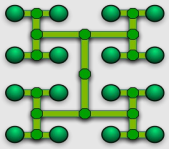
Universidade de Brasília

Modelo Comportamental

Ricardo Jacobi
Departamento de Ciência da Computação
Universidade de Brasília
jacobi@unb.br



LAICO

Nível Comportamental

- Antigamente, era confundido com o nível RT
- No modelo RT o projeto deve ser detalhado em nível de estados, condições de transição de estados e quais micro-operações ocorrem nos estados
- No nível comportamental, o funcionamento é expresso usualmente na forma de um algoritmo. O sequenciamento é definido em termos de eventos de entrada e saída, e não ciclos de relógio
- Ambos compartilham, entretanto, a mesma definição de pinos de entrada e saída

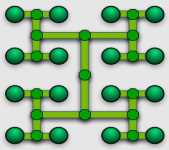
- Ref:
  - System Design with SystemC, T. Grötker et al. Kluwer Academics

Modelo Síncrono

- Neste exemplo será considerado um modelo comportamental síncrono:
  - Eventos sequenciados e sincronizados por um relógio
  - Uso de sinais para comunicação
  - Neste nível, um ciclo de relógio pode corresponder a diversos ciclos em uma implementação RT
- Processos:
  - SC_THREAD:
  - SC_CTHREAD: clocked thread.
  - Ambos provêem a manutenção do estado da execução, o que permite interromper o processo

SC_CTHREAD

- Similar a SC_THREAD, mas:
  - Sensitividade restrita a uma única borda do relógio
  - Construções auxiliares:
    - wait(): suspende a thread até a próxima transição do relógio
    - wait(n): suspende por *n* ciclos de relógio
    - reset_signal_is(signal, level): provê um reset síncrono na thread
  - SC_CTHREADs não são executadas no passo de inicialização da simulação



Ciclos de E/S

- Um ciclo de E/S é o intervalo de tempo no qual não ocorre troca de informação entre o processo e o meio externo
- wait() permite sincronizar o processo com sinais externos
  - Os sinais são amostrados nas bordas do relógio
  - Reação dos processos não é imediata, mas sincronizada com a próxima borda do relógio

Algoritmo de Euclides

- Exemplo de modelagem em nível comportamental
- Definição: calcular o máximo divisor comum (GCD) de dois naturais
  - Dados a e b, onde a ≥ 0 e b >0,
    - Se b divide a, então gcd(a, b) = b;
    - Senão, gcd(a, b) = gcd(b, r), onde r é o resto de a / b

GCD

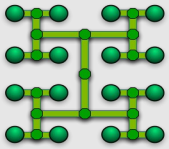
- Módulo tem entradas A e B, naturais e saída C
- Sinal de handshake: ready
  - Quando ready é acionado, dado na saída é válido
  - Mecanismo simplificado: ready é mantido ativo por um ciclo e no ciclo seguinte, desativado e as entradas amostradas para iniciar um novo ciclo de cálculo

SC_MODULE(gcd)

```
SC_MODULE(gcd) {
  sc_in_clk              clk;
  sc_in<bool>            rst;
  sc_in<unsigned>        A, B;
  sc_out<unsigned>       C;
  sc_out<bool>           ready;

  void compute();
  SC_CTOR(gcd) {
     SC_CTHREAD(compute, clk.pos());
     reset_signal_is(rst, true);  // condição reset síncrono
    }
};
```

Universidade de Brasília



LAICO

SC_MODULE(gcd) ...

```
void gcd::compute() {
  unsigned tmp_a = 0, tmp_b;      // reset section
  while (true) {
    C.write(tmp_a);                  // escreve em C
    ready.write(true);   // saida valida
    wait();                          // outro ciclo de E/S
    tmp_a = A.read();                // amostra entradas
    tmp_b = B.read();
    ready.write(false);
    wait();
    ...
```

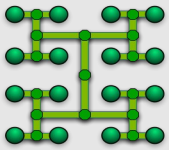
SC_MODULE(gcd) ...

```
// nao ocorre ciclo de E/S durante o calculo do GCD.
while (tmp_b != 0) {
   unsigned r = tmp_a;
   tmp_a = tmp_b;
   r = r % tmp_b;
   tmp_b = r;
   }
 }
}
```

Análise

- Utiliza-se o operador % para calcular diretamente o módulo
- Para síntese, uma forma mais comum seria:

```
while (r >= tmp_b) {
    r = r - tmp_b;
}
```



Análise

- wait()´s são necessários para dar tempo para os sinais serem amostrados (READY)
- Um protocolo mais robusto poderia ser como segue:

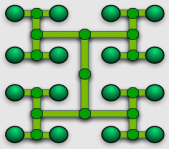
```
C.write(tmp_a);
READY.write(true);
do { wait(); } while (START != true);
```

- Laços infinitos devem ter sempre um wait()
- wait_until() foi depreciado e foi substituído por `do wait(); while (cond);`

Análise ...

- RESET só é monitorado nos ciclos de E/S
- No gcd, durante o laço interno o reset não é monitorado
  - Para ter um comportamento mais detalhado do *reset* teria que introduzir wait()´s no laço
  - Entretanto: wait()´s implicam em perda de velocidade (salvamento do contexto)

Universidade de Brasília



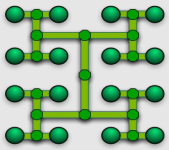
LAICO

Exercício

- Introduzir uma temporização aproximada no código, utilizando o comando *wait(n)* de forma a simular o número de ciclos que o gcd demora para executar

Exercício

- Tranformar a descrição algorítmica em C de um módulo que calcula a raiz quadrada inteira de x em uma descrição SystemC comportamental
  - prever um *reset* síncrono e um sinal de *start* para início do cálculo
  - módulo escreve o resultado, aciona *ready* e espera um novo sinal de *start* para recomeçar



Exercício...

- Raiz quadrada inteira (C):

```
int intsqr1 (unsigned x) {
 int r = 1; d = 2; s = 4;
 while (s <= x) {
  r++; d += 2; s += (d+1);
 }
 return (r);
}
```