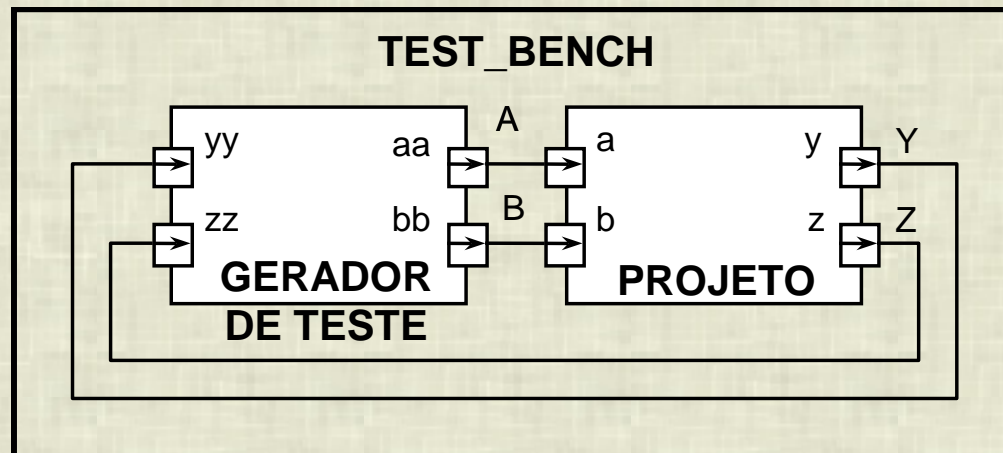


Fundamentos de sistemas digitais

Test-bench

Validação por Simulação

- Utilizar um circuito de teste: *test-bench*
 - Contém um circuito **gerador de teste** e uma instância do projeto
 - O **gerador de teste** deve gerar um conjunto de estímulos para o circuito que permitam que o comportamento do circuito seja verificado
 - O **gerador de teste** normalmente tem uma descrição comportamental não sintetizável, enquanto o projeto tem uma descrição sintetizável
 - O test_bench **não** contém portas de entrada/saída



Validação por Simulação

```
library IEEE;
use IEEE.std_logic_1164.all;
```

Bibliotecas básicas

```
entity tb is
end tb;
```

O test_bench não contém portas na interface externa

Seleção da entidade

```
architecture Tb of tb is
    signal A, B, Y, Z: std_logic;
begin
```

Sinais internos ao test-bench

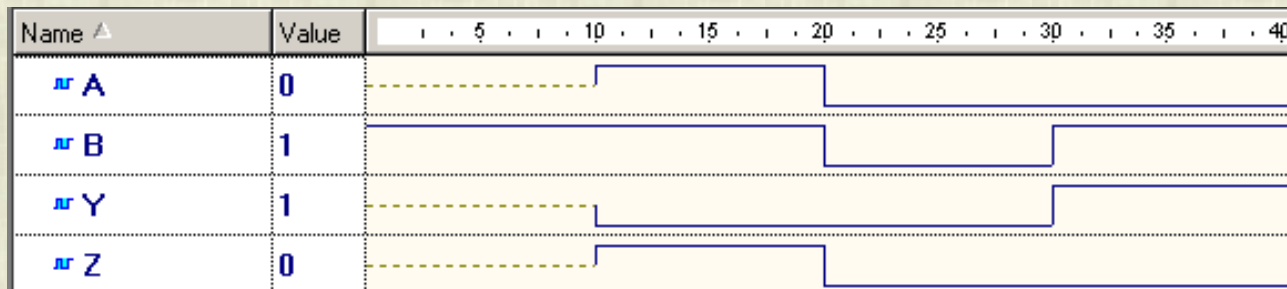
```
    id1: entity work.Entidade port map (a=>A, b=>B, y=>Y, z=>Z);
```

Instanciação do projeto (circuito) a ser testado

```
    A <= '1' after 10ns, '0' after 20ns;
    B <= '1', '0' after 20ns, '1' after 30ns;
```

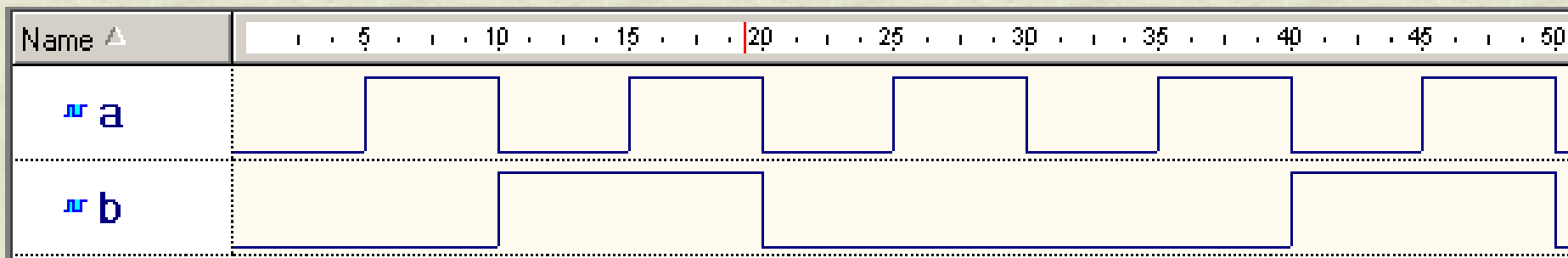
Gerador de teste

```
end Tb;
```



Uso de Processos em Test-Benchs

```
architecture Processos of Processos is
    signal a, b: std_logic;
begin
    process
    begin
        a <= '0', '1' after 5ns;
        wait for 10ns;
    end process;
    process
    begin
        b <= '0', '1' after 10ns, '0' after 20ns;
        wait for 30ns;
    end process;
end Processos;
```

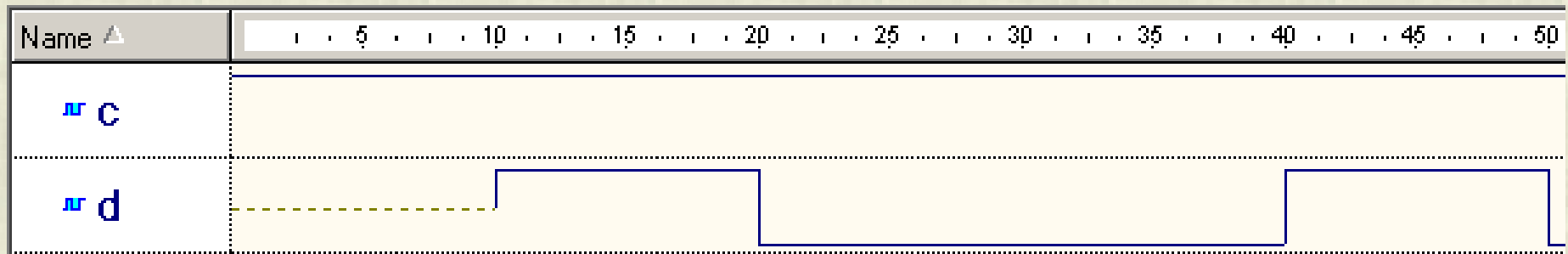


Uso de Processos em Test-Benchs

```

architecture Processos of Processos is
    signal c, d: std_logic;
begin
    process
    begin
        c <= '1', '0' after 20ns;
        wait for 20ns;
    end process;
    process
    begin
        d <= '1' after 10ns, '0' after 20ns;
        wait for 30ns;
    end process;
end Processos;

```








Uso de Processos em Test-Benchs

```

architecture Processos of Processos is
    signal K: std_logic_vector(3 downto 0);
    signal Y: std_logic_vector(3 downto 0) := (others => '0');
begin
    process
    begin
        K <= K + '1' after 10ns;
        wait for 30ns;
    end process;
    process
    begin
        Y <= Y + '1' after 10ns;
        wait for 30ns;
    end process;
end Processos;

```

Name 	<div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> </div>									
  K	U	X								
  Y	0	1	2	3	4					

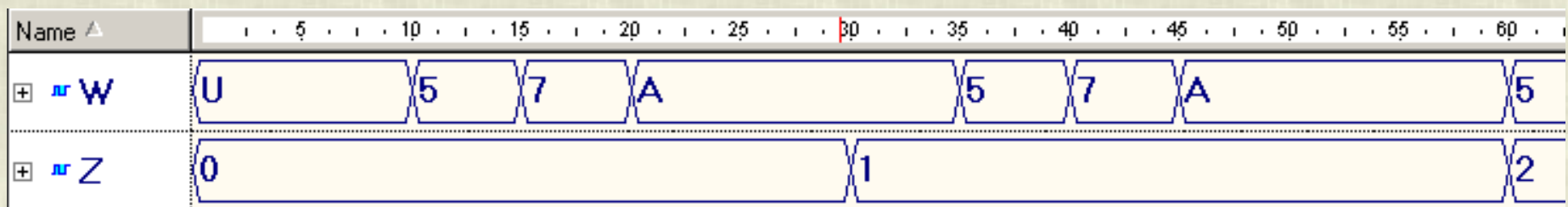
Uso de Processos em Test-Benchs

```

architecture Processos of Processos is
    signal Z: std_logic_vector(3 downto 0) := (others => '0');
    signal W: std_logic_vector(3 downto 0);
begin
    process
    begin
        Z <= Z + '1' after 30ns;
        wait for 30ns;
    end process;

    process
    begin
        W <= "0101" after 10ns, x"7" after 15ns, x"A" after 20ns;
        wait for 25ns;
    end process;
end Processos;

```



Exercícios de Uso de Processos em Test-Benchs

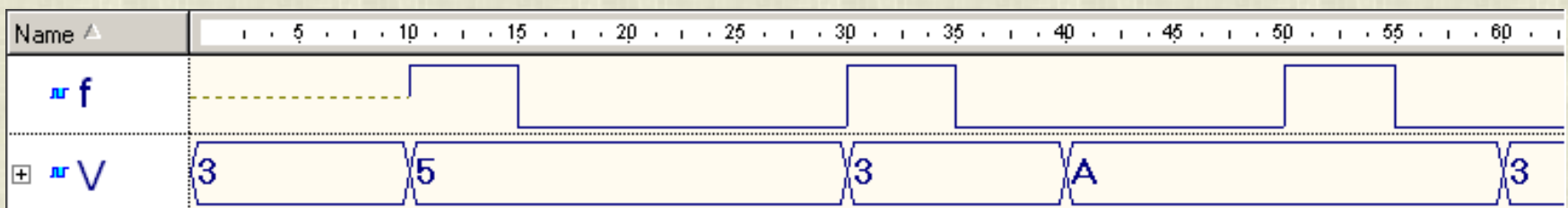
1. Dado os processos abaixo, obtenha formas de onda equivalentes

```

architecture Processos of Processos is
    signal M: std_logic_vector(3 downto 0) := "0011";
    signal N: std_logic_vector(3 downto 0);
begin
    process
    begin
        M <= M + M after 30ns;
        wait for 30ns;
    end process;
    process
    begin
        N <= "0101" + M after 10ns, "0111" after 15ns;
        wait for 25ns;
    end process;
end Processos;

```

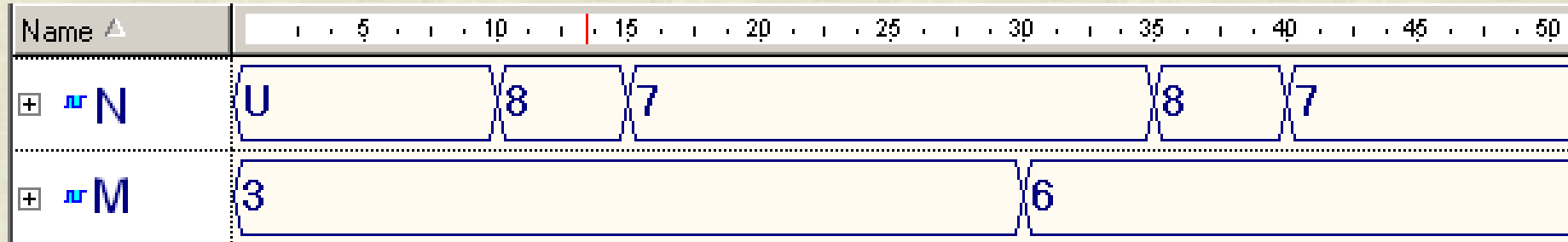
2. Dadas as formas de onda abaixo, obtenha processos que as gerem



Soluções de Exercícios

Uso de Processos em Test-Benchs

1. Dado os processos abaixo, obtenha formas de onda equivalentes



2. Dadas as formas de onda abaixo, obtenha processos que as gerem

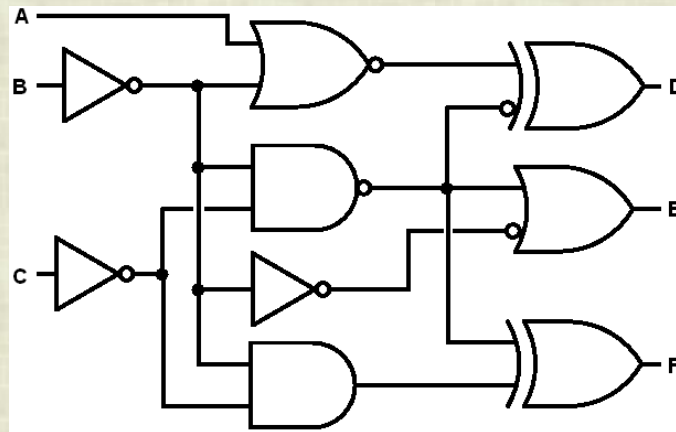
```

architecture Processos of Processos is
    signal f: std_logic;
    signal V: std_logic_vector(3 downto 0) := (others => '0');
begin
    process
    begin
        f <= '1' after 10ns, '0' after 15ns;
        wait for 20ns;
    end process;
    process
    begin
        V <= "0011", V + "0101" after 10ns;
        wait for 30ns;
    end process;
end Processos;

```

-
- **Exercício 1 – Gere um testbench que contenha os sinais A, B e C. Estes sinais devem conter um estímulo que corresponda a seguinte especificação:**
 - Sinal A:
 - Deve iniciar com o valor '0'
 - Deve assumir o valor '1' depois de 20 ns e permanecer assim até o final da simulação
 - Sinal B:
 - Deve iniciar com o valor '0' e assumir este valor a partir de 20ns novamente
 - Deve assumir o valor '1' a partir dos 10ns e novamente após os 30 ns
 - Sinal C:
 - Deve iniciar com o valor '0' e reassumir tal valor aos 10, 20 e 30 ns
 - Deve assumir o valor '1' a partir dos tempos 5, 15, 25 e 35ns

- **Exercício 2 – Construa um módulo em vhdl para representar o circuito representado graficamente abaixo.**



-
- **Exercício 3 – Inclua o circuito gerado anteriormente no testbench gerado no exercício 1 e apresente forma de onda gerada para os sinais D, E e F.**

Somador Completo de 1 bit

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity SomalBit is
    port
    (
        cin: in std_logic;
        a: in std_logic;
        b: in std_logic;
        cout: out std_logic;
        s: out std_logic
    );
end SomalBit;

architecture Somador1Bit of SomalBit is
begin
    cout <= (a and b) or (a and cin) or (b and cin);
    s <= a xor b xor cin;
end Somador1Bit;
```