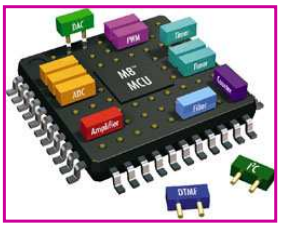
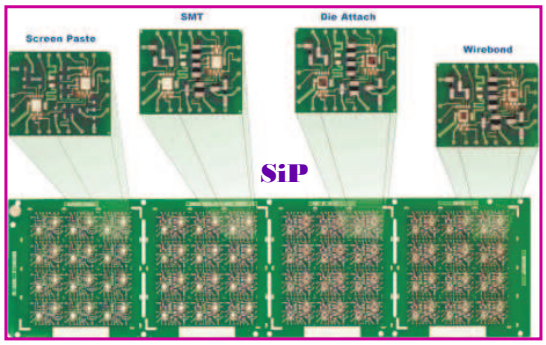
PARTICIONAMENTO

Alternativas de implementação

- SoC ou System-on-Chip: *é a idea de integrar todos os componentes de um sistema embutido num único chip;*
- SiP ou System-in-Package: *quando os recursos disponíveis num único chip não são suficiente para a implementação do sistema embutido, usa-se então vários chips interconectados.*



SoC



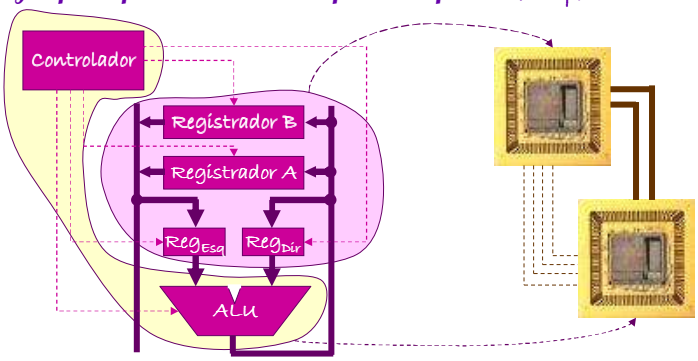
PARTICIONAMENTO

- *A funcionalidade de um sistema embutido é implementada utilizando um conjunto de componentes físicos interconectados;*
- *A implementação de um sistema embutido consiste em:*
  - *Selecionar o conjunto de componentes ou obter uma alocação;*
  - *Distribuir a funcionalidade do sistema entre os componentes alocados ou obter uma partição;*
  - *A alocação e a partição devem ser escolhidas de tal maneira que a implementação satisfaça todas a restrições de projeto como custo, desempenho, área e consumo de energia.*
- *Existem duas abordagem para particionamento do sistema:*
  - *Partitionamento estrutural*
  - *Particionamento funcional*

PARTICIONAMENTO

Particionamento Estrutural

- *O sistema é projetado utilizando o modelo estrutural que é uma interconexão de objetos de hardware como portas lógicas, flip-flops, registradores, somadores, contadores, processadores, etc.*
- *Os objetos de hardware usados são particionados em grupos*
- *Cada grupo representa um componente físico (chip) do sistema.*



PARTICIONAMENTO

- *O particionamento estrutural se tornou popular por várias razões:*
  - *Aproximação simples da área necessária para implementar os componentes físicos que formam o sistema:*

$$T_C = \Sigma_{c \in C} \; T_c$$

  - *Aproximação simples da pinagem dos componentes que formam o sistema:*

$$P_C = \Sigma_{\forall c_1 \in C, \forall c_2 \notin C} \; W_{c_1, c_2}$$

  - *Uso dos métodos formais da teoria de grafos para transformar o problema de particionamento de hardware em um outro de particionamento de grafo.*

PARTICIONAMENTO

- Um grafo é definido por um conjunto de nós e um conjunto de arcos aonde cada arco pode conectar um subconjunto dos nós. Nós e arcos são associados com valores, chamados pesos;
- O grafo representa a estrutura do sistema, os nós os componentes de hardware, e os arcos as interconexões; O peso associado à um nó representa a sua área e aquele associado à um arco é a largura da interconexão.
- Um cluster consiste num subconjunto de nós e tem como peso a soma dos pesos dos nós nele contidos e como cutsize a soma dos pesos dos arcos que conectam um nó do cluster com pelo menos um nó fora dele.
- Um cluster representa um componente físico ou chip.
- O tamanho do chip deve ser igual ou maior do que o peso do cluster associado;
- A pinagem disponível num chip deve ser igual ou maior do que o cutsize do cluster associado.

PARTICIONAMENTO

- O paticionamento estrutural tem várias limitações:
  - Compromissos de tamanho/desempenho são difíceis porque decisões feitas durante a fase de projeto podem ser anuladas pelo particionamento;
  - Projetos com um número grande de objetos de hardware tornam o processo de particionamento muito complexo e portanto levam a soluções inferiores
  - Projetos de somente hardware.

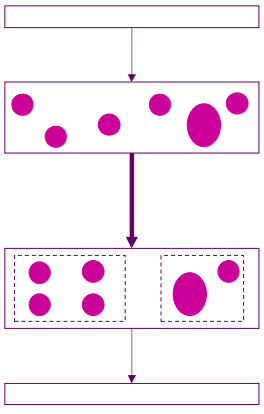
PARTICIONAMENTO

Particionamento Funcional

- Primeiro, o sistema é decomposto em várias funcionalidades primárias, chamadas objetos funcionais;
- Segundo, os objetos funcionais são particionados entre os componentes do sistema;
- Terceiro, as funcionalidades dos componentes são implementadas em hardware ou em software;
- O particionamento funcional tem várias vantagens:
  - Balanços tamanho/desempenho mais fáceis;
  - Número de objetos é geralmente pequeno;
  - Pode ser usado para particionamento software/hardware.

PARTICIONAMENTO

- Existem vários aspectos que são importantes para a realização de particionamento funcional:



PARTICIONAMENTO

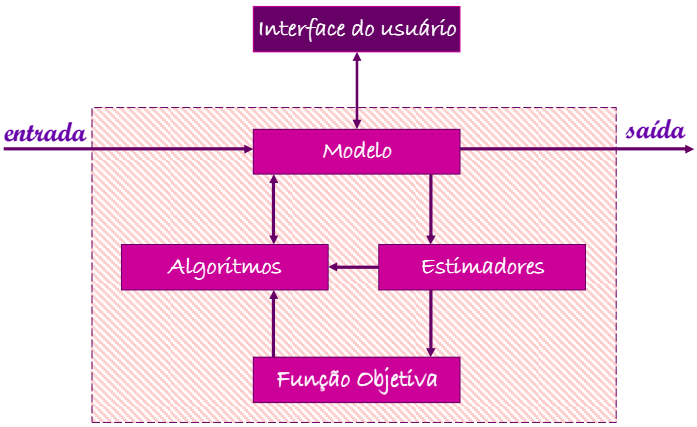
- *Nível de abstração da especificação*
  - *É informalmente mensurado pelo número de objetos primários permitidos;*
  - *Um nível de abstração baixo indica um número alto de objetos primários;*
  - *Esquemáticos tem um nível de abstração muito baixo;*
  - *Especificações em VHDL tem um nível de abstração muito alto;*
- *O nível de abstração define a granularidade do particionamento;*
- *Para níveis de abstração baixos corresponde uma granularidade fina dos objetos (fine-granularity) e para níveis de abstração altos corresponde uma granularidade grossa (coarse-granularity).*

PARTICIONAMENTO

- *A granularidade dos objetos tem impacto direto na computação das estimativas de tamanho de partições.*
- *Granularidade fina impõem processos de estimativas complexos e granularidade grossa impõem processos de estimativas razoáveis.*
- *Níveis adequados, permitindo um particionamento funcional eficiente podem ser baseadas em:*
  - *Grafos de fluxo de dados entre funções;*
  - *Grafos de fluxo de dados entre operações aritméticas e lógicas;*
  - *Máquinas de estados do sistema;*
  - *Grafos de transferência entres registradores;*
- *Um particionamento funcional baseado no modelo estrutural não é nada mais do que o particionamento estrutural.*

PARTICIONAMENTO

Configuração de um sistema de particionamento



PARTICIONAMENTO

Definição formal do problema de particionamento

Dado um conjunto de objetos: $\mathcal{O} = \{o_1, o_2, ..., o_n\}$, *determine a partição* $\mathcal{P} = \{p_1, p_2, ..., p_m\}$ *tal que:*

a) $m << n$

b) $p_i \subset \mathcal{O}, 1 \leq i \leq m$

c) $p_1 \cup p_2 \cup ... \cup p_m = \mathcal{O}$

d) $p_i \cap p_j = \emptyset, \forall i, j, i \neq j$

e) $\mathcal{F}_{obj}(\mathcal{P})$ *é mínima*

O problema de particionamento de Hardware/Software é um caso particular do problema de particionamento com $m = 2$.

PARTICIONAMENTO

- *Método de Agrupamento Hierárquico*

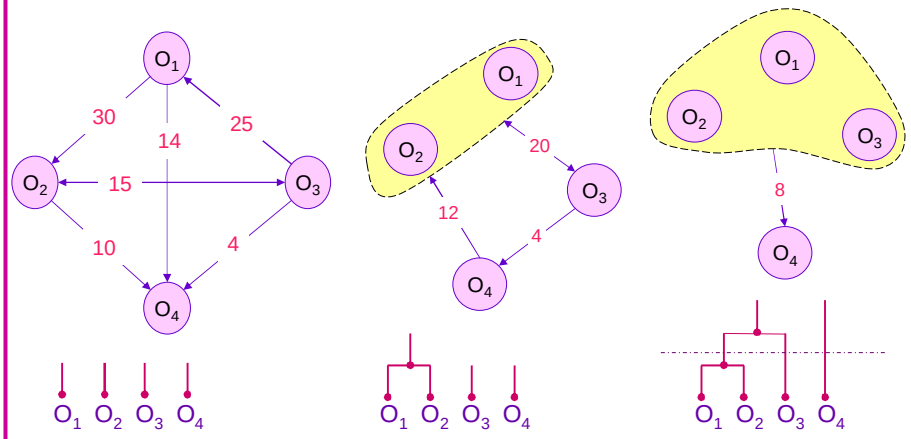
```
Algoritmo Hierarchical_Clustring(𝒪);
   P := ∅; C := ∅;
   Para cada objeto o_i ∈ 𝒪 Faça
      p_i := o_i;
      P := P ∪ {p_i};
   Para cada partição p_i ∈ P Faça
      Para cada partição p_j ∈ P\{p_i} Faça
         c_i,j := Proximidade(p_i, p_j);
         C := C ∪ {c_i,j};
   Enquanto não CondiçãoTérmino(P) Faça
      (p_i, p_j) := ObjetosMaisPróximo(P,C);
      P := P\{p_i, p_j} ∪ {p_ij};
      C := C \{c_xy | (x=i) ou (x=j) ou (y=i) ou (y=j);
      Para cada objeto p_k ∈ P Faça
         c_ij,k := Proximidade(p_ij, p_k);  {soma/média/max/min}
   Retorna P;
```

PARTICIONAMENTO

- *Função* `Proximidade(p,q)` *retorna uma medida de proximidade dos objetos* `p` *e* `q` *no sistema. A medida deve ser proporcional ao número de conexões entre* `p` *e* `p` *assim como a largura destas;*
- *Função* `ObjetoMaisPróximo(P,C)` *retorna os dois objetos mais próximos em* `P`*, considerando a suas respectivas proximidades em* `C`*. Observe que colocar objetos que tem grande interação numa mesma partição aumenta o desempenho do sistema.*
- *Função* `CondiçãoTérmino(P)` *deve determinar quando o processo de particionamento deve parar. As condições de parada mais comuns são:*
  - *O número de clusters formados é maior de um certo limite estabelecido;*
  - *A proximidade dos clusters formados é menor de um certo limite, geralmente chamado de threshold de proximidade.*

PARTICIONAMENTO

- *Exemplo de particionamento com* `Hierarchical_Clustring(0)`



Threshold de proximidade = 10
P = {[O_1, O_2, O_3], [O_4]}

PARTICIONAMENTO

- *Método de agrupamento hierárquico com multi-estágios*
  - *Etapa 0: Construir o grafo do objeto usando a função de proximidade $\mathcal{F}_0$;*
  - *Etapa 1: Para todas as funções de proximidades $\mathcal{F}_i$, i = 0, ...*
    - *Etapa 1.1: Aplicar o agrupamento hierárquico com a função de proximidade $\mathcal{F}_i$;*
    - *Etapa 1.2: Obter a árvore completa de clusters*
    - *Etapa 1.3: Se possível, dividir a árvore de clusters utilizando uma cutline senão parar o processo*
    - *Etapa 1.4: Construir o novo grafo de objetos considerando cada sub-árvore abaixo da cutline como cluster e usando uma nova função de proximidade $\mathcal{F}_i$;*
    - *Etapa 1.5: Repetir o processo.*

PARTICIONAMENTO

- *Método de migração de grupos ou min-cut*

```
Algoritmo Min-Cut(𝒪);
  P := (𝒪,∅);
  Repita
    P_antes := P; Custo_antes:= ℱobj(P); Custo_mpart := ∞;
    Para cada Oi Faça Oi.mudou := 0;
    Para i de 1 a n Faça
      Custo_mmud := ∞;
      Para cada Oj | Oj.mudou = 0 Faça
        Custo := ℱobj(Muda(P, Oj));
        Se custo < custo_mmud Então
          custo_mmud := custo; objeto_mmud := Oj;
      P := Muda(P, objeto_mmud); objeto_mmud.mudou := 1;
      Se custo_mmud < custo_mpart Então
         melhor_part := P; custo_mpart := custo_mmud;
    Se custo_mpart < Custo_antes Então
      P := melhor_part;
    Senão Retorna P_antes;
```

PARTICIONAMENTO

- *O algoritmo de Min-Cut tenta minimizar o número de conexões entre as partições (corte ou cutsize) mas pode ser utilizado com outras métricas de qualidades também;*
- *O algoritmo apresentado fornece o resultado como 2 partições somente mas pode ser estendido para M partições;*
- *A estratégia consiste em:*
  - *primeiro determinar para cada objeto O a redução de custo que a mudança de O para a partição opostas ocasionaria;*
  - *Depois, colocar o objeto que introduziu a maior diminuição ou o menor aumento de custo na partição oposta;*
  - *Assim, este algoritmo consegue evitar alguns mínimos locais.*
- *Exercício: Modifique o algoritmo de Min-Cut para permitir o particionamento dos objetos em m partições!*

PARTICIONAMENTO

- ***Min-Cut vs. Ratio-Cut***

Este método aplica o algoritmo do Min-Cut utilizando uma nova função objetiva, chamada Ratio-Cut que permite minimizar o cutsize e no mesmo tempo manter um número balanceado de grupos sem impor nem limite nem um threshold de proximidade.

Considere uma partição $P = \{p_1, p_2, \ldots, p_m\}$***:***

$$Ratio(P) = \frac{Cut(P)}{\prod_{i=1}^{m} Tamanho(p_i)}$$

Cut(P) ***retorna a soma dos pesos das conexões entre as partições;***
Tamanho(p) ***retorna a área requerida para implementação do grupo*** *p*