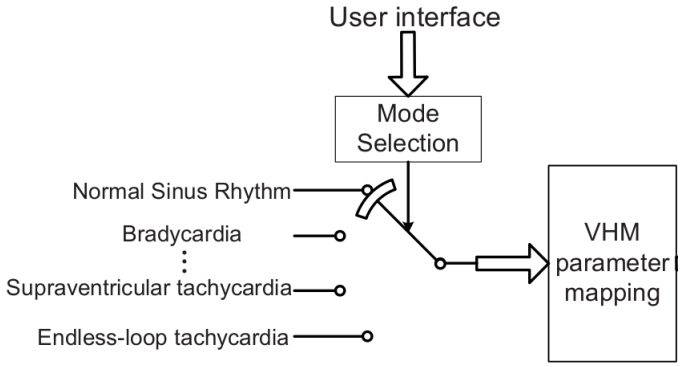


Figura 27: Opções de patologias cardíacas (fonte: [2] modificado).

Esse modelo foi encapsulado em um única unidade e gerado o seu modelo Verilog equivalente usando a ferramenta HDL Coder do Simulink. Os fontes Verilog foim gerados e simulados no ModelSim, criando um simples testbench para estimular o circuito e verificar seu comportamento e comparar com o modelo de referência do Simulink. Em resumo, o teste apenas executa o modelo sem patologias e posteriormente envia sinais de PAC e PVC.

Esse teste gerou os *waveforms* apresentados nas Figuras 28 e 29. No primeiro caso, pode-se observar a propagação dos sinais do nó, que se inicia pelo NA1Out (ou nó SA) que é um comportamento esperado, também vericou-se a propagação desse sinal para as demais unidades (NA1Out, NA2Out, etc) e verificou-se as transições nos estados dos autômatos, em todos os casos apresentando um comportamento de acordo com o modelo de referência.

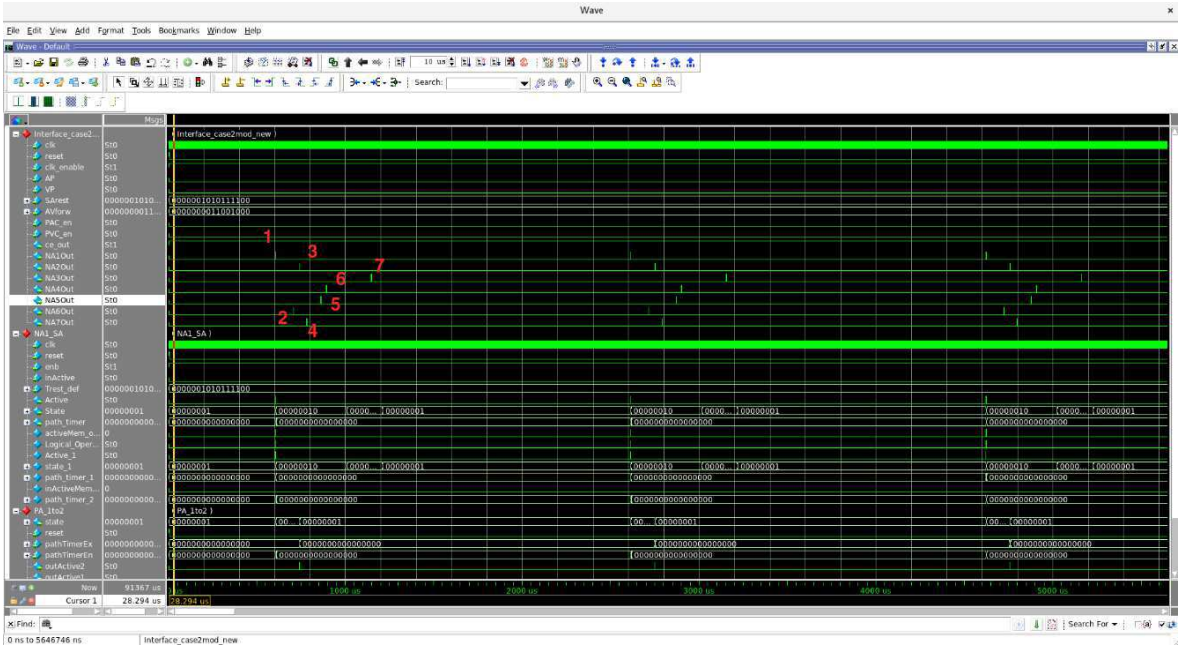


Figura 28: Simulação de modelo gerado pelo Simulink no ModelSim em condições normais.

Em seguida foi avaliado o comportamento do modelo com estímulos PAC e o resultado foi satisfatório como pode ser observado na Figura 29. Pode-se verificar que no ciclo seguinte do batimento cardíaco após a ativação do sinal PAC_en o modelo executa o ciclo compĺeto de batimento em um intervalo de tempo curto, indicando uma contração atrial prematura.

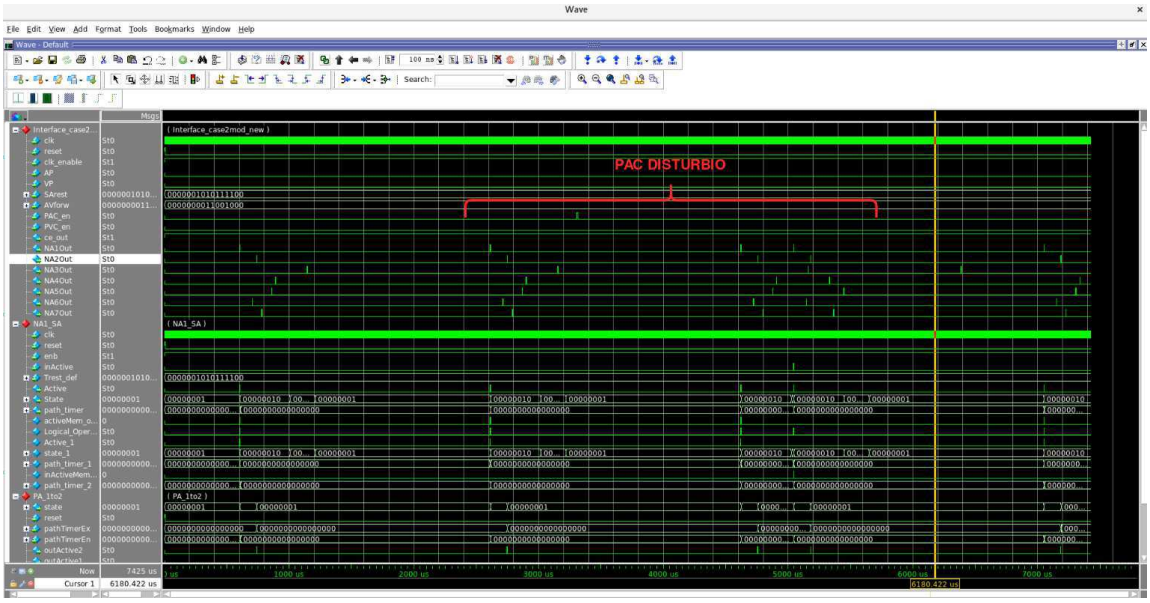


Figura 29: Simulação de modelo gerado pelo Simulink no ModelSim com PAC.

A simulação do modelo gerado demonstrou-se coerente com o modelo de referência, porém, alguns pontos devem ser investigados com mais cautela, como: a localização de cada autômato no modelo do coração, pois sabe-se apenas que o NA1Out implementa o nó SA e o NA3Out implementa o nó AV, não conhecendo quais partes do coração os demais autômatos modelam; também é preciso verificar como construir modelos alternativos sejam eles mais ou menos complexos e verificar qual o impacto essa escolha causa na modelagem do coração e consequentemente na avaliação de marca-passos. Entretanto, preferiu-se seguir o fluxo de desenvolvimento para tentar reproduzir o projeto HoC.

Com o modelo Verilog construído, é possível sintetizá-lo em uma plataforma FPGA e com isso conectar marca-passos reais a ela por meio de uma interface, para avaliar o comportamento do marca-passo real e coração virtual, reprozindo o HoC. Para isso, primeiramente foi verificado se o código gerado pelo Simulink seria sintetizável em FPGA, e isso foi feito sintetizando o circuito para a FPGA Cyclone V com a ferramenta Quartus 17.1 Lite Edition da Intel. Obtendo sucesso, como pode ser observado na Figura 30 e 31.

O próximo passo seria construir uma infraestrutura adicional de hardware para proporcionar a conexão entre marca-passo e FPGA. Felizmente, não foi preciso desenvolver essa infraestrutura, pois um projeto chamado Pacemaker Verificar System (PVS) já o fez. Esse projeto é a parte final para a construção do HoC e será discutida na próxima seção.

Figura 30: Log de síntese gerado pelo Quartus do projeto exemplo do VHM Simulink.

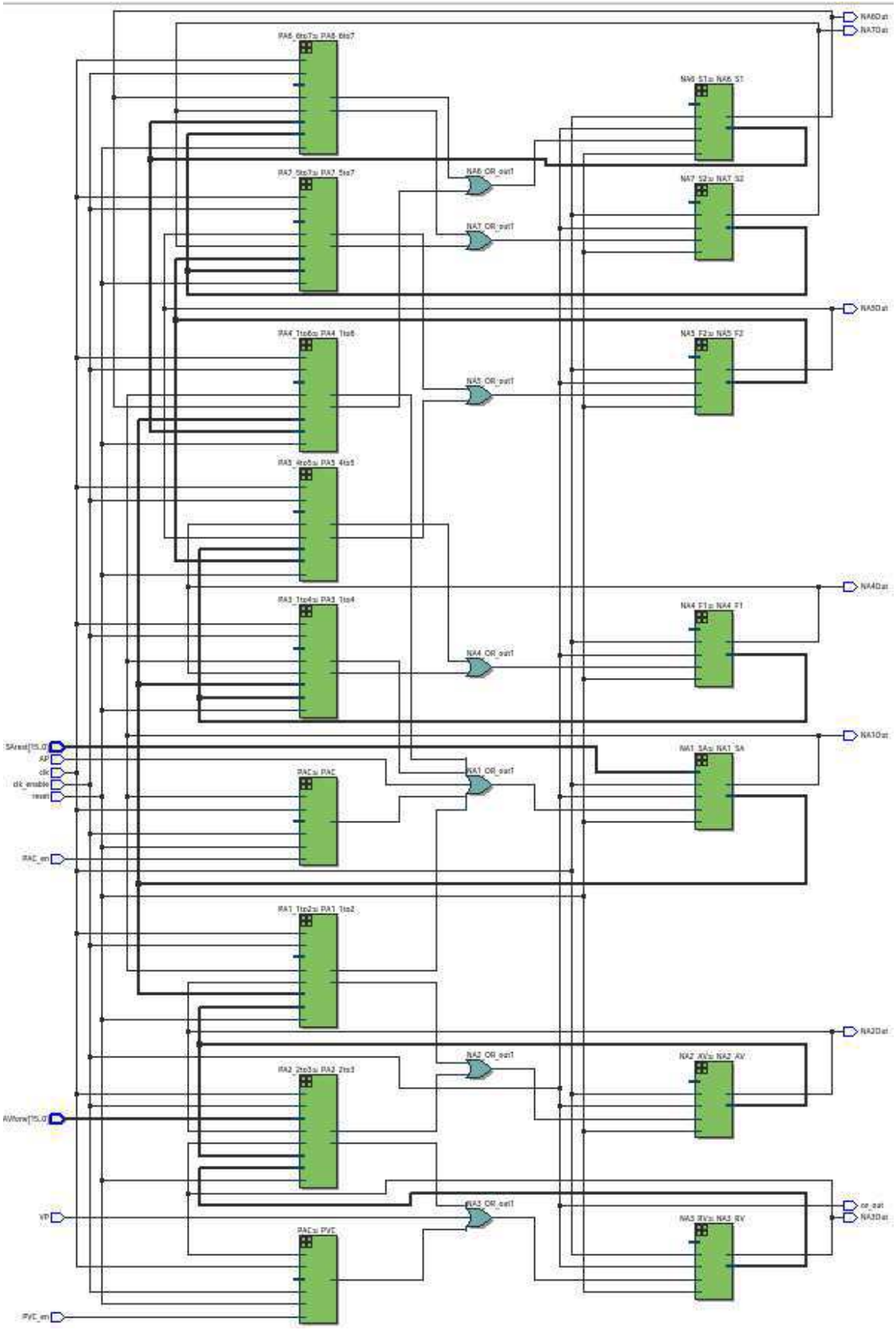


Figura 31: RTL gerado pelo Quartus do projeto exemplo do VHM Simulink.

5 PVS: Pacemaker Verification System

O PVS é um projeto aberto que serve para complementar o *framework* VHM Simulink. Ele cria uma infraestrutura de hardware sobre o projeto desenvolvido com VHM Simulink, provendo comunicação serial entre *host* e FPGA, permitindo alterar os parâmetros dos autômatos processados na FPGA em tempo de execução e aquisição de dados do modelo em hardware para serem apresentados em uma interface gráfica e permitir uma melhor interpretação da execução, entre outras opções, que serão discutidas ao decorrer da seção. O projeto está disponível para download no link `https://github.com/mlab-upenn/pvs`.

5.1 Visão geral do PVS

O projeto contém os diretórios apresentados na Figura 32. O diretório GUI, contém os scripts MATLAB que implementam a interface do PVS e o driver da serial. Ao executar o código PVS_GUI.m no MATLAB, a interface apresentada na Figura 34 é exibida, essa interface se comunica com a FPGA via FPGA e pode ser usada para modificar o modelo do coração, enviar sinais PAC ou PVC e visualizar o sinal sintético do ECG gerado pelo modelo executado na FPGA.

Figura 32: Diretórios do repositório PVS.

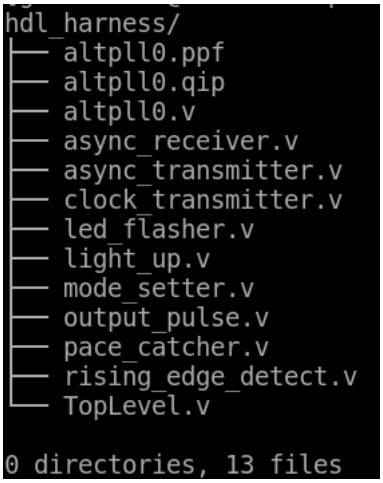


Figura 33: Fontes Verilog do projeto PVS.

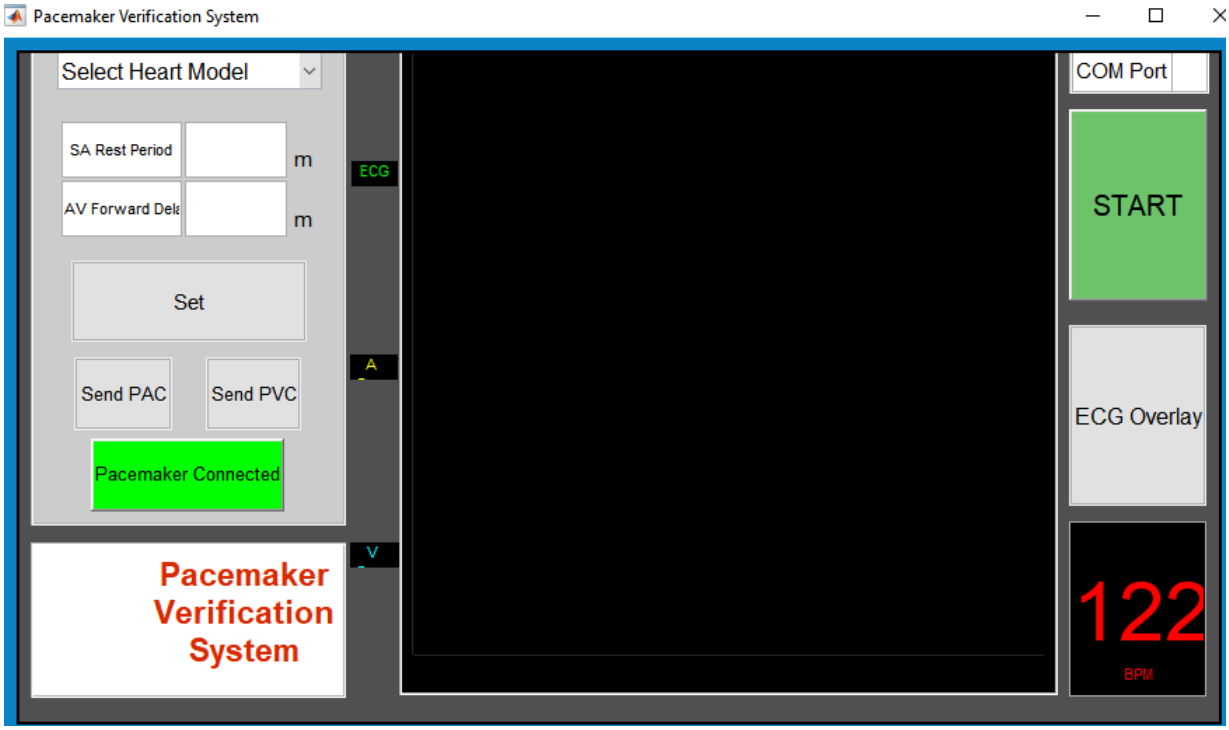


Figura 34: Interface PVS.

O diretório SimpleModels e LookupTable contém alguns modelos contruídos usando o *framework* VHM Simulink (estudado na Seção 4). O repositório hdl_harness contém os fontes Verilog para encapsular os modelos de Verilog gerados pelo VHM Simulink. Grosso modo, o TopLevel.v faz a conexão de um modelo de coração gerado pelo VHM Simulink e os demais componentes presentes no repositório, sendo eles, os seguintes: a serial que é implementada nos códigos async_receiver.v, async_transmitter.v, clock_transmitter.v e model_setter,v; o fonte output_pulse.v que recebe os pulsos dos nós do coração e os mantém por alguns milésimos de segundo; o altpll.v implementa um relógio de 1.5kHz que é a base de frequêcia usada para os autômatos computar os seus atrasos. Enfim, os outros códigos, são usados para acionar LEDs e alguns pinos externos da FPGA DE0-Nano da Intel, com sinais do modelo do coração de interesse, como por exemplo os sinais de ativação dos nós (NA1Out, NA2Out, etc).

O mesmo modelo do coração contruído na Seção 4, com o *framework* VHM Simulink, foi usado com o PVS. Esse modelo é chamado de case2mod_new e nas próximas subseções serão apresentadas como construir e integrar esse modelo no repositório PVS, permitindo inclusive uma melhor compreensão desse projeto.

5.2 Desenvolvimento do PVS em FPGA

5.2.1 Geração do modelo Verilog

O modelo de coração usado no desenvolvimento é similar ao estudado na Seção 4 e ele está reimplementado no repositório PVS no fonte SimpleModels/case2mod_new.mdl. É um modelo reduzido, possuindo apenas 7 nós e 7 caminhos, diferente do modelo do MATLAB com 33 autômatos de nó e 34 de caminho. O case2mode_new.mdl é compatível com o módulo topo (hdl_harness/TopLevel.v) do projeto PVS e por isso foi usado, porém, pode-se alterar esse modelo para um maior, precisando ajustar todos os fontes hdl_harness.

A geração do HDL foi feita no Simulink usando a ferramenta HDL Coder, da seguinte forma: 1) abriu-se o arquivo case2mod_new.mdl; 2) na aba **Code** da janela aberta escolheu-se o ítem HDL Workflow Advisor; 3) Na janela aberta, executou-se **Run All**, do item 1 ao item 3, e por fim Generate HDL. O resultado disso foi o modelo do coração, similar ao desenvolvido na Seção 4, e pronto para se conectar com o resto da infraestrutura em hdl_harness.

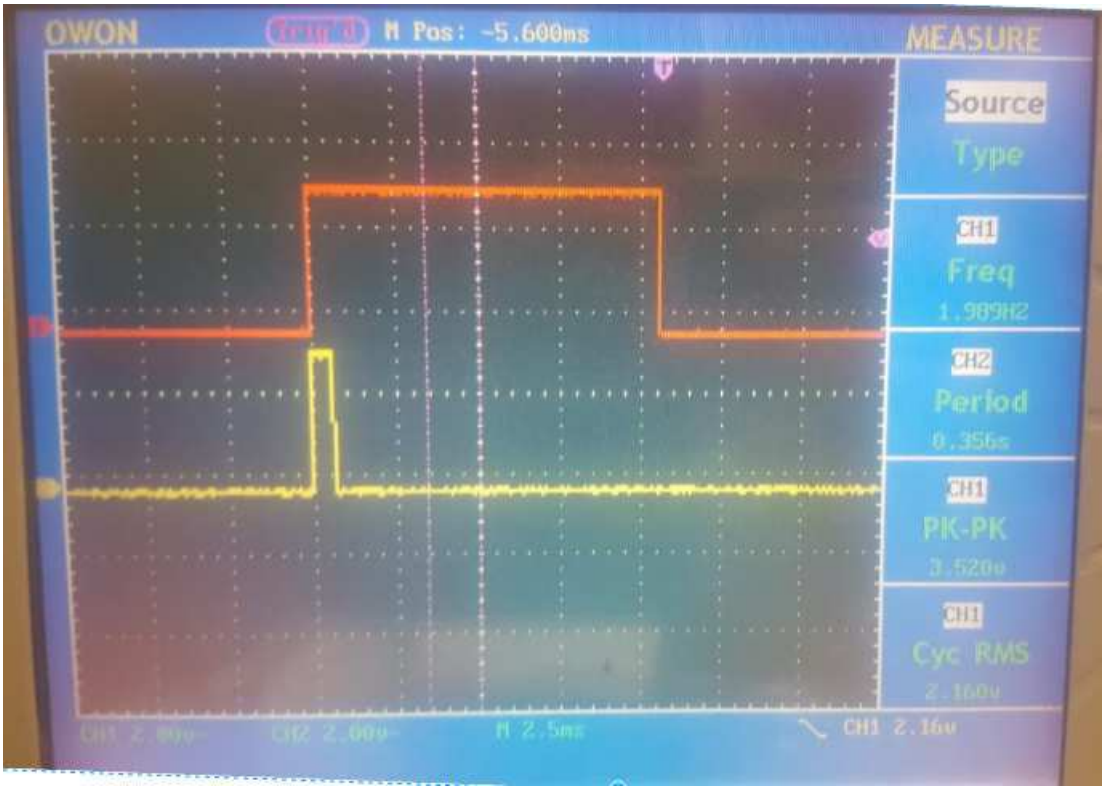


Figura 35: Comparação de um pulso de nó e o mesmo prolongado pelo PVS.

5.2.2 Interface do circuito

O modelo Verilog do coração gerado na Seção precedente possuí uma interface externa e seus sinais são apresentados e descritos na Tabela 3. Perceba que alguns sinais de entrada apresentados na Tabela, estão presentes na interface do PVS, como: SArest, AVforw, PAC_en e PVC_en. Na verdade, quando estabelecida a conexão entre FPGA e *host* via serial, esses valores podem ser alterados. A serial na FPGA recebe os valores e atualiza os parâmetros desejados em tempo de execução. Os outros sinais como AP e VP são pinos para conexão de um marca-passo real e os nós (Nó X) são os sinais que os marca-passos devem ler. Esses sinais, nó modelo orignal geram pulsos digitais em um ciclo de relógio, mas o PVS prolonga esses pulsos de nó por 10 milésimos de segundos (10ms), como pode ser visualizado na Figura 40, não se sabe ainda o motivo da escolha de 10ms. Esse atraso é causado pelo módulo output_pulse.v do PVS.

Entradas	**Descrição**
AP	Atrial Pacing ou Estímulo Auricular, ou seja, sinal vindo de possível marca-passo para estimular átrios. Em outras palavras, quando esse sinal está em nível alto ele ativa o nó SA (NA1_SA).
VP	Ventriculo Pacing ou Estímulo Ventricular, ou seja, sinal vindo de possível marca-passo para estimular ventrículo. Em outras palavras, quando esse sinal está em nível alto ele ativa o nó RV (NA3_RV).
SArest	Tempo de descansto de SA, pode ser controlado pela hierarquia superior. No caso da demo o TopLevel.v em hdl_harness faz o controle desse parametro.
AVforw	Tempo de propagação do caminho, pode ser controlado pela hierarquia superior. No caso da demo o TopLevel.v em hdl_harness faz o controle desse parametro.
PAC_en	Habilita autômato PAC, modelando efeito da doença PAC.
PVC_en	Habilita autômato PVC, modelando efeito da doença PVC.
Saídas	
Nó X	Pequena região X do coração monitorada. X pode ser SA, AV, RV, etc. No exemplo em questão o nó varia do 1 ao 7.

Tabela 3: Descrição dos sinais do modelo de coração (demo/case2mod_new) no Simulink.

5.2.3 Simulação do modelo gerado com hdl_harness

Um script fora feito no ModelSim para simular o sistema hdl_harness junto ao modelo do coração construído. Foi conseguido simular, sem erros de compilação, porém para avaliar as funcionalidades desse projeto seria preciso construir um testbench que simule o comportamento da serial e dos demais componentes do PVS e conseguir avaliar o comportamento do circuito. Devido ao tempo, foi escolhido realizar primeiramente a síntese do circuito (hdl_harness integrado com modelo do coração), deixando o desenvolvimento do testbench para ser feito em um trabalho anterior.

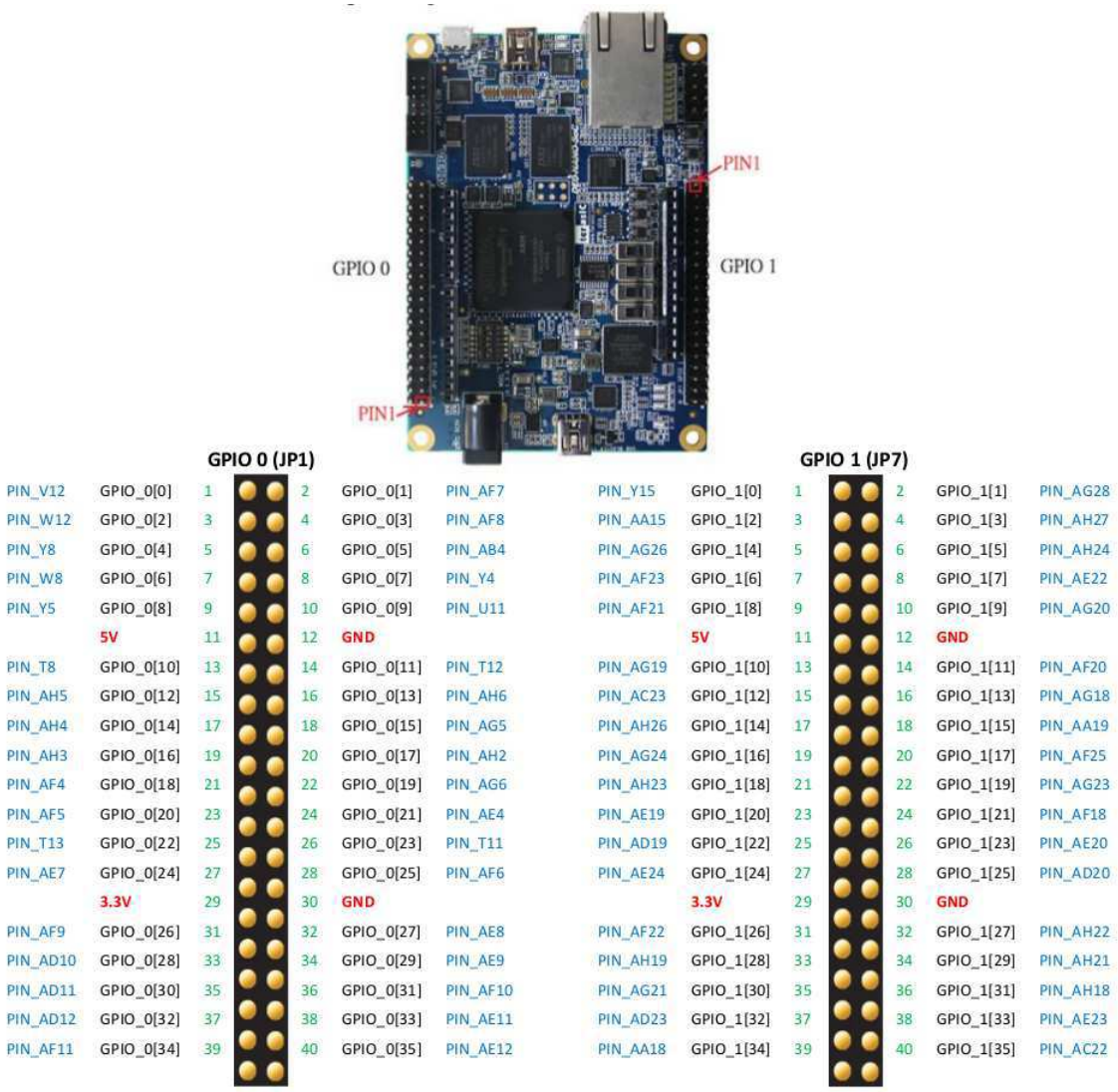
5.2.4 Síntese FPGA

O projeto PVS foi construído originalmente na plataforma DE0-Nano que usa a FPGA Cyclone IV. O projeto usa o IP Altera PLL para gerar um relógio de baixa frequencia (1.5kHz). Porém, para o desenvolvimento em questão foi usado o kit DE0-Nano-SoC com FPGA Cyclone V, e essa FPGA não possuí o IP Altera PLL de baixa frequência, abaixo de 1MHz. Devido a isso, primeiramente criou-se um divisor de frequência em Verilog para gerar um relógio de 1.5kHz e substituir o IP PLL. Não foi encontrado nem descoberto nenhum argumento para se fazer a escolha da frequência do sistema com esse valor.

Com isso, foi feita a síntese do sistema, isto é, dos códigos hdl_harness, do modelo de coração e do divisor de frequência desenvolvido. Além disso, modificou-se o fonte TopLevel.v, adaptando-o sua interface para os pinos do kit de desenvolvimento FPGA DE0-Nano-SoC Altera. O CLOCK_50 foi substituido por FPGA_CLK2_50, o PIN foi substituido por GPIO_0, o PIN_IN foi substituido por SW (switches) e os LEDs mantiveram-se os mesmos. O GPIO_0 contém os pinos de serial e de saída dos nós. A conexão desses sinais com os pinos do kit FPGA DE0-Nano-SoC foi feita usando como referência a Figura 36.

O circuito foi sintetizado no Quartus 17.1 Lite Edition com sucesso, gerando como saída o RTL apresentado na Figura 37 e o log apresentado na Figura 38. Comparando o log da Figura 38 com o da Figura 30 é possível perceber um incremento nos recursos da FPGA, sendo um resultado esperado tendo em vista que o circuito foi aumentado.

Em seguida, foi embutido o bitstream gerado pelo Quartus no kit FPGA usado. Na próxima seção é apresetando em mais detalhes o componente serial do PVS e os testes realizados com essa interface para validar o seu funcionamente e seguir para etapa de conexão da plataforma criada no host.



PIN_V12	GPIO_0[0]	1	2	GPIO_0[1]	PIN_AF7		PIN_Y15	GPIO_1[0]	1	2	GPIO_1[1]	PIN_AG28		
PIN_W12	GPIO_0[2]	3	4	GPIO_0[3]	PIN_AF8		PIN_AA15	GPIO_1[2]	3	4	GPIO_1[3]	PIN_AH27		
PIN_Y8	GPIO_0[4]	5	6	GPIO_0[5]	PIN_AB4		PIN_AG26	GPIO_1[4]	5	6	GPIO_1[5]	PIN_AH24		
PIN_W8	GPIO_0[6]	7	8	GPIO_0[7]	PIN_Y4		PIN_AF23	GPIO_1[6]	7	8	GPIO_1[7]	PIN_AE22		
PIN_Y5	GPIO_0[8]	9	10	GPIO_0[9]	PIN_U11		PIN_AF21	GPIO_1[8]	9	10	GPIO_1[9]	PIN_AG20		
	5V	11	12	GND				5V	11	12	GND			
PIN_T8	GPIO_0[10]	13	14	GPIO_0[11]	PIN_T12		PIN_AG19	GPIO_1[10]	13	14	GPIO_1[11]	PIN_AF20		
PIN_AH5	GPIO_0[12]	15	16	GPIO_0[13]	PIN_AH6		PIN_AC23	GPIO_1[12]	15	16	GPIO_1[13]	PIN_AG18		
PIN_AH4	GPIO_0[14]	17	18	GPIO_0[15]	PIN_AG5		PIN_AH26	GPIO_1[14]	17	18	GPIO_1[15]	PIN_AA19		
PIN_AH3	GPIO_0[16]	19	20	GPIO_0[17]	PIN_AH2		PIN_AG24	GPIO_1[16]	19	20	GPIO_1[17]	PIN_AF25		
PIN_AF4	GPIO_0[18]	21	22	GPIO_0[19]	PIN_AG6		PIN_AH23	GPIO_1[18]	21	22	GPIO_1[19]	PIN_AG23		
PIN_AF5	GPIO_0[20]	23	24	GPIO_0[21]	PIN_AE4		PIN_AE19	GPIO_1[20]	23	24	GPIO_1[21]	PIN_AF18		
PIN_T13	GPIO_0[22]	25	26	GPIO_0[23]	PIN_T11		PIN_AD19	GPIO_1[22]	25	26	GPIO_1[23]	PIN_AE20		
PIN_AE7	GPIO_0[24]	27	28	GPIO_0[25]	PIN_AF6		PIN_AE24	GPIO_1[24]	27	28	GPIO_1[25]	PIN_AD20		
	3.3V	29	30	GND				3.3V	29	30	GND			
PIN_AF9	GPIO_0[26]	31	32	GPIO_0[27]	PIN_AE8		PIN_AF22	GPIO_1[26]	31	32	GPIO_1[27]	PIN_AH22		
PIN_AD10	GPIO_0[28]	33	34	GPIO_0[29]	PIN_AE9		PIN_AH19	GPIO_1[28]	33	34	GPIO_1[29]	PIN_AH21		
PIN_AD11	GPIO_0[30]	35	36	GPIO_0[31]	PIN_AF10		PIN_AG21	GPIO_1[30]	35	36	GPIO_1[31]	PIN_AH18		
PIN_AD12	GPIO_0[32]	37	38	GPIO_0[33]	PIN_AE11		PIN_AD23	GPIO_1[32]	37	38	GPIO_1[33]	PIN_AE23		
PIN_AF11	GPIO_0[34]	39	40	GPIO_0[35]	PIN_AE12		PIN_AA18	GPIO_1[34]	39	40	GPIO_1[35]	PIN_AC22		

Figura 36: Arranjo de pinos do kit de desenvolvimento DE0-Nano-SoC Altera.

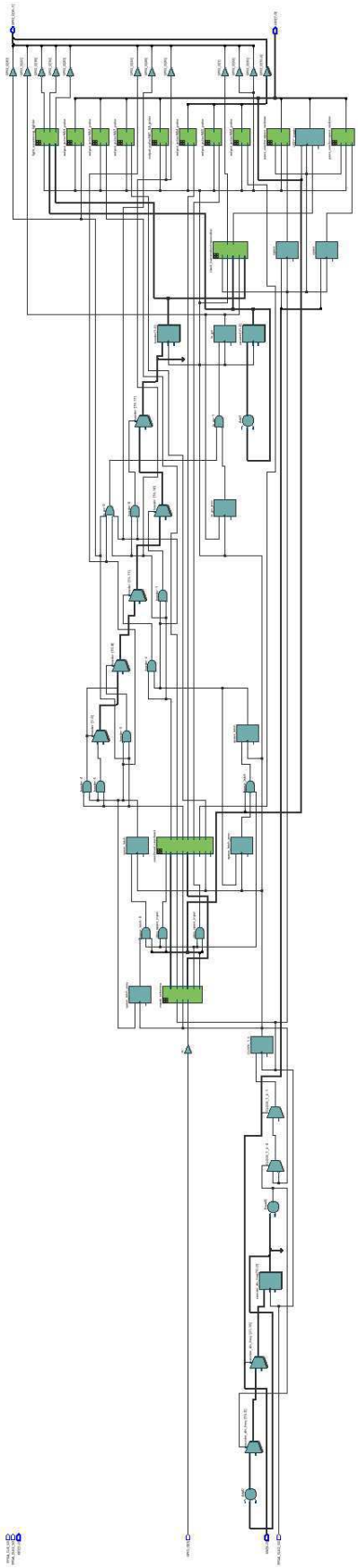


Figura 37: RTL gerado pelo Quartus como resultado de síntese hdl_harness.

Figura 38: Log gerado pelo Quartus como resultado de síntese hdl_harness.

5.2.5 Interface serial e testes

Verificando os códigos da serial do projeto PVS, fontes async-trasmitter.v e async_ receiver.v, descobriu-se que a serial implementa o protocolo RS-232 com baudrate 115200, 8 bits, sem paridade, 1 bit de stop e sem controle de fluxo.

Como foi discutido anteriormente, a interface serial serve para configurar parâmetros do sistema em tempo de execução, isto é, os valores enviados pelo host para mudar os parâmetros do modelo do coração, para enviar PAC, PVC, etc, é possível graças a serial. Já os dados que a FPGA envia para o host são construídos respeitando um protocolo [5], o primeiro byte enviado representa um cabeçalho definindo alguns estados para ser interpretado pelo MATLAB, apresentados na Tabela **??** (veja página 50 do [5], para mais informações).

Valor de byte	**Descrição**
0	Valor ruim.
1	Ativação nó SA sem APace;.
2	Sem ativação nó SA e ativação APace.
3	Ativação nó SA e APace.
4	Ativação nó RV sem VPace.
5	Sem ativação nó RV e ativação de VPace.
6	Ativação nó RV e VPace.

Tabela 4: Descrição dos sinais de cabeçalho enviados pela serial.

Esses valores são lidos pelo host e servem como base para apresentação do processamento da FPGA na interface gráfica PVS, da seguinte forma, após o primeiro byte, são enviados 4 bytes (sem sinal, little-endian) que contém um valor de tempo, então o cabeçalho informa a que evento esse valor do contador está sendo transmitido, veja página 24 do [5] para mais detalhes, e com isso a interface PVS consegue gerar um sinal sintético de ECG, que será visto posteriormente.

Dando continuidade, foram conectados aos pinos da FPGA que recebem os valores Tx e Rx da serial um cabo FTDI, idêntico ao da Figura 39. Da seguinte forma:

- Vdd do cabo FTDI conectado ao Vdd (5V) da FPGA;
- GND do cabo FTDI conectado ao GND da FPGA;
- Sinal Rx do cabo FTDI (fio branco) conectado com GPIO_0[1], Tx da FPGA;
- Sinal Tx do cabo FTDI (fio verde) conectado com GPIO_0[0], Rx da FPGA.

Figura 39: Cabo FTDI usado em experimento.

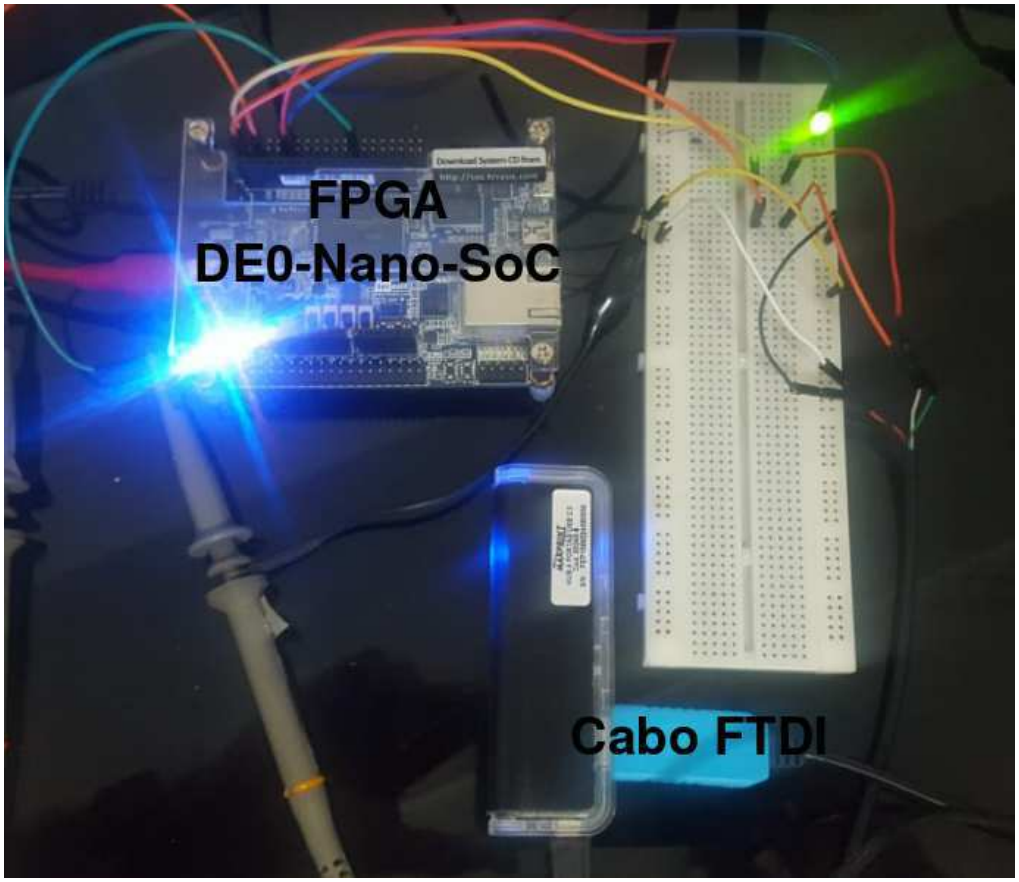


Figura 40: Circuito usado FPGA, Cabo FTDI USB para Serial e protoboard.

Com a FPGA processando o projeto, foi verificado em um osciloscópio os pinos quem implementam as funções Tx e Rx da serial. Um dos *frames* é apresentado na Figura 41 e com ele conclui-se que o sinal de relógio está sendo gerado de forma correta e que existe um sinal Tx estão sendo transmitido.

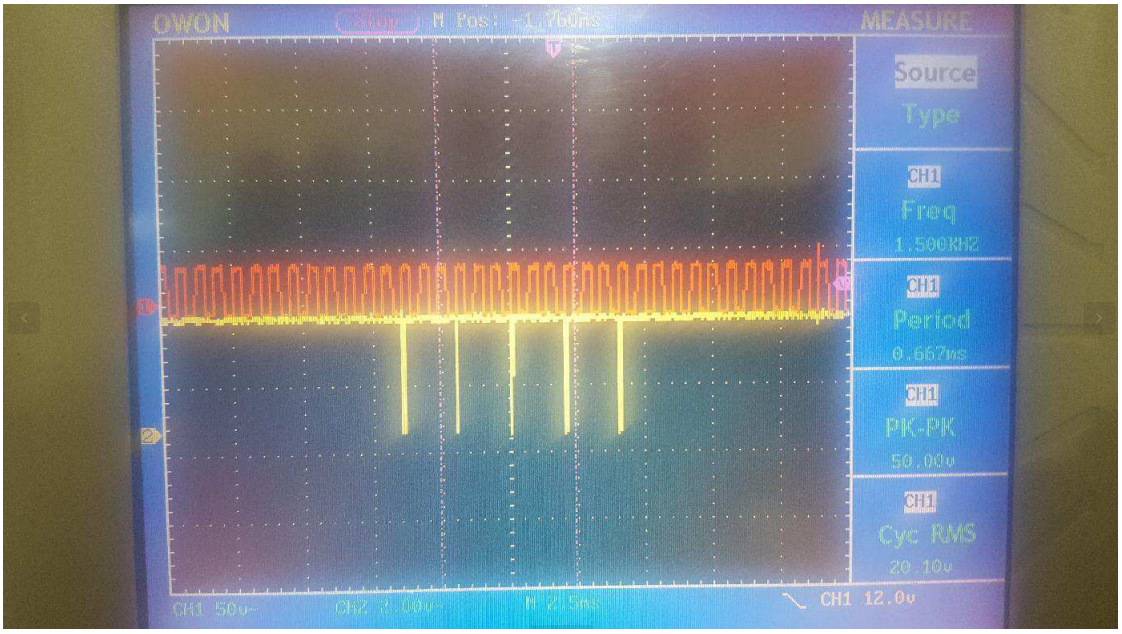


Figura 41: Frame de máquina serial RS-232, relógia 1.5kHz e sinal Tx.

```
[gvillanova@localhost report]$ sudo jpnevulator --ascii --timing-print --tty /dev/ttyUSB0 --read
2018-11-05 23:42:18.543802:
01 6E 3A 21 00                                  .n:!.
2018-11-05 23:42:19.087737:
01 A4 3D 21 00                                  ..=!.
2018-11-05 23:42:19.639724:
40 21 00                                        @!.
2018-11-05 23:42:20.263712:
21 00 04 8B 44 21 00                            !...D!.
2018-11-05 23:42:21.279761:
00                                              .
2018-11-05 23:42:21.831724:
01 B2 4D 21 00                                  ..M!.
2018-11-05 23:42:22.375704:
01 E8 50 21 00                                  ..P!.
2018-11-05 23:42:23.007736:
01 1E 54 21 00 04 99 54 21 00                   ..T!...T!.
2018-11-05 23:42:23.471711:
57 21 00                                        W!.
2018-11-05 23:42:24.023793:
01 8A 5A 21 00                                  ..Z!.
2018-11-05 23:42:24.567749:
01 C0 5D 21 00                                  ..]!.
2018-11-05 23:42:25.119726:
01 F6 60 21 00                                  ..`!.
2018-11-05 23:42:25.751753:
```

Figura 42: Monitoramento de bits transmitidos pelo módulo serial do Hoc.

Assim, sabendo o protocolo da serial e a velocidade de transmissão, o próximo passo foi verificar se os dados são recebidos pelo host. Para isso foi utilizado o software gratuito jpnevulator que monitora os valores de uma porta serial. O resultado obtido da conexão é apresentado na Figura 42, do lado esquerdo da saída no terminal tem-se os valores em hexadecimal recebidos da FPGA, no canto direito esses valores são decodificados como caractéres ASCII. Entretanto, como a informação é formatada como um cabeçalho de 1 byte e um corpo little endian sem sinal de 4 bytes que contém o valor do temporizador, os dados apresentados como ASCII devem ser ignorados, pois os dados não são caractéres, apenas valores. Esse experimento mostrou que a serial está conseguindo transmitir dados para o host, com isso, vançou para conexão da FPGA no MATLAB, discutidos na próxima seção.

5.2.6 Conexão FPGA-MATLAB e sinais gerados pela plataforma

O último passo foi conectar a FPGA no MATLAB, para isso bastou abrir o MATLAB, entrar no diretório pvs/GUI e executar "run PVS_GUI.m". Para versões mais novas do MATLAB deve-se comentar a função wavread.

Conhecendo a porta serial, basta colocá-la na interface e apertar o botão do start e os sinais transmitidos pela serial são transformados em um ECG sintético, como pode ser observado na Figura 43. Todas as funcionalidas da interface foram testadas e funcionaram. As linhas APace e VPace representam os sinais que devem ser enviado pelo marca-passo, porém, foram colocadas no pinos SW (switches) da FPGA e os sinais foram gerados manualmente, verificando funcionalidade.

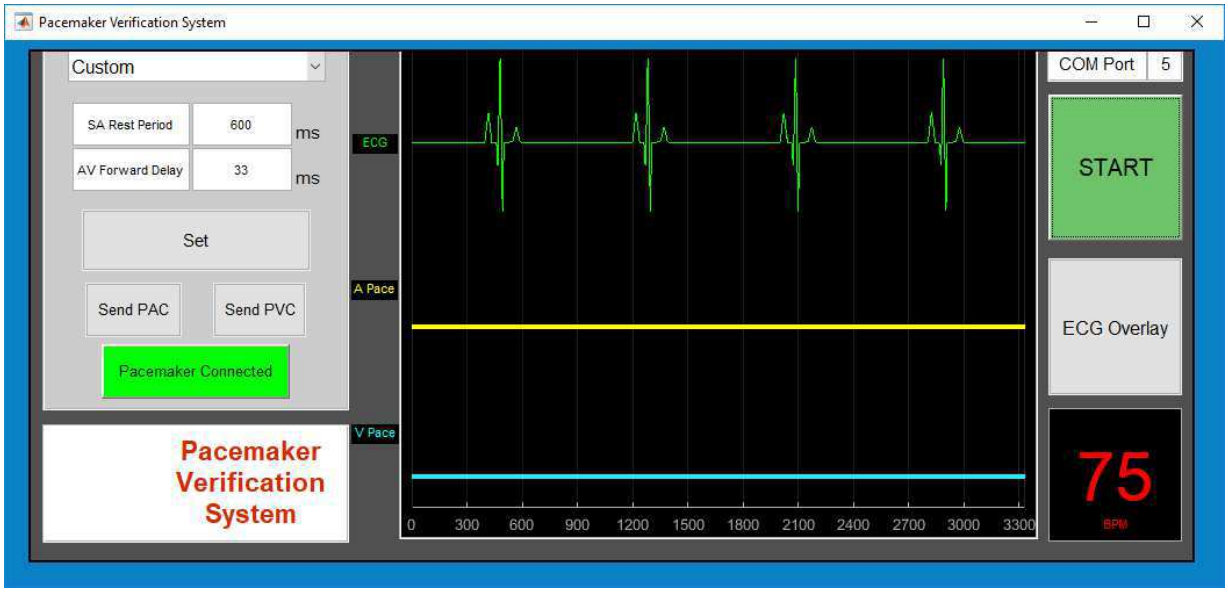


Figura 43: Resultado de conexão de FPGA com PVS no Matlab.

Por fim, foram verificados com auxílio de um osciloscópio os pulsos gerados pelos autômatos de nó. As Figuras 44, 45, 47, 46 e 48 apresenta os sinais capturados da plataforma, verifou-se que os pulsos de nós simulados na Seção 4 se comportam de forma equivalente na plataforma, porém aqui o sinal é prolongado por 10ms, como já fora discutido anteriormente.

Pode-se observar que foi conseguido criar a plataforma VHM em hardware e aproveitar toda a infraestrutura que é fornecida pelo projeto. Aproxima etapa, então, foi desenvolver uma interface analógica digital para permiter que um marca-passo real se conecte com o coração virtual e com isso avaliar o comportamento de ambos, para posteriormente, criar uma plataforma automatizada de validação de marca-passos. Na próxima seção é apresentado a interface A/D desenvolvida.

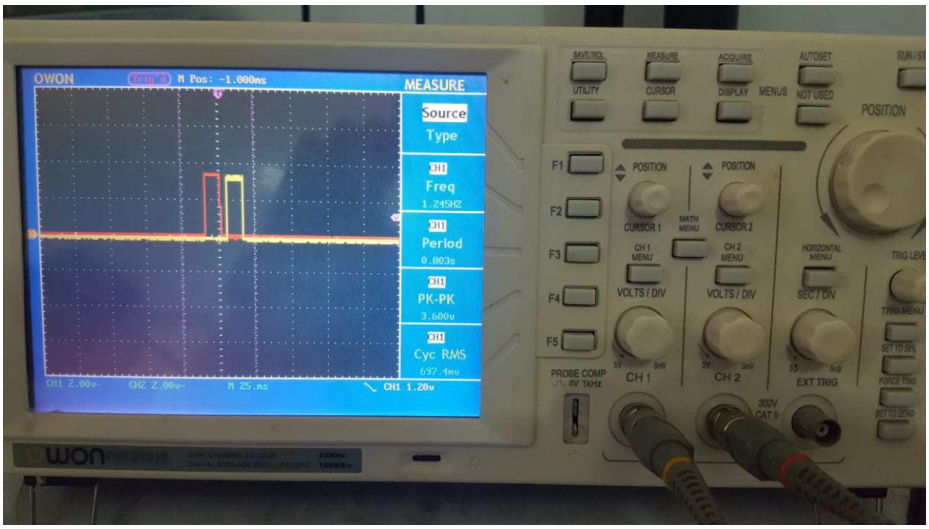


Figura 44: Captura dos pulsos gerados pelos nós NA1 e NA2 - (A).

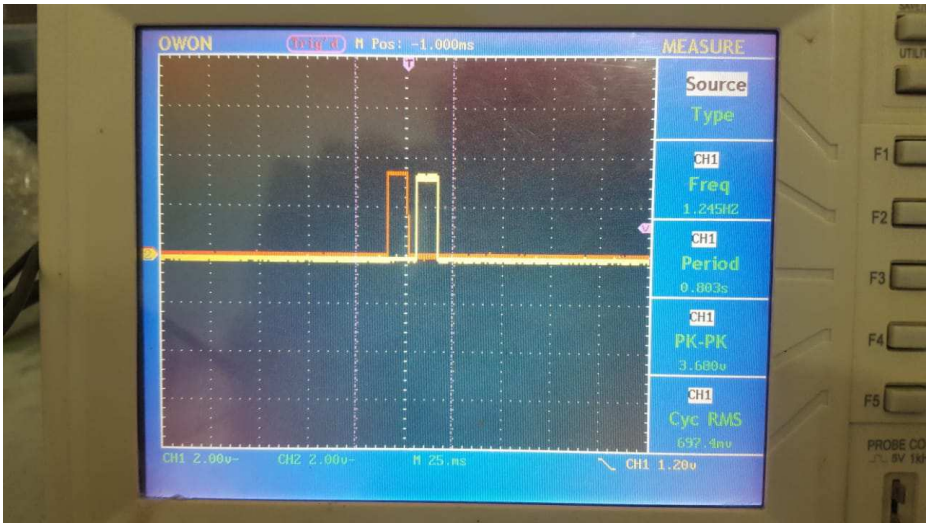


Figura 45: Captura dos pulsos gerados pelos nós NA1 e NA2 - (B).

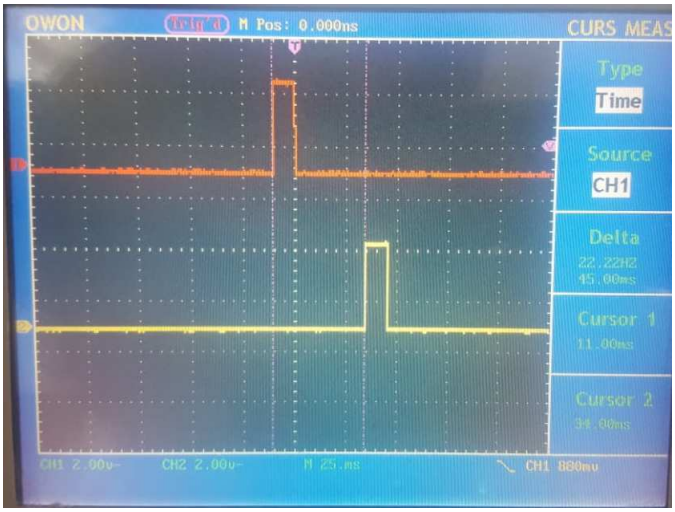


Figura 46: Diferença de 45ms entre os nós NA1 e NA2.

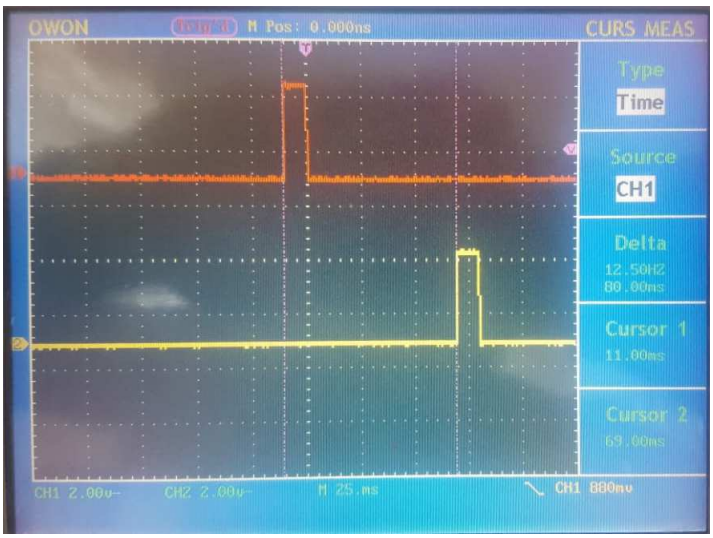


Figura 47: Diferença de 80ms entre os nós NA1 e NA3.

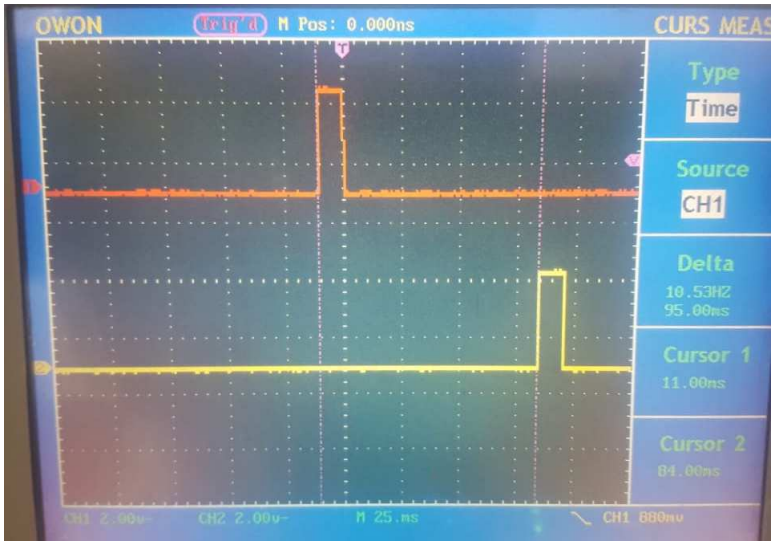


Figura 48: Diferença de 95ms entre os nós NA1 e NA7.

6 Interface A/D e resultados finais

Para conectar um marca-passo real a FPGA, é preciso criar uma interface A/D que condicione o sinal digital para o nível analógico que a interface do marca-passo suporata, e também fazer o inverso, isto é, condicionar o sinal analógico do marca-passo para o nível digital de entrada suportada pela FPGA.

Um circuito com essa finalidade foi desenvolvido e é apresentado em [5], nos Anexos A, B e C estão todas as informações contidas sobre o circuito, sendo elas: lista de materiais, esquemático do circuito e desenho da PCI. Com base nessas informações foi desenvolvido uma projeto de PCI para implementar essa funcionalidade. O projeto foi feito no software aberto KiCAD, sendo a lista de materiais, esquemático e desenho do projeto PCI apresentadas no Apêndice A, B e C respectivamente. A placa usou apenas dois layers (*Front* e *Back*) e ficou com dimensão final de 5,2cm por 6,2cm. .

A Figura 49 apresenta o modelo 3D da placa. Infelizmente não houve tempo para implementar esse circuito e com isso não pode terminar a construção da plataforma, consequentemente os próximos passos que seriam, conectar marca-passo a plataforma, iniciar o desenvolvimento com base nas normas da ABNT e finalmente construir uma plataforma automatizada de testes para validação de marca-passos não fora feita.

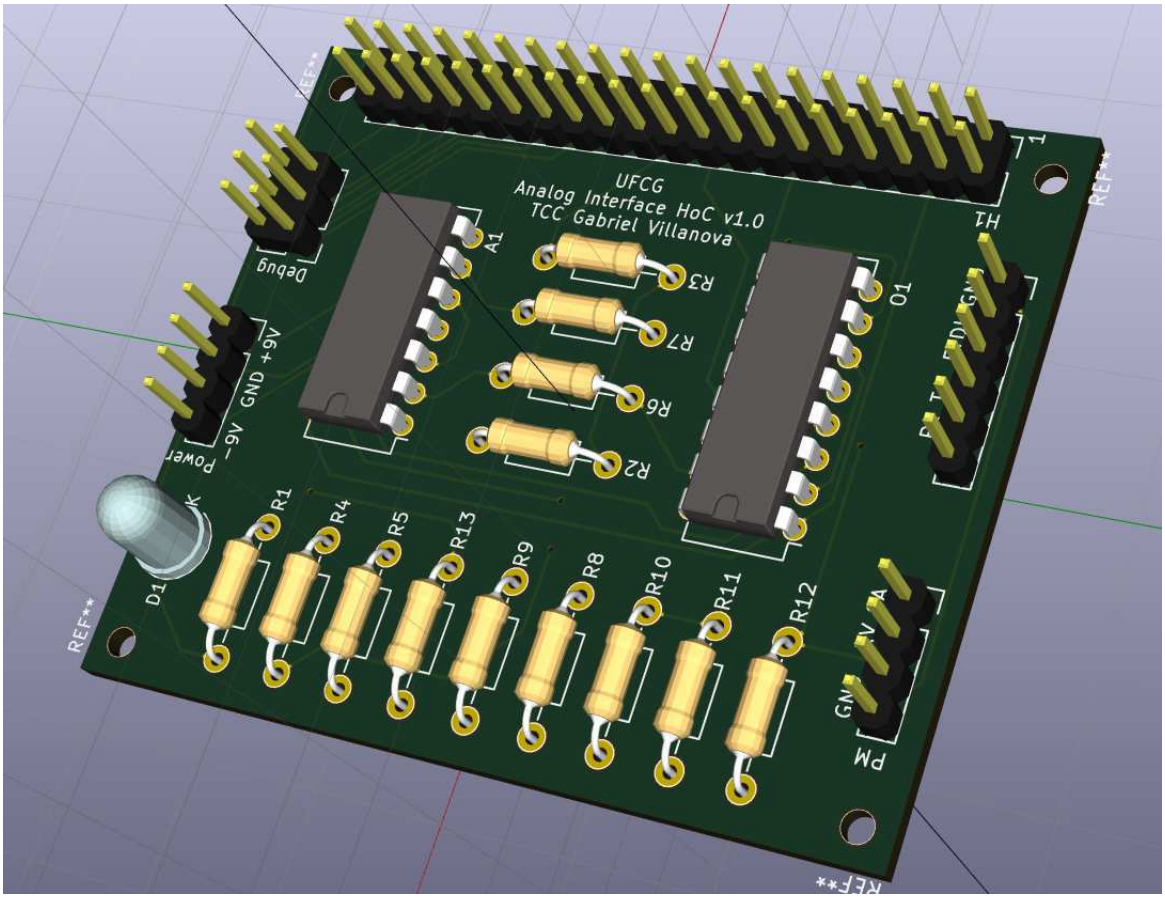


Figura 49: Modelo 3D da PCI desenvolvida para interface A/D.

7 Conclusão

Foi bastante satisfatório o trabalho realizado, muitas áreas de conhecimento foram usadas, como: modelagem em tempo real, modelagem com autômatos, desenvolvimento de hardware digital e analógico, simulação de circuitos digitais e eletrofisiologia do coração humano. Tornando esse trabalho um excelente exercício das habilidades adquiridas ao longo do curso de Engenharia Elétrica da UFCG.

O objetivo final do trabalho, de construir uma plataforma automatizada para validação de marca-passos não foi realizada, sendo um resultado esperado devido a complexidade de se desenvolver um projeto assim. Entretanto, o trabalho será continuado no mestrado da UFCG pelo mesmo autor e será desenvolvido ao longo de dois anos.

Pretende-se, ao longo desses dois anos, estudar mais profundamente todos os assuntos abordados nesse trabalho, o formalismo matemático, o porque de cada tomada de decisão no projeto, propor melhorias, estudar as Normas Brasileiras para equipamentos médicos implantáveis, desenvolver e evoluir as partes pendentes do trabalho e com isso construir uma plataforma de validação de marca-passos automatizada, embasada por todos os formalismos necessários e em conformidade com a ABNT.

Referências

[1] Zhihao Jiang, Miroslav Pajic, Allison T. Connolly, Sanjay Dixit, and Rahul Mangharam, "Real-time Heart Model for Implantable Cardiac Device Validation and Verification", . March 2010.

[2] Zhihao Jiang, Miroslav Pajic, and Rahul Mangharam, "Cyber-Physical Modeling of Implantable Cardiac Medical Devices", Proceeding of IEEE Special Issue on Cyber-Physical Systems 100(1). December 2011. http://dx.doi.org/10.1109/JPROC.2011.2161241.

[3] Z. Jiang, M. Pajic, and R. Mangharam. Closed-loop Verification of Medical Devices with Model Abstraction and Refinement. In ICCPS'11: ACM/IEEE 2nd Intl. Conf. on Cyber-Physical Systems, 2011.

[4] Zhihao Jiang and Rahul Mangharam, "Modeling Cardiac Pacemaker Malfunctions With the Virtual Heart Model", 33 nd International Conference on IEEE Engineering in Medicine and Biology Society (IEEE EMBC) , 263-266. June 2011. http://dx.doi.org/10.1109/IEMBS.2011.6090051.

[5] PVS: Pacemaker Verification System. Senior Design Paper. May, 2012.

[6] M. Pajic, Z. Jiang, I. Lee, O. Sokolsky and R. Mangharam, "From Verification to Implementation: A Model Translation Tool and a Pacemaker Case Study," 2012 IEEE 18th Real Time and Embedded Technology and Applications Symposium, Beijing, 2012, pp. 173-184. doi: 10.1109/RTAS.2012.25

[7] University of Pennsylvania Develops Electrophysiological Heart Model for Real-Time Closed-Loop Testing of Pacemakers. By Zhihao Jiang and Rahul Mangharam, University of Pennsylvania. MathWorks.

[8] Zhihao Jiang, Sriram Radhakrishnan, Varun Sampath, Shilpa Sarode, and Rahul Mangharam, "Heart-on-a-Chip: A Closed-loop Testing Platform for Implantable Pacemakers", July 2014.

[9] Ecg Essencial - Eletrocardiograma na Prática Diária - 7ª Ed. 2013 - Malcolm S. Thaler.

[10] Eletro-Fármaco-Fisipatologia Dos Canais Iônicos Cardiacos E Sua Repercusão No Eletrocardiograma, Novembro 2009.

[11] Lieb WR, Stein WD (1986). "Chapter 2. Simple Diffusion across the Membrane Barrier". Transport and Diffusion across Cell Membranes. San Diego: Academic Press. pp. 69–112. ISBN 0-12-664661-9.

[12] Physiology of cardiac conduction and contractility. Greg Ikonnikov and Dominique Yelle. July, 2005.

[13] L. A. Tuan, M. C. Zheng and Q. T. Tho, "Modeling and Verification of Safety Critical Systems: A Case Study on Pacemaker," 2010 Fourth International Conference on Secure Software Integration and Reliability Improvement, Singapore, 2010, pp. 23-32.

[14] M. Ryzhii and E. Ryzhii, "Simulink heart model for simulation of the effect of external signals," 2016 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), Chiang Mai, 2016, pp. 1-5.

[15] S. N. Narayanan, S. Sutha and G. Divya, "Development of non-linear model for human heart," 2017 Trends in Industrial Measurement and Automation (TIMA), Chennai, 2017, pp. 1-5. doi: 10.1109/TIMA.2017.8064818

[16] N. Majma, S. M. Babamir and A. Monadjemi, "Runtime verification of pacemaker using fuzzy logic and colored Petri-nets," 2015 4th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS), Zahedan, 2015, pp. 1-5.

[17] D. Halperin, T. S. Heydt-Benjamin, K. Fu, T. Kohno and W. H. Maisel, "Security and Privacy for Implantable Medical Devices," in IEEE Pervasive Computing, vol. 7, no. 1, pp. 30-39, Jan.-March 2008.

[18] L. K. Holley and C. F. Hughes, "Medical device development practices: empirical survey and legal implications," Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society. Vol.20 Biomedical Engineering Towards the Year 2000 and Beyond (Cat.No.98CH36286), Hong Kong, China, 1998, pp. 3327-3330 vol.6.

Anexo A: BOM para concepção de circuito analógico HoC.

Quantidade	Descrição
1	Altera/Terasic DE0-Nano FPGA Board
1	Sparkfun FTDI Basic Breakout Board (3.3V)
1	Toshiba TLP-624 Opto-isolation IC
1	LF347 Quad JFET Operational Amplifier IC
1	Boston Scientific Altrua S606 Pacemaker

Tabela 5: BOM (Bill of Materials) do circuito analógico HoC original.

Anexo B: PCI do circuito analógico do HoC.

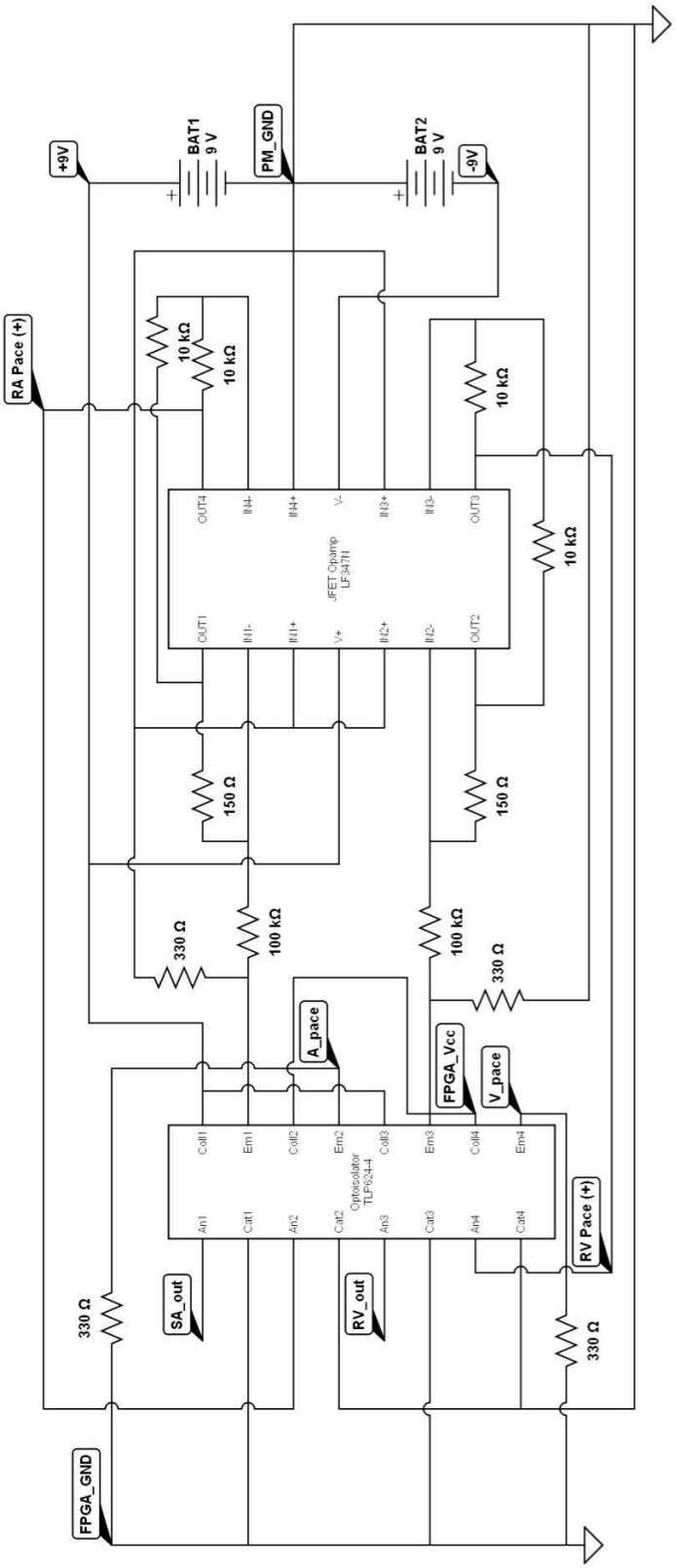


Figura 50: PCI do circuito analógico HoC.

AnexoCB: Modelo 3D de circuito analógico do HoC.

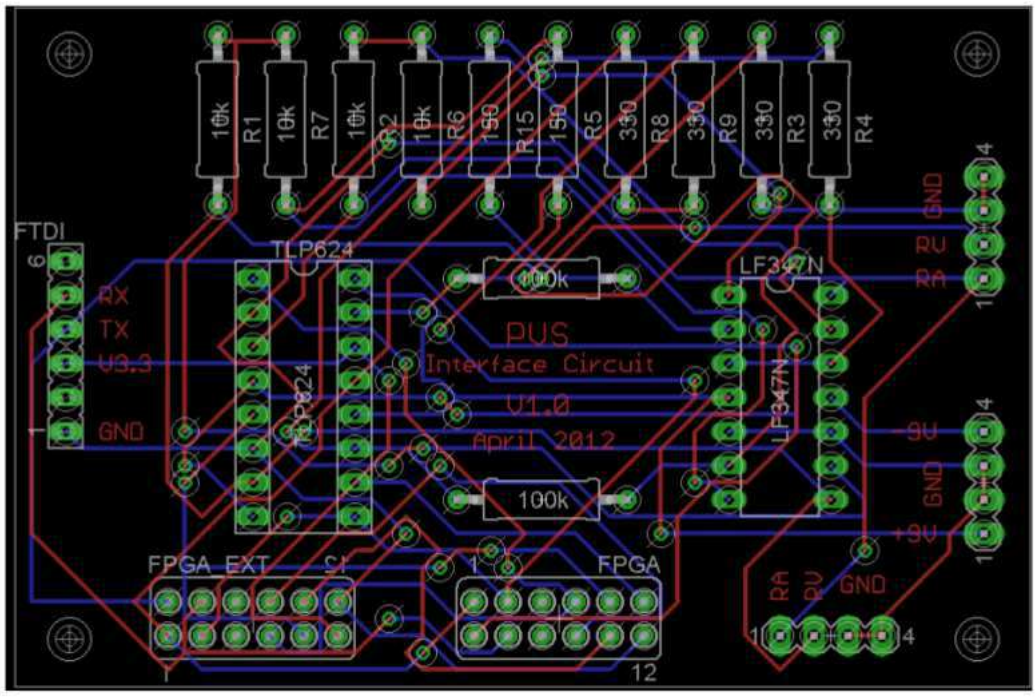


Figura 51: Modelo PCI do circuito analógico HoC.

Apêndice A: BOM PCI UFCG-HoC

Qnt	Item	Manufacturer	Descrição
1	732-5310-ND Würth Elektronik	61304021121	CONN HEADER VERT 40POS 2.54
2	732-5317-ND Würth Elektronik	61300411121	CONN HEADER 4 POS 2.54
1	732-5319-ND Würth Elektronik	61300611121	CONN HEADER 6 POS 2.54
1	732-5295-ND Würth Elektronik	61300621121	CONN HEADER VERT DUAL 6POS 2.54
5	CF14JT330RCT-ND Stackpole Electronics	CF14JT330R	RES 330 OHM 1/4W 5% AXIAL
4	CF14JT10K0CT-ND Stackpole Electronics	CF14JT10K0	RES 10K OHM 1/4W 5% AXIAL
2	CF14JT100KCT-ND Stackpole Electronics	CF14JT100K	RES 100K OHM 1/4W 5% AXIAL
2	CF14JT150RCT-ND Stackpole Electronics	CF14JT150R	RES 150 OHM 1/4W 5% AXIAL
1	296-7138-5-ND Texas Instruments	LF347N	IC OPAMP JFET 3MHZ 14DIP
1	M3AAA-4006J-ND	3M M3AAA-4006J IDC	CBL - HHSC40H/AE40G/HHSC40H
1	TLP624-4(F)	Toshiba	TLP624-4(F)
4	732-12665-ND	Würth Elektronik	960040010

Tabela 6: BOM (Bill of Materials) do circuito analógico UFCG-HoC.

Apêndice B: Esquemático UFCG-HoC

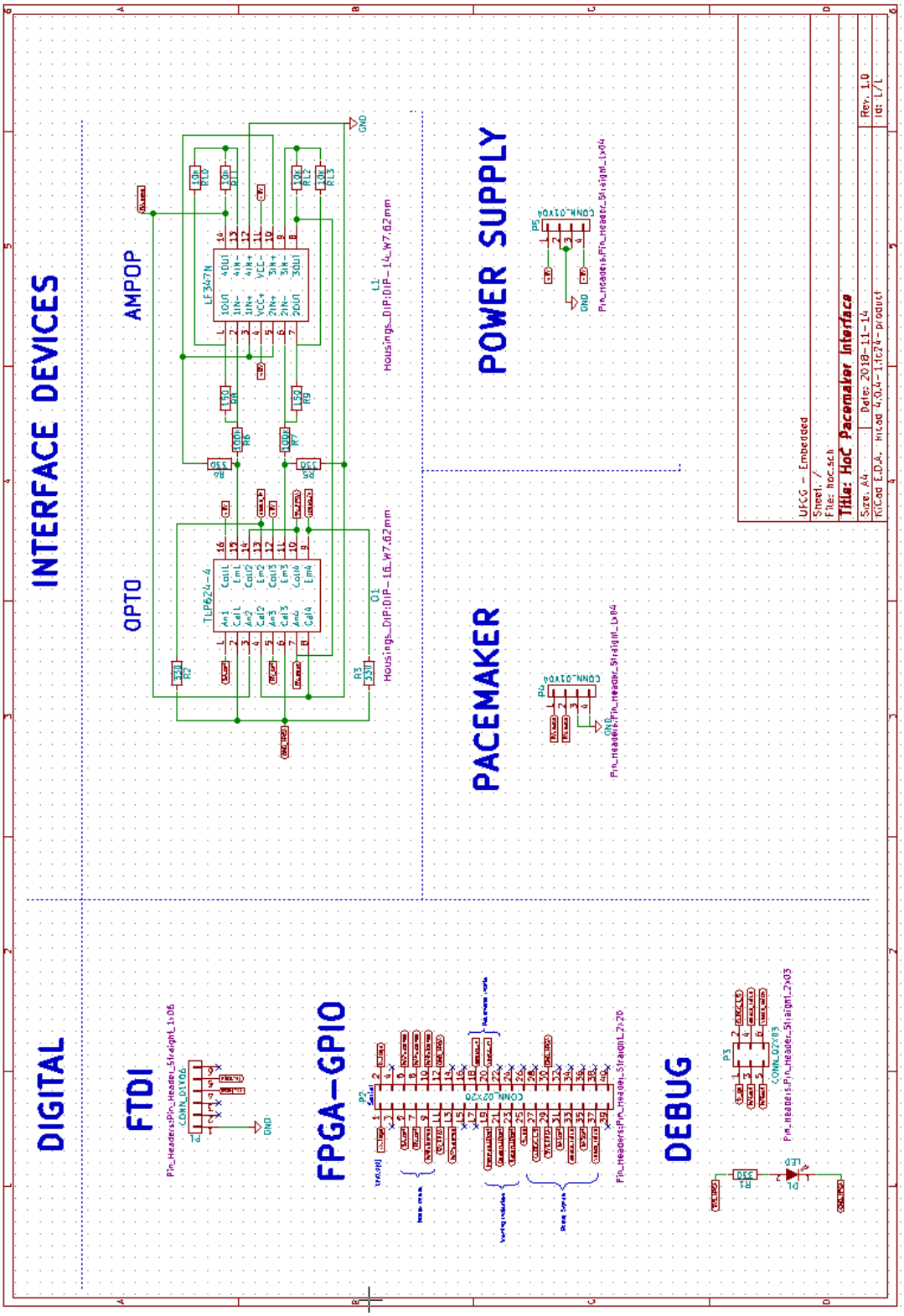


Figura 52: Esquemático do circuito analógico UFCG-HoC.

Apêndice B: PCI UFCG-HoC

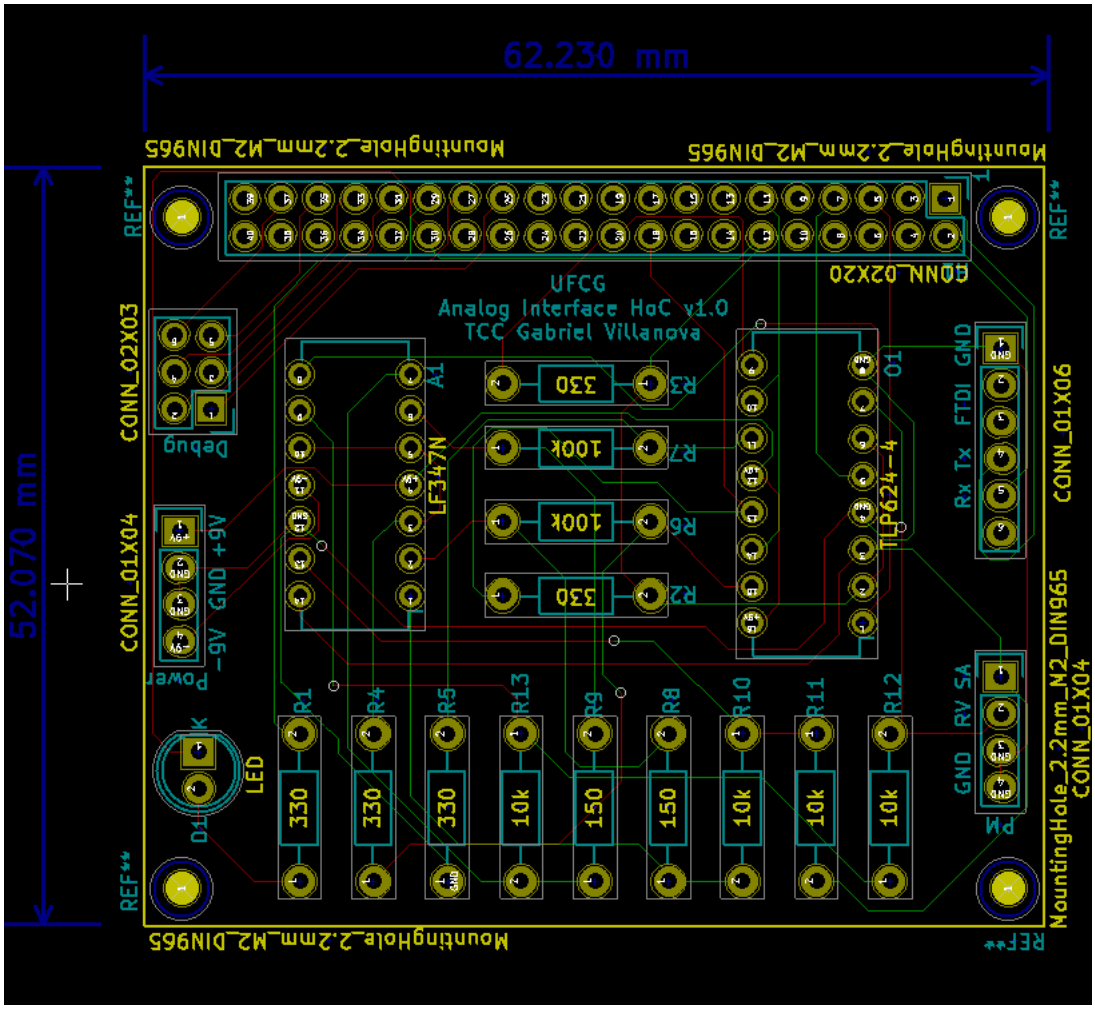


Figura 53: PCI do circuito analógico UFCG-HoC.