Universidade Federal de Santa Catarina - UFSC
Centro Tecnológico - CTC
Departamento de Informática e Estatística - INE
Curso de Ciências da Computação - CCO

INE5351 - Tópicos Especiais em Arquitetura de Computadores I
Princípios de Projeto de Sistemas Computacionais Embutidos

Modelagem de IP Utilizando SystemC RTL
JDiv RoundUp

# Manual do IP

Gustavo Henrique Nihei
Léo Willian Kölln

Florianópolis, Julho de 2007

# 1 Identificação do IP

**Nome:** JDIV Round-Up.

**Função:** Efetua divisão entre números inteiros, retornando quociente inteiro arredondado para cima.

**Uso:** Utilizado para efetuar cálculos de divisões inteiras, como por exemplo no algoritmo para manipulação de imagens no formato JPEG, que é a aplicação funcional neste projeto.

# 2 Interface do IP

## 2.1 Tabela de entradas e saídas

**Tipo:** Controle / Status / Dados / Clock

**Direção:** Entrada / Saída

| Nome | Tipo | Direção | Função |
|---|---|---|---|
| ENABLE | Controle | Entrada | Porta de controle utilizada para habilitar o IP. |
| READY | Status | Saída | Porta de status utilizada para informar da conclusão da operação requisitada. |
| dividend | Dados | Entrada | Porta de dados para informar o valor do dividendo da operação. |
| divisor | Dados | Entrada | Porta de dados para informar o valor do divisor da operação. |
| quotient | Dados | Saída | Porta de dados onde fica disponível o valor da operação. |
| clk | Clock | Entrada | Porta para sinal de clock. |

## 2.2 Mecanismo de sincronização entre IP e wrapper TLM-RTL

A sincronia do adaptador TLM-RTL com o IP é feita através de uma troca de sinais. O adaptador põe os dados no barramento, e chama o IP ativando o sinal ENABLE. Enquanto espera a conclusão dos cálculos do IP, o adaptador entra em loop. O IP, quando terminados os cálculos, retorna o resultado e ativa o sinal de READY. Ao perceber a ativação de READY, o adaptador sai do loop e desativa o ENABLE. O adaptador tem sua própria thread, e a mesma é ativada a cada ciclo de clock.

# 3 Interface do wrapper TLM-RTL

## 3.1 Tabela de entradas e saídas

**Tipo:** Controle / Status / Dados / Clock

**Direção:** Entrada / Saída

| Nome | Tipo | Direção | Função |
|---|---|---|---|
| ENABLE | Controle | Saída | Porta de controle utilizada para habilitar o IP. |
| READY | Status | Entrada | Porta de status utilizada para informar da conclusão da operação requisitada. |
| a | Dados | Saída | Porta utilizada para comunicação com a porta "dividend" do módulo JDIV. |
| b | Dados | Saída | Porta utilizada para comunicação com a porta "divisor" do módulo JDIV. |
| c | Dados | Entrada | Porta utilizada para comunicação com a porta "quotient" do módulo JDIV. |
| REQ | Dados | Entrada | Protocolo de comunicação "protocol::REQ", utilizado para as requisições ao JDIV. |

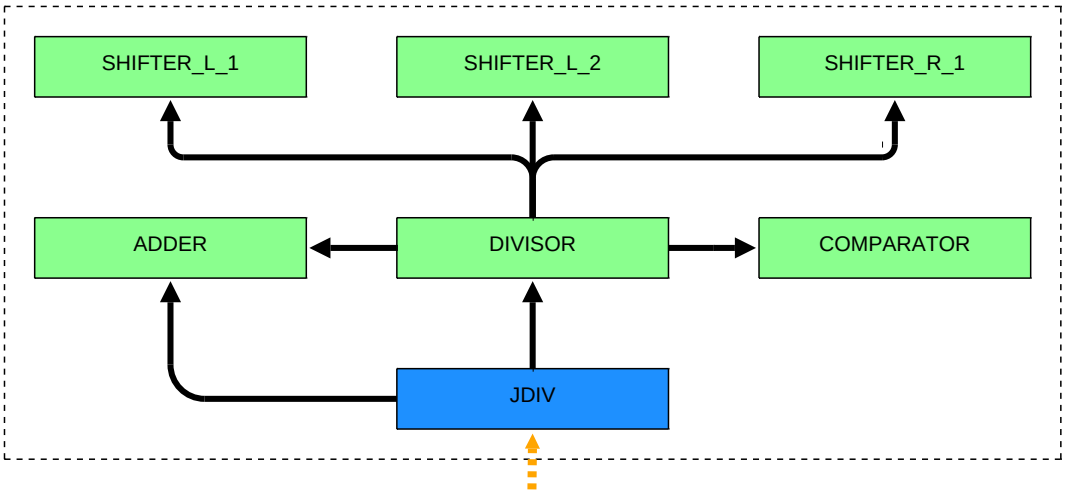
| | | | |
|---|---|---|---|
| RSP | Dados | Saída | Protocolo de comunicação “protocol::RSP”, utilizado para as respostas provenientes do JDIV. |
| clk | Clock | Entrada | Porta para sinal de clock. |

### 3.2 Mecanismo de sincronização entre wrapper TLM-RTL e wrapper TLM-SimpleBus

Como o mecanismo de controle do IP está modelado em nível RTL, no mesmo {além de módulos que com ele se comunicam}não podem existir chamadas “wait()”. Portanto, um outro método de sincronismo teve de ser buscado pra sincronizar os dois adaptadores. Optamos por utilizar a implementação de filas do SystemC, sc_fifo, pois a mesma se auto-sincroniza.
O adaptador, em loop infinito, inicia uma leitura da fila de requisições. Através do mecanismo de auto-sincronia da sc_fifo, a programa somente prosseguirá quando for possível ler um elemento da fila. Então, envia a requisição obtida para o adaptador TLM-RTL, de onde aguarda uma resposta. Após retornada a resposta, salva-a numa fila de respostas, e então ativa o sinal READY para comunicação interna. A fila de respostas será lida quando o resultado do cálculo do IP tiver de ser enviado para o SimpleBus.

## 4 Estrutura do IP

### 4.1 Diagrama de blocos




## 5 Principal algoritmo do IP

### 5.1 Código do algoritmo de divisão

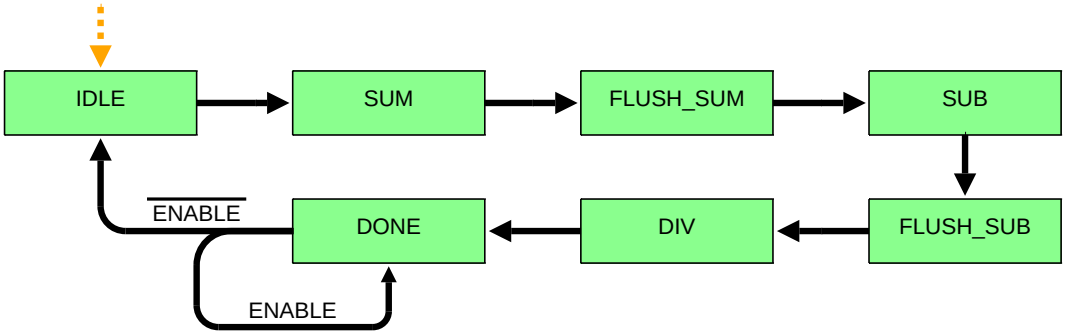
**Referência:** http://www.bearcave.com/software/divide.htm

```
01      void unsigned_divide(unsigned int dividend, unsigned int divisor,
unsigned int &quotient, unsigned int &remainder)
02      {
03            unsigned int t, num_bits;
04            unsigned int q, bit, d;
05            int i;
06
07            remainder = 0;
08            quotient = 0;
```

```
09
10            if (divisor == 0)
11                  return;
12
13            if (divisor > dividend)
14            {
15                  remainder = dividend;
16                  return;
17            }
18
19            if (divisor == dividend)
20            {
21                  quotient = 1;
22                  return;
23            }
24
25            num_bits = 32;
26
27            while (remainder < divisor)
28            {
29                  bit = (dividend & 0x80000000) >> 31;
30                  remainder = (remainder << 1) | bit;
31                  d = dividend;
32                  dividend = dividend << 1;
33                  num_bits--;
34            }
35
36            dividend = d;
37            remainder = remainder >> 1;
38            num_bits++;
39
40            for (i = 0; i < num_bits; i++)
41            {
42                  bit = (dividend & 0x80000000) >> 31;
43                  remainder = (remainder << 1) | bit;
44                  t = remainder - divisor;
45                  q = !((t & 0x80000000) >> 31);
46                  dividend = dividend << 1;
47                  quotient = (quotient << 1) | q;
48                  if (q)
49                        remainder = t;
50            }
51      }
```

# 6 Máquina de estados da unidade de controle do IP

## 6.1 Diagrama de transição entre estados




## 6.2 Tabela de ações disparadas

| Estado | Ações | UF's |
|---|---|---|
| IDLE | READY=false;<br>reg_a=dividend;<br>reg_b=divisor; | |
| SUM | READY=false;<br>ADDER_ENABLE=true;<br>adder_a=reg_a;<br>adder_b=reg_b; | 1 Somador |
| FLUSH_SUM | READY=false;<br>ADDER_ENABLE=false;<br>ADDER_RESET=true;<br>reg_aux=adder_sum; | 1 Somador |
| SUB | READY=false;<br>ADDER_ENABLE=true;<br>adder_a=reg_aux;<br>adder_b=-1L; | 1 Somador |
| FLUSH_SUB | READY=false;<br>ADDER_ENABLE=false;<br>ADDER_RESET=true;<br>reg_aux=adder_sum; | 1 Somador |
| DIV | READY=false;<br>DIV_ENABLE=true;<br>div_dividend=reg_aux;<br>div_divisor=reg_b; | 1 Divisor |
| DONE | READY=true;<br>DIV_ENABLE=false;<br>quotient=div_quotient; | |