Circuitos Elétricos II – 2º semestre de 2018 - Trabalho

Prof. Antônio Carlos Moreirão de Queiroz

Escrever um programa que analise circuitos no domínio do tempo, contendo elementos lineares e não lineares, usando análise nodal modificada, o método "backward" de Euler e o de Newton-Raphson.

O programa deverá analisar um circuito composto pelos elementos:

- Fontes de corrente e de tensão independentes.
- Resistores, capacitores e indutores.
- As quatro fontes controladas.
- Amplificadores operacionais ideais, de 4 terminais.
- Portas lógicas AND, NAND, OR e NOR
- Flip-flops D e monoestáveis.

O programa deve ler um "netlist" descrevendo o circuito, inicializar a análise no domínio do tempo com cálculo com condições iniciais ou de ponto de operação, e então fazer a análise com o tempo total e o tamanho do passo, fixo, dados. Os resultados devem ser salvos em uma tabela em arquivo, de forma que possam ser lidos por outro programa que plote as curvas, como o MNAE. Uma linha de comando no "netlist" define os parâmetros necessários. A primeira linha desta tabela deve listar os nomes de todas as variáveis na tabela, iniciando pelo tempo "t", com correntes citadas como "j" seguidas do nome do elemento onde estão. Ex: t 1 2 3 jH1 jF2.

O programa pode ser baseado no programa exemplo MNA1, que implementa a análise pedida para um circuito resistivo linear. Uma versão gráfica (para o Borland C++ builder 6) está também disponível:

http://www.coe.ufrj.br/~acmq/cursos/mna1.zip
http://www.coe.ufrj.br/~acmq/cursos/mna1gr.zip

Formato para o "netlist":
O "netlist" pode ser gerado pelo programa EDFIL, a partir do diagrama esquemático. Veja os programas em http://www.coe.ufrj.br/~acmq/cursos

Primeira linha: Comentário (o editor EDFIL coloca o número de nós nesta linha).
Linhas seguintes: Descrição do circuito, com um elemento por linha. A primeira letra determina o tipo de elemento.

Resistor: R<nome> <nó1> <nó2> <Resistência>
Indutor: L<nome> <nó1> <nó2> <Indutância> [IC=<corrente inicial>]
Capacitor: C<nome> <nó1> <nó2> <Capacitância> [IC=<tensão inicial>]
Fonte de tensão controlada a tensão: E<nome> <nóV+> <nóV-> <nóv+> <nóv-> <A_v>
Fonte de corrente controlada a corrente: F<nome> <nóI+> <nóI-> <nói+> <nói-> <A_i>
Fonte de corrente controlada a tensão: G<nome> <nóI+> <nóI-> <nóv+> <nóv-> <G_m>
Fonte de tensão controlada a corrente: H<nome> <nóV+> <nóV-> <nói+> <nói-> <R_m>
Fonte de corrente: I<nome> <nó+> <nó-> <parâmetros>
Fonte de tensão: V<nome> <nó+> <nó-> <parâmetros>
Amplificador operacional ideal: O<nome> <nó saída+> <nó saída-> <nó entrada+> <nó entrada->
AND:)<nome> <nósaída> <nóentrada> <nóentrada> <Parâmetros>
NAND: (<nome> <nóentrada> <nóentrada> <nósaída> <Parâmetros>
OR: }<nome> <nóentrada> <nóentrada> <nósaída> <Parâmetros>
NOR: {<nome> <nóentrada> <nóentrada> <nósaída> <Parâmetros>
Flip-Flop: %<nome> <nóQ+> <nóQ-> <nóD> <nóCk> [<Reset>] <Parâmetros>
Monoestável: @<nome> <nóQ+> <nóQ-> <nóTrigger> <nóReset> <Parâmetros>
Reset: !<nome> <nóSet> <nóReset> <Parâmetros>
Comentário: *<comentário>

(Notar que <xxx> significa colocar o valor xxx sem <>.)

Atualizado em 26/9/2018

Os parâmetros para as fontes são de acordo com o formato do SPICE, como implementado no programa MNAE. Devem ser suportadas fontes contínuas, senoidais e em pulsos.

Fonte contínua: DC <valor>
Fonte senoidal: SIN <nível contínuo> <amplitude> <frequência em Hz> <atraso> <amortecimento> <defasagem em graus> <número de ciclos>
Fonte pulsada: PULSE <amplitude 1> <amplitude 2> <atraso> <tempo de subida> <tempo de descida> <tempo ligada> <período> <número de ciclos>

A fonte senoidal vale:

$$x(t) = A_0 + Ae^{-\alpha(t-t_a)}\,\mathrm{sen}\left(2\pi f\,(t-t_a) + \frac{\pi}{180}\varphi\right)$$

onde A_0 é o nível contínuo, A a amplitude, f a frequência, t_a o atraso, α o amortecimento e φ a defasagem. Antes de $t = t_a$ ou após o número de ciclos, tem o valor inicial ou final respectivamente, de forma a não criar descontinuidades. (Há casos em que é mais útil que volte ao valor contínuo. O programa pode permitir esse modo de operação.) A fonte pulsada começa na amplitude 1, e fica aí até o fim do tempo de atraso. Então muda para a amplitude 2 variando linearmente dentro do tempo de subida, fica na amplitude 2 durante o tempo ligada, volta à amplitude 1 dentro do tempo de descida, e repete tudo com o período e o número de ciclos especificados. Termina na amplitude 1. Os tempos de subida e de descida podem ser nulos. O programa pode usar o tempo do passo então.

As correntes nos indutores devem ser calculadas. As nos capacitores não.

As direções para fontes são de acordo com a ordem dos nós e as direções convencionais associadas, sendo o primeiro nó o positivo.

Os parâmetros para as portas lógicas são: <V> <R> <C> <A>.
A tensão V é a tensão de saída máxima, R a resistência da saída, C a capacitância de entrada, para cada entrada, e o ganho A é a derivada da tensão de saída em aberto em relação à variação da tensão em qualquer entrada, antes da saturação da saída, em torno de V/2. É usado um modelo Norton para a saída.

Os parâmetros para o flip-flop são <V> <R> <C>, como nas portas lógicas.
Para o monoestável são <V> <R> <C> <T>, T sendo o tempo em que fica ligado com Q+=V e Q-=0.
Os modelos destes elementos são comportamentais:
Para o flip-flop: Copia a entrada D para a saída, ou V ou 0 dependendo de se a entrada D está acima ou abaixo de V/2, quando a entrada Clock muda de abaixo de V/2 para acima de V/2. Para o monoestável, muda a saída Q+ para V e a Q- para 0 quando a entrada Trigger passa de abaixo para acima de V/2, e a mantém por T segundos.
Os modelos das saídas são com equivalentes Norton, como em todos os circuitos lógicos.

O flip-flop pode se referir a um "bloco de set/reset" que realiza estas funções e aparece antes no netlist, pelo seu nome. O bloco tem parâmetros <V> <C> apenas. (Assim todos os elementos só tem 4 terminais no máximo). O monoestável tem seu próprio terminal de "reset".

Exemplo:

!xxx 2 3 3.3 10e-12
%yyy 5 9 12 15 !xxx 3.3 1000 10e-12
@zzz 5 6 8 1 15 1000 10e-12 1e-6

O programa deve ler as instruções de como tratar o "netlist" de uma linha de comando no próprio "netlist", no formato abaixo. Não deve ser necessário fornecer qualquer outro parâmetro ao programa além do arquivo de entrada, embora o programa possa ter outros meios de configuração, fora dos parâmetros normais, por exemplo para "debug". UIC significa usar condições iniciais.

.TRAN <tempo final> <passo> BE <passos por ponto na tabela> [UIC]

No método de Newton-Raphson, caso não ocorra convergência em um número razoável de iterações (20-50), use a técnica de estimar uma nova solução com valores randômicos para as variáveis que não convergem. Conte quantas vezes o ciclo de Newton-Raphson é executado para determinar se a randomização é necessária, e quantas vezes a randomização foi usada, desistindo por não convergência se este limite for atingido. Opcionalmente, o programa pode verificar se estão ocorrendo oscilações entre duas soluções durante a convergência, e partir diretamente para a randomização em vez de esperar um número grande de tentativas.
Opcionalmente, o próprio programa pode plotar seus gráficos. O programa MNAE pode ser usado para plotar os gráficos a partir das tabelas. Este programa faz a mesma análise. Uma versão incluindo flip-flops e monoestáveis será preparada em breve.
http://www.coe.ufrj.br/~acmq/programs/mnae.zip

O programa deve ser escrito preferencialmente em uma linguagem compilada como C, C++ ou Pascal. O programa deve preferencialmente rodar em ambiente gráfico Windows. Evite usar apenas uma interface de console. Um arquivo .zip com tudo o que for necessário para o programa, inclusive fontes, arquivo executável, documentação, bibliotecas e exemplos não deve ter mais de 5 Mbytes. Evite sistemas de desenvolvimento que requeiram extensas bibliotecas instaladas.

Sugere-se partir do programa exemplo MNA1, que já tem o algoritmo completo da análise nodal modificada em C, implementar a análise no tempo com as fontes de sinal, e então implementar os elementos reativos. Por fim implementar os não lineares com o ciclo de Newton-Raphson.

Grupos de 3 alunos, no máximo. O programa deverá ser apresentado e demonstrado, completamente funcional, por todo o grupo, e entregue com um relatório (em arquivo, não impresso) com comentários e exemplos significativos e originais verificados, até (entenda-se antes de) duas semanas antes da segunda prova.

Note-se que o trabalho é bastante extenso, e deve ser começado imediatamente. Trabalhos incompletos serão devolvidos para serem completados, só sendo aceitos completos.