

# ECE382M.20: System-on-Chip (SoC) Design

---

## Lecture 4 – HW/SW Partitioning

*Sources:*

*Prof. Margarida Jacome, UT Austin*

*Prof. Lothar Thiele, ETH Zürich*

Andreas Gerstlauer

Electrical and Computer Engineering

The University of Texas at Austin

gerstl@ece.utexas.edu



The University of Texas at Austin

Chandra Department of Electrical  
and Computer Engineering

Cockrell School of Engineering

---

1

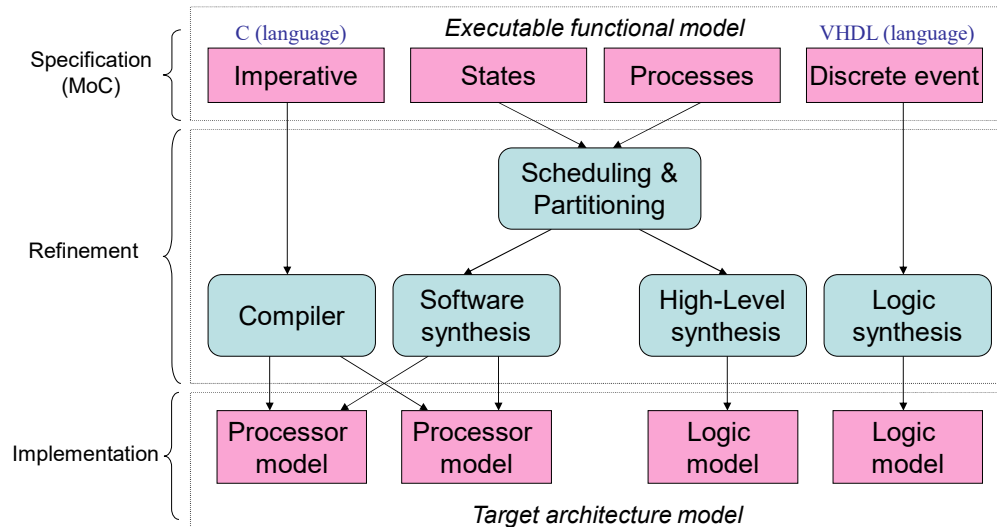
## Lecture 4: Outline

---

- **HW/SW co-design tasks**
  - Design synthesis
  - System mapping
- **HW/SW partitioning**
  - Decomposition
  - Constructive heuristics
  - Iterative heuristics

2

## Design Synthesis



ECE382M.20: SoC Design, Lecture 4

© Margarida Jacome, UT Austin

3

3

## Synthesis Tasks

- **Mapping**
  - Allocate resources (hardware/software processors)
  - Bind computations to resources
  - Schedule operations in time
  - Partitioning = (allocation +) binding
  - Mapping = binding + scheduling
- **Allocation, scheduling and binding interact, but separating them helps**
  - Alternatively allocate, bind, then schedule

ECE382M.20: SoC Design, Lecture 4

© Margarida Jacome, UT Austin

4

4

### Mapping Example

Task graph

Hardware platform

ECE382M.20: SoC Design, Lecture 4

© Margarida Jacome, UT Austin

5

5

### Example Cost Model

- Process execution times

|    | M1 | M2 |
|----|----|----|
| P1 | 5  | 5  |
| P2 | 5  | 6  |
| P3 | -- | 5  |

Task graph

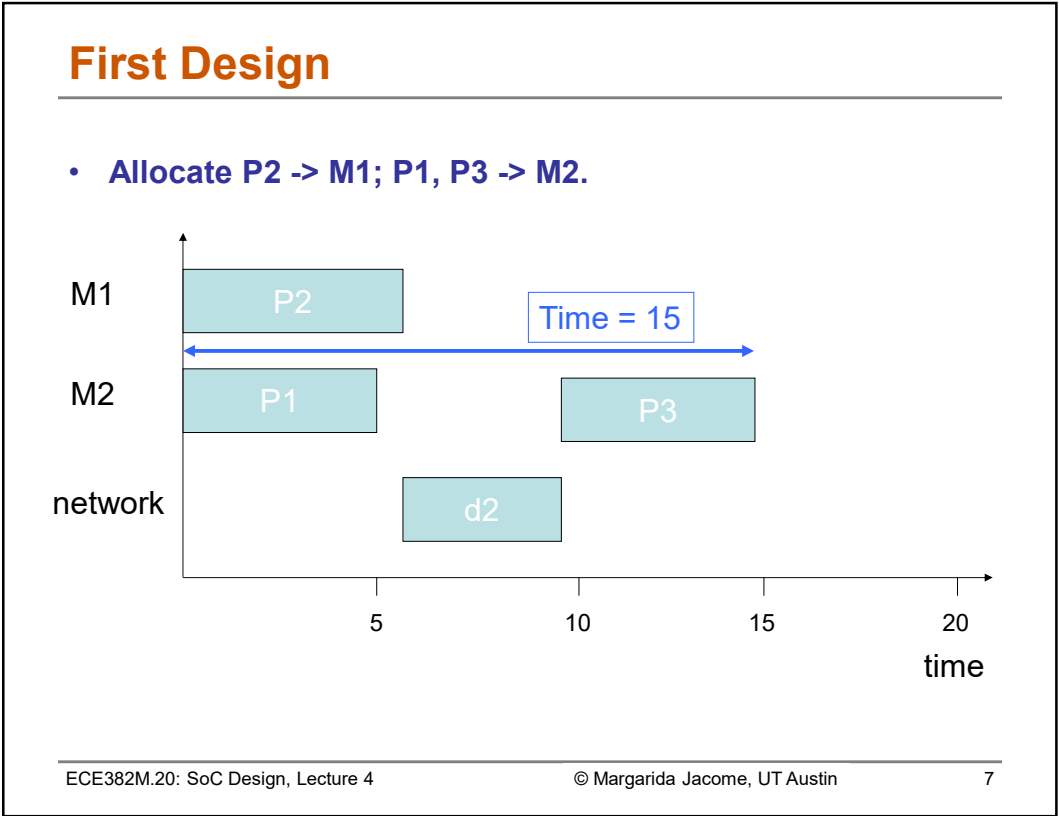
- Communication cost
  - Assume communication within PE is free
  - Cost of communication from P1 to P3 is  $d1 = 2$
  - Cost of P2 to P3 communication is  $d2 = 4$

ECE382M.20: SoC Design, Lecture 4

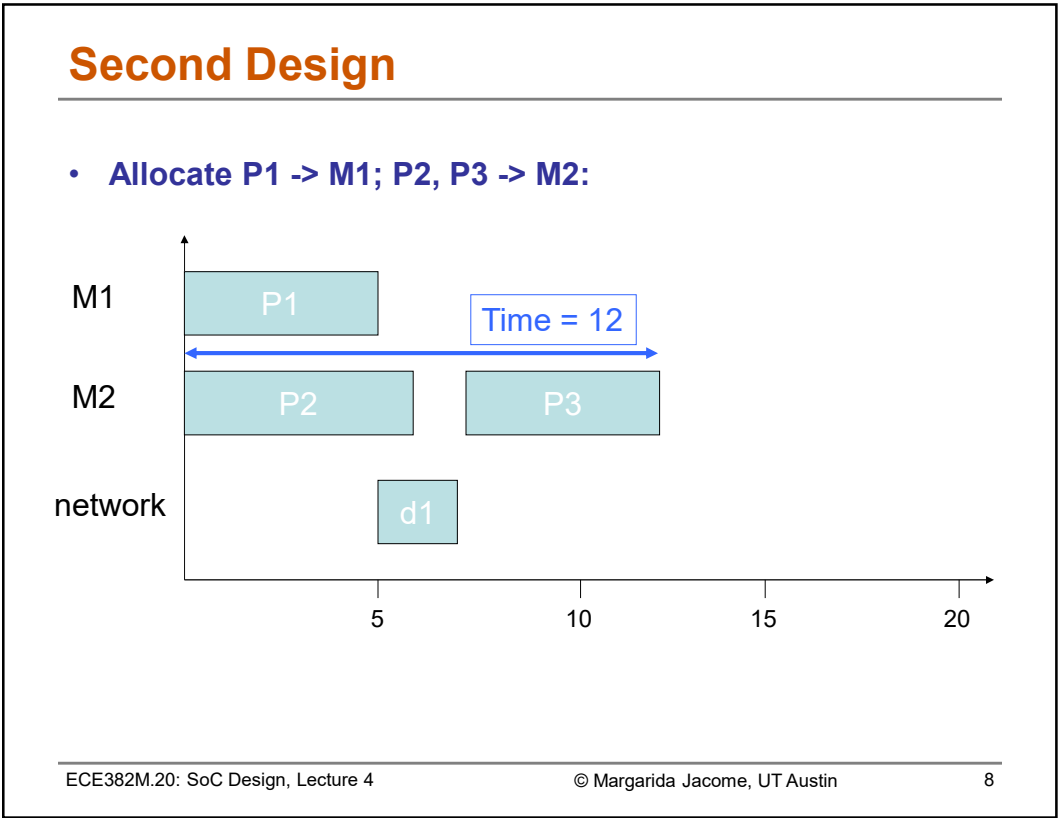
© Margarida Jacome, UT Austin

6

6



7



8

## Lecture 4: Outline

- ✓ **HW/SW co-design tasks**

- ✓ Design synthesis
- ✓ System mapping

- **HW/SW partitioning**

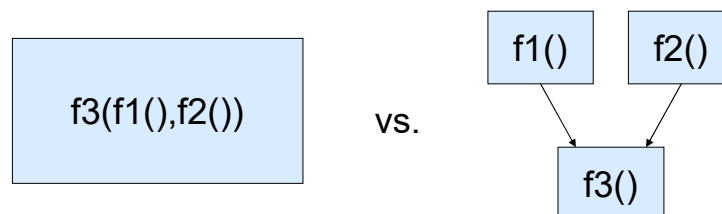
- Decomposition
- Constructive heuristics
- Iterative heuristics

## Decomposition

- **Divide functional specification into modules**

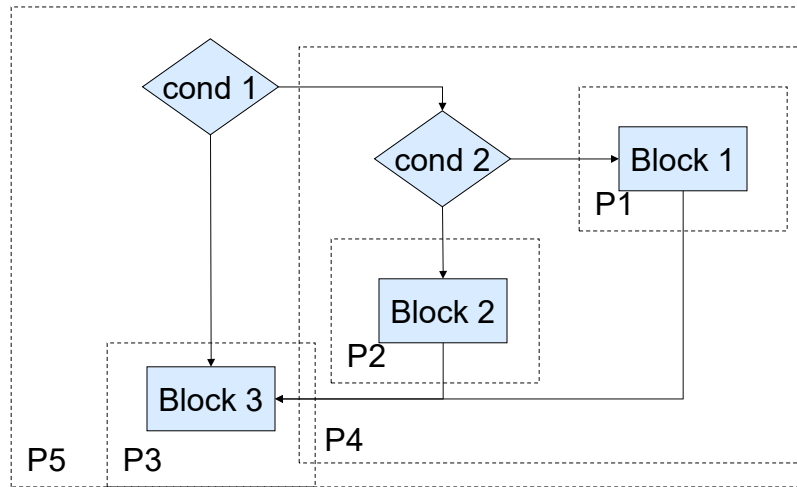
- Map units onto PEs
- Units may become processes

- **Determine proper level of parallelism**



## Decomposition Example

- Divide program into Control-Data Flow Graph (CDFG)
- Hierarchically decompose CDFG to identify partitions



ECE382M.20: SoC Design, Lecture 4

© Margarida Jacome, UT Austin

11

11

## Partitioning

- **Assign tasks (objects) to processors (partitions) such that all objects are assigned to unique partitions**
  - Minimize communication cost (graph partitioning)
  - Minimize partition count (bin packing)
  - Partition size, partition count, etc. constraints
- **Exact methods**
  - Exhaustive enumeration, integer linear programming (ILP)
- **Constructive heuristics**
  - Random mapping, hierarchical clustering
- **Iterative heuristics**
  - Hill climbing, Kernighan-Lin, simulated annealing

ECE382M.20: SoC Design, Lecture 4

© Lothar Thiele, ETH Zürich

12

12

## Constructive Methods

- **Construct solution one by one**
  - Visit every object once
  - Can generate a starting partition for iterative methods
  - Shows the difficulty of finding proper closeness functions
- **Random mapping**
  - Each object is assigned to a block randomly
- **Hierarchical clustering**
  - Stepwise grouping of objects
  - Closeness function determines how desirable it is to group two objects

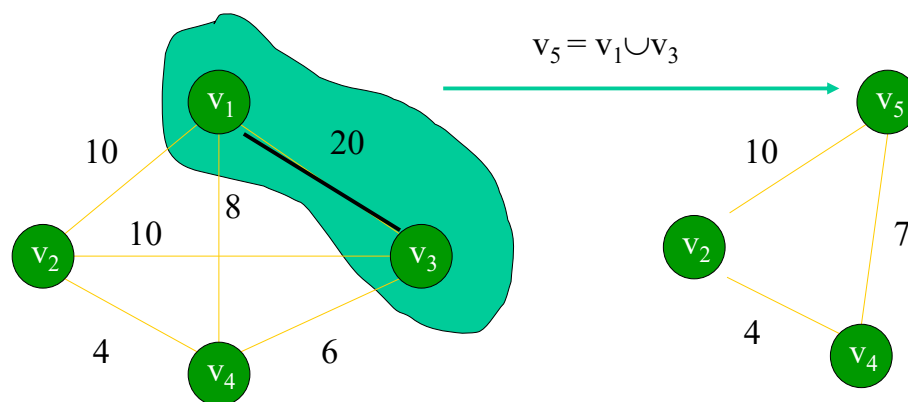
ECE382M.20: SoC Design, Lecture 4

© Lothar Thiele, ETH Zürich

13

13

## Hierarchical Clustering - Example (1)



Closeness function: arithmetic mean of weights

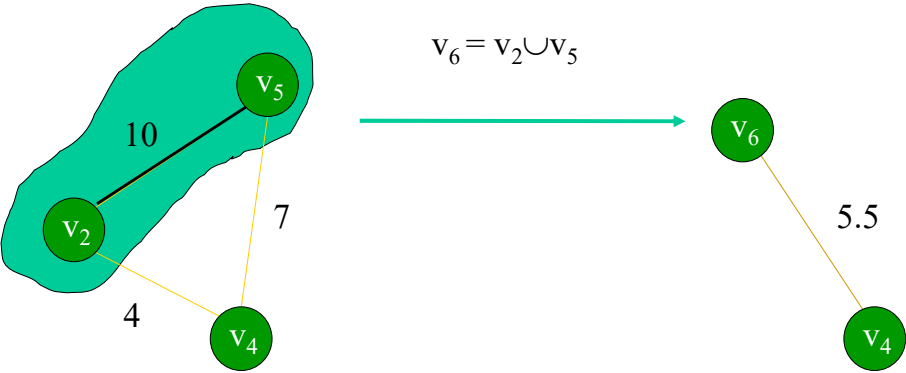
ECE382M.20: SoC Design, Lecture 4

© Lothar Thiele, ETH Zürich

14

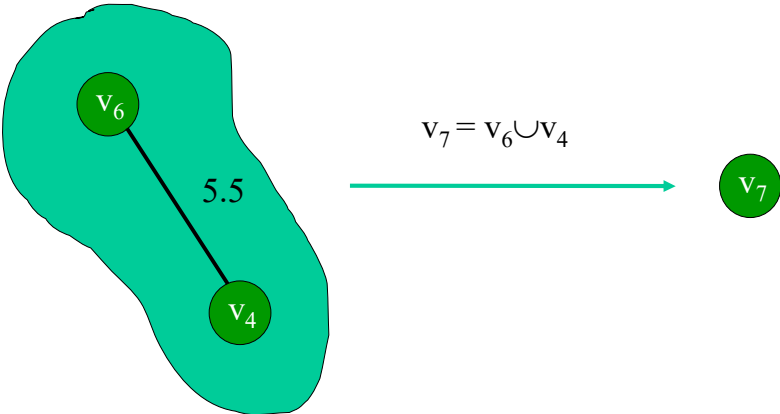
14

Hierarchical Clustering - Example (2)



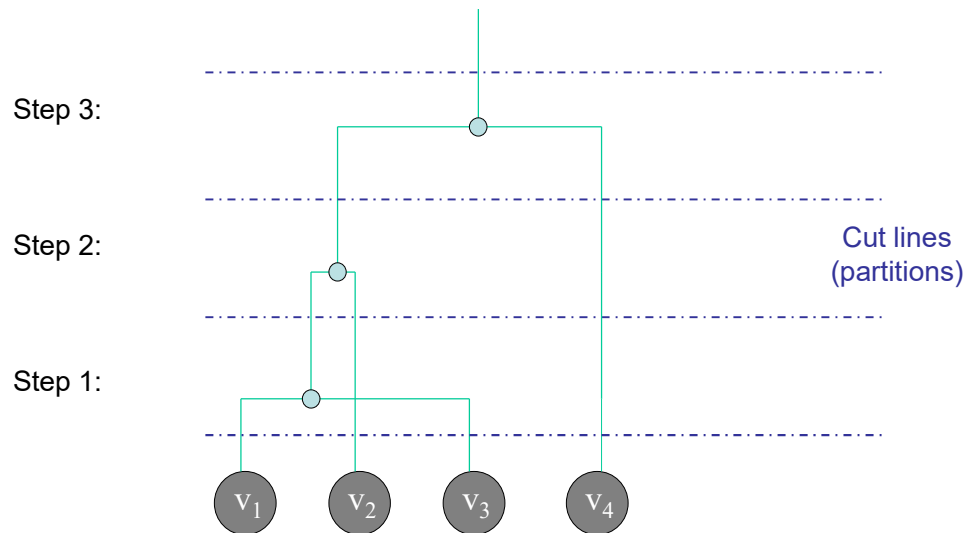
15

Hierarchical Clustering - Example (3)



16

## Hierarchical Clustering - Example (4)



ECE382M.20: SoC Design, Lecture 4

© Lothar Thiele, ETH Zürich

17

17

## Iterative Methods

- **Principles**

- Start with some initial solution
- Search neighborhood (similar solutions), select candidate and make local change based on fitness/cost function
- End on stopping criterion

- **Simple iterative improvement or “hill climbing”**

- Select candidate with best improvement in cost
- Stop when no candidate with lower cost is found

- **Kernighan-Lin**

- More exhaustive search to escape local optima

ECE382M.20: SoC Design, Lecture 4

© Lothar Thiele, ETH Zürich

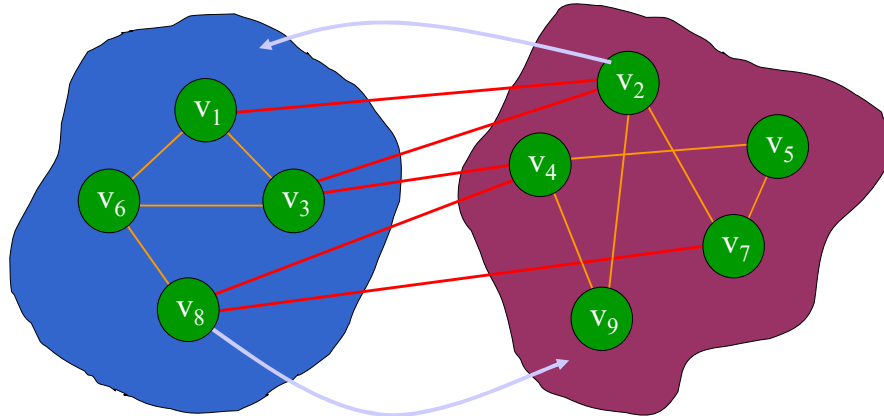
18

18

## Iterative Improvement (Hill Climbing)

- **Simple greedy heuristic**

- Until there is no improvement in cost: re-group a pair of objects which leads to the largest gain in cost



Example: Cost = number of edges crossing the partitions  
Before re-group: 5 ; after re-group: 4 ; gain = 1

ECE382M.20: SoC Design, Lecture 4

© Lothar Thiele, ETH Zürich

19

19

## Kernighan-Lin

- **Problem**

- Simple greedy heuristic can get stuck in a local minimum

- **Kernighan-Lin algorithm**

- As long as a better partition is found
  - From all possible pairs of objects, virtually re-group the “best” (lowest cost of the resulting partition)
  - From the remaining not yet touched objects, virtually re-group the “best” pair, etc.,
  - Continue until all objects have been re-grouped
  - From these  $n/2$  partitions take the one with smallest cost and actually perform the corresponding re-group operations.  $O(n^2 \log n)$  complexity
- Still can get stuck in local minimum
  - Among sequences of moves

- **More complex strategies**

- Randomize search, e.g. simulated annealing

ECE382M.20: SoC Design, Lecture 4

© Lothar Thiele, ETH Zürich

20

20

## Lecture 4: Summary

---

- **HW/SW co-design tasks**
  - Design synthesis
  - System mapping
- **HW/SW partitioning**
  - Decomposition
  - Constructive heuristics
  - Iterative heuristics

### ➤ Scheduling