Ferramentas para Síntese Automática de Circuitos Integrados

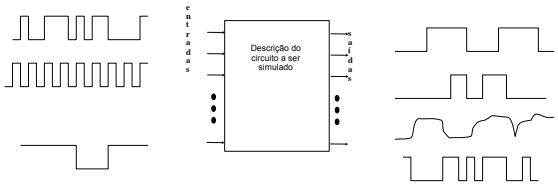
Professores Marcelo Johann e Ricardo Reis

Simulação e Verificação

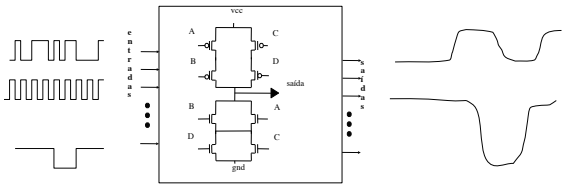
Simulation and Verification

- Introduction
- Concept and application
- Abstraction Levels
- Gate-Level Modeling and Simulation
  - Compiler Driven
  - Event Driven
- Verification

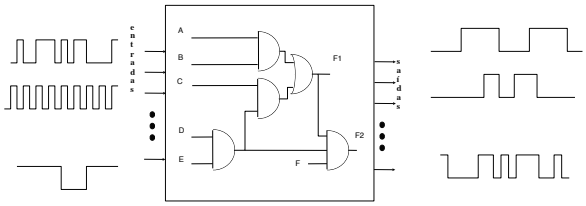
Ferramentas de Simulação



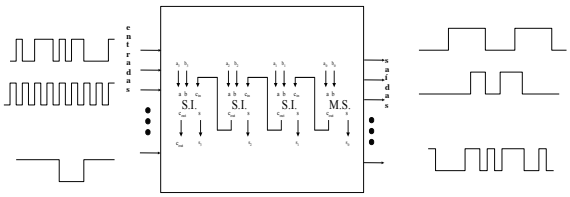
Simuladores Elétricos



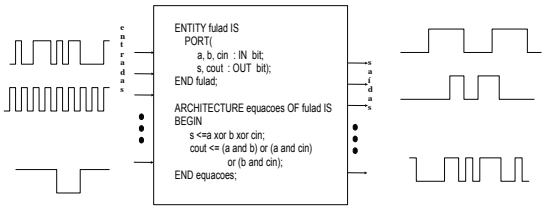
Simuladores Lógicos



Simuladores Funcionais



Simuladores Comportamentais



Concept and Application

- Model (vs. the real thing)
- Experiment
  - what happened in one run…
- Application is on verification
  - But what does it mean?

Abstraction Levels

- Device-level
- Circuit-level (or electrical simulation)
- Timing-level and macro-level
- Switch-level
- Gate-level (or logic simulation)
- Register-transfer-level (RTL)
- System-level

Other issues…

- Analog
- Digital
- Mixed-level
- Mixed-Mode
- HW-SW co-simulation

- Modules:
  - kernel, input, stimuli, results

Gate-level Modeling and Simulation

- Signal Modeling
- Gate Modeling
- Delay Modeling
- Connectivity Modeling

- Compiler-driven simulation
- Event-driven simulation

Signal Modeling

- Binary, or Boolean…
- 0,1,X,U,Z,weaks,don't care,up,down, etc…
  - Up to 99 values were already used!!!
- Try to determine Truth Tables for that!
- User defined data types for signal (VHDL)

- Nowadays IEEE 1164 is a common language

IEEE 1164

Value	Interpretation
U	Uninitialized
X	Forcing Unknown
0	Forcing 0
1	Forcing 1
Z	High Impedance
W	Weak unknown
L	Weak 0
H	Weak 1
-	Don´t care

Gate Modeling

- Outputs must be computed as a function of inputs
- Truth Tables
  Look up in a table
- Subroutine
  Uses machine instructions for speed up

Delay Modeling

- The propagation delay model
  - Fixed delay for each gate type
  - Zero delay model
  - Unit delay model
  - a+nb, for a constant, n gates, b fanout related
- The rise/fall delay model
- The inertial delay model
  - Requires a minimum pulse width

Compiler-driven Simulation

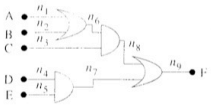


Figure 10.3 A simple circuit composed of logic gates.

$n_1 \leftarrow$ A;
$n_2 \leftarrow$ B;
$n_3 \leftarrow$ C;
$n_4 \leftarrow$ D;
$n_5 \leftarrow$ E;
$n_6 \leftarrow$ OR(n_1, n_2);
$n_7 \leftarrow$ AND(n_4, n_5);
$n_8 \leftarrow$ AND(n_6, n_3);
$n_9 \leftarrow$ OR(n_7, n_8);
F $\leftarrow$ n_9;

Figure 10.4 Code for the zero-delay simulation of the circuit of Figure 10.3.

Compiler-driven Simulation

- Glitches:

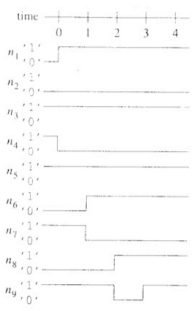


Figure 10.5 The appearing of a glitch on net n_9 in the circuit of Figure 10.3 when using unit-delay simulation.

Event-driven Simulation

- Only a few signals are changing at most times
- Event = singal change in value
- Need to keep propagated events in the right order
- Different delay models are easy to incorporate

Event list (or queue)

- Array implementation with discrete time

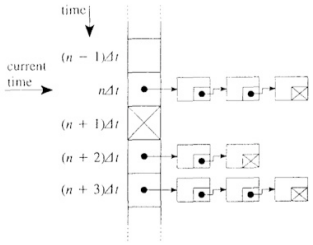


Figure 10.8 The array implementation of the event queue.

Event list (or queue)

- Time wheel and overflow lists

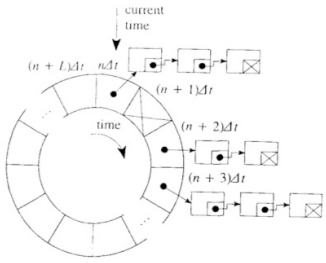


Figure 10.9 The time-wheel implementation of the event queue.

Event list (or queue)

- Code

```
event_driven_simulation ()
{
  struct event_queue *Q;
  Q ← new_queue();
  "insert stimuli in Q";
  "initialize: all network nodes connected to a memory to 'U' and
               all other nodes to 'X'";
  for (t ← t_start; t < t_end;) {
    current_event ← first_event(Q);
    t ← current_event->time;
    "process current_event and add new events to Q at
     time t + appropriate delay";
  }
}
```

Figure 10.10 Pseudo-code of the event-driven simulation function.

Ferramentas de Verificação

Objetivos:

Comparação de equivalência entre descrições intermediárias de um circuito sendo sintetizado.

Teste de consistência de uma descrição individual (verificação de regras).

Classificação:

- Verificadores de Regras de Projeto
- Verificadores de Regras elétricas
- Extratores elétricos
- Extratores lógicos
- Comparadores de netlists
- Verificadores de equivalência lógica
- Verificadores de timing

GME UFRGS

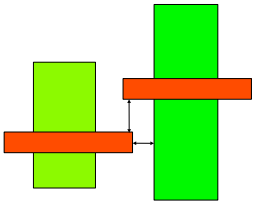
Verificação de Regras de Projeto

Objetivos:

Verificar se as máscaras do circuito obedecem as regras geométricas de projeto.

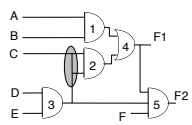
Classificação das regras:

- Distância mínima,
- Largura mínima,
- Composição de camadas,
- Superposição mínima,
- etc...

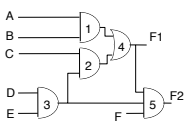


GME UFRGS

Verificação de Regras Elétricas



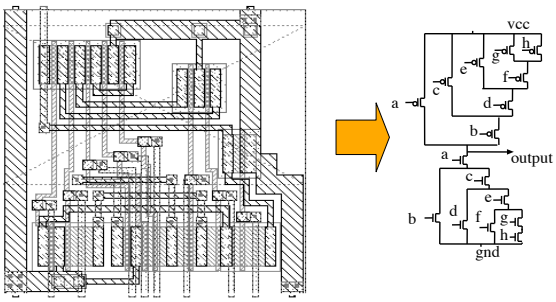
a) Circuito combinacional contento erro



b) Circuito combinacional corrigido

GME UFRGS

Extratores Elétricos

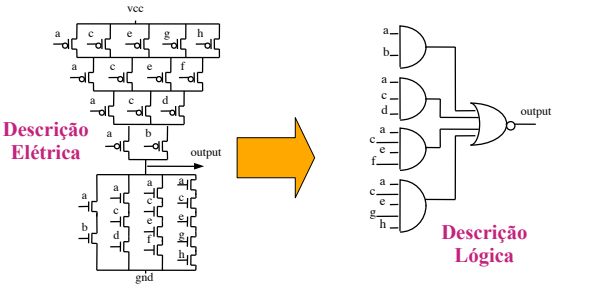


GME

UFRGS

Ferramentas de CAD

Extratores Lógicos

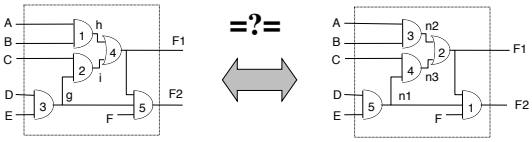


GME

UFRGS

Ferramentas de CAD

Comparadores de Netlists



GME

UFRGS

Ferramentas de CAD

Verificação de Equivalência Lógica

Objetivos:

Verificar a equivalência lógica entre dois circuitos, ou seja determinar se eles implementam a mesma função Booleana.

Exemplo:

As duas equações abaixo são equivalentes?

$$x1 = (a+b)\bullet(c+d)\bullet(\bar{a}+\bar{b}+\bar{c}+\bar{d})$$

$$x2 = a\bullet\bar{b}\bullet c+\bar{a}\bullet b\bullet d+a\bullet\bar{c}\bullet d+b\bullet c\bullet\bar{d}$$

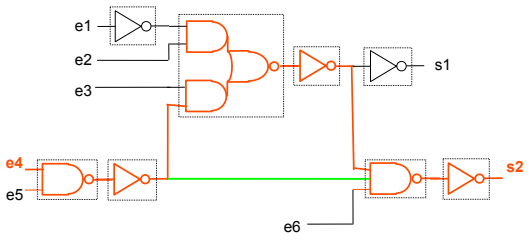
GME

UFRGS

Verificação de Timing

Objetivo:

Determinar o atraso máximo de um determinado circuito.



GME

UFRGS

Ferramentas de Auxílio ao Teste

Objetivos:

Facilitar o teste do circuito depois da fabricação, de modo a determinar as peças que poderão ser comercializadas e aquelas que deverão ser destruídas devido à defeitos de fabricação.

Classificação:

Geração de vetores de teste

Inserção de Boundary-Scan

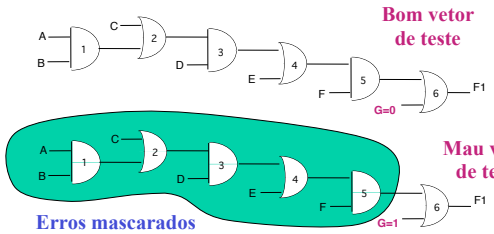
Geração de BIST

GME

UFRGS

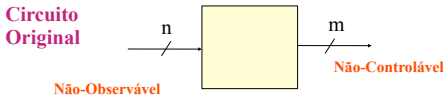
Geração de Vetores de Teste

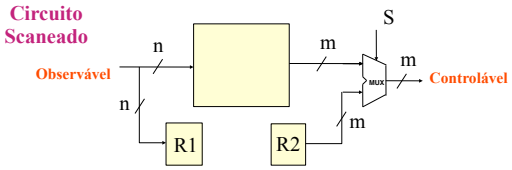
Objetivo:
Encontrar vetores que testem grande parte do circuito.



GME

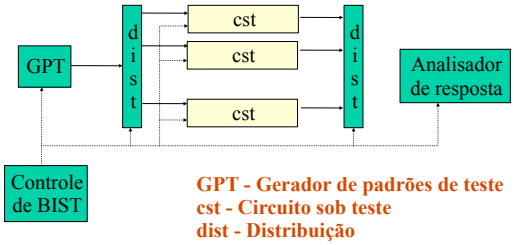
Inserção de Boundary-Scan





GME

Geração de BIST



GPT - Gerador de padrões de teste
cst - Circuito sob teste
dist - Distribuição

GME

Ferramentas para Síntese Automática de Circuitos Integrados

Professores Marcelo Johann e Ricardo Reis

Simulação e Verificação