X.2.2/15

4.º CONGRESSO BRASILEIRO DE AUTOMÁTICA — CAMPINAS — 1982

A AUTOMAÇÃO DO PROJETO E DESENVOLVIMENTO DE SISTEMAS DIGITAIS: O SPD COMO FERRAMENTA DE OBTENÇÃO AUTOMÁTICA DE "CROSS-SOFTWARE" PARA SISTEMAS DE COMPUTAÇÃO

JOÃO JOSÉ NETO

Escola Politécnica de São Paulo
Caixa Postal 8.174 - S.P. Brasil

Resumo

A tarefa de projetar sistemas digitais complexos exige uma considerável infraestrutura automática de apoio, sem a qual se torna inviável a obtenção rápida de resultados que permitam iniciar novas fases do projeto. Com o objetivo de fornecer ao projetista este tipo de recurso, foi criado o SPD - sistema automático de apoio ao projeto e desenvolvimento de sistemas digitais aqui apresentado, de maneira compacta, nos seus aspectos estrutural e funcional, enfatizando-se sua aplicabilidade como poderosa ferramenta de projeto em inúmeras áreas. Uma breve discussão das instruções disponíveis nas linguagens e entrada procura apresentar ao leitor os principais recursos oferecidos pelo sistema.

DIGITAL SYSTEMS DESIGN AUTOMATION AND DEVELOPMENT : THE SPD AS A TOOL FOR THE AUTOMATIC CONSTRUCTION OF COMPUTING SYSTEMS CROSS-SOFTWARE.

Abstract

The task of designing complex digital systems demands considerable automatic support background, without which it becomes hard to obtain rapidly the results needed for beginning new phases of a project. SPD - an automatic support system to the project and development of digital systems - has been designed with the intent of providing these kinds of tools to the designer. The present paper describes in a short way the structural and functional aspects of SPD, emphasizing it as a powerful design tool in several application fields. The presentation of SPD's facilities is made through a brief discussion on the instructions available in its input languages.

1. INTRODUÇÃO

A presente publicação tem por objetivo apresentar e descrever o funcionamento e a estrutura do SPD, um sistema desenvolvido no Laboratório de Sistemas Digitais da Escola Politécnica da Universidade de São Paulo com a finalidade de fornecer um ferramental automático para auxiliar a pesquisa e o desenvolvimento na área de projeto e utilização de sistemas digitais.

O projeto SPD [4,5] foi motivado e idealizado como conseqüência das atividades de projeto dos primeiros computadores nacionais, na década passada: Para estes projetos foi necessário o desenvolvimento de uma série de programas auxiliares com a finalidade de viabilizar tarefas que não seriam realizáveis manualmente. Entre estes programas, os mais importantes foram os de documentação de circuitos, traçado de rotas de circuito impresso, fiação, simuladores e "cross-assemblers".[1,6,9,10,11]

A tarefa de construção de simuladores e "cross assembler" mostra ser particularmente repetitiva e trabalhosa, exigindo um constante trabalho de manutenção por parte da equipe que os desenvolve para que possam cumprir sua finalidade ao longo do desenvolvimento do projeto de uma nova máquina.[2,3,5]

Com a intenção de reduzir o investimento da equipe neste sentido, idealizou-se o SPD, cuja missão principal é a de automatizar e facilitar a construção de tais módulos.[4,5]

O SPD apresenta atualmente, como principal recurso, a capacidade de assimilar descrições, em metalinguagem de alto nível, da estrutura e do funcionamento de máquinas digitais, produzido automaticamente, a partir de tais descrições, programas que perfazem a simulação das máquinas descritas, no computador hospedeiro Burroughs B6700, disponível na Universidade de São Paulo. Outro recurso oferecido pelo sistema é a gera-

ção automática de "cross-assemblers", voltados para as necessidades do usuário, a partir de uma descrição funcional destes programas, também realizado através da mesma metalinguagem. Estes "cross-assemblers" podem ser utilizados em conjunto com os simuladores construídos pelo SPD, o que facilita sobremaneira o desenvolvimento de "software" básico e de aplicação para máquinas novas e/ou não disponíveis.

Atualmente, o SPD está completamente implementado, e testado com sucesso em várias aplicações. Encontra-se em andamento, além disso, o projeto de extensões do SPD para incorporar a geração automática de compiladores. Algumas pesquisas satélites foram iniciadas, visando a elaboração de outros trabalhos acadêmicos relacionados com o SPD.[6,7]

2. OBJETIVOS DO SPD

A implantação, por parte de uma equipe de "software", do conjunto de programas de apoio (montador e simulador, especialmente), tem a característica de evoluir de acordo com o andamento do projeto da máquina, exigindo uma constante manutenção de tais programas. Este caráter dinâmico requer a constante dedicação de uma equipe para manter atualizados tais programas, pois estes são essenciais para o andamento de todo o projeto. Esta dificuldade propricia a tendência a um divórcio entre os grupos de "hardware" e de "software" do projeto, já que muitas vezes as duas equipes não podem tolerar os atrasos simultâneos necessários às devidas atualizações, o que acarreta a elaboração de dois projetos, paralelos, ao invés de um único, integrado. Com o uso do SPD, estas necessidades são supridas por um conjunto de ferramentas com as seguintes atribuições:

2.a - Documentação: através de uma linguagem de alto nível (LD - Linguagem de Descrição) a equipe mantém sempre autalizada uma descrição formal da estrutura e funcionamento da máquina podendo-se inclusive dispor de descrições que a apresentem em diferentes graus de profundidade e detalhe, à disposição da equipe.

2.b - Automação da simulação: através do processamento da LD, o SPD constrói automaticamente um simulador para a máquina cuja descrição é fornecida, reduzindo a tarefa de gerar um novo simulador para a máquina a uma simples modificação na documentação formal, toda vez que for necessário efetuar alterações no funcionamento e/ou estrutura da máquina.

2.c - Automação da obtenção de montadores: analogamente, o SPD fornece meios para a geração automática de montadores permitindo, com um mínimo de esforço, adaptar uma descrição já existente às modificações impostas pela equipe do projeto, fornecendo assim a baixo custo, um novo montador para a máquina em desenvolvimento.

2.d - Infraestrutura para o desenvolvimento do "software" básico: com o conjunto montador-simulador, é possível à equipe efetuar o desenvolvimento do "software" básico já nos primeiros estágios do projeto da máquina, de modo que estejam disponíveis os programas de "software" básico, devidamente testados, assim que se obtiver o primeiro protótipo operante da máquina.

2.e - Ferramenta para o projeto e depuração de arquitetura: paralelamente à sua utilização no desenvolvimento de máquinas reais, o SPD pode ser utilizado na criação de novas arquiteturas, viabilizando o teste de novas idéias e concepções, sem a necessidade da construção de protótipos, efetuar observações no comportamento da nova máquina, e, a partir dos fatos observados, modificar seu projeto já nas primeiras etapas de sua concepção.

2.f - Sistematização do procedimento de projeto: sendo um sistema de caráter formal, o SPD impõe ao projetista algumas regras na elaboração do seu trabalho. Assim, técnicas de desenvolvimento "top-down", ou por refinamentos sucessivos, incorporam-se ao projeto de máquina. Esta ganha também uma documentação completa, precisa e centralizada, propiciando um rendimento alto à equipe de projeto, permitindo uma rápida realimentação dos resultados de mudanças e melhorando a qualidade final do produto. Além disto a integração da equipe de projeto, propiciada pela utilização do SPD, pode ser considerada como uma das principais vantagens que tal metodologia oferece.

2.g - Ferramenta de desenvolvimento: O SPD não é usado apenas no projeto de novas máquinas. Todo o desenvolvimento de "software" e do estudo da arquitetura de máquinas existentes, porém não disponíveis imediatamente, pode ser efetuado com o uso do SPD. A construção prévia de simuladores para diversas máquinas como recursos de "biblioteca" permite sua imediata utilização como componentes básicos de novos projetos, à disposição do projetista.

2.h - Ferramenta pedagógica: Uma das finalidades mais importantes do SPD, pelas suas próprias características, é o de suporte para o ensino de arquitetura, "software" básico, simulação, linguagens de programação, estruturas de dados, compiladores, documentação, avaliação de desemepnho de sistemas, microprogramação, etc. Pelo fato de que o SPD dispensa, até um certo ponto, o uso de equipamentos e componentes e montagens de circuitos em laboratório, exigindo apenas tempo de computador, torna-se muito conveniente para o ensino de inúmeros tópicos relacionados com computadores a baixo custo. Como as descrições da máquina podem, no SPD, ser efetuadas em níveis diversos, é possível utilizar o sistema, integralmente ou não, em vários cursos, com a vantagem de poder o aluno dedicar-se, em cada curso, aos assuntos mais relevantes apenas, sem a necessidade de explorar aspec

tos desnecessários em profundidade, fato que não ocorre no caso do uso, para o mesmo fim, de protótipos de laboratório.

3. DESCRIÇÃO DO SPD

Este sistema automático tem como finalidade gerar, a partir de informações fornecidas ao sistema por meio de uma programação em linguagem de alto nível específica, um ou mais blocos de programa denominados módulos com os quais o usuário pode efetuar a simulação da máquina descrita, a montagem de programas escritos em linguagem "assembler" definida pelo usuário, ou outras operações simulares.

Uma segunda linguagem, de controle, permite ao usuário efetuar o acionamento dos vários módulos definidos e de outras rotinas do sistema, responsáveis pelos recursos que o SPD coloca à disposição do usuário para o desenvolvimento e para a utilização dos seus módulos.

O sistema pode ser dividido em duas fases, que são executadas sucessivamente a saber (fig. 1):

3.a - Fase de geração: nesta fase, é efetuada a leitura da descrição dos módulos, descrição esta que é feita por meio da linguagem de descrição (LD). Este texto é fornecido ao compilador LD (gerador propriamente dito), que efetua a tradução dos módulos descritos no texto LD, para a linguagem ALGOL, cujo compilador é disponível no computador hospedeiro utilizado (o B6700) gerando ainda saídas impressas e arquivos de informações em memória de massa, para uso na fase de execução.

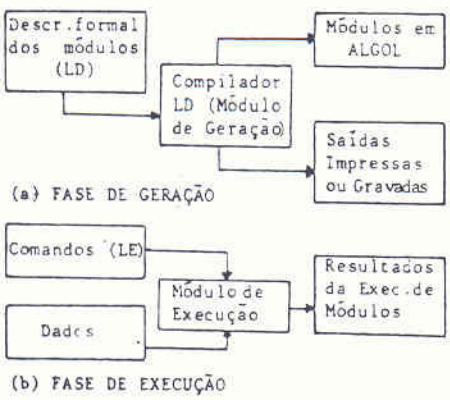

Fig. 1: Esquema geral do SPD

3.b - Fase de Execução: os módulos construídos pelo compilador na fase de geração são incorporados a programas já prontos, que são as rotinas de biblioteca do sistema e o supervisor de execução dos módulos. A execução deste último é regida, por sua vez, através de comandos de controle fornecidos pelo usuário (interativamente ou em "batch") por meio da linguagem de execução (LE).

Do ponto de vista do usuário, o SPD compõe-se, portanto, de duas linguagens de entrada: a LD (Linguagem de Descrição de Sistemas Digitais), através da qual descrições do "cross-software" desejado são fornecidas ao sistema, e que é uma metalinguagem dedicada ao especialista (projetista da máquina, ou um elemento que tenha conhecimentos de arquitetura e de técnicas de simulação) e a LE (Linguagem de Execução dos programas do SPD) dedicada ao usuário final dos programas construídos. Esta é apenas uma linguagem de comandos, de fácil utilização e baixa complexidade, o que não se dá com a LD, que exige um treinamento maior para a sua utilização.

Para processar tais linguagens de entrada, o SPD conta pois com dois programas: um um Compilador da Linguagem LD (o Módulo Geração) e um Interpretador da Linguagem LE (o Módulo de Execução).

3.1. Esquema da Utilização do SPD

A utilização do SPD (fig.2) inicia-se com a descrição do sistema digital que se deseja simular, em linguagem LD, a qual deve ser submetida ao compilador LD para análise e tradução. Uma vez livre de erros de sintaxe, o texto LD fornecido é convertido pelo compilador LD em um programa equivalente, em linguagem ALGOL do B6700. Incorpora-se em seguida o texto ALGOL assim obtido ao programa que implementa o interpretador LE (disponível em biblioteca), submetendo-se o conjunto ao compilador ALGOL do B6700, através do qual o conjunto é tranformado em um programa executável. O programa assim obtido é o simulador desejado, equipado com recursos que lhe permitem ser operado através de comandos LE. Assim, o usuário final do SPD tem condições de ativar o programa simulador obtido através do fornecimento de comandos LE convenientes; outros comandos LE podem ser utilizados para auxiliar eventuais tarefas de depuração. Como consequência, obtém-se como saídas do SPD resultados de simulação, informações de controle e relatórios solicitados pelo usuário.

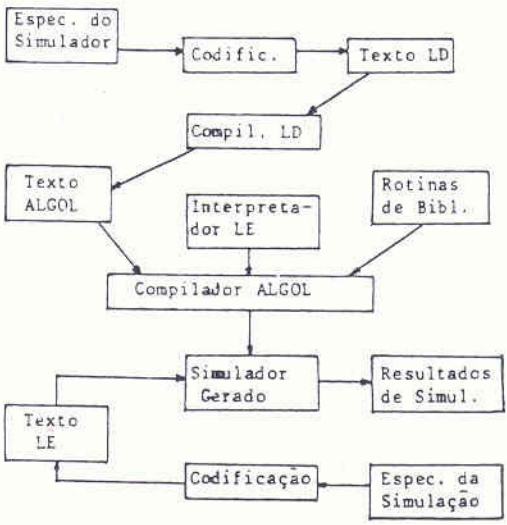

Fig. 2: Esquema do uso da SPD

3.2. Estrutura dos Programas Componentes do SPD

3.2.1. Compilador LD

O programa compilador propriamente dito (fig. 3) é a parte central do módulo de geração, sendo o bloco que efetua a tradução das descrições dos módulos, fornecidas por meio da linguagem de descrição (LD) ao sistema.

O programa de descrição dos módulos é separado em seus componentes pelo compilador, e analisado sintática e semanticamente pelo mesmo. Para a análise sintática, o compilador LD conta com o auxílio de um arquivo de especificação da linguagem de descrição, o qual contém todas as informações a respeito da sintaxe da mesma bem como a indicação das ações semânticas que devem ser executadas em cada situação.

O programa compilador efetua uma interpretação de produções de Floyd-Evans modificadas (contidas no arquivo de especificação da linguagem LD) sob a direção dos elementos básicos da linguagem de descrição, extraídos do texto fonte LD pelo seu analisador léxico.

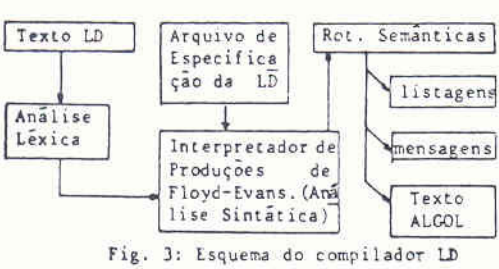

Fig. 3: Esquema do compilador LD

Eventuais erros de sintaxe ou incoerências semânticas detetadas durante a análise da descrição dos módulos são acusados, e mensagens neste sentido são fornecidas pelo compilador. Adicionalmente, e de acordo com especificações fornecidas pelo usuário, listagens de vários tipos são geradas pelo sistema.

3.2.2. Interpretador LE

O controle de execução do SPD é sempre iniciado pelo supervisor de execução, que é basicamente um interpretador de comandos, que forma o núcleo do módulo de execução (fig. 4).

Este supervisor interpreta comandos LE que lhe são fornecidos, promovendo a execução das operações solicitadas, entre as quais as mais importantes são a ativação de módulos e a ativação de rotinas de biblioteca (para a execução de tarefas auxiliares tais como "dumps" de memória, carregamento de dados em elementos de armazenamento pertencentes aos módulos, ou o controle de rastreamento de "pontos de teste" internos aos módulos).

Logo, o interpretador LE recebe do usuário uma seqüência de comandos (em LE), que instruem o sistema sobre as ações a serem executadas (preparação de dados, posicionamento de parâmetros do sistema, ações de simulação a serem tomadas) e produz para o usuário mensagens de erro (sintaxe, consistência), resultados de simulação, informações solicitadas pelo usuário em comandos LE, mensagens (de erro) de execução (especificadas em LD pelo usuário ou detetadas pelo sistema).

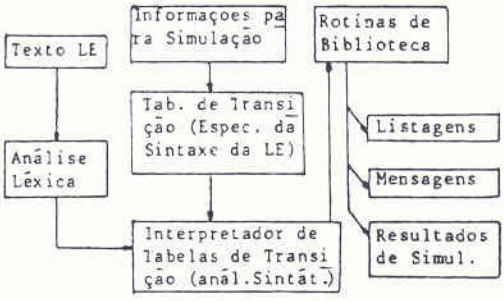

Fig. 4: Esquema do simulador gerado

3.3. As linguagens de entrada do SPD

A seguir, é feito um estudo paralelo dos recursos oferecidos pelas linguagens de entrada do SPD, enfatizando-se as peculiaridades e as relações existentes entre as mesmas. Como se sabe, a LD é uma linguagem de alto nível, com estrutura de blocos, apresentando declarações e comandos executáveis, enquanto a LE é composta apenas de comandos de controle, destinados a promover a execução de funções do SPD, quer programadas em LD pelo usuário, quer oferecidas pelo sistema.

Ambas as linguagens oferecem formato livre ao programador, e suas instruções devem ser separadas por ponto e vírgula (;), sendo identificadas pela presença obrigatória de palavras reservadas.

A estrutura de blocos da LD é caracterizada pela ocorrência de regiões envolvidas pelo par de palavras reservadas BEGIN-END. Nestas regiões, é possível utilizar quaisquer identificadores que tenham sido declarados em seu interior. É possível definir dessa maneira: programa, blocos com ou sem nome, funções e procedimentos, todos com a conotação habitual utilizada em linguagens de programação, bem como os módulos, que são blocos especiais que definem entidades acessíveis em tempo de execução através dos seguintes comandos da LE: comando RUN (para iniciar a execução do módulo), comando ABORT (para descontinuá-lo), comando CONTINUE (para prosseguir a execução de um módulo interrompido). Dois outros comandos da LE atuam sobre os módulos, mas têm sua ação estendida aos blocos com nome, procedimentos e funções, de maneira similar: os comandos ENABLE e DISABLE, que permitem autorizar ou inibir, respectivamente, a passagem do controle de execução para a LE ao ser atingido o final do bloco. Estes dois comandos atuam também sobre pontos da LD definidos como CHECKPOINTS controlando a passagem da execução por tais pontos, e sobre os EVENTOS declarados na LD, permitindo monitorar e controlar sua ocorrência.

É importante notar que, qualquer que seja a classe de elementos a que se deseja ter acesso na LE, é necessário que este seja explicitamente declarado como acessível na LE atra-

vés da sua declaração como EXTERNAL em LD.

Com estes recursos, é possível controlar a execução dos programas do SPD. No entanto, sua finalidade principal é a de manipular elementos de armazenamento declaráveis em LD: flip-flops, registradores, memórias, pilhas, filas, tabelas e arquivos. Para cada uma dessas classes, existem na LE três tipos de comandos: um responsável por seu preenchimento (LOAD), outro pela sua inicialização (CLEAR) e outro pela sua apresentação (DUMP), totais ou seletivos. Na LD, sua manipulação ocorre especialmente no comando MOVE, responsável pelo preenchimento de um componente com o resultado de uma expressão. Comandos auxiliares da LD permitem manipulações mais restritas (comandos SET, CLEAR, COMPLEMENT, INCREMENT, DECREMENT), ou acessos a tabelas (comandos FIND, INSERT, REMOVE, REPLACE). Adicionalmente, a LD oferece comandos simples de entrada e saída (comandos READ e PRINT), que operam sobre as variáveis e matrizes da LD (não acessíveis em LE), um conjunto de comandos de controle de fluxo (comandos REPEAT, IF, CASE, EXIT), que permitem controlar a lógica dos algoritmos escritos em LD. Os procedimentos podem ser chamados com ou sem parâmetros através do comando EXECUTE, e as funções, através de seu uso em expressões. Complementando a LD, é possível definir eventos assíncronos (monitoráveis pela LE) e programar sua ocorrência através dos comandos CAUSE, SET,CLEAR, EBABLE e DISABLE. O tempo simulado é controlado através do comando DELAY, que também promove todo o controle de ocorrência dos eventos na LD.

A LE possui uma série de comandos adicionais que facilitam o usuário na manipulação dos elementos de armazenamento da LD em tempo de execução: são os comandos COMPLEMENT, DECREMENT, INCREMENT, SET e RESET (que operam sobre registradores), os comandos PUSH e POP (para pilhas), e o comando REWIND (para arquivos).

Como recursos de documentação impressa dos resultados de execução do SPD, a LE dispõe de comandos que permitem comentar o próprio texto LE (COMMENT), ou as saídas do SPD (comando MESSAGE), manipular variáveis de execução não acessíveis à LD (comando LET) e imprimí-las (comando DISPLAY), ligar e desligar listagens (comando LIST), controlar espaçamento de impressão (comandos PAGE e SPACE), controlar a base de numeração a ser usada na LE (comando BASE), ligar e desligar o rastreamento dos comandos do programa LD (comando TRACE) e de alterações do conteúdo de elementos de armazenamento da LD, selecionáveis (comando MONITOR). São disponíveis além disto comandos que facilitam a depuração do próprio SPD (comandos ACCUMULATORS, CONTROLS, TOKENS, TRANSITIONS) e dos módulos criado (comandos SAVE, RESTORE e REMOVE, através dos quais é possível salvar e recuperar estados particulares do processamento, e remover o "backup" criado em tempo de execução). Além disto, a LE pode operar em modo programado, oferecendo para isto alguns recursos simples de programação: o comando LABEL, para a definição dos rótulos que podem ser utilizados em comandos de desvio (Comando incondicional GOTO, e desvio para subrotina GOSUB). O retorno de subrotina é feito através do comando RETURN. Um comando condicional (IF) permite testar o valor de uma expressão booleana da LE ou então a causa da última passagem do controle de execução para a LE. Caso a condição testada seja verdadeira, um comando de quebra de seqüência de execução (GOTO, GOSUB, RUN, ABORT, CONTINUE ou RETURN) é executado. Finalmente, o comando END indica o final físico de um texto LE. Com estes recursos oferecidos pelas linguagens de entrada, o projetista fica capacitado a definir e utilizar seus programas com muita comodidade, tornando-os operacionais em curto espaço de tempo, graças às ferramentas de desenvolvimento e de depuração disponíveis, extensivas também aos programas desenvolvidos, o que favorece sobremaneira o usuário final do sistema.

4. PERSPECTIVAS DE EXPANSÃO DO SPD

O projeto SPD está completo e implementado, encontrando-se atualmente em fase de expansão. Está, adicionalmente, em fase avançada de implementação, um conjunto de programas que compõem o substrato de um compilador de compiladores a ser incorporado ao SPD[6,7]. É prevista, para isso, uma extensão das linguagens de entrada, visando-se adapatá-la para esta finalidade. Uma alteração adicional está prevista, no sentido de incluir no SPD um conjunto de recursos voltados para o teste automático de consistência estrutural das arquiteturas descritas em LD. Uma terceira modificação de monta no SPD está prevista para quando forem introduzidos, no sistema, recursos de coleta e análise de dados estatísticos sobre o desempenho do próprio SPD, das arquiteturas e dos montadores e compiladores cujas descrições lhe são fornecidas.

5. CONCLUSÕES

Os objetivos citados anteriormente foram plenamente alcançados na atual versão do SPD, todos eles voltados para a aplicação do SPD como ferramenta de projeto, tendo sido observados alguns resultados relevantes na utilização da mesma:

5.a - Redução substancial do tempo de desenvolvimento do software de apoio (simuladores e montadores): o tempo necessário para descrever uma máquina do porte de um microprocessador usual é de algumas horas apenas, permitindo à equipe de projeto programar em linguagem de montagem e testar seus programas quase imediatamente.

5.b - Redução do texto fonte: com o uso de linguagens de programação de alto nível não específicas o texto necessário para construir os programas de apoio é muitas vezes mais longo que o texto correspondente em LD.

5.c - Aumento da clareza do texto fonte - Por meio de uma linguagem dedicada como a LD, é possível representar idéias com maior precisão e de maneira mais concisa, tornando mais simples sua compreensão.

5.d - Facilidade de utilização: constatou-se, através do uso do SPD por parte de elementos não familiarizados que suas linguagens são de muito fácil aprendizado, produzindo descrições compactas e completas das arquiteturas ensaiadas, bem como permitindo seu uso através de procedimentos de execução muito simples.

5.e - Eficácia como ferramenta didática: verificou-se, através de usuários nao especialistas em simulação e arquitetura de computador, os quais utilizaram experimentalmente o SPD, sua utilidade, como ferramenta eficaz de ensino de engenharia de sistemas digitais, arquitetura, microprocessadores, sistemas digitais, etc.

6. REFERÊNCIAS

[1] Cunha Jr., N.M, "Automação de Projeto de Sistemas Digitais - Partição de Circuitos Lógicos", Dissertação de Mestrado, EPUSP, 1976.

[2] Ferreira, M.A.G.V., "Sistema de Programação 'Cross-Montador' para um Minicomputador", Dissertação de Mestrado, EPUSP, 1974.

[3] José Neto, J., "Aspectos do projeto do software de um minicomputador", Dissertação de Mestrado, EPUSP, 1975.

[4] José Neto, J., "SPD - Um sistema de Suporte para o projeto e desenvolvimento de Sistemas Digitais", Publicação Interna nº 2 do Departamento de Engenharia de Eletricidade da EPUSP, 1978.

[5] José Neto, J., "SPD - Um sistema automático de Apoio ao Projeto e Desenvolvimento de Sistemas Digitais", Tese de Doutorado, EPUSP, 1980.

[6] José Neto, J. e Magalhães, M.E.S. "Um Gerador Automático de Reconhecedores Sintáticos para o SPD", VIII Semish, Florianópolis, 1981.

[7] José Neto, J. e Magalhães, M.E.S., "Reconhecedores Sintáticos - Uma alternativa Didática para o uso em cursos de engenharia", XIV Congresso Nacional de Informática, São Paulo, 1981.

[8] Martini, J.S.C., "Automação do Projeto de Sistemas Digitais - Rotas de Circuitos Impressos", Dissertação de Mestrado, EPUSP, 1975.

[9] Massola, A.M.A., "Simulação em Nível de Portas Lógicas", Tese de Doutoramento, EPUSP, 1974.

[10] Massola, A.M.A., "Automação do Projeto de Sistemas Digitais - Arquivo Central", Tese de Livre Docência, EPUSP 1976.

[11] Melnikoff, S.S.S., "Automação do Projeto de Sistemas Digitais - Posicionamento de Componentes", Dissertação de Mestrado, EPUSP, 1977.