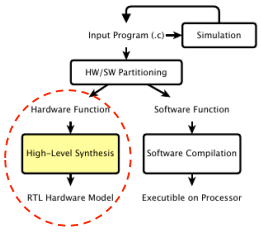
Prototipação de Circuitos Integrados Digitais

Síntese de alto-nível

DI-UFPE/GRECO

Síntese de Alto-nível(HLS)



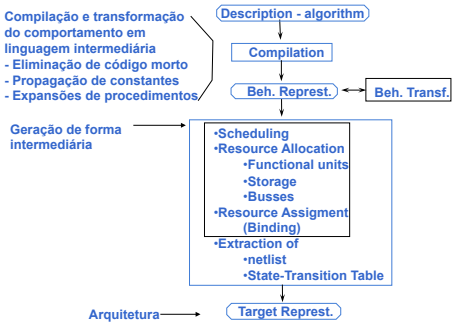
Síntese de Alto-nível

- **Definição**
  - É a fase do projeto de um sistema digital na qual uma arquitetura é definida para a descrição funcional deste sistema baseado em um conjunto de restrições e objetivos.
  - Na síntese algorítmica um projeto é especificado em termos de passos computacionais compostos de um conjunto de operações executadas entre dois sucessivos I/O e pontos de sincronização. Um passo de computação pode necessitar de vários ciclos de relógio.
  - A principal função da síntese de alto-nível é partir estes passos de computação em um conjunto de ciclos de relógio.

Síntese de Alto-nível

- O resultado da síntese de alto-nível é em geral a descrição de um sistema síncrono no domínio arquitetural no nível RT (Register Transfer) composto por um *Data path (caminho de dados)* e *Controller (controlador)*
  - **Parte de Processamento de dados (data path) - manipula entrada de dados para gerar a saída desejada**
  - **Parte de controle(Controller) - controla a seqüência e o tipo de manipulação de dados**

Síntese de Alto-nível

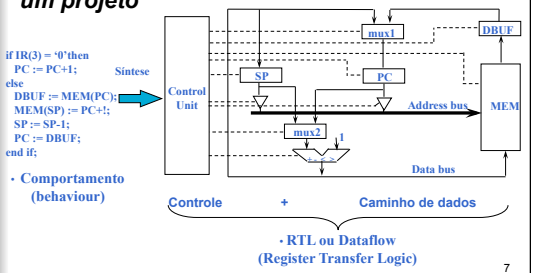


Síntese de Alto-nível

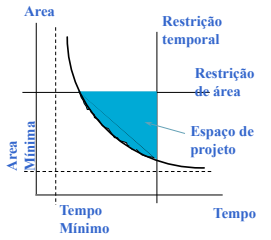
- Vantagens
  - Redução no tempo de projeto
  - Diminuição do número de erros
  - Possibilidade de "pesquisar" o espaço de projeto
  - Documentação
  - Tornar "mais fácil" o desenvolvimento de circuitos digitais

Síntese de Alto-nível

- ***Diferentes Vistas de um projeto***



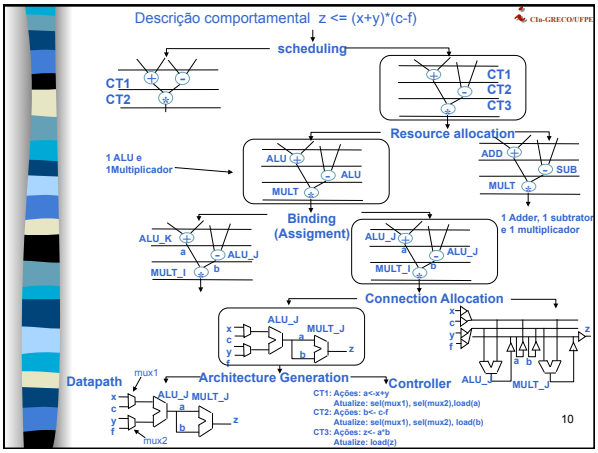
Síntese de Alto-nível (hardware)

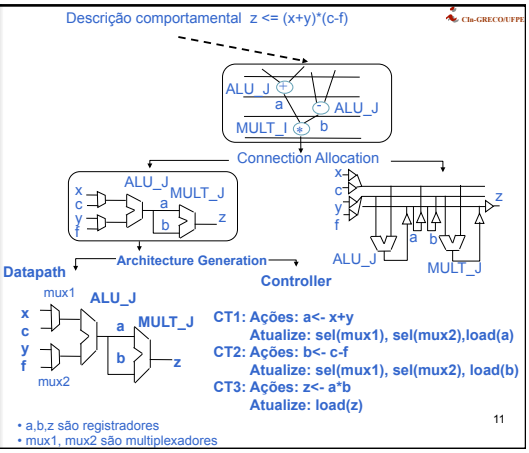


- Escalonamento voltado à área
  - Recursos - fixos (unidades funcionais)
  - Delay - variável
- Escalonamento voltado à tempo
  - Delay - fixo
  - Recursos - variável

Síntese de alto-nível

- Tarefas
  - Scheduling (Escalonamento)
    - É a alocação de operações a períodos de tempo sujeitos a certos restrições e minimização de uma função custo.
  - Resource Allocation (Alocação de recursos)
    - É a determinação do tipo e número de recursos existentes para sintetizar o sistema.
      - » Número e tipos de unidades funcionais (adicionadores, multiplicadores, ...).
      - » Número e tipo de elementos de armazenamento.
      - » Número e tipo de barramentos (busses).
  - Resource Assignment (Binding) (atribuição de recursos)
    - É a fase na qual os elementos são instanciados no sistema digital.
      - Operações para unidades funcionais.
      - Valores a serem armazenados a instâncias de elementos de armazenamento.
      - Transferência de dados para instância de barramento.



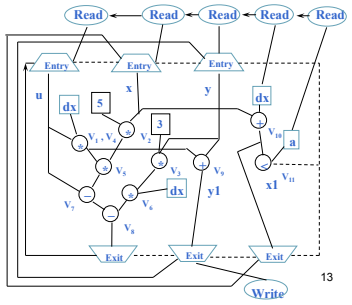


Scheduling

- **Se recursos são disponíveis $|R_k|$ -> arbitrariamente grande, para o mínimo tamanho do schedule, S corresponde ao tamanho crítico.**
- **Dado um certo tamanho de schedule e permitindo recursos sem limites, operações podem ser executadas em diferentes posições que residem entre:**
  σ_{ASAP} -> mais breve
  σ_{ALAP} -> mais tárdio
  Os dois valores σ_{ASAP} e σ_{ALAP} dependem das restrições de precedência.
- **Definição de tempo ASAP e ALAP**
  Dado um DFG(V,E) acíclico, um schedule de tamanho S e recursos ilimitados,
  - **tempo ASAP é o tempo mais breve para execução de uma operação**
  - **tempo ALAP é o tempo mais distante para a execução de uma operação**

Exemplo: Síntese de uma equação diferencial

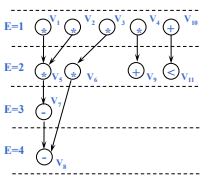
```
while (x<a) loop
   x1:= x+dx;
   u1:=u-5*x*(u*dx)-3*y*dx;
   y1:=y+(u*dx)
   x:=x1; y:=y1;
end loop;
```



ASAP Scheduling

ASAP
Dado um DFG(V,E) acíclico, um schedule de tamanho S e um número ilimitado de recursos, o tempo ASAP representa o mais recente passo de controle possível para a implementação de uma tarefa (DGF).

- **Implementação usando algoritmo ASAP**



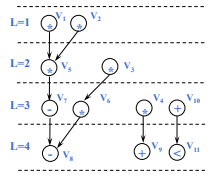
ASAP Algorithm:
for each node $v_i \in V$ do
if Pred $v_i = \phi$ then
$E_i = 1$;
$V = V - \{v_i\}$;
else
$E_i = 0$;
endif
end for

while $V \neq \phi$ do
for each node $v_i \in V$ do
if ALL_NODES_SCHED(Pred v_i, E) then
$E_i = MAX(Pred\ v_i, E) + 1$;
$V = V - \{v_i\}$;
endif
end for
end while

ALAP Scheduling

ALAP
Dado um DFG(V,E) acíclico, um schedule de tamanho S e um número ilimitado de recursos, o tempo ALAP representa o passo de controle mais longo possível para a implementação de uma tarefa (DGF).

- Implementação usando algoritmo ALAP



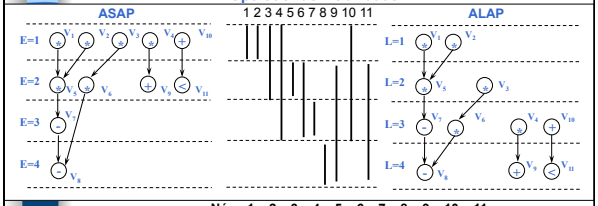
Exemplo: O mínimo nível sucessor de V1 é o nível 2. Assim o nível de v1=nível 2 -1.

ALAP Algorithm:
for each node $v_i \in V$ do
if Succ $v_i = \phi$ then
$L_i = T$;
$V = V - \{v_i\}$;
else
$L_i = 0$;
endif
end for

while $V \neq \phi$ do
for each node $v_i \in V$ do
if ALL_NODES_SCHED(Succ v_i, L) then
$L_i = MIN(Succ\ v_i, L) - 1$;
$V = V - \{v_i\}$;
endif
end for
end while

Mobilidade

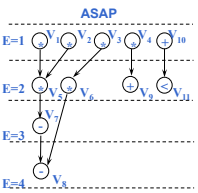
- Mobilidade de operações é o número de passos de controle do mais próximo ao mais distante passo possível de controle no qual uma operação pode começar a execução, tal que o caminho crítico não seja aumentado.
- **$M_o = \sigma_{ALAP} - \sigma_{ASAP}$**

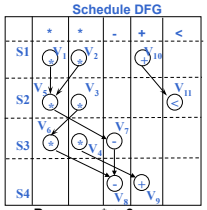


Nó:	v1	v2	v3	v4	v5	v6	v7	v8	v9	v10	v11
Operação:	*	*	*	*	*	*	-	-	+	+	<
Mobilidade:	0	0	1	2	0	1	0	0	2	2	2

Scheduling
Mobilidade - Restrição de Recursos

- **Lista de prioridades**
  - ASAP+Função prioridade (mobilidade) para cada recurso
  - Conflitos de recursos resolvido por função prioridade
  - Método construtivo; schedule, reavaliar prioridade, sort, ..





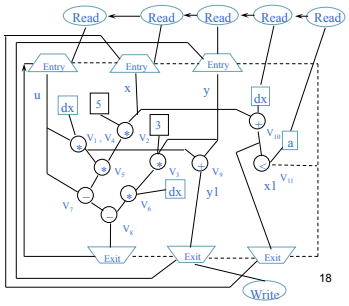
Lista de prioridades: (*): v1<0>,v2<0>,V3<1>,v4<2>
(+): v10<2>
(-): Nenhum
(<): nenhum

Recursos: * -> 2
+ -> 1
- -> 1
< -> 1

Síntese de uma equação diferencial

- $\frac{d^2y}{dx^2} + 5\frac{dy}{dx}x + 3y = 0$

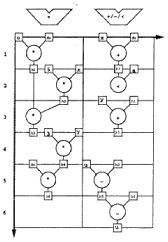
```
Use work.sys.pack.all;
ENTITY differ IS
PORT (inport: IN integer;
      outport: OUT integer;
      sysclock: IN bit );
END differ;
ARCHITECTURE his_synthesis OF differ
IS
BEGIN
    PROCESS
      VARIABLE a, dx, x, u, y: Integer;
      VARIABLE x1, y1: integer;
    BEGIN
      cycles (sysclock, 1)
      a:= inport;
      cycles (sysclock, 1)
      dx:= inport;
      cycles (sysclock, 1)
      y:= inport;
      cycles (sysclock, 1)
      x:= inport;
      cycles (sysclock, 1)
      u:= inport;
    LOOP
      cycles(sysclock, 7)
      x1:= x+dx;
      y1:=y+(u*dx)
      u:=u-5*x*(u*dx)-3*y*dx;
      x:=x1; y:=y1;
    EXIT WHEN NOT (x1 < a)
END LOOP;
outport <= y;
END PROCESS;
END his_synthesis;
```



Mapeamento da Equação Diferencial

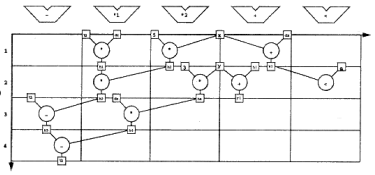
Design 1

Modelo com o mínimo de recursos (ALAP)



Design 2

Modelo com o Máximo de recursos (ASAP)



Síntese da Equação Diferencial

Etapas da síntese

- Alocação das Unidades funcionais
- Escalonamento baseado nas limitações de recursos de hardware
- Atribuição das Unidades Funcionais em cada estágio de Scheduling
- Atribuição de registradores
- Extração de multiplexadores

Síntese da Equação Diferencial

- **Allocation**

Component	Delay ns
ALU(+,-,<)	40
Adder	35
Subtracter	35
Comparator(<)	10
Parallel Multiplier	80
Register	2
2:1 Multiplexor	2
Tristate Driver	2
2-Stage Pipelined Multiplier	90

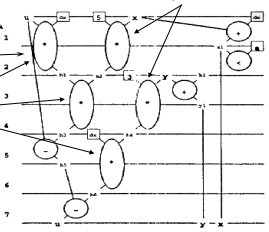
- Freqüência de operação=1/50ns
- **Scheduling**

Passos de controle

Multiplicador M2

Unidades funcionais:
- 2 multiplicadores paralelos
- 1 ALU

Multiplicador M1



- **Assigment**

Func. Units	Mult. M1	Mult. M2
Oper.	1, 3, 5	2, 4

Alocação de registradores

- Tempo de vida(lifetime) de uma variável é o tempo no qual a variável fica guardada em um registrador.

Obs: levar em conta áreas de interconexão.

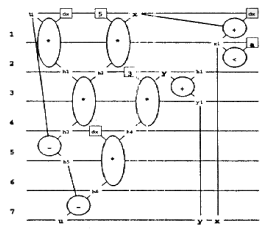
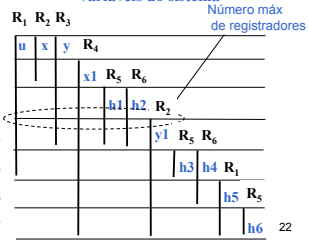


Tabela de tempo de vida das variáveis do sistema

Número máx de registradores



Decomposição de Registradores

Atribuição de Recursos

Registradores	R1	R2	R3	R4	R5	R6
Values	u h5	x y1	y	x1	h1 h3 h6	h2 h4

- Resultado final

Extração de Multiplexadores (exemplo)

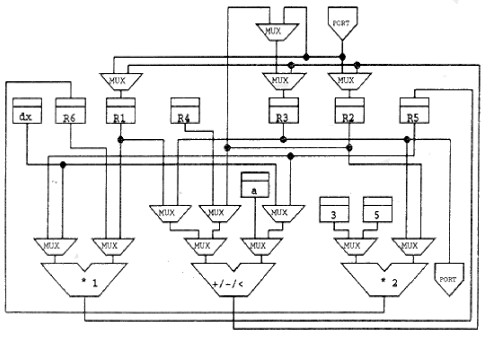
- **Entradas do multiplicador M1**

Control step	1	2	3	4	5	6	7
Register left input of M1	R1	R1	R5	R5	Rdx	Rdx	-
Register right input of M1	Rdx	Rdx	R6	R6	R6	R6	-

- **Neste caso em particular observamos que o multiplexador do lado esquerdo do multiplicador M1 deve possuir três entradas, enquanto que o multiplexador direito deve ter duas entradas.**

Alocação de registradores e multiplexação da via de dados

Arquitetura com via dados multiplexado

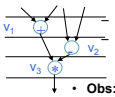


Representação Interna

- **Modelos**
  - **Formas Intermediárias Orientadas a Linguagem**
    - Data-Flow Graph
    - Control-Flow Graph
    - Control-Data-Flow Graph
  - **Forma Intermediária Orientada a Arquitetura**
    - FSMD - Finite State Machine com Modelo Data Path
    - FSMC - Finite Statae Machine com Modelo Co-Processor

Representação Interna

- **Data-Flow Graph**
  - Este tipo de representação é o mais popular para programas em síntese de Alto-nível. Seus nós representam operações do programa e arcos representam valores.
  - A função do nó é gerar um novo valor na sua saída em função de suas entradas.
  - Definição:
    - **Um DFG é definido por um grafo G=(V,E), onde:**
      - **V = {v_1,..., v_n} é um conjunto composto por nós.**
      - **E ⊂ V x V é uma relação assimétrica de fluxo de dados, cujos elementos são arcos dirigidos.**
      - **Os nós representam operações.**
      - **Um arco dirigido (e_{ij}) entre dois nós v_i e v_j ε V existe se o dado produzido pela operação o_i (representado por v_1) é consumido pela operação o_j (representado por v_2).**
    - **Obs: Este modelo é poderoso para representarmos expressões. No entanto, ele não é apropriado para representar estruturas de controle.**



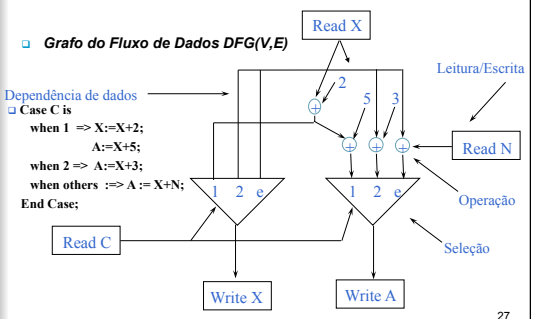
Representação Interna

- ***Grafo do Fluxo de Dados DFG(V,E)***

Dependência de dados

- Case C is

when 1 => X:=X+2;
A:=X+5;
when 2 => A:=X+3;
when others :=> A := X+N;
End Case;



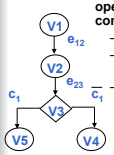
Representação Interna

- **Control-Flow Graph**
  - É a representação mais adequada para modelar controle em projetos que contenham loops, sincronização, etc., ou seja, características que reflitam propriedades inerentes de controladores.
  - **Definição:**
    - **Um CFG é definido por um grafo G=(V,E), onde:**
      - **V = {v_1,..., v_n} é um conjunto composto por nós.**
      - **E ⊂ V x V é uma relação de controle de fluxo composto por uma seqüência de setas dirigidas.**
    - **Os nós em um grafo pertencem a duas classes:**
      - **Nós para operações do tipo atribuição, operações aritméticas e chamadas de procedimentos representadas (Vo)**
      - **Nós de desvio modelam declarações condicionais tais como If, Case e loops (Vb)**
      - **Onde V = Vo U Vb**
    - **Obs: Os nós de operações têm apenas um sucessor e os nós de desvio podem ter mais que um sucessor.**

Representação Interna

- **Control-Flow Graph**
  - Características de arcos direcionados
    - **Um arco direcionado representa a relação de precedências entre dois nós v_i e v_j . Um arco tem um peso que está diretamente relacionado a uma condição $Cond_{ij}$, que significa que uma operação representada por v_j será executada se v_i é executada e a condição $Cond_{ij}$ é verdadeira.**
      - **Propriedades dos arcos:**
      - **Se v_i é um nó representando uma operação e v_j é um nó imediatamente sucessor de v_j então $Cond_{ij}$ =1 (verdadeiro).**
      - **Se v_j é um nó representando um desvio, e (v_{j1},.....,v_{jk}) são k sucessores imediatos de v_j então:**
        $Cond_{j,im} \cap Cond_{j,ih}$ = 0, para todos m,h ε (1, ...,k) e m ≠ h, e $\sum_{i=1}^{k} Cond_{j,ih}$ = 1.
        Esta propriedade assegura que o CFG é determinístico.

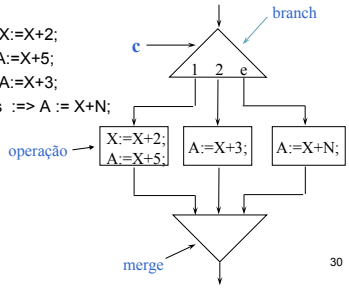
Obs: Embora CFG modele bem estruturas de controle, ele é limitado à análise e transformações de fluxo de dados.



Representação Interna

- ***Grafo de Fluxo de Controle CFG(V,E)***
  - Case C is

when 1 => X:=X+2;
A:=X+5;
when 2 => A:=X+3;
when others :=> A := X+N;
End Case;

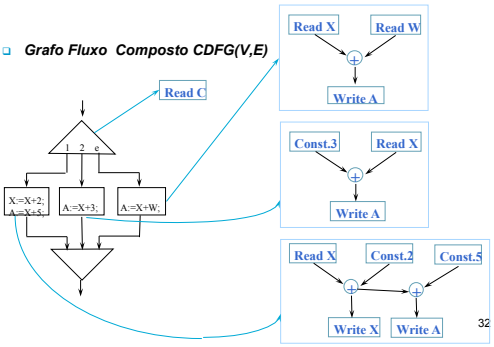


Representação Interna

- **Control-Data-Flow Graph (CDFG)**
  - Este tipo de representação é a mais comum para representar síntese comportamental, desde que ela estende às características do DFG nós e controle.
  - Representações dos CDFG
    - **Uma primeira representação é uma extensão dos grafos baseados em fluxo de dados (DFG) para suportar controle. Nesta representação são adicionados ao modelo, tipos de nós de controle como desvios, e merges.**
    - **A segunda representação trata com blocos básicos que representam seqüências de operações no qual o fluxo de controle está presente. Para cada bloco básico, uma DFG é feito. Neste esquema os nós de fluxo de controle são colocados a parte dos nós de fluxo de dados.**

Representação Interna

- ***Grafo Fluxo Composto CDFG(V,E)***

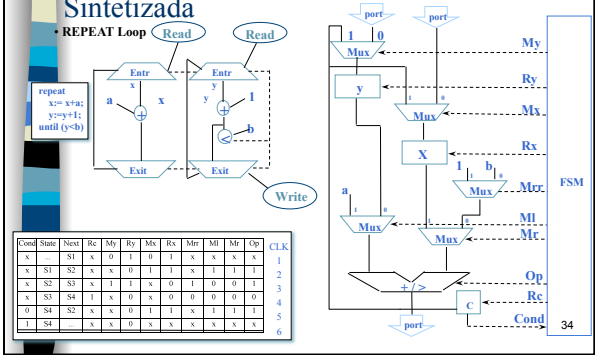


Representação Interna da Estrutura Sintetizada

- Máquina de estados
  - FSM é função de S, x, y, f, g
    - **S = s_1, ..., s_n - Conjunto de Estados**
    - **Y = y_1, ..., y_n - Saídas de Controle**
    - **X = x_1, ..., x_n - Entradas de Condição**
    - **f: $s(t_{n+1}) = f(s(t_n), x(t_n))$ - Função de Transição**
    - **g: - Função de Saída**
      - **$y(t_n) = g(s(t_n))$ - Moore**
      - **$y(t_n) = g(s(t_n), X(t_n))$ - Mealy**

Representação Interna da Estrutura Sintetizada

- **REPEAT Loop**

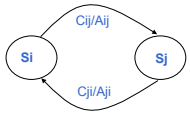


Representação Interna

- **Forma Intermediária Orientada a Arquitetura**
  - Este tipo de representação é a mais próxima da arquitetura produzida pela síntese comportamental que para descrição comportamental.
  - Nesta forma o caminho de dados e controle podem ser representados explicitamente.
  - O caminho de dados é modelado como um conjunto de atribuições e expressões sobre dados.
  - O controle é geralmente atribuído a uma FSM.
  - A unidade de controle é em geral expressa em duas formas:
    - **FSMD - Introduzida por Gajski como uma forma universal para representação de projetos de hardware. FSMD é uma FSM estendida para dados.**
    - **FSMC - É um FSMD com operadores executados por co-processadores.**

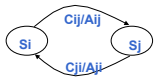
Representação Interna

- FSMD
  - FSMD é definida como:
    - um conjunto de variáveis V
    - um conjunto de expressões E={f(x,y,z,...))|x,y,z ε V}
    - Um conjunto de armazenamento de dados A={x<= e|x ε V, e ε E}
    - Um Status também é definido para armazenar a condição de execução das expressões

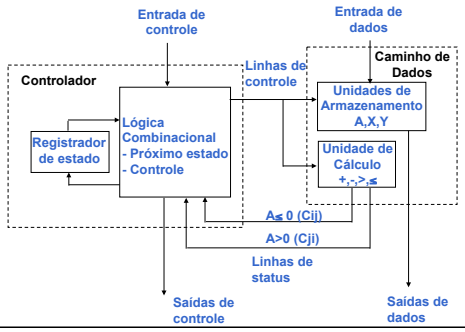


Cij: A ≤ 0;
Cji: A > 0;
Aij: X:= A+Y; Output <='1';
Aji: X:= A-Y; Ouput <='0';

FSMD



Cij: A ≤ 0;
Cji: A > 0;
Aij: X:= A+Y; Output <='1';
Aji: X:= A-Y; Ouput <='0';

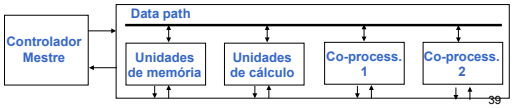


FSMD

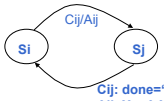
- Definição
  - FSMD é formulado por uma 5-tupla:
    - **<S,I x SS,O x A,f,h >, onde:**
    - **S: os conjunto de estados da FSMD**
    - **I x SS: O conjunto de entradas da FSMD. Entradas estendidas com expressões de status.**
    - **OxA: O conjunto de saídas da FSMD. Saídas estendidas com atribuições de variáveis.**
    - **f: função de próximo estado, mapeando S x (I x SS) -> S**
    - **h: Função de saída, mapeando S x (I x SS) -> (O x A)**
  - A FSMD computa novos valores para variáveis armazenadas no caminho de dados e produz saídas.

Representação Interna

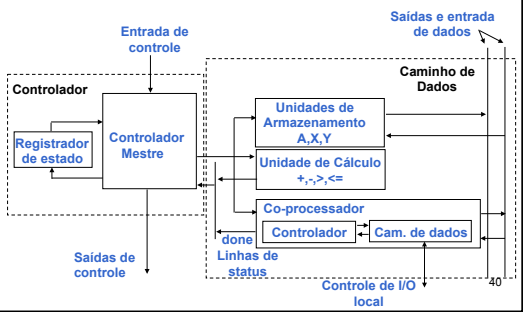
- FSMC (FSM com modelo Co-Processador)
  - FSMC é uma FSMD com operações executadas em co-processadores.
  - As expressões podem incluir operações complexas executadas sobre unidades específicas chamadas co-processadores.
  - Uma FSMC é definida como uma FSMD mais um conjunto de N co-processadores.
  - Cada processador é definida também como uma FSMC.



FSMC



Cij: done='0' and A ≤'0';
Aji: X:= A-Y; call(co-processor,operation);



FSMC

- Definição
  - FSMC é formulado por uma 5-tupla:
    - **<S x $\Pi^N_{i=1}$ S_{ci}, I x SS x $\Pi^N_{i=1}$ I_{ci}, O x A x $\Pi^N_{i=1}$ O_{ci},f,h), onde:**
    - **S_{ci} : O conjunto de estados do co-processador i.**
    - **I_{ci} : O conjunto de entradas do co-processador i.**
    - **O_{ci}: O conjunto de saídas do co-processador i.**
    - **S x $\Pi^N_{i=1}$ S_{ci} : O conjunto de estados da FSMC. O conjunto da FSMC é definido como o produto do estado da FSM e o número de estados locais dos co-processadores.**
    - **I x SS x $\Pi^N_{i=1}$ I_{ci} : As entradas da FSMC. Entradas são estendidas com status (SS) e entradas locais de co-processadores.**
    - **O x A x $\Pi^N_{i=1}$ O_{ci} : As saídas da FSMC. Saídas são estendidas com atribuições e saídas locais de co-processadores.**