PSI-3451 Projeto de CI Lógicos Integrados

Aula 6- Conceitos relacionados aos circuitos modelados em VHDL

Nesta aula o aluno receberá 2 arquivos relativos ao projeto ***Snake***:

- Máquina de estados com dados (*fsmd_1*): uma máquina de estados descrita no modelo comportamental, sendo as operações definidas na forma de transferências entre registradores (RT).
- Máquina de estados com *datapath* (*fsmd_2*): uma máquina de estados descrito de forma mista: a FSM no modelo comportamental sendo as operações descritas na forma estrutural.

A seguir apresentamos uma breve descrição a respeito dos arquivos.

1. A funcionalidade das Máquinas de Estados com Dados e com Datapath

Uma máquina de estados finitos cujas transições decorrem de operações não Boleanas é designada pela sigla FSMD (observe o D final). Numa FSM todas as transições só dependem de operações Booleanas. O projeto de uma FSM somente requer ferramentas de síntese lógica (combinatória e sequencial). O projeto de uma FSMD é dividido em duas partes separadas e interativas. Uma parte é a unidade de cálculo das operações, designada por *datapath* que pode ser sintetizado através de ferramentas de síntese RTL (não serão usadas neste curso). A outra parte é a unidade de controle que constitui uma FSM tradicional. A interação entre estas duas partes se dá pelos sinais designados por *flags*.

Os dois arquivos fornecidos apresentam exatamente a mesma funcionalidade. A *fsmd_1* é descrita inteiramente em VHDL comportamental enquanto a *fsmd_2* apresenta uma descrição mista comportamental/estrutural.

A FSMD na qual se baseiam os arquivos desta aula 6 corresponde ao módulo *FSM_Food*, responsável pelo posicionamento do alimento no tabuleiro do jogo, como já apresentado no documento sobre a funcionalidade do *Snake*. O seu diagrama de estados simplificado é reapresentado na Figura 1. O aluno poderá reparar que o estado adicional de *Food_OK* foi adicionado ao modelo correspondente do documento do *Snake* para que a máquina seja modelada como do tipo Moore.

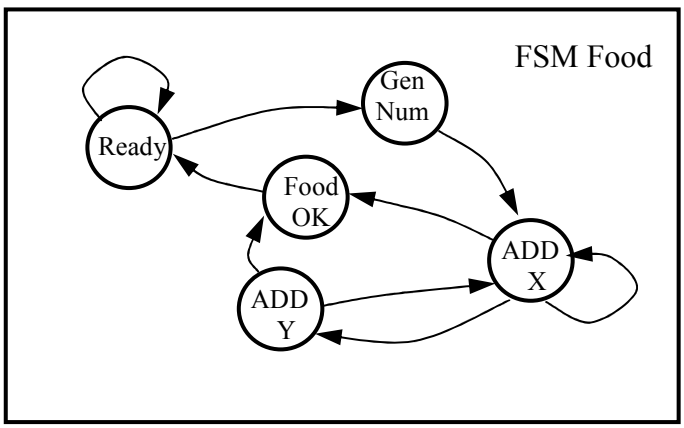

Figura 1. Diagrama de Estados do FSM_Food

É importante observar que no texto originalmente fornecido sobre o projeto *Snake* (na aula 1 deste curso), a máquina *FSM_Food* é uma das FSM do módulo *Control Snake*. Neste mesmo texto as operações são realizadas nas unidades funcionais do módulo *datapath*.

Nesta aula, parte das unidades funcionais do *datapath* serão incorporados, implicitamente (*fsmd_1.vhd*) ou explicitamente (*fsmd_2.vhd*). Desta forma, tanto o controle como as operações serão englobadas.

O módulo topo nos 2 casos será o mesmo e a *entity* conterá apenas os sinais de controle indicados na Figura 2. O sinal de entrada *fsm_m_start* é checado no estado *Ready*, indicando se a *FSM_Main* (veja texto do projeto *Snake*) coloca a *FSM_Food* em operação. Na saída existe o *flag fsm_m_done* a ser enviado para *FSM_Main* indicando se a tarefa de posicionar o alimento no tabuleiro do *Snake* foi realizada ou não.

A transição entre os demais estados é realizada dependendo de dois sinais internos: os *flags ofc_of_x* e *comp_body_flag*. O primeiro verifica no estado *Add_X* se ocorreu overflow na soma ao se calcular o endereço (da memória) da nova coluna x (para a atual linha y). Em caso positivo passa-se para o estado *Add_Y* para realizar o incremento da coordenada y (avança para a próxima linha). Por sua vez, *comp_body_flag* é verificado em *Add_X* e *Add_Y*, identificando se a posição do tabuleiro está livre. Se estiver o alimento é colocado no tabuleiro no estado *Food_OK*. Estes *flags* serão ainda abordados posteriormente.

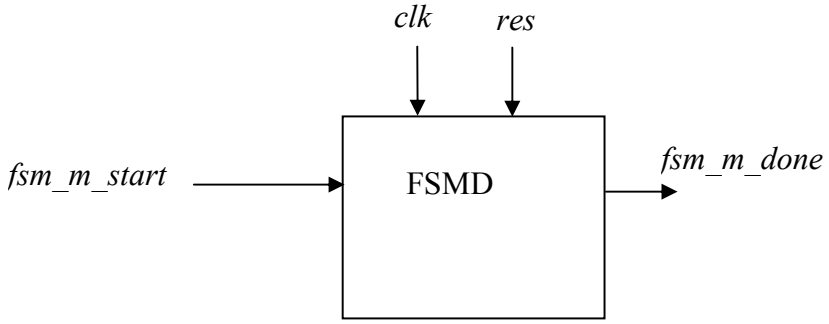

Figura 2. Entradas e saídas dos módulos FSM_Food com as operações

2. Máquinas de Estados com Dados

O arquivo *fsm_1.vhd* contém o modelo VHDL do módulo *fsm_food_with_data.* Segundo o *template* apresentado nos slides desta aula, a descrição consiste numa série de RTs (transferência entre registradores). As transferências necessárias para o módulo *fsm_food_with_data* são apresentadas na Figura 3. Há as seguintes observações:

1) Cada registrador *nome_reg* tem associado os seguintes nomes à sua entrada e saída, respectivamente: *nome_reg_next* e *nome_reg*.

2) A memória ram é um caso particular de registrador. Para a sua descrição RT, os valores de todos os seus endereços são atualizados a cada ciclo de relógio (*ram*← *ram_next*). Trata-se de uma memória ***dual-port para escrita e leitura***. Dado um valor de endereço, a leitura é realizada automaticamente em todos os ciclos, independentemente deste valor de saída ser utilizado de fato. Por outro lado, a escrita deve ser sinalizada no ciclo desejado. Nas RTs de uma FSMD usa-se um sinal de habilitação *wren* (*write enable*) a fim de conectar o dado a ser escrito com o *ram_next* do endereço correspondente.

3) Os fluxos de operações são montados a partir dos trechos combinacionais enumerados de 1 a 12. Por exemplo, para que *mem_addr* receba o valor de (*reg2*+16) (*reg2*+1 *shift 4*), os trechos 4 e 6a devem ser conectados.

4) Os trechos 9, 10 e 12, marcados em azul ocorrem sempre, independente do estado.

5) Há dois blocos de atraso para que a escrita de "10000000" (código correspondente a alimento na memória) seja realizada. Ela é realizada no estado *Food_OK*, porém o endereço é computado dois ciclos antes.

6) A operação reg_2+1 que aparece na figura não será utilizada. Esta opção existe no projeto *Snake*, porém será utilizada apenas na máquina FSM_Step.

A Figura 4 apresenta o ASMD correspondente do módulo *fsm_food_with_data*, realizado no modelo Moore. Nos blocos dos estados aparecem as atribuições de valores de saída e as conexões combinacionais (para estabelecer valores de entrada de registradores, em *_next). Estas são descritas como a combinação dos trechos enumerados da figura 3.

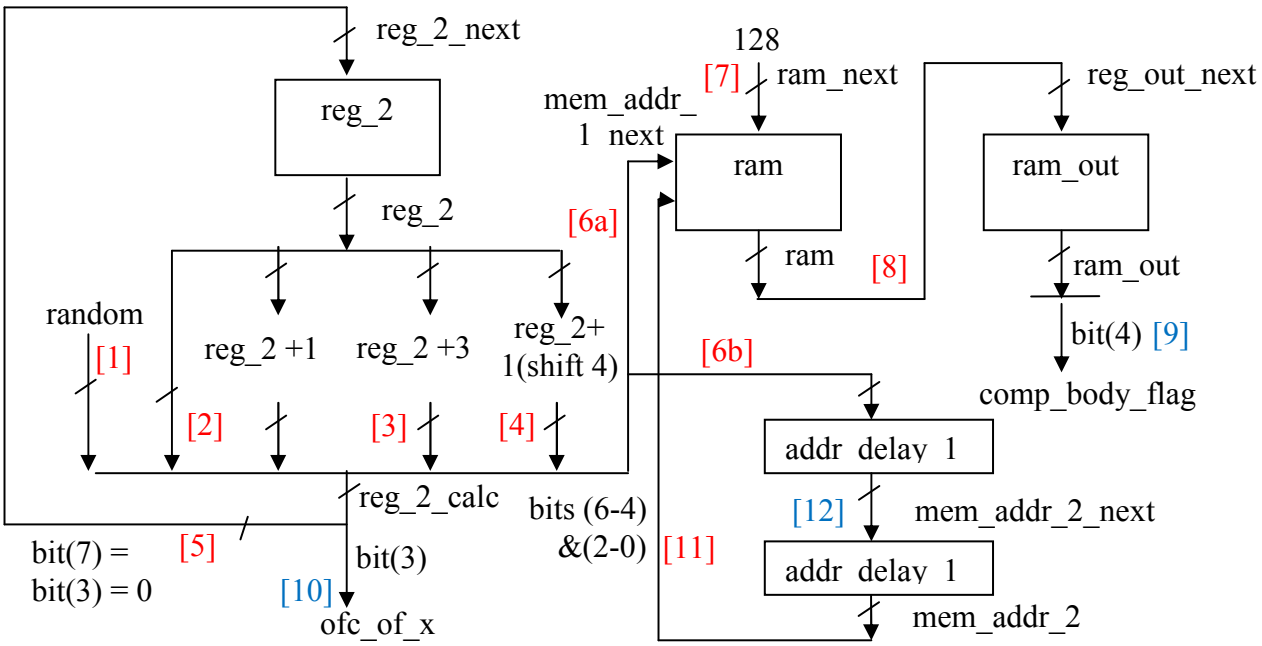

Figura 3. Registradores e transferências previstas no *fsm_food_with_data*

3. Máquinas de Estados com Datapath

3.1 Composição Estrutural

O arquivo *fsmd_2.vhd* contém o modelo VHDL do módulo *fsm_food_with_datpath* que seguirá a figura da descrição mista (comportamental e estrutural) apresentada no último slide desta aula. O módulo representa uma *entity* topo contendo dois submódulos, *fsm_2* e *datapath*_2. O *datapath* conta com os seguintes sub-módulos: registradores, unidades funcionais e muxes.

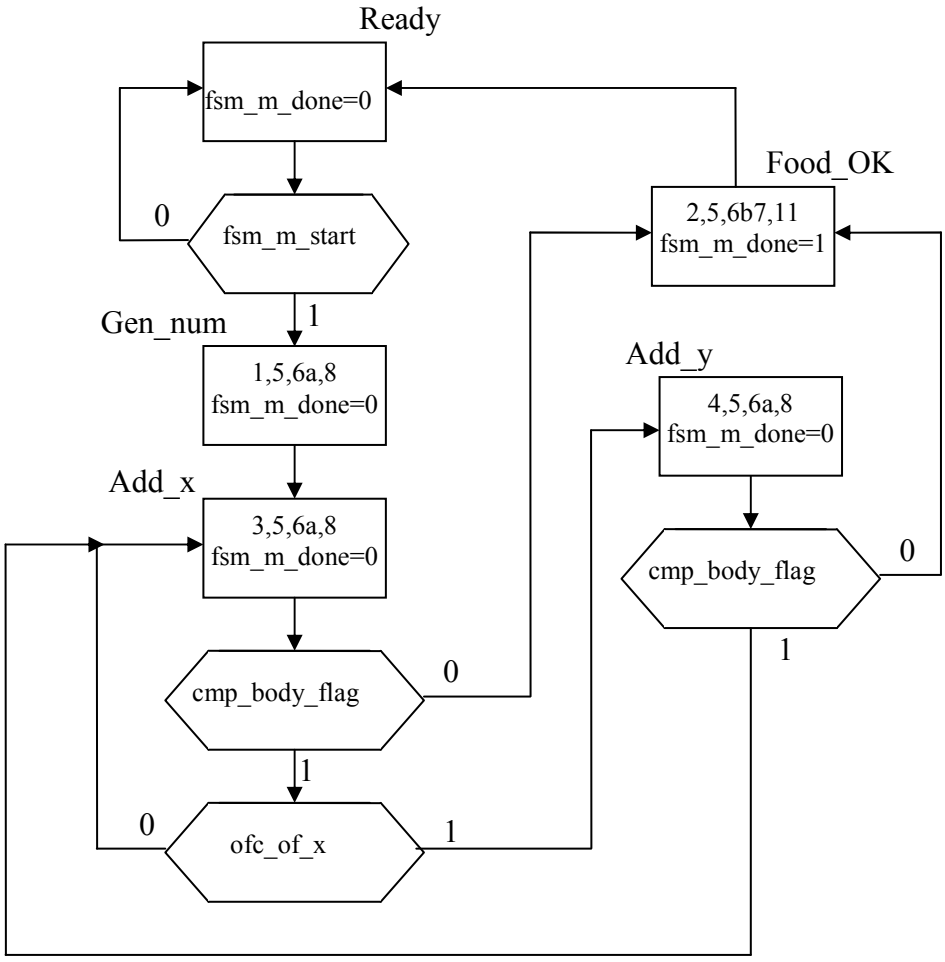

Figura 4. ASMD correspondente ao módulo *fsm_food_with_data*

A. Registradores

Os registradores correspondem a vetores de registradores tipo D. A descrição VHDL é idêntica à de um FF de um bit, porém as transferências de valores são estendidas a todos os seus bits. Para a sua utilização no *datapath*, um registrador deve ter um sinal de *load* (*enable* ou *clear* também são denominações encontradas). Como ilustrado na Figura 5, se o sinal de *load* estiver ativado, na ativação do *clock*, ocorre a transferência do vetor de entrada *d* para a saída *q*. Já quando o sinal de *load* está desativado, o vetor de saída permanece com o valor anterior, independente do valor em *d*.

Logo, o sinal de *load* de um registrador *R* é um sinal de controle que a FSM deve enviar ao *datapath* sempre que uma operação de escrita no registrador *R* seja especificada em um estado.

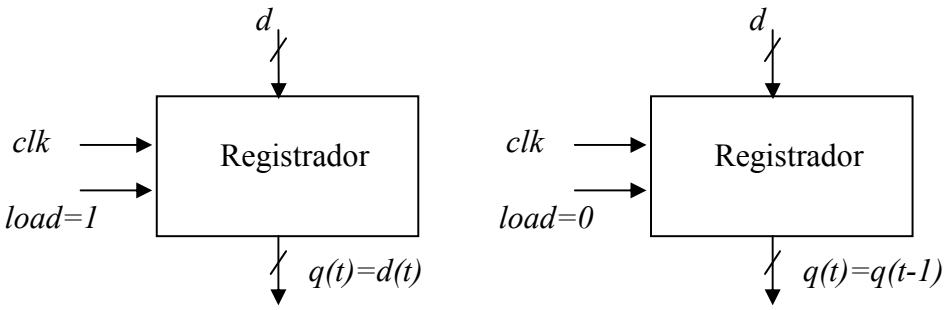

Figura 5. Registrador

B. Unidades Funcionais

As unidades funcionais são as responsáveis pelas operações lógicas, aritméticas ou simplesmente de encaminhamento de dados. No projeto usaremos os blocos *gen_num* e ALU já vistos em aula anteriores (com seus respectivos sinais de controle de muxes). Adicionalmente utilizaremos os seguintes blocos:

- Memória: o módulo ram é uma descrição comportamental de uma memória com funcionalidade *dual-port* já descrita na seção 2. A Figura 6 mostra as suas entradas e saídas. Observar que o sinal *wren_a* (*write enable*) deve ser ativado somente nos estados da FSM em que um dado deva ser escrito neste registrador. Nos demais este sinal deve permanecer desativado.

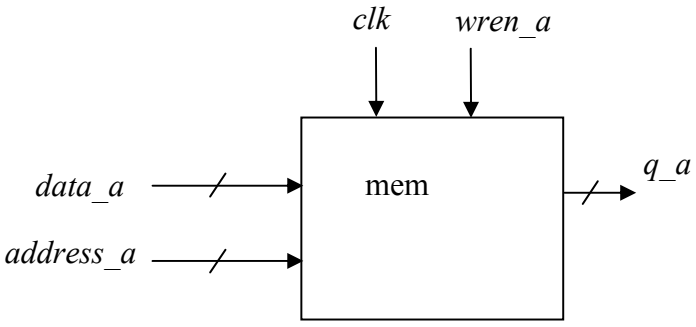

Figura 6. Interface da memória

- Correção de bits de overflow: o módulo combinacional *ofc_correction* realiza a "correção" do valor de endereços após a realização da soma (na ALU) cujo resultado tem os seus bits 3 e 7 (caso de palavra de 8 bits) zerados. O objetivo é eliminar a sinalização de *overflow* que, por ventura, tenha ocorrido nestes bits (para permitir a detecção de novas ocorrências de *overflow*)

- Geração de código: o módulo *code_gen* é um bloco que realiza a conversão para o tipo *byte* de valores dados no tipo CODE. Por exemplo, no caso que nos interessa neste momento, o valor FOOD (um valor do tipo CODE) deve ser escrito em alguma posição na memória. O módulo o traduz para "10000000".

- Comparador: o módulo *comparator* verifica se o dado armazenado na posição de memória de interesse é parte da cobra. Para isto verifica se o bit 4 é igual a 1.

C. Muxes

Os muxes que fazem parte da rede de roteamento (*routing network*), apresentado nos slides da aula, serão implementados no estilo *dataflow*, como testado nas aulas anteriores.

A Figura 7 mostra o *datapath* do *fsmd_2*, com os blocos e seus sinais de controle. Observe-se que os blocos *num_gen* e ALU já foram testados em outras aulas. Os sinais internos com a nomenclatura usada no arquivo *datapath.vhd* estão em cor azul.

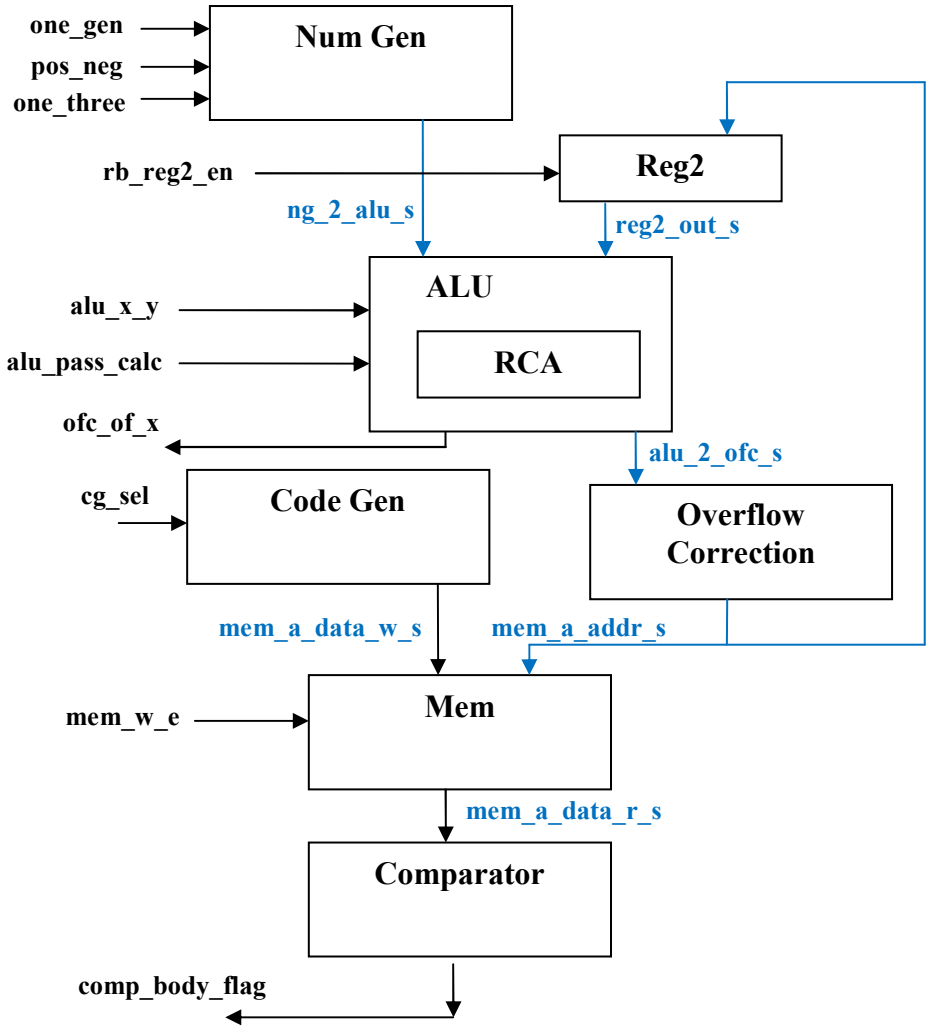

Figura 7. O *datapath* da *fsmd_2*

3.1 ASM a partir do ASMD

Dado que, agora, as RTs da ASMD utilizada em *fsmd_1* são efetuadas no *datapath* da Figura 7, a ASMD é substituída por uma ASM equivalente, levando-se em conta os sinais de controle correspondentes. Cada RT da ASMD deve ser realizada por um conjunto específico de unidades funcionais, muxes e registradores, sendo os sinais de controle ajustados adequadamente.

A Figura 8 recodifica estas situações, com os *flags* correspondentes aos trechos descritos na ASMD. Ao codificar o VHDL da FSM o projetista deve ter o cuidado de ajustar os valores de todos os *flags*. Por exemplo, nos quadros vermelho, azul e verde, o sinal *load* do registrador *reg_2* deve ser ativado (='1'); isto significa que, para todos os demais estados, o seu valor deve ser ajustado para o valor inverso (='0').

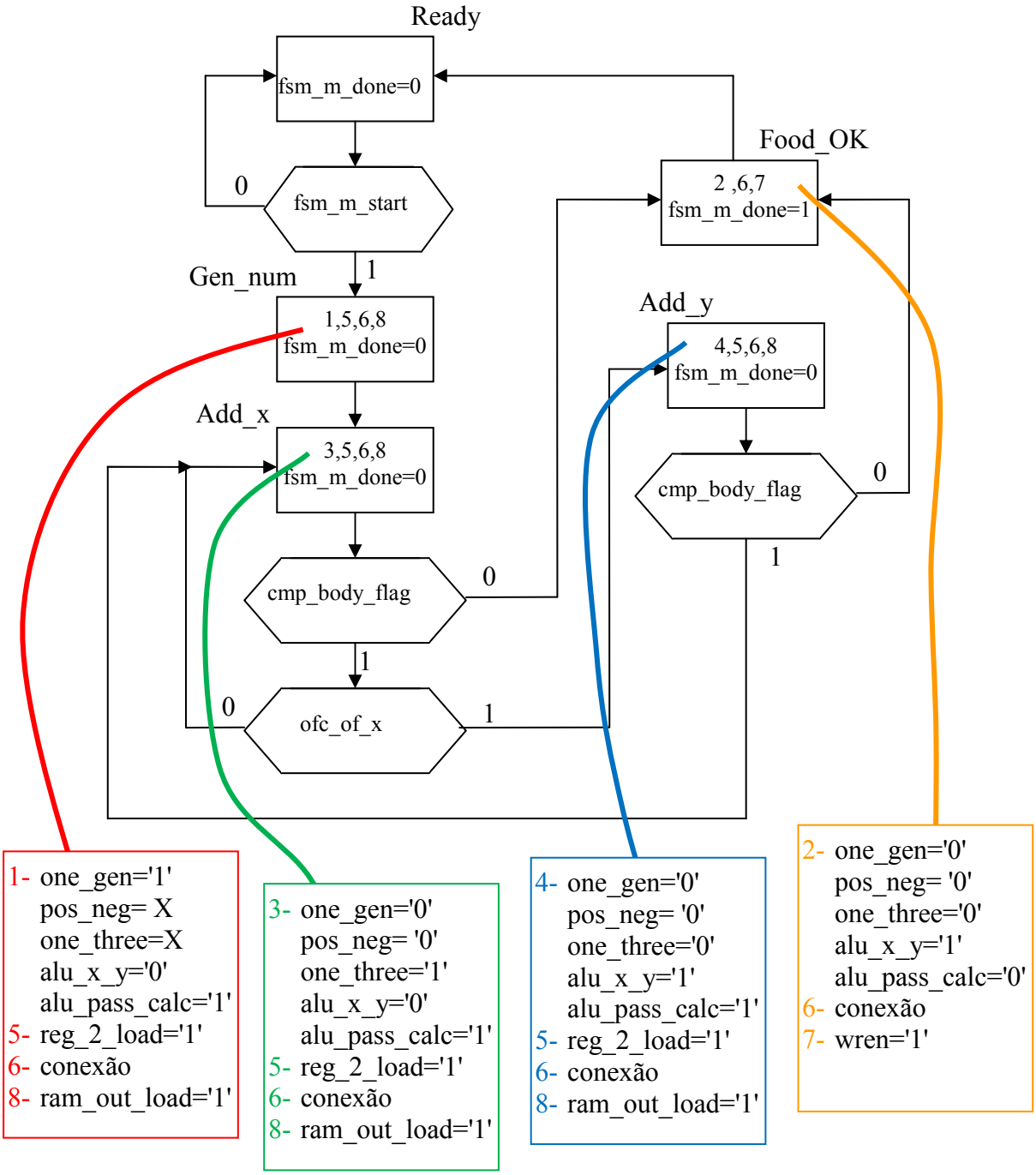

Figura 7. Correspondência entre RTs e ativação de flags