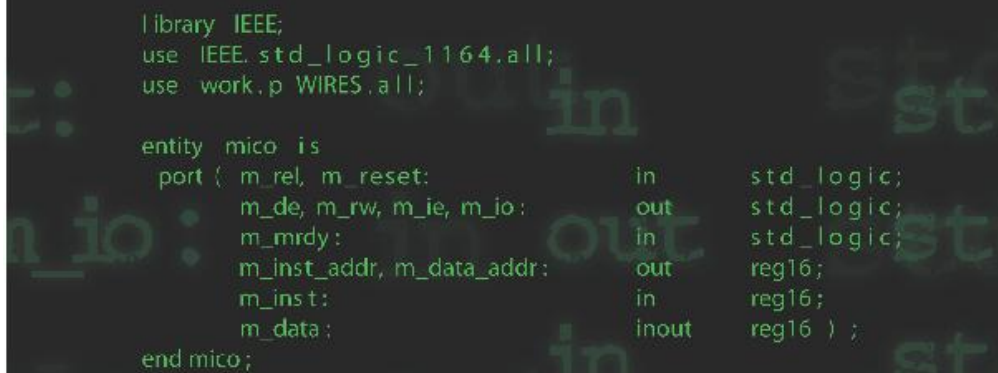
Sistemas Digitais e Microprocessadores

Roberto A. Hexsel



Editora UFPR

Sistemas Digitais e Microprocessadores

Roberto A Hexsel[1]

Departamento de Informática
Universidade Federal do Paraná

14 de junho de 2015

Sumário

Prefácio

Sistemas digitais e microprocessadores são, em larga medida, ubíquos em ambientes domésticos, profissionais e de lazer. Por exemplo, numa sala de estar típica de classe média (em 2012) há microprocessadores na televisão e em seu controle remoto, vários processadores digitais de sinais e microprocessadores no aparelho de DVD e em seu controle remoto, um ou mais no aparelho de *home theater* e em seu controle remoto. Dependendo do nível de sofisticação do automóvel, há ao menos um microprocessador em cada um dos freios ABS, painel de instrumentos, injeção eletrônica, um em cada *air bag*, rádio ou toca-CD/DVD/MP3. Na cozinha e área de serviço encontram-se microprocessadores no forno de micro-ondas, na geladeira, na lavadora de roupas, no condicionador de ar e em seu controle remoto. Quanto mais sofisticado é o aparelho de telefonia móvel, maior o número de microprocessadores: um para controlar o teclado, tela, e aplicativos 'simples' como agenda e calendário, um para controlar as funções de rádio, um ou mais para a câmera e tratamento de imagens, um para controlar o acesso à Internet e processar os aplicativos sofisticados.

O computador pessoal de mesa em que este texto está sendo editado contém uma unidade de processamento com dois processadores (*cores*), um microprocessador de 8 bits em cada uma das interfaces USB, no *mouse*, no teclado, ao menos um microprocessador de 32 bits nos dispositivos mais sofisticados como monitor, gravador de DVDs, interface de rede, interface de vídeo, disco rígido, e *modem* ADSL. Um único equipamento contém, ao menos, 16 microprocessadores de variadas capacidades de processamento e custo.

Exceto pela unidade de processamento do computador pessoal, todos os outros sistemas digitais mencionados são considerados "sistemas embarcados" ou "sistemas embutidos" (*embedded systems*). Para todos os sistemas mencionados, foi necessário desenvolver *software* do nível de aplicação e de sistema, e a maior parte do código é 'novo' para cada aplicação e/ou dispositivo. Portanto, comparando-se o número de linhas de código fonte 'novo' escritas para os sistemas mencionados acima, com o código fonte 'novo' escrito para os programas de aplicação do computador pessoal, a proporção é da ordem de 5-10 para 1, numa estimativa conservadora. Isso significa que as oportunidades abertas ao

projetista de *software* embarcado são algo maiores e mais interessantes do que aquelas no mercado de aplicativos para sistemas *desktop*.

Este texto contém material que é fundamental para quem pretende trabalhar no projeto e desenvolvimento de sistemas digitais e de sistemas embarcados, tanto no desenvolvimento de circuitos e sistemas de *hardware*, quanto no desenvolvimento de *software* de aplicação e de sistemas operacionais, embora não trate explicitamente destes sistemas.

O conteúdo e a apresentação deste texto são o resultado das experiências do autor ao lecionar a disciplina de Projetos Digitais e Microprocessadores no Bacharelado em Ciência da Computação da Universidade Federal do Paraná. O autor pressupõe que o leitor já teve contato com os fundamentos de lógica proposicional, circuitos combinacionais e sequenciais, em textos tais como Mano & Kime [Man02] e Katz & Borriello [KB04], bem como com uma linguagem de programação em alto nível como C. Para a compreensão do material não são necessários conhecimentos de eletromagnetismo nem de eletrônica para além daqueles adquiridos em um bom curso de Física no Ensino Médio.

Uma característica importante do texto é a ênfase na interface entre o *hardware* e o *software*, porque o texto é uma ferramenta para a formação de cientistas da computação, analistas e programadores, para quem o *hardware* é uma plataforma de trabalho, mais do que um fim em si mesmo. Portanto, o Capítulo 5 descreve cuidadosamente o conjunto de instruções do processador Mico, antes de descrever uma implementação daquele conjunto. O conjunto de instruções é a *application programming interface* (API) do processador, e este é uma implementação daquela API. O Capítulo 6 discute dois aspectos importantes na programação em C, que são o acesso a estruturas de dados e o suporte a funções, com ênfase no mapeamento entre as abstrações da linguagem C e suas traduções para o *assembly* do Mico. O Capítulo 8 introduz quatro periféricos simples como sendo interfaces entre o processador e o mundo externo àquele. O capítulo propõe um sistema de interrupções, tanto do ponto de vista do *hardware* como do *software*, e para as interfaces serial e paralela, apresenta os periféricos propriamente ditos, sua ligação ao processador, e mostra versões primitivas para os *drivers* destes dispositivos.

Esta forma de exposição do material enfatiza as APIs entre distintos níveis de abstração – sistema digital, código *assembly*, código C, componentes do sistema operacional – porque tal enfoque determina uma simetria entre distintas visões, aquela do projetista dos circuitos e sistemas digitais, e aquela do programador que se utiliza daqueles sistemas. Com esta apresentação, o estudante é colocado alternadamente na posição de quem implementa uma API, e na posição de quem a utiliza, e esta simetria é extremamente útil do ponto de vista didático [HC06].

Com a popularização das linguagens de descrição de *hardware* (LDH), tais como VHDL e Verilog, e de ferramentas para a simulação e síntese de circuitos descritos nestas linguagens, tornou-se viável estudar as disciplinas de projeto de

sistemas digitais inteiramente *in abstracto*, com os projetos desenvolvidos somente em termos de descrições em uma LDH, com todos os circuitos abstraídos para sinais e processos. Na experiência do autor, e do ponto de vista didático, esta abordagem só é efetiva se o estudante possui uma sólida compreensão do funcionamento concreto de circuitos digitais, e de seu comportamento temporal. Sem a compreensão do concreto, a visão abstrata pode tornar-se demasiado distante da realidade. Frequentemente, as consequências deste distanciamento são projetos irrealizáveis.

A organização dos conteúdos obedece a sequência usual no estudo de Sistemas Digitais: uma abstração do comportamento dos sinais elétricos baseada em Álgebra Booleana, circuitos combinacionais, e circuitos sequenciais. Circuitos de memória são introduzidos a seguir e então o projeto e a implementação do microprocessador, seguidos por uma introdução à interface entre linguagem de alto nível e o *hardware* que a executa. São estudados vários projetos de barramentos, em ordem crescente de desempenho e complexidade, e são apresentados quatro periféricos simples, frequentemente encontrados em sistemas embarcados. O último capítulo apresenta um pequeno subconjunto da linguagem VHDL como veículo para a modelagem do microprocessador Mico. Os conteúdos de cada capítulo são apresentados no que se segue.

O Capítulo 1 introduz a abstração de *sinais como bits*, introduz operações sobre bits, define suas propriedades, e introduz um formalismo para especificar circuitos, inspirado na Linguagem Z. Segue-se a descrição e especificação formal de três circuitos combinacionais básicos: multiplexadores, demultiplexadores e seletores. Exemplos de circuitos combinacionais complexos ilustram o uso de componentes da família TTL-74xx. O capítulo se encerra com a descrição, no nível funcional, de circuitos de tecnologia CMOS. Outros exemplos de circuitos combinacionais complexos – três projetos de somadores e uma Unidade de Lógica e Aritmética – são discutidos no Capítulo 9, no contexto da modelagem em VHDL de componentes de processadores.

Circuitos Sequenciais são investigados no Capítulo 3, com uma revisão sobre o comportamento de circuitos com memória, com ênfase em *flip-flops*. O comportamento de vários contadores e registradores de deslocamento é formalizado, empregando-se como exemplo componentes da família TTL-74xx. São então apresentados vários conceitos relacionados à temporização de circuitos sequenciais síncronos e discute-se o projeto de máquinas de estados finitas. Tomando como base o projeto de uma máquina de vender chocolates, são apresentadas duas implementações para o seu controlador, sendo que na segunda emprega-se a técnica de implementação com microcontroladores. Dentre os exemplos complexos discutidos no final do capítulo destaca-se o bloco de registradores que é empregado no processador.

O Capítulo 4 introduz circuitos de memória, apresentando de forma relativamente abstrata o projeto de circuitos de memória de grande capacidade. A

interface destes circuitos com o mundo externo a eles é discutida num nível suficiente de detalhe para facilitar a compreensão do material do Capítulo 5. Este tópico, a interface entre memória e processador, é retomado e expandido no Capítulo 7.

O microprocessador Mico é apresentado no Capítulo 5. Este processador é uma adaptação para 16 bits do MIPS R3000 descrito em Patterson & Hennessy [PH09]. O conjunto de instruções do Mico é precisamente definido, e uma implementação daquele conjunto é apresentada em detalhe. Esta implementação emprega uma disciplina de temporização na qual cada instrução é executada num certo número de ciclos (curtos) de relógio, que é chamada de "disciplina multiciclos". O Capítulo 9 apresenta um modelo VHDL do processador projetado com a "disciplina de temporização ciclo longo", sob a qual cada instrução é executada em um único ciclo (longo) de relógio. Este capítulo ainda introduz duas técnicas de projeto para as interfaces do processador com memória e periféricos.

O Capítulo 6 faz a ligação de dois níveis de abstração da programação, (i) entre programas escritos na linguagem de alto nível C, e (ii) sua tradução para a linguagem de montagem (*assembly*) do Mico. Discute-se a relação entre o endereçamento implícito de componentes de estruturas de dados em C, e o endereçamento que deve ser explicitado no código em *assembly*. Um protocolo para o código que permite a implementação de funções é apresentado. O capítulo se encerra com o estudo sobre o comportamento do programa recursivo que computa o fatorial.

Com base na descrição do processador, versões mais sofisticadas para a interface entre o sistema de memória e o processador são introduzidas no Capítulo 7. São apresentados projetos de barramentos com níveis crescentes de concorrência entre as transferências, portanto com melhor desempenho e maior complexidade.

O Capítulo 8 introduz os periféricos mais simples que são encontrados em sistemas de pequeno porte, tais como interfaces paralela e serial, contadores e conversores analógico-digitais. Um sistema realista de sinalização e tratamento de interrupções é apresentado, com exemplos simplificados de tratadores de interrupções das interfaces paralela e serial.

A modelagem de sistemas digitais em VHDL é introduzida no Capítulo 9. Um subconjunto relativamente pequeno das construções da linguagem é apresentado como um efeito colateral da discussão sobre modelos de circuitos realistas, tais como somadores rápidos com cadeia de adiantamento de vai-um, e do circuito de controle de fluxo de execução do Mico, que compreende o contador de programa e o circuito de decodificação das instruções de saltos e desvios. O Apêndice contém o código dos programas de teste dos circuitos detectores de bordas – que são o veículo para a discussão sobre modelagem de tempo –, e do modelo funcional do Mico.

Os exercícios não são agrupados ao final dos capítulos, mas estão distribuídos ao longo das seções porque eles são uma parte importante do texto e foram escolhidos para reforçar a compreensão do material bem como expandir os conteúdos e ideias apresentadas.

Agradecimentos

O autor agradece aos alunos da disciplina Projetos Digitais de Microprocessadores, das turmas de 1997 a 2004, e após 2010, que foram, ao mesmo tempo, a inspiração para este texto, e as cobaias de sua aplicação em Projetos. Agradeço também aos alunos das turmas de Organização e Arquitetura de Computadores, de 1995 a 2011, por propiciarem a oportunidade, o aprendizado, e a experiência de tentar expandir o que aqueles haviam apreendido em Projetos. É o desejo do autor que eventuais danos que lhes possa ter infligido nestas disciplinas sejam perdoados com o passar dos anos.

As aulas espetacularmente claras de Anatólio Laschuk, na Escola de Engenharia da UFRGS, serviram de inspiração para este trabalho – espero lhes ter feito justiça. Com Jeff Sanders aprendi o valor de especificar clara e precisamente o que se tenta projetar e implementar.

Agradecimentos são devidos aos colegas do Departamento de Informática da UFPR, pelo ambiente de trabalho cordial e produtivo ao longo destes anos, em especial, aos colegas da área de Sistemas: Aldri L. dos Santos, André L. P. Guedes, Armando L. N. Delgado, Bruno Müller Jr, Cristina D. Murta, Eduardo Todt, Elias P. Duarte Jr, Fabiano Silva, Heinz A. Niederheitmann Jr, Hélio Pedrini, Heraldo F. Madeira, Luis Allan Kunzle, Luis Carlos E. de Bona, Luiz Carlos P. Albini, Michele N. Lima, Nelson Suga, Renato J. S. Carmo, e Wagner N. Zola.

A forma de apresentação da abstração de sinais como bits resultou de uma sugestão de Renato Carmo. Maria Cristina Périgo fez um excelente trabalho ao revisar o texto. Evidentemente, todos os erros remanescentes são de responsabilidade do autor.

Referências Bibliográficas

[Ash96] Peter J Ashenden. *The Designer's Guide to VHDL*. Morgan Kaufmann, 1996. ISBN 1558602704.

[Bac86] Maurice J Bach. *The Design of the UNIX Operating System*. Prentice-Hall, 1986. ISBN 0132017997.

[Bal03] Mark Balch. *Complete Digital Design*. McGraw-Hill, 2003. ISBN 0071409270.

[BJ97] Gerrit A Blaauw and Frederick P Brooks Jr. *Computer Architecture: Concepts and Evolution*. Addison-Wesley, 1997. ISBN 0201105578.

[Bob87] Leonard S Bobrow. *Elementary Linear Circuit Analysis*. Holt, Rinehart & Winston, 1987. ISBN 0030072980.

[Bro04] Stuart Brorson. Circuit simulation using gEDA and SPICE – HOWTO. Páginas html, Electroniscript, inc., 2004. Disponível em `<http://www.brorson.com/gEDA/SPICE/intro.html>`. Acesso em 8/5/2012.

[Byb12] Anthony Bybell. GTKWave – visualization tool for VCD, LXT, LXT2, VZT, FST, and GHW files, 2012. Disponível em `<http://gtkwave.sourceforge.net>`. Acesso em 11/7/2012.

[CH05] Tadeu Carmona and Roberto A Hexsel. *Universidade Redes*. Editora Digerati, 2005. ISBN 8589535797.

[Cla80] Wesley A Clark. From electron mobility to logical structure: A view of integrated circuits. *ACM Computing Surveys*, 12(3):325–356, Sep 1980.

[CM06] Ney Calazans and Fernando Moraes. Simulação VHDL do processador MRStd. Páginas html, Faculdade de Informática, PUC-RS, 2006. Disponível em `<http://www.inf.pucrs.br/~calazans/undergrad/orgcomp_EC/mips_mono/mips_v0.vhd>`. Acesso em 8/5/2012.

[Fle97] William I Fletcher. *An Engineering Approach to Digital Design*. Prentice Hall, 1997. ISBN 9780132776998.

[Gin12] Tristan Gingold. GHDL – G Hardware Design Language, 2012. Disponível em `<http://ghdl.free.fr>`. Acesso em 11/7/2012.

[HC06] Roberto A Hexsel and Renato Carmo. Ensino de Arquitetura de Computadores com enfoque na interface hardware/software. In *WEAC'06: Workshop sobre Educação em Arquitetura de Computadores*, pages 9–16, 2006.

[Hex01] Roberto A Hexsel. Redes de dados: Tecnologia e programação. Relatório Técnico RT-DINF 005/2001, Depto. de Informática, UFPR, 2001. Disponível em `<http://www.inf.ufpr.br/roberto/rt005_01.pdf>`. Acesso em 8/5/2012.

[HP90] John L Hennessy and David A Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 1st edition, 1990. ISBN 1558600698.

[HP12] John L Hennessy and David A Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 5th edition, 2012. ISBN 9780123838728.

[HU79] John E Hopcroft and Jeffrey D Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979. ISBN 020102988X.

[JNW08] B L Jacob, S W Ng, and D T Wang. *Memory Systems: Cache, DRAM, Disk*. Morgan Kaufmann, 2008. ISBN 0123797513.

[Kat94] Randy H Katz. *Contemporary Logic Design*. Benjamin-Cummings, 1994. ISBN 0805327037.

[KB04] Randy H Katz and Gaetano Borriello. *Contemporary Logic Design*. Prentice Hall, 2004. ISBN 978-0201308570.

[KL96] Sung-Mo Kang and Yusuf Leblebici. *CMOS Digital Integrated Circuits: Analysis and Design*. McGraw-Hill, 1996. ISBN 0070380465.

[Koh78] Zvi Kohavi. *Switching and Finite Automata Theory*. Tata McGraw-Hill, 2nd edition, 1978. ISBN 0070993874.

[Kor01] Israel Koren. *Computer Arithmetic Algorithms*. A K Peters, 2nd edition, 2001. ISBN 1568811608.

[KR88] Brian W Kernighan and Dennis M Ritchie. *The C Programming Language*. Prentice Hall, 2nd edition, 1988. ISBN 9780131103628.

[Man02] M Morris Mano. *Digital Design*. Prentice Hall, 3rd edition, 2002. ISBN 01306211218.

[MC80] Carver Mead and Lynn Conway. *Introduction to VLSI Systems*. Addison-Wesley, 1980. ISBN 0201043583.

[MK00] M Morris Mano and Charles R Kime. *Logic and Computer Design Fundamentals*. Prentice Hall, 2nd edition, 2000. ISBN 0130124680.

[Mou01] Arnaldo V Moura. *Especificações em Z*. Editora da Unicamp, 2001. ISBN 8526805754.

[PD11] Larry L Peterson and Bruce S Davie. *Computer Networks: A Systems Approach*. Morgan Kaufmann, 5th edition, 2011. ISBN 9780123850591.

[Per93] Douglas L Perry. *VHDL*. McGraw-Hill, 2nd edition, 1993. ISBN 0070494347.

[Per02] Douglas L Perry. *VHDL: Programming By Example*. McGraw-Hill, 4th edition, 2002. ISBN 0071400701.

[PH05] David A Patterson and John L Hennessy. *Computer Organization & Design: The Hardware/Software Interface*. Morgan Kaufmann, 3rd edition, 2005. ISBN 1558606041.

[PH09] David A Patterson and John L Hennessy. *Computer Organization & Design: The Hardware/Software Interface*. Morgan Kaufmann, 4th edition, 2009. ISBN 9780123744937.

[PT96] David Pellerin and Douglas Taylor. *VHDL Made Easy!* Prentice Hall, 1996. ISBN 0136507638.

[RCN03] Jan M Rabaey, Anantha Chandrakasan, and Borivoje Nikolic. *Digital Integrated Circuits – A Design Perspective*. Prentice Hall, 2nd edition, 2003. ISBN 0130909963.

[S+99] Richard Stallman et al. GCC, the GNU Compiler Collection. Páginas html, Free Software Foundation, 1999. Disponível em `<http://gcc.gnu.org>`. Acesso em 28/7/2013.

[San90] Jeff W Sanders. Lectures on the foundations of hardware design. Lecture notes, Programming Research Group, Oxford University Computing Laboratory, 1990.

[SCO96] The Santa Cruz Operation SCO. *System V Application Binary Interface – MIPS RISC Supplement*. 3rd edition, 1996.

[Sha96] Tom Shanley. *Pentium Pro Processor System Architecture*. MindShare Addison-Wesley, 1996. ISBN 0201479532.

[Spi89] J M Spivey. *The Z Notation*. Prentice Hall, 1989. ISBN 013983768X.

[SS90] Adel S Sedra and Kenneth C Smith. *Microeletronic Circuits*. Holt, Rinehart & Winston, 3rd edition, 1990. ISBN 003051648X.

[Swe07] Dominic Sweetman. *See MIPS Run – Linux*. Morgan Kaufmann, 2nd edition, 2007. ISBN 0120884216.

[Tan92] Andrew S Tanenbaum. *Modern Operating Systems*. Prentice-Hall, 1992. ISBN 0135881870.

[Tan99] Andrew S Tanenbaum. *Organização Estruturada de Computadores*. Livros Técnicos e Científicos, 4th edition, 1999. ISBN 8521612532.

[TS89] Herbert Taub and Donald Schilling. *Digital Integrated Electronics*. McGraw-Hill, 1989. ISBN 007Y857881.

[WE92] Neil H E Weste and Kamran Eshraghian. *Principles of CMOS VLSI Design: A Systems Perspective*. Addison-Wesley, 1992. ISBN 0201533766.

[WH10] Neil Weste and David Harris. *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison-Wesley, 4th edition, 2010. ISBN 0321547748.

Índice Remissivo