Proposta de um agente cognitivo para gerenciar as regras de roteamento e *stackup* de uma placa de circuito impresso

Rodrigo Giroto Borges

***Abstract*— O projetos de *hardware* e PCB (*Printed Circuit Board*) tornam-se cada vez mais complexos em virtude da evolução da capacidade de processamento e interfaces de comunicação. Desta forma, o roteamento do PCB segue esta mesma tendência de complexidade sendo necessário adicionar um maior número de *layers* para roteamento e especificar uma série de *constraints* para que os circuitos funcionem adequadamente e atenda ainda a critérios de fabricação. A proposta deste artigo é criar um agente com uma mente artificial utilizando uma arquitetura cognitiva com capacidade de gerenciar as regras para roteamento da placa e realizar os *trade-offs* necessários para atender os requisitos elétricos e de fabricação.**

***Index Terms*— PCB, agente cognitivo, arquiteturas cognitivas**

I. Introdução

DURANTE o processo de desenvolvimento de *hardware*, o engenheiro de projeto deve lidar com uma série de restrições de ordem mecânica, elétrica, de fabricação, montagem, *signal integrity*, *power integrity*, dentre outros, para que o projeto atenda os requisitos e funcione de maneira esperada no final. Muitas dessas restrições interferem no projeto de maneira mútua, o que exige uma análise de *trade-off* entre as restrições para que haja um balanço sem comprometer o resultado final do projeto. Como exemplos de projeto desta natureza e utilizados no nosso dia a dia, podemos citar os equipamentos ligados à área de *consumer*, como é o caso dos *laptops, smartphones* e *tablets*. Em uma análise rápida destes produtos, é possível observar um número cada vez maior de funcionalidades, sensores e circuitos (GPS, acelerômetros, giroscópio, *light sensor, proximity sensor*) por área, uma redução do tamanho dos equipamentos (pelo menos quando comparado aos computadores) e aumento da capacidade de processamento, já prenunciado por Gordon Moore[Moore 1965].

Como consequência, os projetos de PCB(*printed circuit board*) destes equipamentos tornam-se cada vez mais complexos dentro deste contexto exigindo que o engenheiro eletrônico seja capaz de lidar com condições cada vez mais restritas de projeto de placa.

Neste contexto, os engenheiros utilizam as ferramentas EDA (*Electronic Design Automation*) para o desenvolvimento de *hardware*, especificamente os projetos que exige o *design* de PCB, projetos que serão tratados com maior ênfase neste trabalho. Exemplos de ferramentas EDA largamente utilizadas incluem as ferramentas da Cadence, Mentor Graphics, Altium Designer, Zuken e Eagle, dentre outras. Estes *softwares* oferecem com maior ou menor grau de maturidade, um ambiente com ferramentas próprias para lidar com diversos aspectos do projeto, incluindo o *design* do esquemático, *design* da placa, simulação de *signal integrity* e *power integrity, mechanical restrictions, Bill of Materials* até a geração de arquivos de manufatura (*gerbers files*)e montagem da placa. Muitos desses aspectos são tratados e controlados dentro da ferramenta EDA na forma de *constraints* ou restrições de projeto. Desta forma, temos *constraints* de ordem elétrica, relacionado ao *placement*, roteamento, manufatura, SMT (*Surface mounting technology*), planos de alimentação, pontos de testes, máscaras, linhas de *high spees* e *signal integrity*. A Fig.1 [PCB Rules Altium] mostra os *constraints* ou PCB *rules* da ferramenta Altium Designer extraído do manual.

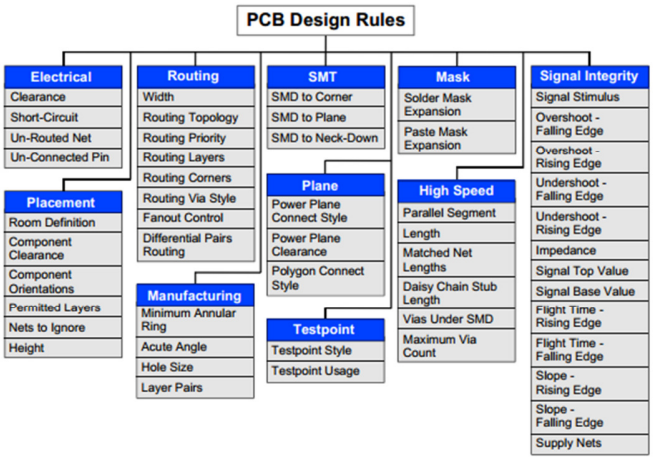

Fig. 1: exemplo de regras de PCB

A Fig. 2 mostra, em linhas gerais como é o fluxo de projeto para transformar a solução eletrônica (desenho esquemático) no projeto de PCB dentro da ferramenta EDA.

R.G. Borges (e-mail: rodrigogiroto@yahoo.com.br).

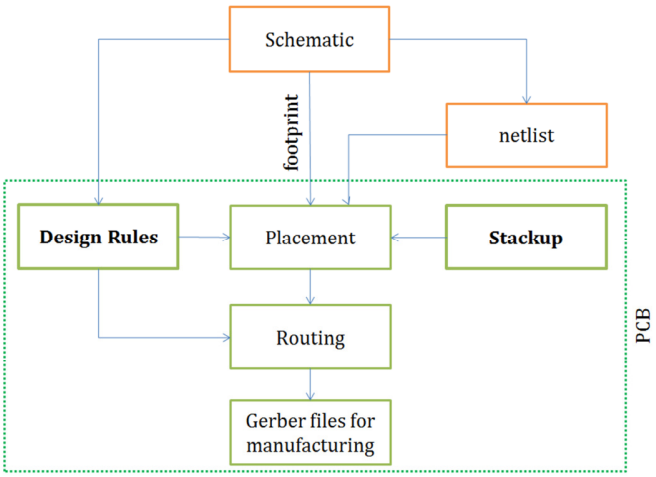

Fig.2: Fluxo de trabalho para gerar o *design* do PCB a partir do esquemático

- *Schematic*: desenho contendo a solução eletrônica do circuito;
- *Footprint*: formato físico ou *package* do componente;
- *Netlist*: lista de conexão entre os pinos dos componentes;
- *Design Rules*: módulo onde são inseridos as regras de *design* para realizar os controles do projeto, conforme descrito na Fig. 1;
- *Placement*: posicionamento dos componentes na placa. Alguns dos seus aspectos podem ser controlados pelo módulo de *design rules*;
- *Stackup*: empilhamento dos *layers* da placa;
- *Routing*: processo para realizar o roteamento da placa. Pode ser realizado de forma manual, semi-manual (*interactive routing*) ou automática. É fortemente controlado pelo módulo de *design rules*.
- *Gerber files for manufacturing*: etapa de geração dos arquivos finais para fabricação da placa;

O engenheiro de *hardware*, como detentor do conhecimento do projeto, deve introduzir na ferramenta EDA as informações para obter um projeto de placa confiável. Quanto maior for a sua capacidade de transformar seu conhecimento implícito em regras de projeto e passíveis de ser introduzida na ferramenta EDA na forma de regras, tanto maior será o controle da ferramenta e menor será a chance de erros.

A. *Ciências cognitivas*

Dentro da área de ciências cognitivas e inteligência artificial, diversos trabalhos vêm sendo conduzidos para tentar criar máquinas inteligentes com capacidade cognitiva comparável a um ser inteligente e, de maneira mais ambiciosa, a um humano. Dentro deste esforço, diversos pesquisadores propuseram modelos de arquitetura cognitiva para criação de mentes artificiais. A arquitetura cognitiva pode ser definida como um *framework* de *software* que reúne as áreas essenciais de uma mente de um agente cognitivo e, ao ser programado de forma adequada, será capaz de realizar tarefas e tomar decisões de maneira inteligente o suficiente para atingir o objetivo proposto.

Diversos grupos de pesquisas ao redor do mundo implementaram e estão implementando ainda modelos de arquitetura cognitiva, como é o caso da arquitetura ACT-R desenvolvida por John R. Anderson na *Carnegie Mellon University*, arquitetura Clarion desenvolvida por Ron Sun na *Rensselaer Polytechnic Institute*, arquitetura LIDA desenvolvida por Stan Franklin na *University of Memphis* e a arquitetura SOAR desenvolvido por Allen Newell e John Laird na *Carnegie Mellon University* e *University of Michigan*.

Neste trabalho, o autor propõe na sequencia que, utilizando uma dessas arquiteturas cognitivas, é possível criar um agente com capacidade cognitiva suficiente para realizar parte do trabalho relacionado ao desenvolvimento de *hardware*. Embora boa parte do desenvolvimento dentro do ambiente da ferramenta EDA possa ser controlado por regras de *design*, será discutido na sequencia de forma mais detalhada apenas as regras e procedimentos para realizar o roteamento da placa e definição do *stackup*. Em especial, as regras de roteamento e cuidados ligados às interfaces rápidas que, por apresentarem frequências de comunicação mais altas, possuem regras de projeto mais restritas e são mais criticas no desenvolvimento da eletrônica dos produtos já citados na introdução deste trabalho. No entanto, nada impede que o aprendizado gerado a partir deste trabalho possa ser aplicado na evolução posterior do agente cognitivo para que este assuma o trabalho de outras etapas do projeto, além da definição das regras de roteamento e definição do *stackup*.

B. *Agente cognitivo*

Para que o agente cognitivo possa definir as regras de *design* e o *stackup*, é necessário que o mesmo detenha as informações relacionadas ao projeto e o conhecimento procedural na sua memória. Essas informações podem ser adquiridas através da interação com o mundo real (através da busca de informações disponíveis em *datasheets* e documentos de projeto) ou informações pré-inseridas em sua memória. Trataremos aqui de um caso em que o agente consulta as informações pré-inseridas em sua memoria e questiona o engenheiro de informações adicionais para tomada de decisão, caso necessário.

Para determinar as regras e o *stackup*, é necessário que o agente conheça os procedimentos e as informações que deve levar em consideração. Essa capacidade pode ser introduzida em um agente através de uma memória procedural e uma memoria episódica.

Fig.3: agente cognitivo com memória procedural e memória episódica

Como exemplo, tanto a arquitetura ACT-R como o SOAR possuem estas memorias em seu *framework*.
Exemplos de informações procedurais incluem o procedimento para cálculo de impedância característica de uma linha de transmissão, procedimentos para realizar *trade-offs* de regras e etc.
Exemplos de informações armazenadas na memoria episódica incluem limitações de fabricação, características elétricas das interfaces, nomes e suas variações de sinais de uma interface e etc.
Para que o agente possa interagir com o projeto dentro da ferramenta EDA, conforme apresentado na Fig. 2, uma forma de razoável simplicidade é ler o arquivo *netlist*. A partir desta leitura, ele deve ser capaz de definir as regras de roteamento e o *stackup*. Para isso será necessário que o agente realize algumas etapas intermediarias.
Ao ler um arquivo *netlist*, o agente deve ser capaz de identificar as *interfaces* rápidas do projeto e, a partir dessas informações, inferir as classes de sinais desta *interface* e as características elétricas dos sinais desta *interface*.

Fig.4: Classificações realizadas pelo agente cognitivo como processo intermediário para a definição de regras de roteamento e *stackup.*

Abaixo está listado um exemplo aplicado a um circuito de memoria DDR3.

- *Interface*: memória DDR3;
- Classes de sinais: *Data, Adress, Command* e *Clock*;
- Características elétricas (*Data*): impedância 50 Ohms, 1,65V;

Para realizar a definição do *stackup*, o agente cognitivo deve considerar as informações das *interfaces* rápidas (qual é a impedância dos sinais que exigem impedância controlada) e consultar informações de espessuras de materiais disponíveis na sua memoria episódica, conforme apresentado na Fig. 5.

Fig.5: etapa de definição do *stackup* da placa

Além disso, para que o agente seja capaz de definir um *stackup* de forma confiável, ele pode questionar o usuário sobre algumas informações relevantes para essa decisão como a espessura máxima permitida para a placa, o número de *layers* necessário para realizar o roteamento da placa, quantos destes *layers* serão planos de alimentação ou *ground*, quantos serão para sinais e etc.
Para definir as regras de roteamento, a Fig. 6 mostra as informações mínimas de entrada para que o agente possa realizar esta tarefa.

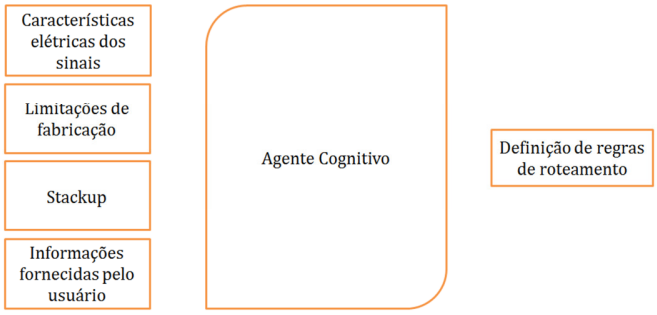

Fig. 6: etapa de definição de regras de roteamento

Durante o processo de definição de regras de roteamento, o agente cognitivo deve realizar ainda uma série de *trade-offs*. A Tabela I demonstra um exemplo para definição de regras de *clearence* e regras de largura de trilhas para uma determinada classe de *nets*.

TABLE I
EXEMPLOS DE TRADE-OFFS NECESSÁRIOS PARA GERAR AS REGRAS DE ROTEAMENTO

Regra	Trade-off	Informações necessárias
Width (classe de nets)	A espessura da trilha tem influencia direta na densidade de roteamento por área, impedância da linha, capacidade de conduzir corrente e dificuldade de fabricação.	- Densidade de conexões; - Limitação de fabricação; - Impedância; - *Stackup*; - Corrente maxima;
Clearence (classe de nets)	O *clearance* entre trilhas tem influência direta na densidade de roteamento por área, *crosstalk* entre trilhas, capacidade de manter a rigidez dielétrica na presença de altas tensões e dificuldade de fabricação.	- Densidade de ligações; - Limitação de fabricação; - Máximo *crosstalk* permitido; - Tensão maxima;

A Fig.7 mostra o diagrama de blocos macro contendo as informações de entradas e ações do agente. Os blocos relacionados à memoria foram omitidos para facilitar a leitura do diagrama.

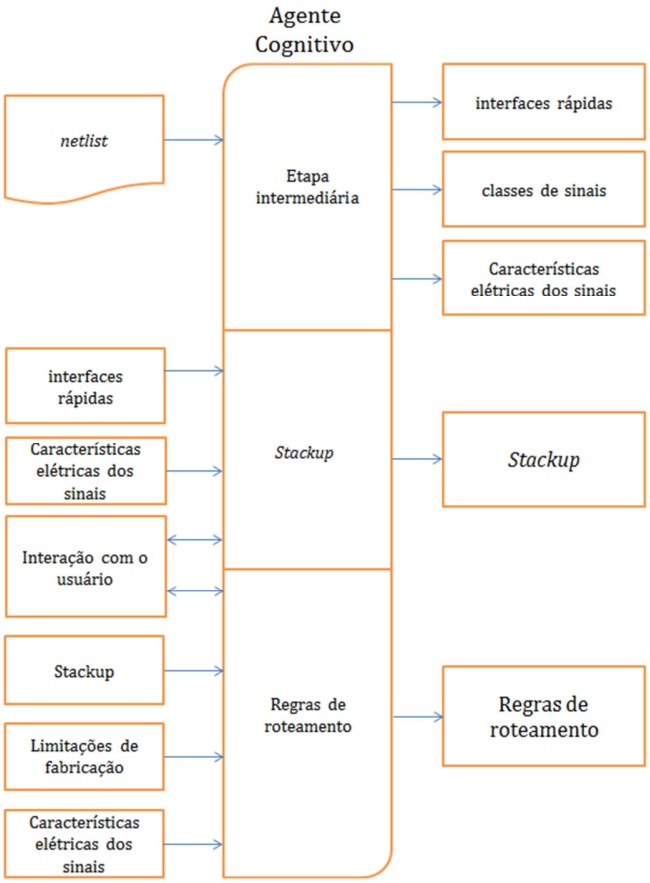

Fig.7: Diagrama de blocos contendo as informações de entrada e ações internas e externas do agente cognitivo

II. Conclusão

Nas ferramentas EDA atuais, o conhecimento existente do projeto é de total responsabilidade do engenheiro e a ferramenta atua como um ambiente de apoio para que o projeto possa ser desenvolvido. Ao longo do tempo, as ferramentas EDA incorporaram uma série de automações para as tarefas repetitivas. Também foram incorporados automações para reuso de blocos de circuitos, permitindo que aqueles circuitos que são exatamente iguais e se repetem possam ser reutilizados em outras placas sem a necessidade de um *redesign*.

No entanto, na essência, essas e outras evoluções que ocorreram neste mercado, estão ligadas apenas a automação ou reutilização de *design*, sem que a ferramenta apresente uma inteligência própria.

O desenvolvimento de agentes cognitivos com capacidade de entendimento do projeto que está executando pode ser uma solução para incorporar inteligência as ferramentas e apresentar uma nova perspectiva para evolução destas ferramentas. Os estudos levantados serão implementado utilizando uma arquitetura cognitiva em uma próxima fase.

References

[Moore 1965] G. E. Moore. (1965, Apr.). Cramming more components onto integrated circuits. Farchild Semiconductor. [Online]. Available: http://web.eng.fiu.edu/npala/eee6397ex/gordon_moore_1965_article.pdf

[PCB Rules Altium] *Design Rules Reference,*v1.8, *Altium Designer*, 2008. [Online].Available: http://www.altium.com/files/altiumdesigner/s08/learningguides/tr0116%20design%20rules%20reference.pdf