



Universidade de Brasília

Departamento de Ciência da Computação

Síntese de Hardware

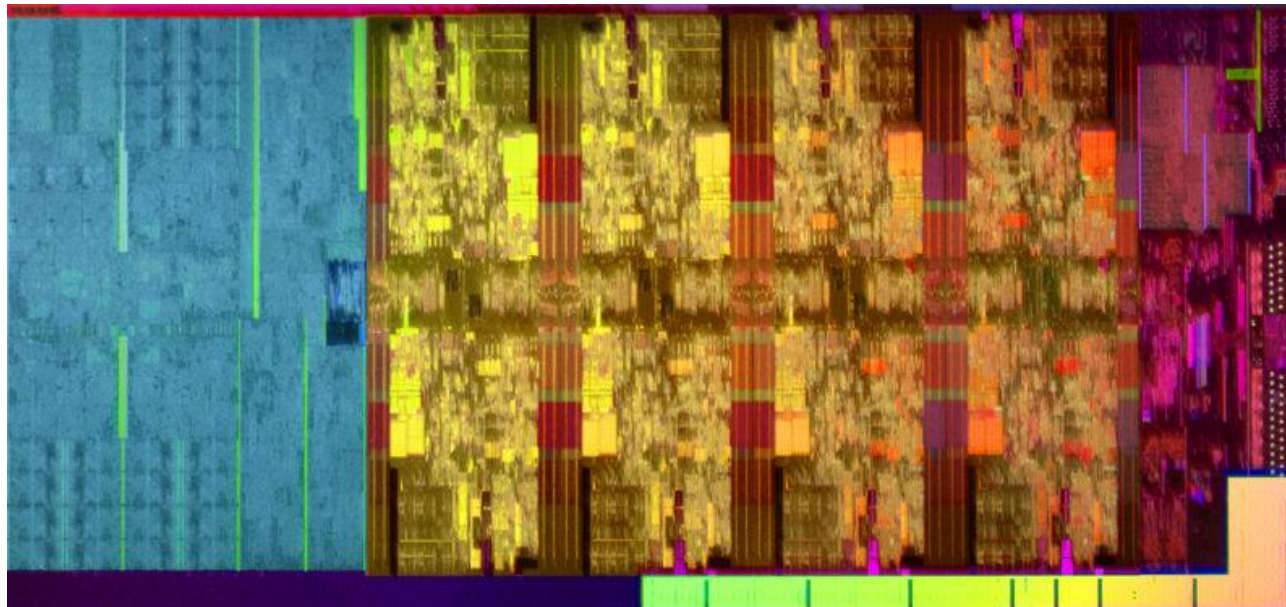
Verilog

Terasic DE2-115

Linguagem de Descrição de Hardware (HDL)

- Motivação:

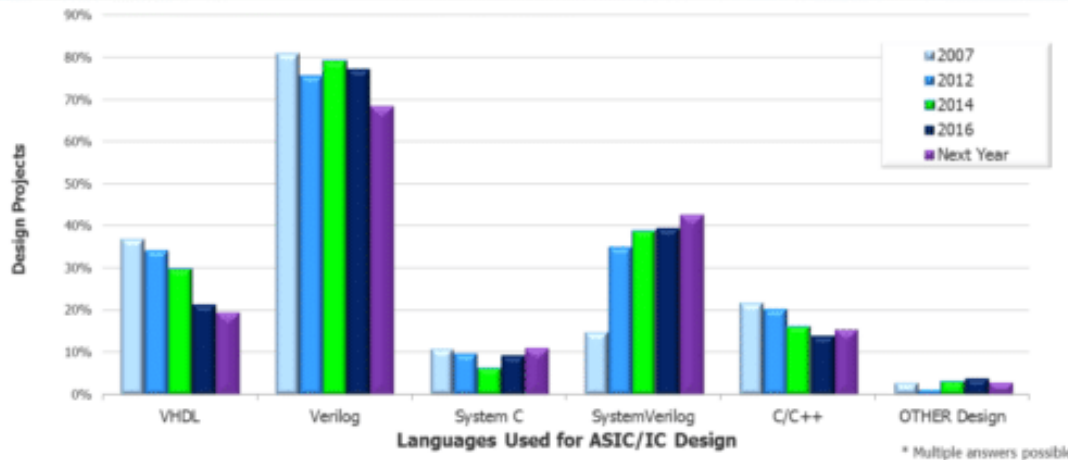
Sistemas Digitais complexos são praticamente impossíveis de estudar e implementar através das técnicas tradicionais de síntese de circuitos digitais usando esquemático ao nível de transistores ou mesmo portas lógicas!



Linguagem de Descrição de Hardware (HDL)

- Juntamente com VHDL, Verilog é uma das mais populares HDLs. Muito usada pela Intel (SystemVerilog).

ASIC/IC Design Language Adoption Trends

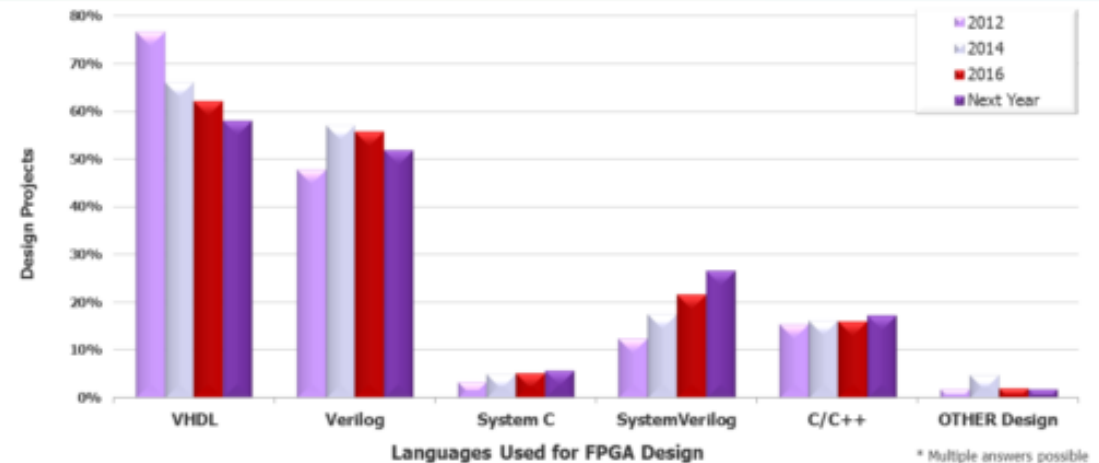


Source: Wilkon Research Group and Mentor Graphics, 2016 Functional Verification Study

© Mentor Graphics Corp. Company Confidential
www.mentor.com



FPGA Design Language Adoption Trends



Source: Wilkon Research Group and Mentor Graphics, 2016 Functional Verification Study

© Mentor Graphics Corp. Company Confidential
www.mentor.com



Fonte: Mentor Graphics ([link](#))



Verilog

■ Histórico

1984/85 – criada por Philip Moorby (Gateway Design System Co.)
adquirida pela empresa Cadence

1990 – Open Verilog International : Domínio público

1995 – Padrão IEEE 1364

2005 – revisão e atualização do padrão, IEEE 1364

2012 – IEEE 1800, SystemVerilog, orientação a objetos - Superset

■ Usada para projetos de circuitos:

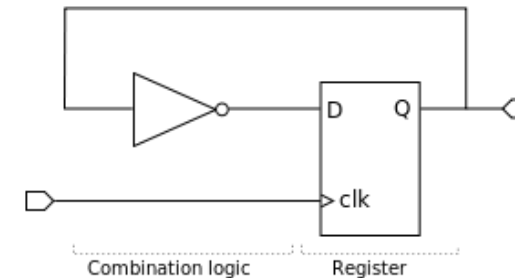
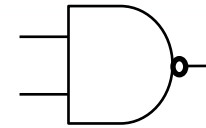
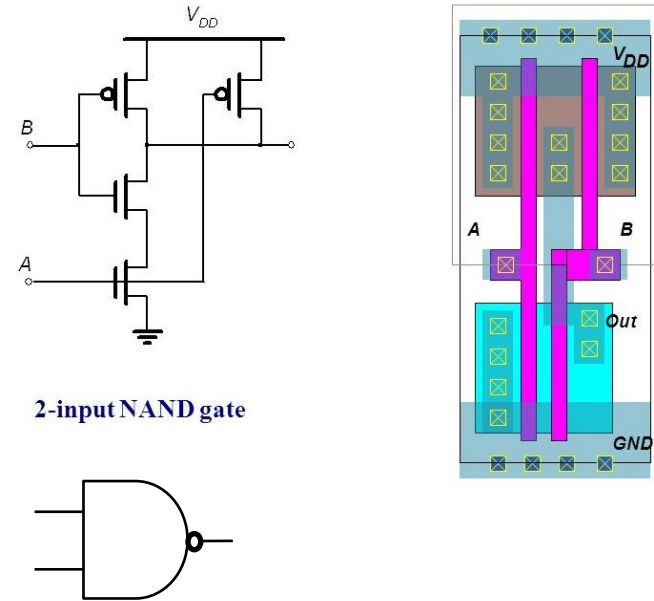
- ASIC (*Application Specific Integrated Circuit*)
- FPGA (*Field Programmable Gate Array*).

Verilog

■ Níveis de Abstração

- Nível de Layout (*Layout Level*)
- Nível de Portas Lógicas (*Gate Level*)
- Nível de Transferência de Registradores (*Register Transfer Level*)
- Nível Comportamental (*Behaviour Level*)


```
assign a = b / c;
```



SystemVerilog – aspectos básicos

■ Representação de Números:

	<tamanho em bits>'<base><número>	dado armazenado
Ex.: Decimal: ('d' ou 'D')	16'd255	0000000011111111
Hexadecimal ('h' ou 'H')	8'h9A	10011010
Binário ('b' ou 'B')	32'b1010	00000000000000000000000000001010
Octal ('o' ou 'O')	8'o21	00_010_001

Números negativos: -8'd3 = -3 em 8 bits em complemento de 2 (11111101)

■ Valores:

níveis lógicos: 0 ou 1
 desconhecido: x (nível lógico não conhecido)
 alta impedância: z (fio não conectado)

Ex.: 32'bz 8'h0x 4'b1z0x

Obs: se o tamanho não for especificado o *default* é 32 bits!

SystemVerilog – aspectos básicos

■ Tipos de Dados

- **wire**: define um ou conjunto de fios

```
Ex.: wire a;           // a é um fio  
      wire [7:0] b;    /* b é um conjunto de 8 fios {b[7],b[6],...,b[1],b[0]}*/
```

- **reg** : define um registrador (síncrono ou assíncrono)

```
Ex.: reg a;           // a é um latch ou flip-flop  
      reg [7:0] b;    /* b é um conjunto de 8 latches ou flip-flops */
```

- **logic** : define um registrador ou wire

```
Ex.: logic a;         // a é um elemento definido pelo sintetizador  
      logic [7:0] b;  /* b é um elemento de 8 bits definido pelo  
                      sintetizador */
```

- Tipos abstratos (32 bits): **integer**, **real** (IEEE 754)

SystemVerilog

■ Características da sintaxe

❖ Case Sensitive :

`reset` é diferente de `Reset` e de `RESET`

❖ Nomes não podem iniciar por números:

`2mux` é um nome inválido! `mux2` é válido

❖ Espaços em branco são ignorados

❖ Comentários iguais à linguagem C

`// comentário de uma linha`

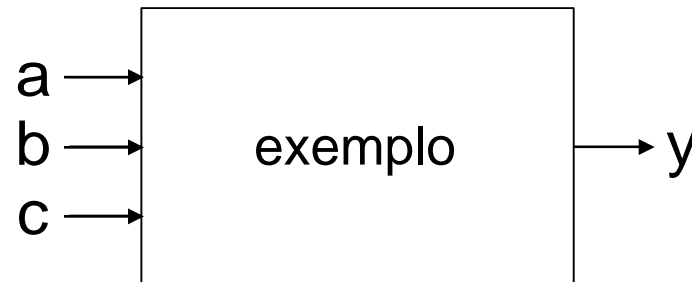
`/* comentário`

`em múltiplas linhas */`

SystemVerilog

■ Estrutura básica de uma descrição Verilog

```
module <nome>(<entradas>, <saídas>);  
    <descrição>;  
endmodule
```



$$y = \bar{a}.\bar{b}.\bar{c} + a.\bar{b}.\bar{c} + a.\bar{b}.c$$

```
module exemplo(input  logic a, b, c,  
               output logic y);  
    assign y = ~a & ~b & ~c | a & ~b & ~c | a & ~b & c;  
endmodule
```



SystemVerilog

■ Operações Básicas do Verilog e precedência

()	Indica prioridade
{}, {{{}}	Concatenação
~	NOT
*, /, %	multiplicação, divisão, módulo
+, -	adição, subtração
<<, >>	deslocamento lógico
<<<, >>>	deslocamento aritmético
<, <=, >, >=	Comparações
==, !=	igual, diferente
&, ~&	AND, NAND
^, ~^	XOR, XNOR
, ~	OR, NOR
?:	operador ternário condicional

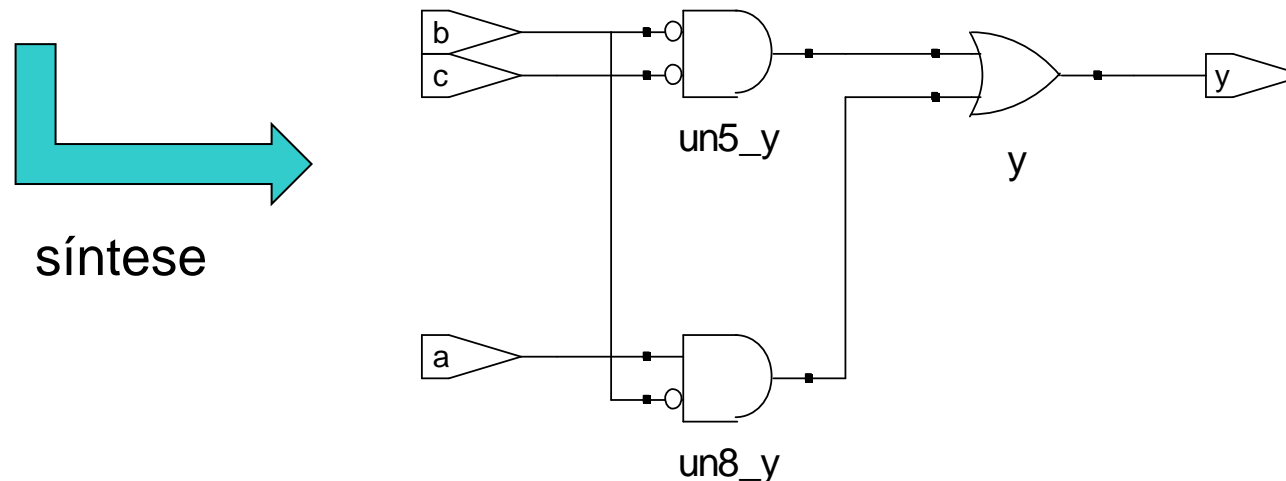
SystemVerilog

■ Síntese

A síntese de um circuito corresponde à tradução (compilação) de uma descrição Verilog em uma netlist (descrição das interconexões dos circuitos) pelo uso de uma ferramenta de síntese (Quartus).

Geralmente, durante a compilação, diversas otimizações são realizadas.

```
module exemplo(input  logic a, b, c,  
                output logic y);  
    assign y = ~a & ~b & ~c | a & ~b & ~c | a & ~b &  c;  
endmodule
```



Atribuições

1) Definindo um circuito combinacional

Neste caso, geralmente y é do tipo `wire`.

```
assign y = ~a & ~b & ~c | a & ~b & ~c;
```

ou

```
always @(*) begin
```

```
    y <= ~a & ~b & ~c | a & ~b & ~c; end;
```

Operador atribuição Não-Blocante

ou

```
always @(*) begin
```

```
    y = ~a & ~b & ~c;
```

```
    y = y | a & ~b & ~c; end;
```

Operador atribuição Blocante



Atribuições

2) Definindo um circuito sequencial

Neste caso, geralmente y é do tipo `reg`.

```
always @(posedge clock) begin
    y <= ~a & ~b & ~c | a & ~b & ~c; end;
```

Operador atribuição Não-Blocante

ou

```
always @(posedge clock) begin
    y = ~a & ~b & ~c;
    y = y | a & ~b & ~c; end;
```

Operador atribuição Blocante



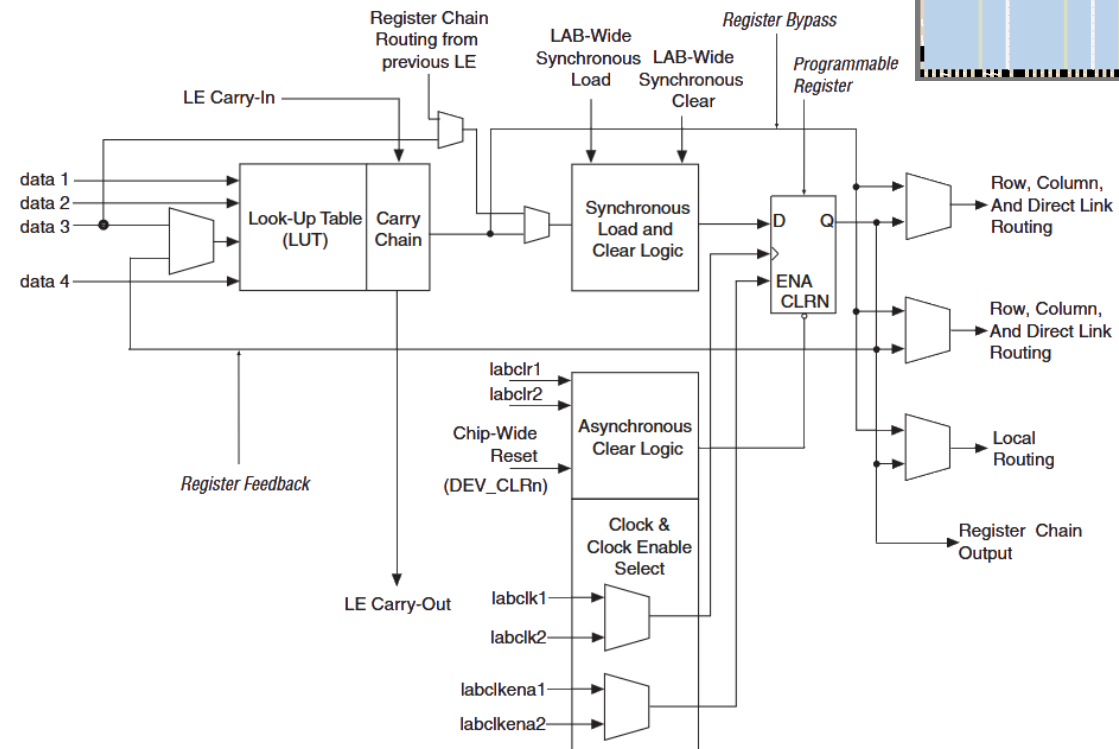
FPGA: Intel Cyclone-IV EP4CE115F29C7

Field Programmable Gate Array

Chip capaz de implementar qualquer sistema digital através da definição da interconexão de seus elementos.

O modelo Cyclone IV possui:

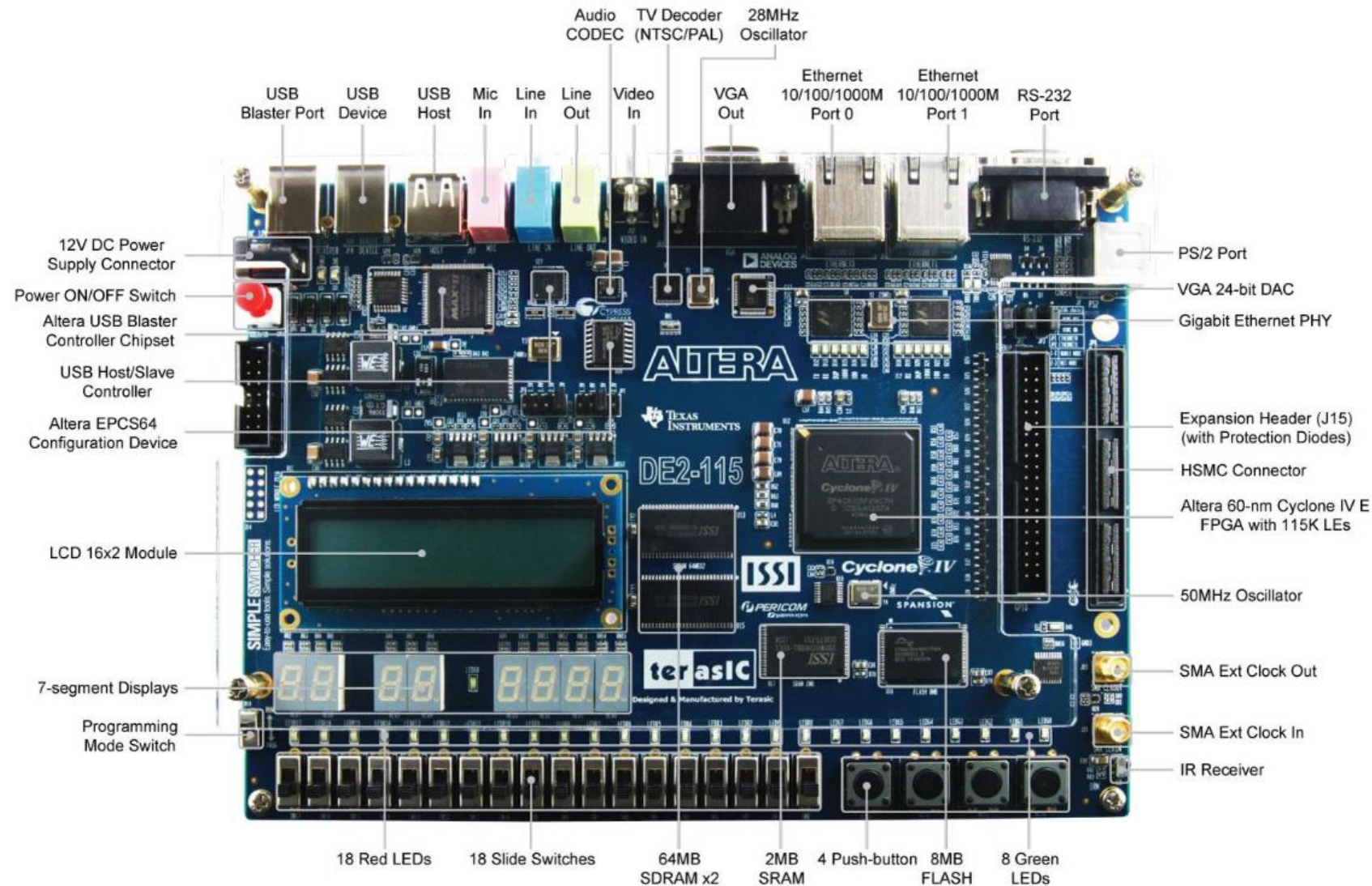
- 780 pinos
- 114.480 *Logic Elements* (LEs)
 - Função lógica de 4 entradas (LUT)
 - 1 registrador
 - Somadores encadeáveis (carry)
 - Roteamento
- Memória RAM 3.981.312 bits
- 532 multiplicadores 9x9
- 4 PLL (*Phase Locked Loop*)
- 528 Pinos de IO para o usuário





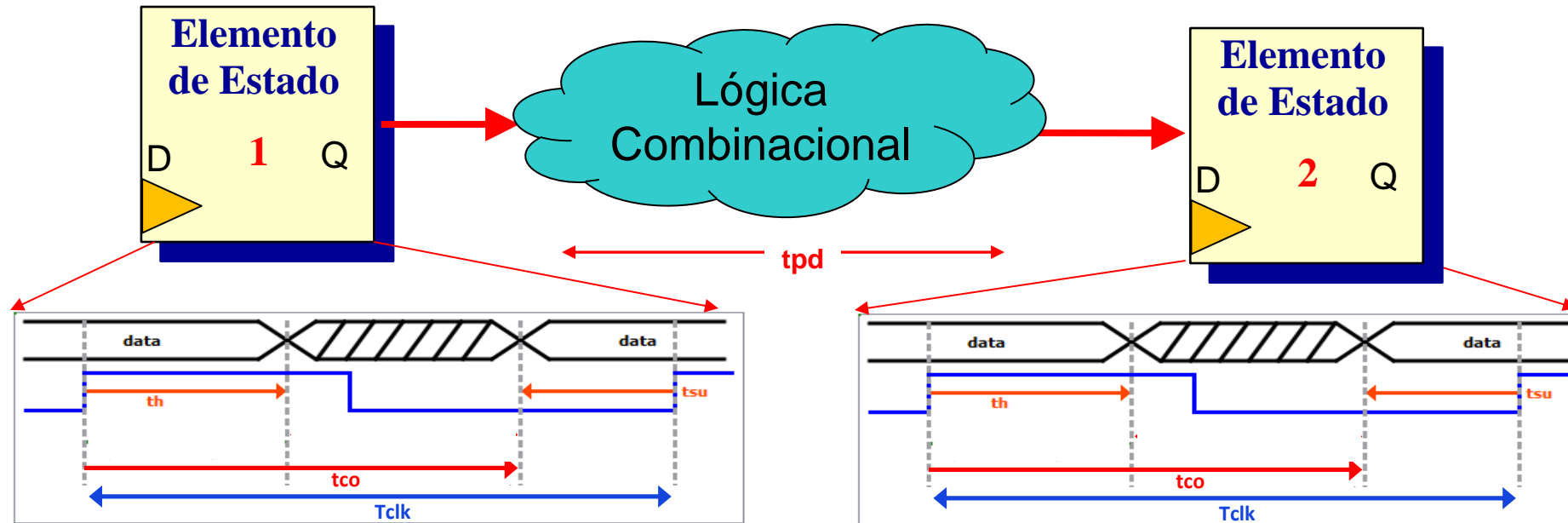
Plataforma de desenvolvimento Intel DE2-115

- Diversos dispositivos de IO já ligados aos pinos do FPGA





Revisão: Temporização em circuitos digitais



- **A escrita no elemento de estado 1 deve respeitar os tempos de:**
 - tempo de pré-carga (*setup time*, tsu) do elemento 1
 - tempo de hold (*hold time*, th) do elemento 1
- **O elemento de estado 2 só pode ser escrito depois que os dados em sua entrada estarem estáveis**
 - atraso de propagação da saída dado o clock (*clock to output*, tco) do elemento 1
 - atraso da lógica combinacional (*propagation delay*, $tpd = \max(tp_{HL}, tp_{LH})$)
 - tempo de pré-carga (*setup time*, tsu) do elemento de estado 2

Desmitificando o sinal de relógio: clock

