Universidade de São Paulo

Departamento de Engenharia de Computação e Sistemas Digitais

PCS

# Introdução a HDL

## Aula 01 - *Hardware Description Languages*

Bruno de Carvalho Albertini

PCS - Departamento de Engenharia de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo

Março, 2020

pcs.usp.br




Olá a todos, sejam bem vindos ao curso de HDL. Eu sou o Prof. Bruno Albertini, do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo. Vamos começar dando uma olhada no que é e por que utilizamos as HDLs.

# Projetando circuitos (pré-HDL)

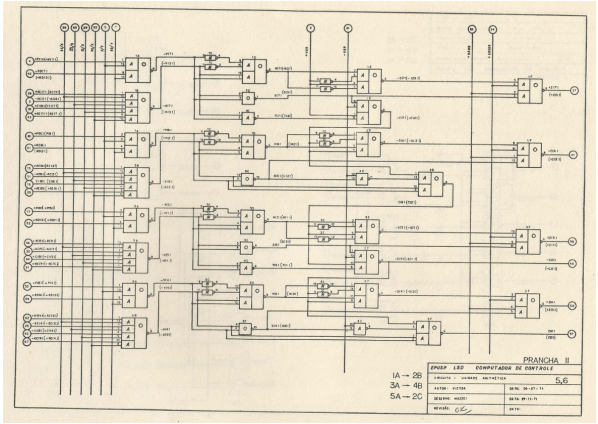



Figura: ULA do Patinho Feio (fonte: dissertação Fregni)





Projetando circuitos (pré-HDL)

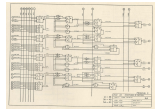


Figura: ULA do Patinho Feio (fonte: dissertação Fregni)

Antigamente, e eu estou falando de década de 70, 80, os circuitos digitais eram projetados usando diagramas de blocos, como esse da figura que você está vendo. Esse da figura é uma Unidade Aritmética do Patinho Feio, o primeiro computador desenvolvido na USP. Esse desenho especificamente foi parte da dissertação de mestrado do Prof. Edson Fregni, aqui da Poli, em 1972, e foi desenhado a mão!

# Projetando circuitos (pré-HDL)

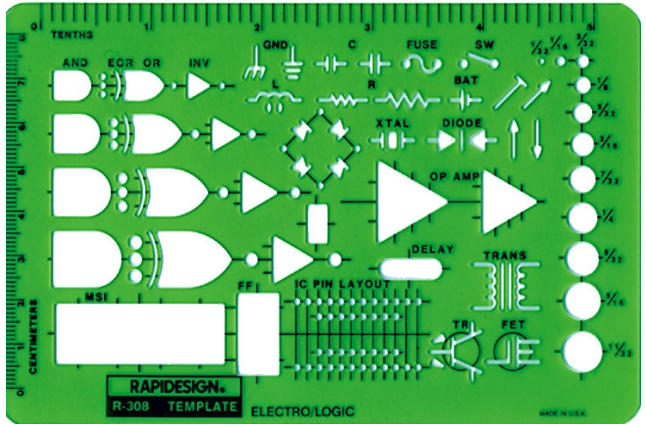



Figura: Regua de projeto (fonte: fabricante)





Projetando circuitos (pré-HDL)

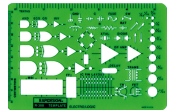


Figura: Regua de projeto (fonte: fabricante)

Os projetistas usavam estes estênceis, também chamado de gabarito, que é um tipo de régua onde você coloca sua caneta nanquim ou lápis dentro do símbolo desejado e segue o padrão, deixando os desenhos bonitinhos.

# Projetando circuitos (pré-HDL)

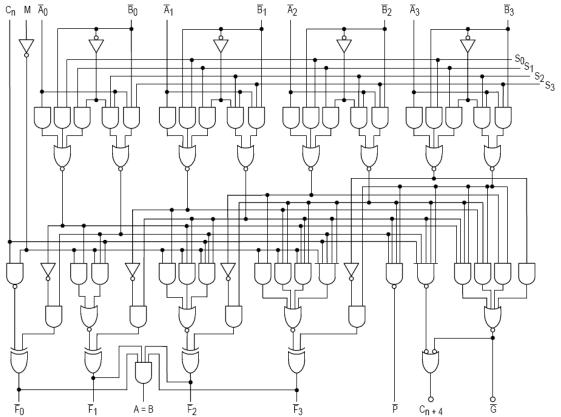



Figura: ULA 74181 (fonte: Texas)





Na década de 80 surgiram os primeiros softwares CADs, que podem ser usados para desenhar circuitos usando um computador. No entanto, o projeto ainda era feito usando desenhos. Esse da figura é uma Unidade Lógica e Aritmética de 4 bits bastante conhecida e utilizada nas décadas de 70 e 80. Ela foi meio que abandonada na década de 90 pois a complexidade dos cálculos exigia uma Unidade Lógica e Aritmética mais poderosa. Nessa época, os circuitos estavam ficando mais baratos, então os projetos creceram bastante em complexidade. Mas como fazer projetos mais complexos?

# Projetando circuitos (pré-HDL)

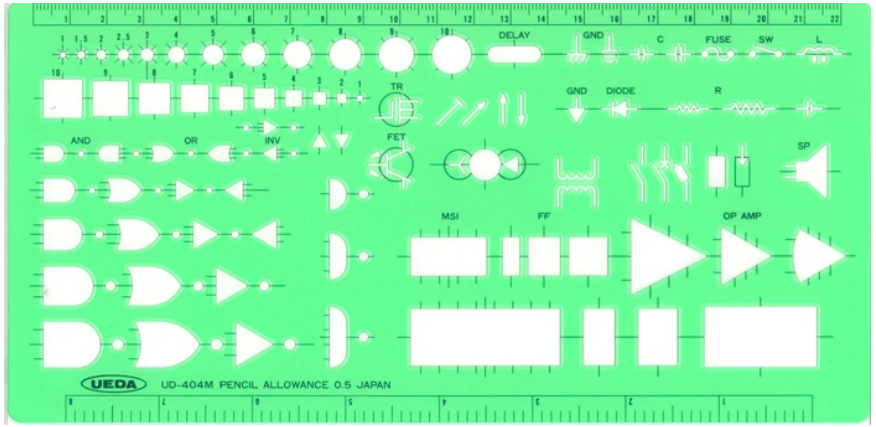
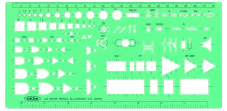



Figura: Regúa de projeto (fonte: fabricante)





Projetando circuitos (pré-HDL)



Dá pra comprar uma reguinha mais complexa [rs] mas obviamente esta solução não é muito eficiente. Na década de 90 surgiram também os dispositivos programáveis, o que tornava o projeto ainda mais complicado com desenho em papel. Com CAD até que dava, mas começaram a aparecer projetos com mais de 1 milhão de componentes, o que é claro que é inviável mesmo com o super gabarito aí da figura. A solução foi tornar a descrição de projeto programática, e aí entram...

# Projetando circuitos (pós-HDL)

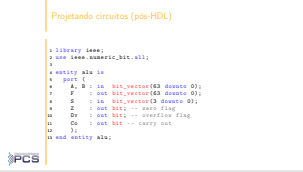


```vhdl
library ieee;
use ieee.numeric_bit.all;

entity alu is
  port (
    A, B : in  bit_vector(63 downto 0);
    F    : out bit_vector(63 downto 0);
    S    : in  bit_vector(3 downto 0);
    Z    : out bit; -- zero flag
    Ov   : out bit; -- overflow flag
    Co   : out bit -- carry out
    );
end entity alu;
```







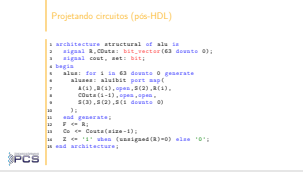

...as HDLs, ou *Hardware Description Languages*, ou em português: Linguagens de Descrição de Hardware. Surgiram na década de 80 devido a necessidade de projetar circuitos complexos, mas só foram realmente adotadas em massa a partir da década de 90. O exemplo que você vê na figura é parte da descrição de uma ULA muito parecida com a 74181 vista anteriormente, só que com 64 bits. Imagina a quantidade de erros que um projeto em papel teria? Uma confusão de um projetista pode deixar o desenho completamente inválido, tipo duas linhas que se cruzam em um desenho e você está em dúvida se elas estão ligadas ou não (só pra constar, existem regras pra isso). Este pedaço de código do slide é um trecho em VHDL que define a interface da ULA com o mundo externo. Normalmente as HDLs separam a descrição da interface da funcionalidade, que pode ser vista...

# Projetando circuitos (pós-HDL)

```
1  architecture structural of alu is
2    signal R,COuts: bit_vector(63 downto 0);
3    signal cout, set: bit;
4  begin
5    alus: for i in 63 downto 0 generate
6      aluses: alu1bit port map(
7        A(i),B(i),open,S(2),R(i),
8        COuts(i-1),open,open,
9        S(3),S(2),S(1 downto 0)
10     );
11   end generate;
12   F <= R;
13   Co <= Couts(size-1);
14   Z <= '1' when (unsigned(R)=0) else '0';
15 end architecture;
```







...aqui. Aqui fica mais claro que essa ULA é de 64 bits. Se fosse um proejto com desenhos, ocuparia várias e várias páginas. A descrição também usa um componente que está descrito em algum outro lugar: uma ULA de 1 bit que pode ser vista na linha 6 sendo instanciada 64 vezes. Esta descrição pode ser sintetizada em um sintetizador de hardware e pode inclusive ser emulada. Ao sintetizá-la, obtemos...

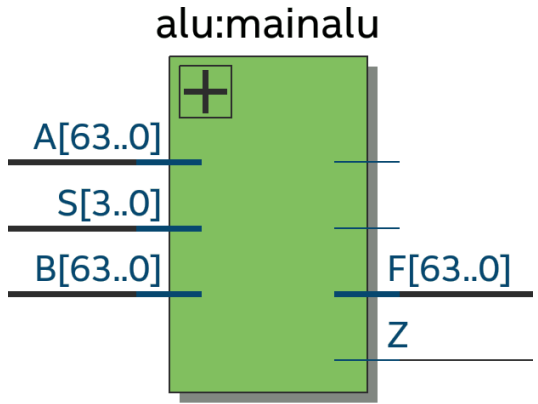
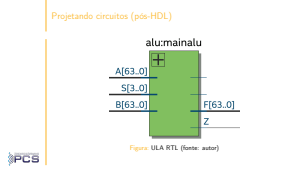



Figura: ULA RTL (fonte: autor)








...algo parecido com esse desenho. Note que as entradas são exatamente as descritas na interface dos slides anteriores (nesse caso o overflow e o carry out não foram usados e não aparecem). Também podemos o que o sintetizador gerou por dentro, mais ou menos assim...

# Projetando circuitos (pós-HDL)

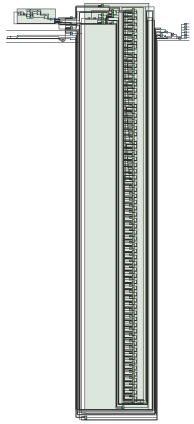


Figura: ULA RTL (fonte: autor)





Projetando circuitos (pós-HDL)

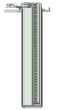


Figura: ULA RTL (fonte: autor)

Não se assuste, essa figura mostra as 64 instâncias da ULA de 1 bit que vimos na descrição em VHDL nos slides anteriores. É só dar um zoom que poderemos vê-la, olha só:

# Projetando circuitos (pós-HDL)

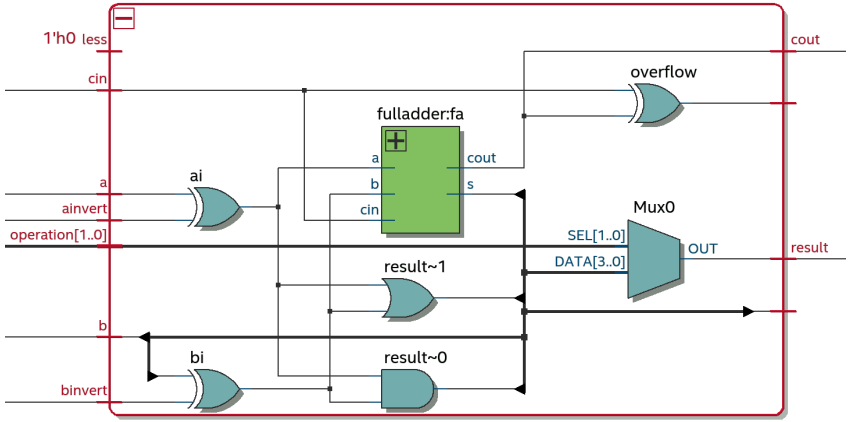
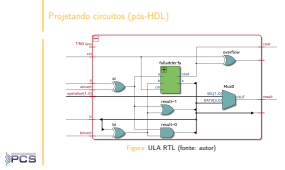



Figura: ULA RTL (fonte: autor)







Essa é a arquitetura interna de uma das ULAs de 1 bit que formam a ULA maior de 64 bits. Vejam que tem algumas portas lógicas e também um módulo somador completo, que pode ser visto no centro da figura.
[MUDAR PARA CAMERA]
Poderíamos continuar explorando o projeto que o sintetizador gerou a partir da nossa descrição em HDL, mas vamos parar por aqui para falar de outras coisas. É bem óbvio que usando uma HDL para descrever seu projeto digital tem suas vantagens, mas quais são elas? Vamos ver.
[MUDAR PARA SLIDE]

# Vantagens de HDL



- Complexidade:
  - Circuitos modernos são infactíveis de **desenhar** mesmo com CAD.
  - É possível expressar um circuito em níveis mais altos de **abstração**.
  - O circuito gerado é facilmente **testável**.
- Padronização:
  - O código de um módulo pode ser facilmente **reutilizado**.
  - As linguagens representam um projeto digital e são **interoperáveis**.
  - Uma descrição em HDL é facilmente **portável** entre ferramentas.





Vantagens de HDL

- Complexidade:
  - Circuitos modernos são infactíveis de **desenhar** mesmo com CAD.
  - É possível expressar um circuito em níveis mais altos de **abstração**.
  - O circuito gerado é facilmente **testável**.
- Padronização:
  - O código de um módulo pode ser facilmente **reutilizado**.
  - As linguagens representam um projeto digital e são **interoperáveis**.
  - Uma descrição em HDL é facilmente **portável** entre ferramentas.

- Pense num projeto de um processador moderno, que você tem no seu celular ou computador. Não dá pra projetar um desses usando desenhos em papel, é simplesmente inviável.
- Quando desenhamos portas lógicas, o que estamos fazendo é um projeto estrutural, com as ligações entre os componentes. Quando usamos HDL, podemos nos beneficiar de abstrações mais altas, como RTL e descrição funcional. Não se preocupe com os termos pois veremos cada um no curso. Basta saber que podemos descrever coisas complexas com menos esforço.
- Mudando para a padronização, as HDLs mais comuns, como VHDL e Verilog, são padronizadas internacionalmente por entidades reconhecidas na Engenharia de Eletricidade e de Computação, como a IEEE. Isso permite que o módulo que você escreve em uma HDL seja usado por qualquer outro projetista, inclusive você mesmo.
- Por serem padrões rígidos e bem feitos, você pode usar um módulo inclusive escrito usando outra linguagem, o que torna o módulo interoperável. Vários sintetizadores suportam mais de uma linguagem, e você pode inclusive usar um módulo escrito em uma em um projeto que usa outra.
- As ferramentas entendem a descrição do mesmo jeito, pois o padrão define também a semântica, então você pode escrever e sintetizar o seu projeto em uma ferramenta, mas simular em outra sem problemas. Não existe essa coisa de ferramenta incompatível, se ambas suportam o padrão, ambas funcionarão com o seu código.

# Desvantagens de HDL



- Descrever um circuito em HDL é diferente de programar.
- É necessário conhecimento de projeto digital.
- Você está descrevendo um circuito.





Desvantagens de HDL
- Descrever um circuito em HDL é diferente de programar.
- É necessário conhecimento de projeto digital.
- Você está descrevendo um circuito.

- Quando você usa uma HDL, você está descrevendo um circuito e não programando. Muitos projetistas iniciantes, especialmente os que possuem formação em computação, se esquecem disso e acabam com descrições muito ruins a ponto de não funcionar direito. Vamos ver mais detalhes já no próximo slide, mas lembre-se sempre que uma descrição em HDL não é um programa.
- Outra coisa importante é que a HDL é uma forma de expressar um projeto de um circuito digital. Se você não sabe fazer um projeto digital, não adianta aprender uma HDL. É como você ensinar francês pra um papagaio, ele vai repetir mas não tem ideia do que está falando. Um bom projetista de hardware é aquele que pensa no circuito que deseja pra resolver o problema e depois se expressa, seja usando HDL ou desenho.
- Por último, a descrição representa um circuito no final das contas, o que significa que não tem sistema operacional, não tem `printf` e não tem depurador como tem em software. Você precisa prever interfaces de depuração e inserir pontos de amostragem caso queira. Não tem osciloscópio ou multímetro também, pois as descrições ainda vão gerar um circuito. Pense assim: dá pra medir a tensão em um ponto em um desenho em papel? Pois é, em HDL é a mesma coisa, não tem essas facilidades até que você materialize o circuito através de um sintetizador. Felizmente existem simuladores pra ajudar.

# Concorrência

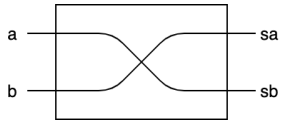


```
1 sa <= a;
2 sb <= b;
```

(a) VHDL

```
1 x = a;
2 a = b;
3 b = x;
```

(b) C/C++




(c) HW

```
1 MOV ecx, eax
2 MOV eax, ebx
3 MOV ebx, ecx
```

(d) Assembly

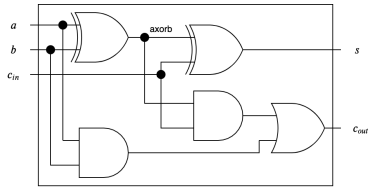




A principal diferença de programar e descrever um circuito em uma HDL é o modelo de concorrência. Em um programa, as instruções geradas são executadas em sequência pelo processador, mas quando se está descrevendo um hardware, você está na verdade ligando portas lógicas, então tudo acontece ao mesmo tempo! Veja o exemplo ao lado, que mostra uma operação de *swap* em VHDL e em C. Em VHDL, o hardware gerado será simplesmente dois fios que fazem a inversão, mas em C a operação gera três instruções e precisa de uma variável temporária extra. O código em VHDL representa um hardware, que funciona o tempo todo, já o em C representa um conjunto de instruções do processador que serão executada em sequencia. Se você tentar usar a mesma abordagem do C em VHDL, irá gerar um curto circuito! Pra ficar mais claro, vamos ver um exemplo mais complexo...

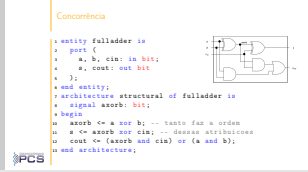
# Concorrência

```
1  entity fulladder is
2    port (
3      a, b, cin: in bit;
4      s, cout: out bit
5    );
6  end entity;
7  architecture structural of fulladder is
8    signal axorb: bit;
9  begin
10   axorb <= a xor b; -- tanto faz a ordem
11   s <= axorb xor cin; -- dessas atribuicoes
12   cout <= (axorb and cin) or (a and b);
13 end architecture;
```







Nesse exemplo vemos um somador completo descrito em VHDL. Note que há uma entidade descrevendo a interface do módulo, chamado de `fa` e em seguida uma arquitetura que descreve seu funcionamento. Quando geramos o hardware através de um sintetizador, o circuito gerado é o que está na figura a direita. Esse circuito funciona o tempo todo, então não existe uma sequencia em que as atribuições precisam ser executadas. Na verdade, se você colocar as atribuições das linhas 10, 11 e 12 em qualquer ordem, o circuito gerado será exatamente o mesmo!

Obrigado!


balbertini@usp.br



[MUDAR PARA CAMERA]

Nessa aula vimos o que é uma HDL, o motivo pelo qual usamos esse tipo de linguagem para descrever hardware e também as principais diferenças de uma HDL e uma linguagem de programação. Espero que tenha entendido tudo e nessa aula vamos parar por aqui, mas esse é só o começo! Para o restante do material siga as orientações na página do curso ou do seu professor. Espero que tenha gostado e até a próxima aula!

# Referências



J. Wakerly.
*Digital Design: Principles and Practices*.
Pearson Education, Incorporated, 5 edition, 2018.





Referências
J. Wakerly.
*Digital Design: Principles and Practices*.
Pearson Education, Incorporated, 5 edition, 2018.