

A Computational Procedure

for

Multi-disciplinary Optimization

Peter Stehlin
SMR SA
P.O. Box 4014
CH-2500 Bienne

Design Manager's Assistant

Objective:

Find a design which satisfies some (possibly multi-valued) goal in an optimal manner

**Input: Design specifications
 Constraints**

Output: Finished blue prints

Example:

Optimize structure

taking into account

**Aerodynamics, aeroelasticity, structural design,
design life, safety, production costs, life cycle cost,
engines, maintenance, etc.,**

in short, everything you can think of.

in

Minimum execution time

Solution:

**Decide which computational modules contribute
to the job
(this step is knowledge based)**

**Execute them in such a sequence that the input
required by each module has been made available
by the modules executed previously
=> Job dependency graph**

**Map the graph onto parallel or distributed
hardware so that the execution time is minimal**

How it is done:

1. Make **list of all processes** involved P1, P2, P3, , Pm

2. Make **list of all data sets** required and produced. S1, S2, S3, , Sn

3. **Associate data sets with processes.** Example:

•
•
•

P8 input sets: 1 3 4 78 output sets: 2 5 79

P9 3 5 79 6 7 10

•
•
•

Input:

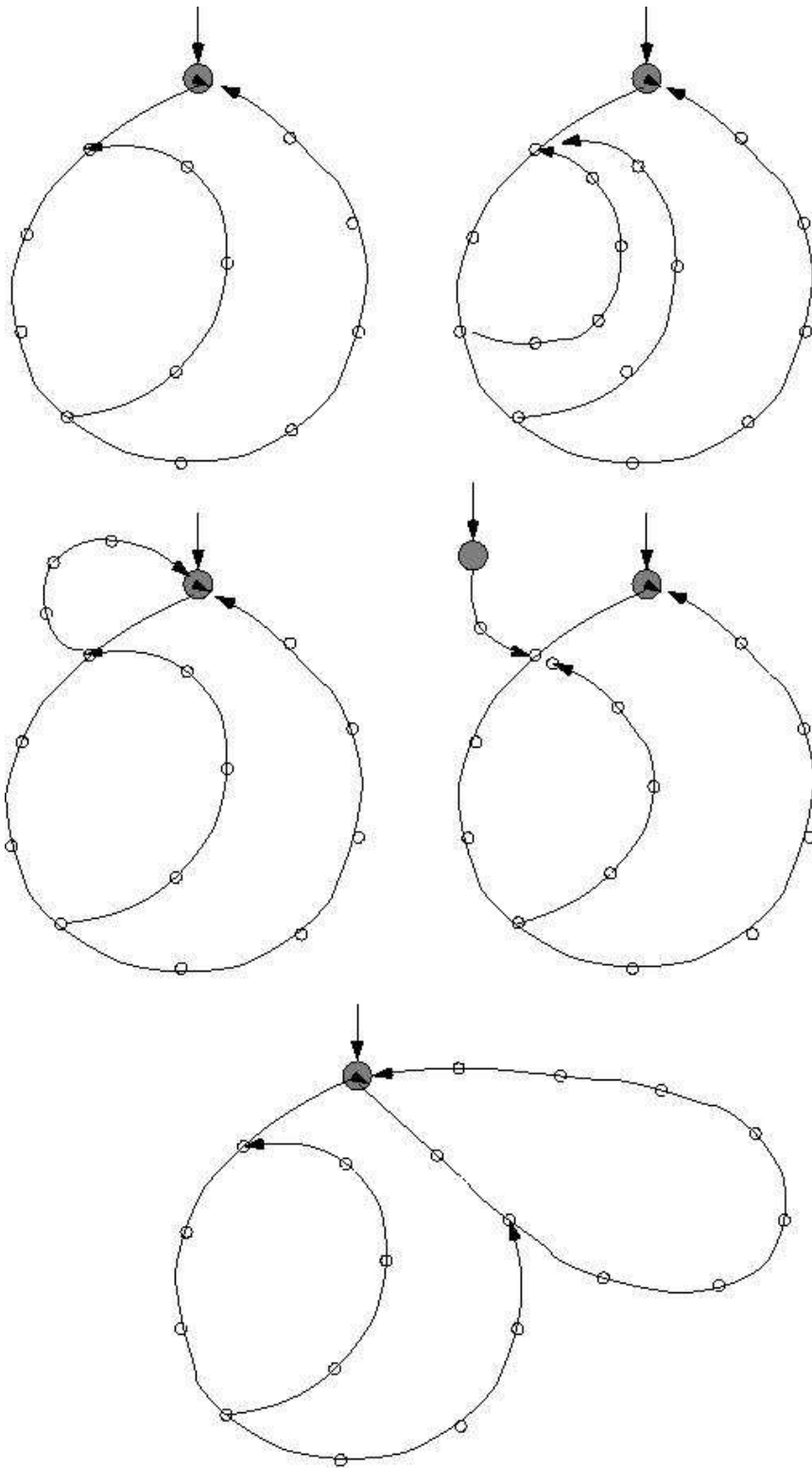
Output:

P8:	Geometry	1	Member sizes	2
	Loads	3	Displacements	5
	Material properties	4	Weight	79
	Constraints	78		

4. **I/O sequencing:** At each stage find processes whose input sets required for execution are available. Observe loops and dynamic iteration conditions (=> **ktool**)

5. Create **directed graph** showing job dependencies and execution sequences

6. **Map** processes onto available hardware (=> **ctmap**)



Ambiguous iteration loops

Simulation engine

(Self-writing program)

Create an **agenda**: List all instructions eligible for execution, beginning with the input processes. Then execute them, one by one, starting at the top:

(I1 arg11 arg12 ...)

(I2 arg21 arg22 ...)

As instructions are executed, they can set up new instructions on the list, re-schedule themselves, change the order of processes on the agenda, etc. When done they are deleted from the agenda. At any moment:

(fm argm1 argm2 argm3 ...)

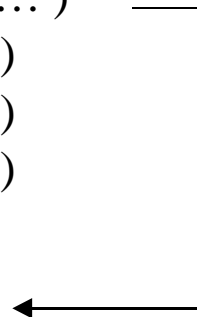
(fn argn1 argn2 argn3 ...)

(fo argo1 argo2 argo3 ...)

(fp argp1 argp2 argp3 ...)

.

.



Job ends when there are no instructions left on the agenda

Legend for the viewfoils showing the demonstration cases:

Horizontal axes: Time in seconds
Vertical axes: In job sequence diagrams: Process number
 In process maps: Processor number

In all demonstration cases, processes were mapped onto five processors. Horizontal line segments (mostly in red) indicate which processes are executing and for how long. Transverse lines (in green) between the horizontal segments show data sets being moved between processes. For clarity, the time required for data transmission has been greatly exaggerated. All processes execute for four seconds at a time, data transmission takes one second per set.

In all real applications, both the execution and transmission times are, of course, quantities which vary dynamically and so is the number of iterations required to achieve convergence in the design loops.

Process to processor mapping:

Of all the eligible processes on the agenda always execute those processes first which have the largest number of child processes

Test case 1: Description of processes and data sets

5

p1

1 6

1

2

p2

2

2

3

p3

2

2

4

p4

2

2

5

p5

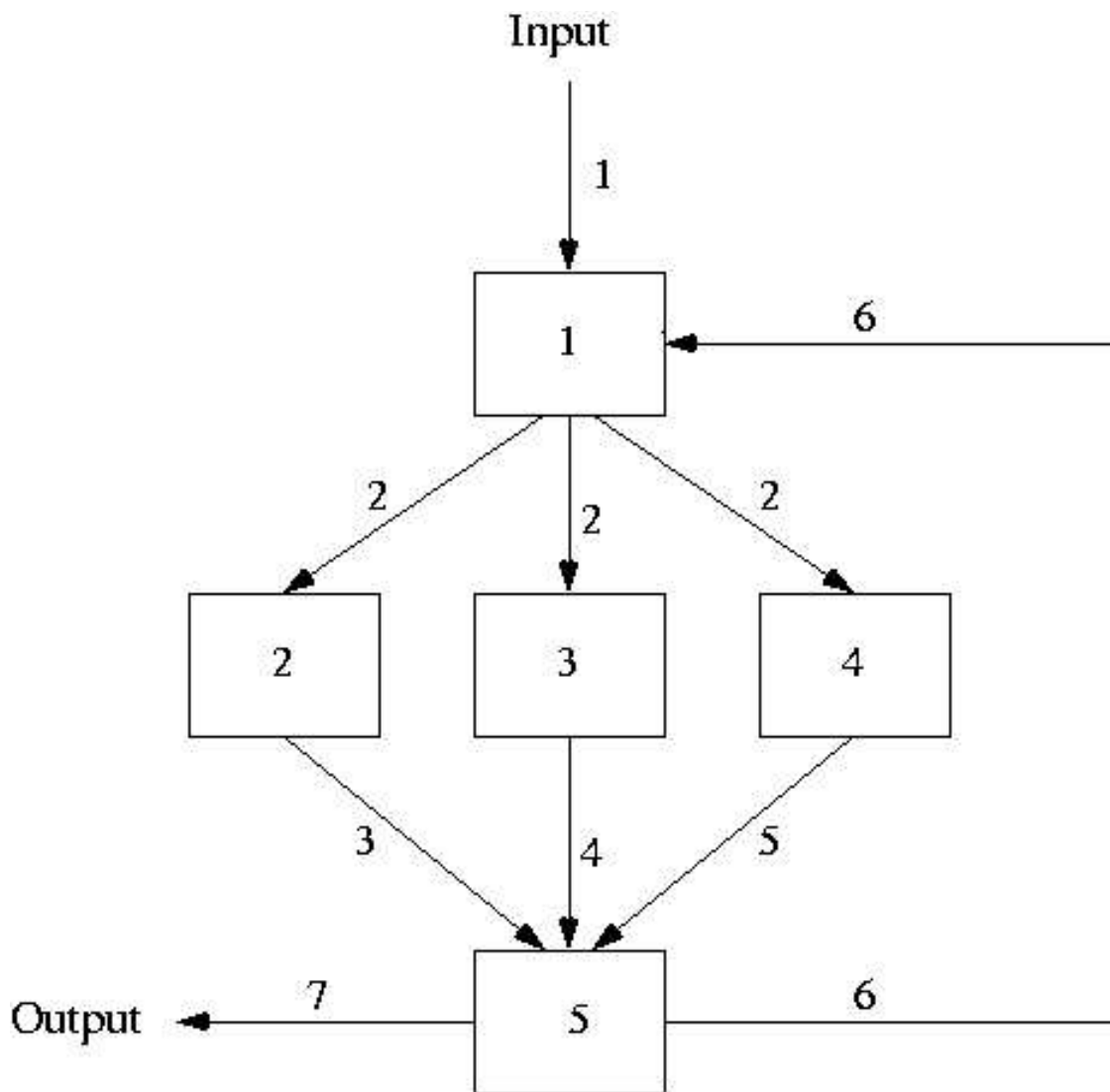
3 4 5

3 4 5

6 7

1 10

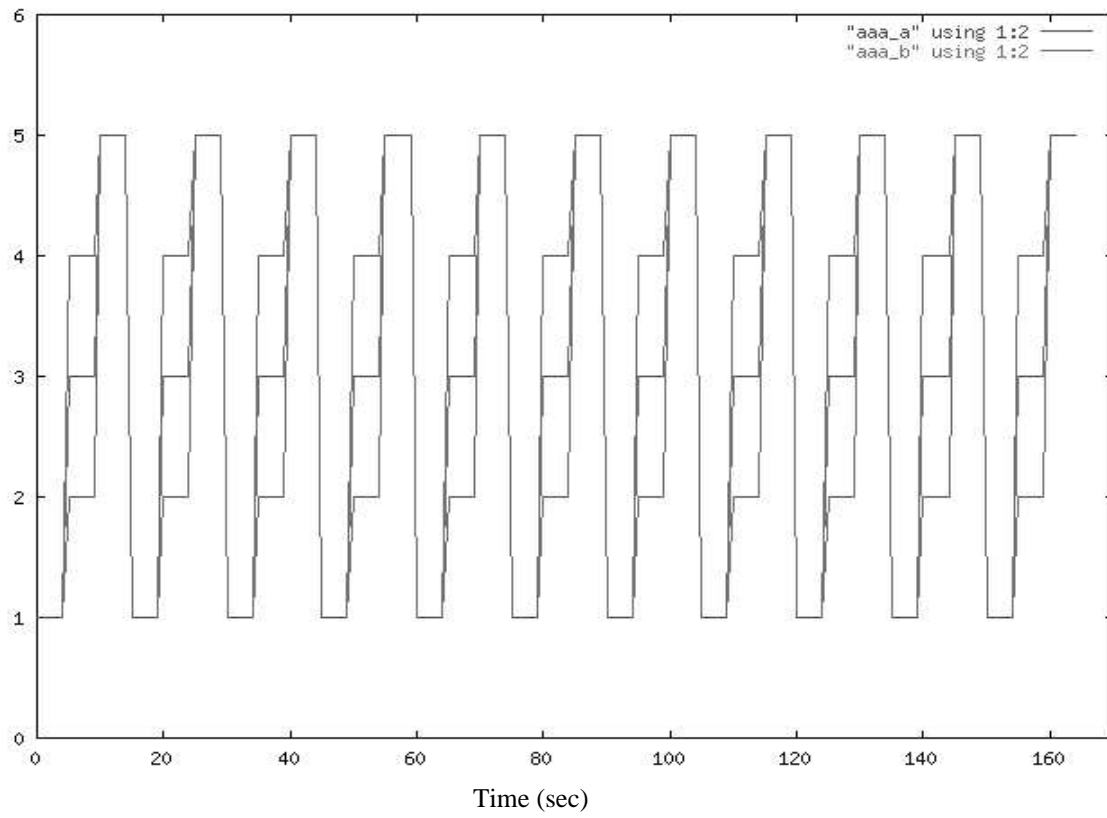
It is usually with such a list that the description of the design process begins.



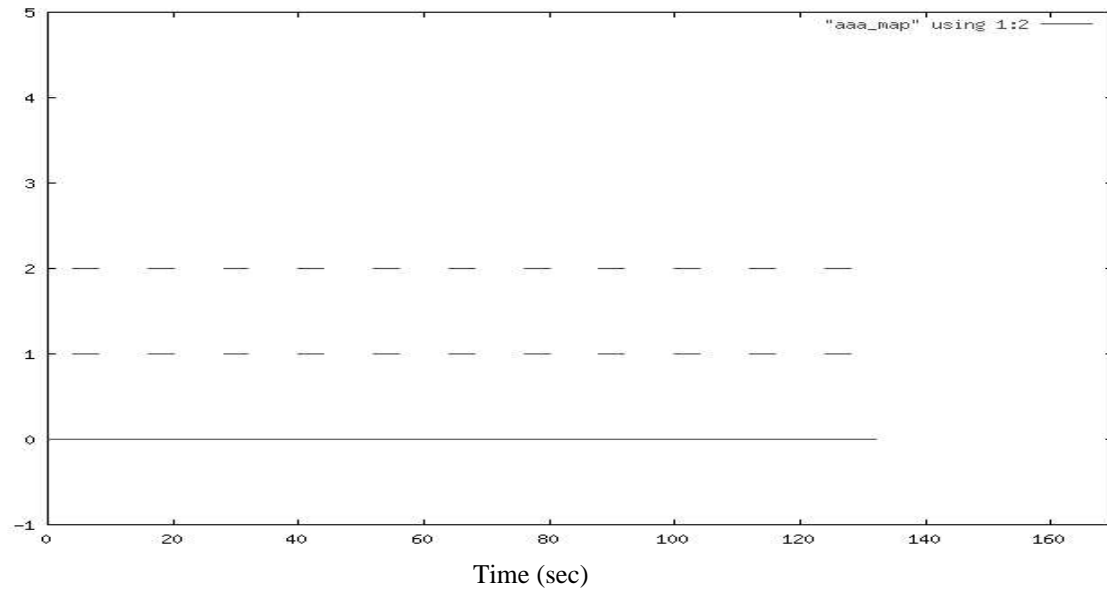
Test case 1: Topology

This diagram is the graphical equivalent of the list in the previous viewfoil

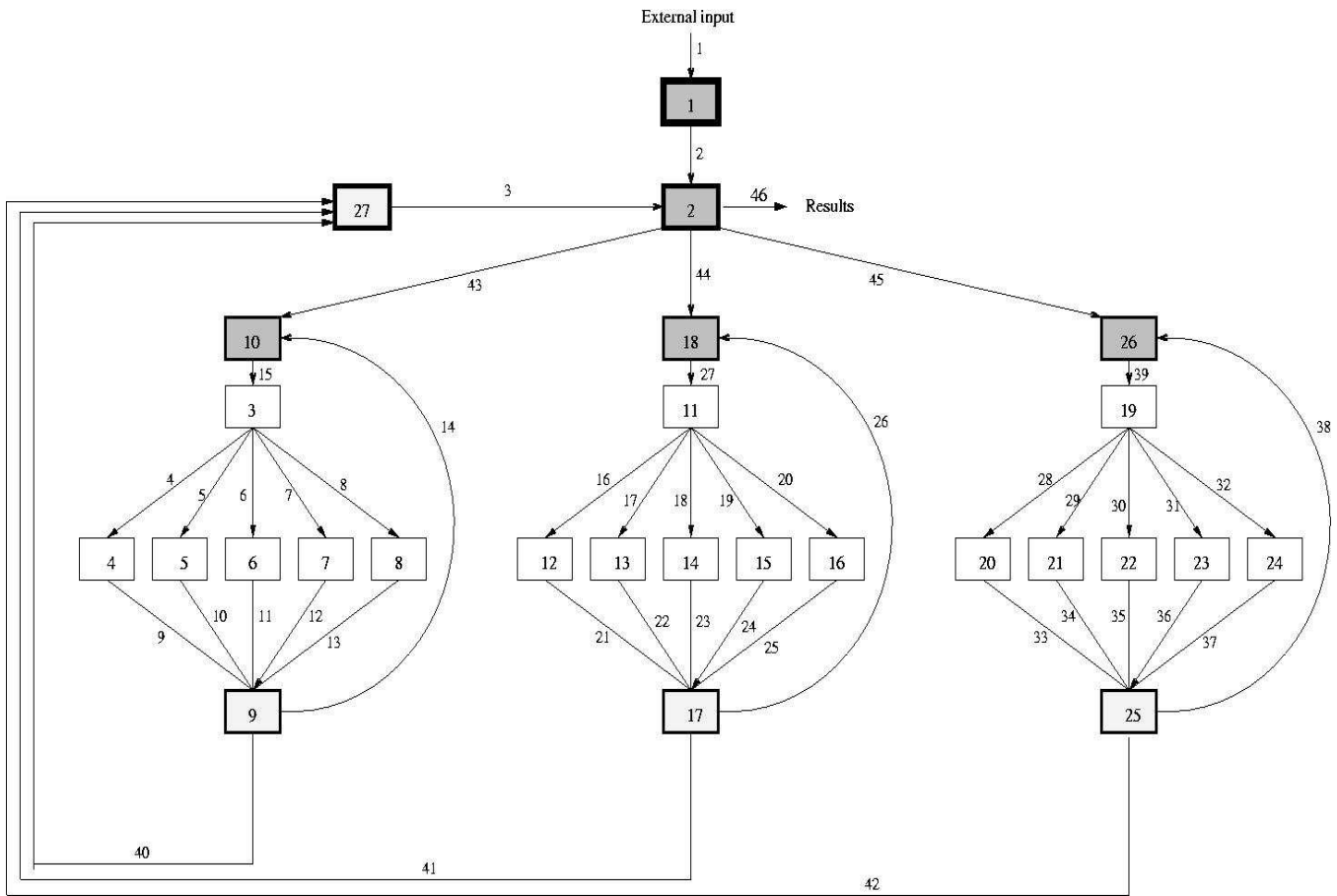
Process



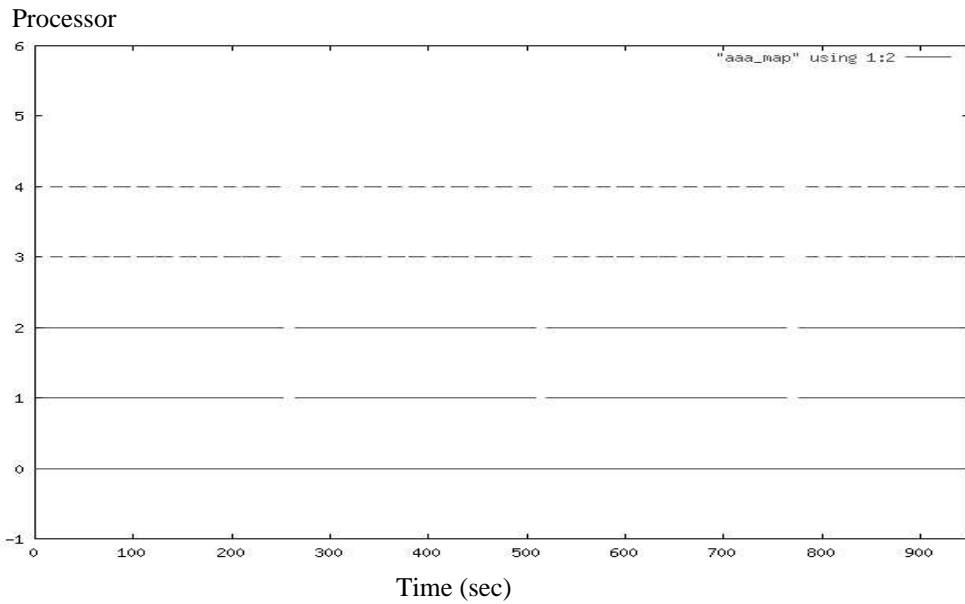
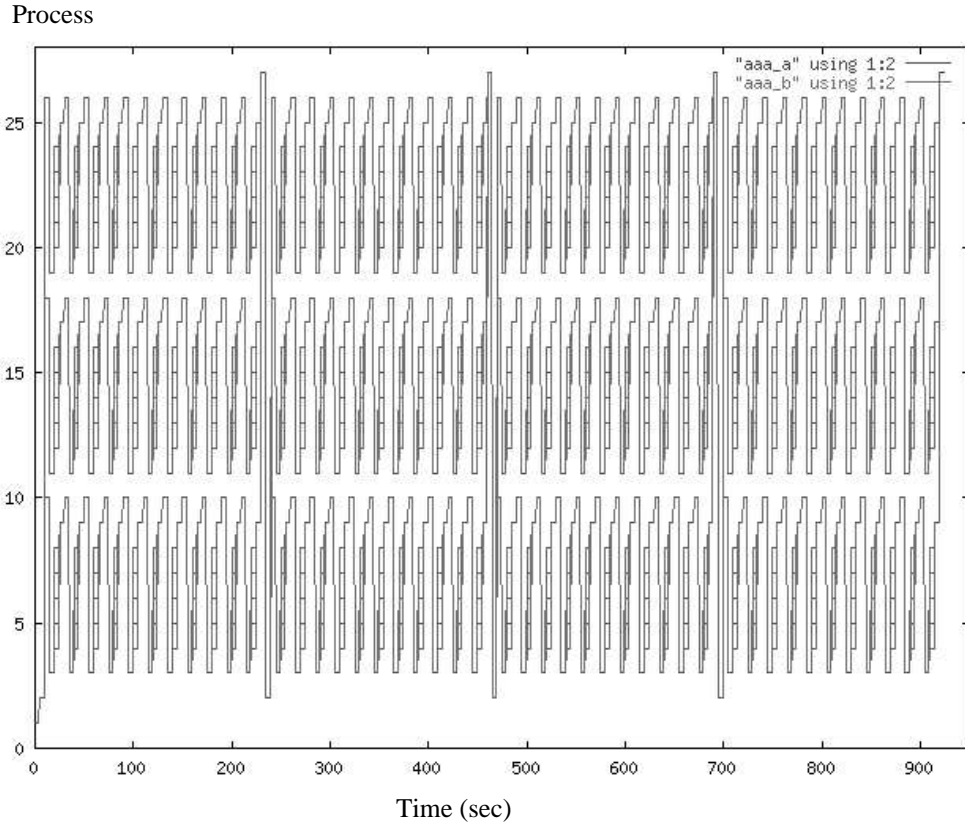
Processor



Test case 1: Job dependency and processor mapping

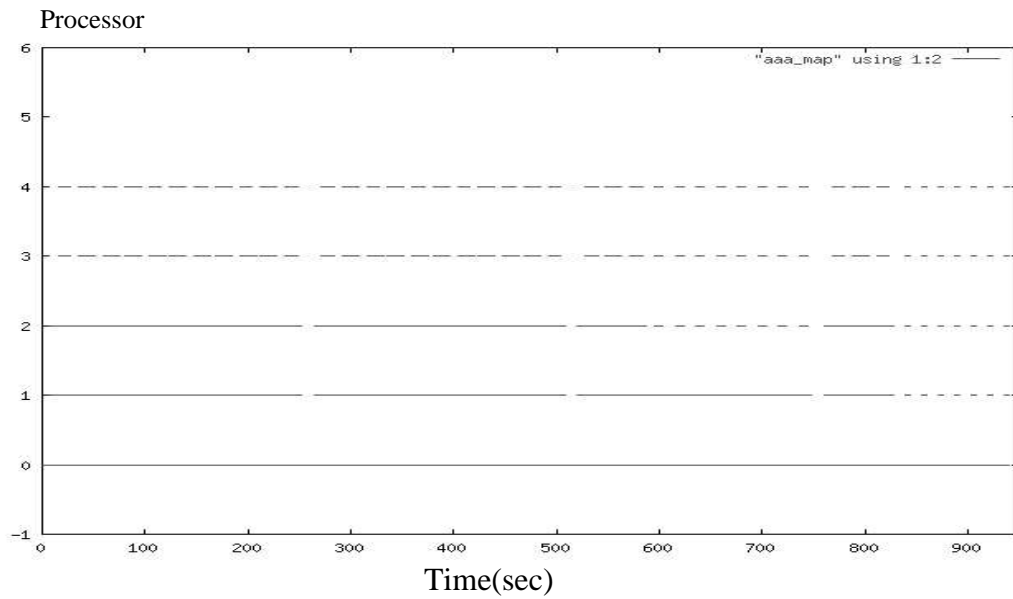
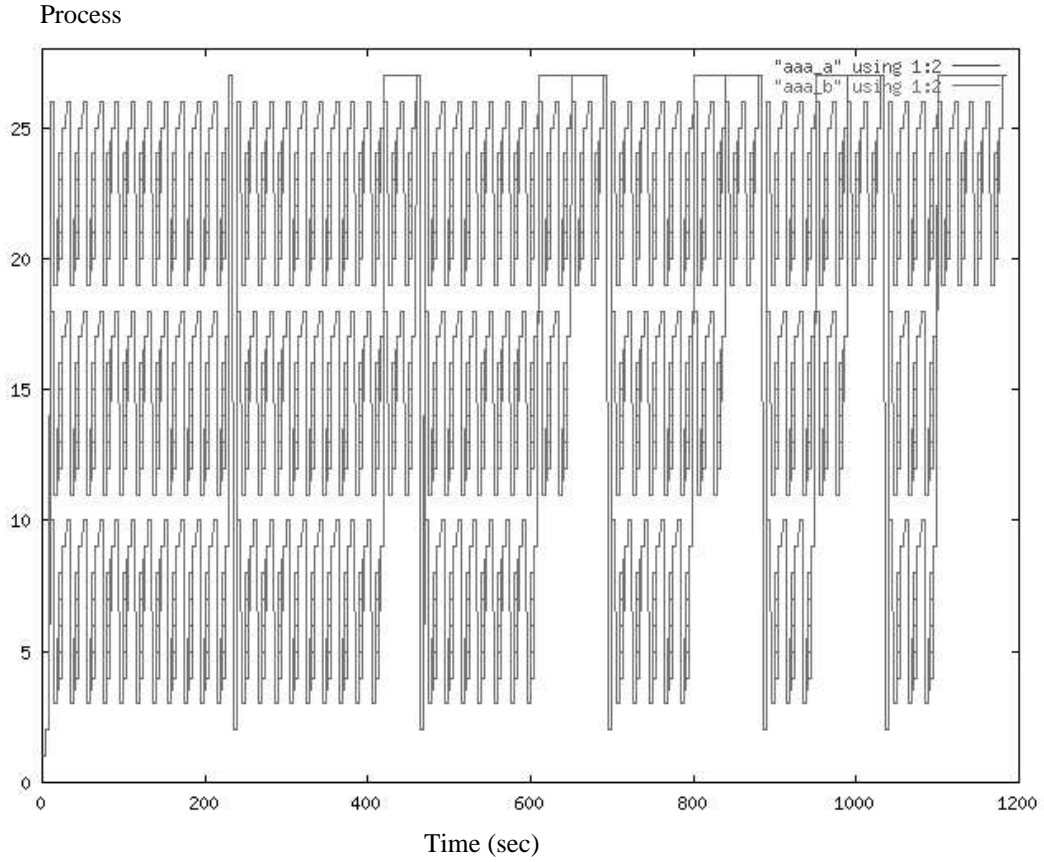


Test case 11: Network topology



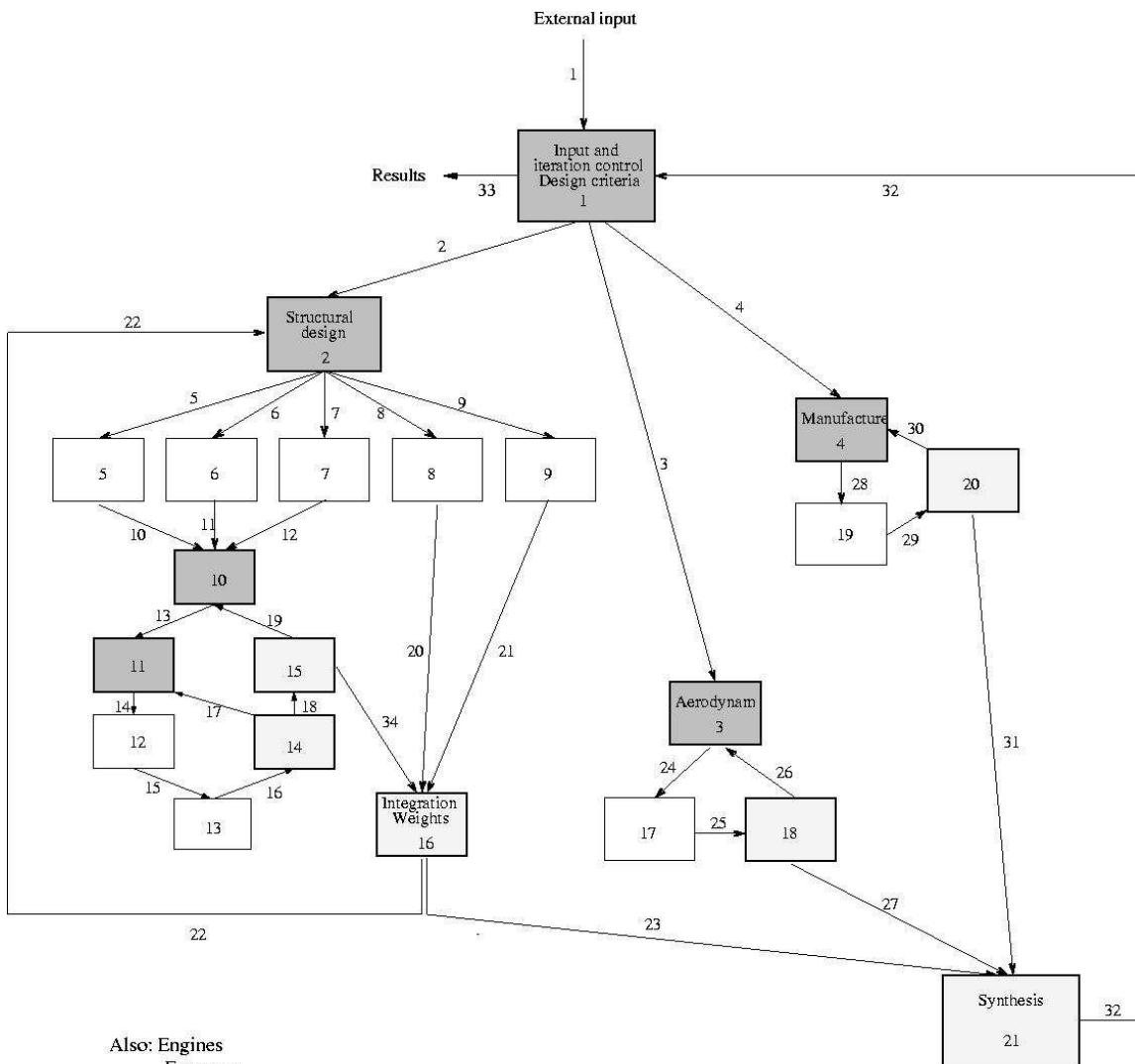
Test case 11: Job dependency and processor mapping

Constant convergence rate in all branches



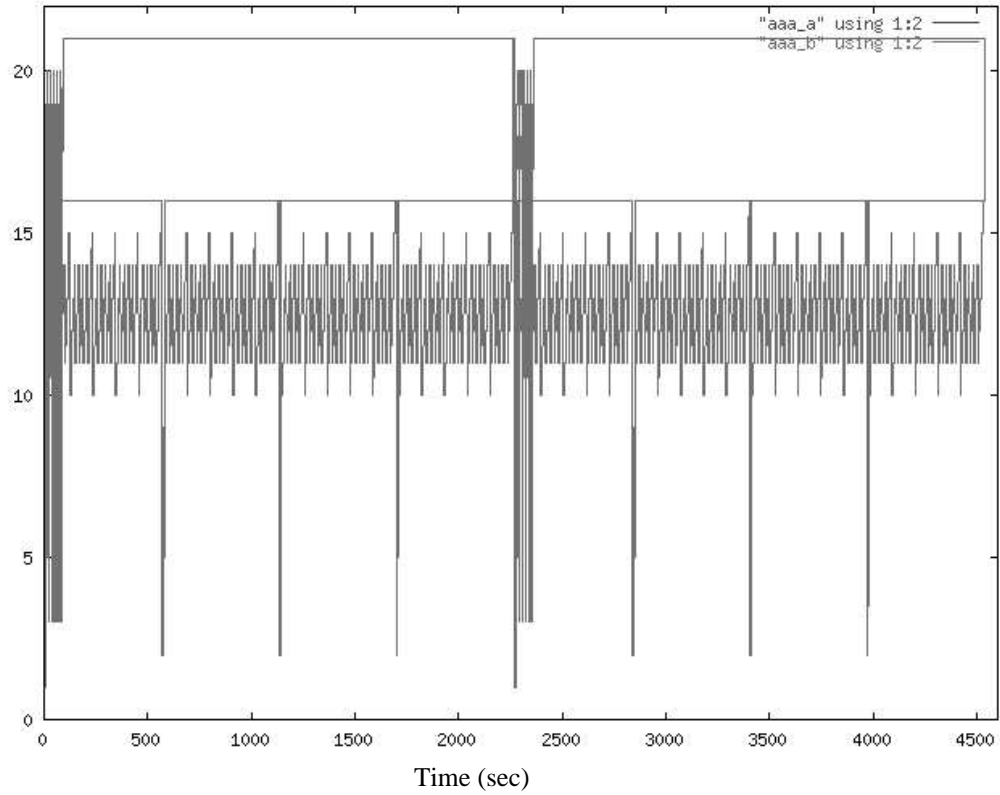
Test case 11: Job dependency and processor mapping

Convergence rate varies between branches as iteration proceeds

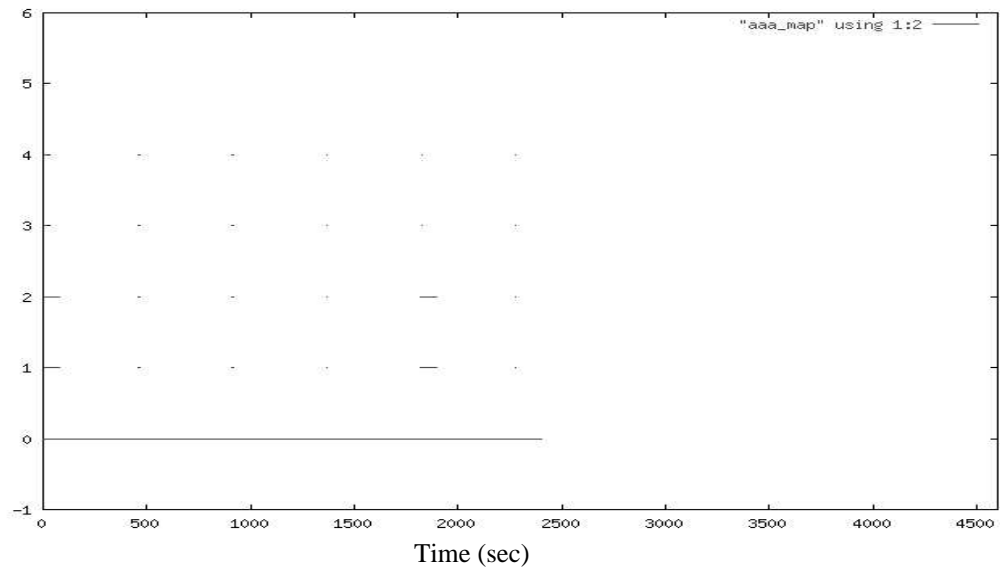


Also: Engines
 Economy
 Maintenance
 Safety
 Fatigue
 Life cycle cost
 etc.

Process

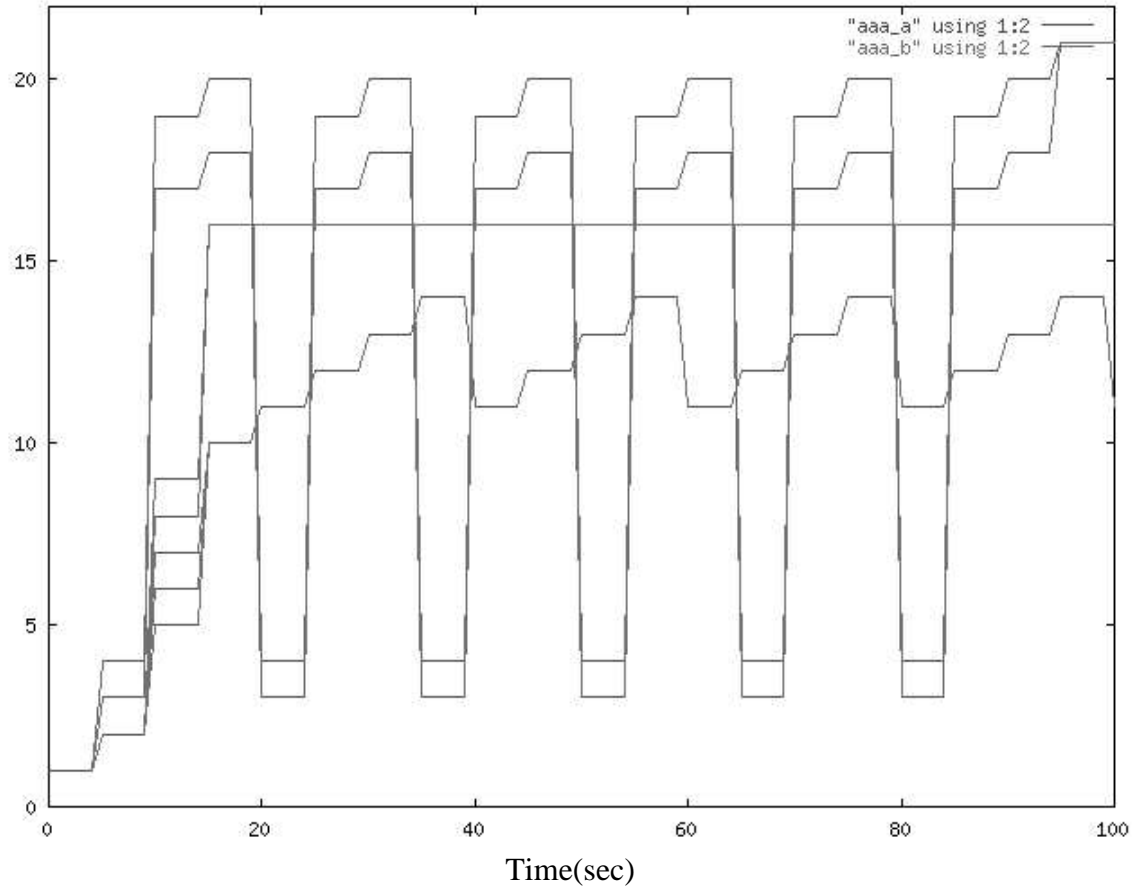


Processor

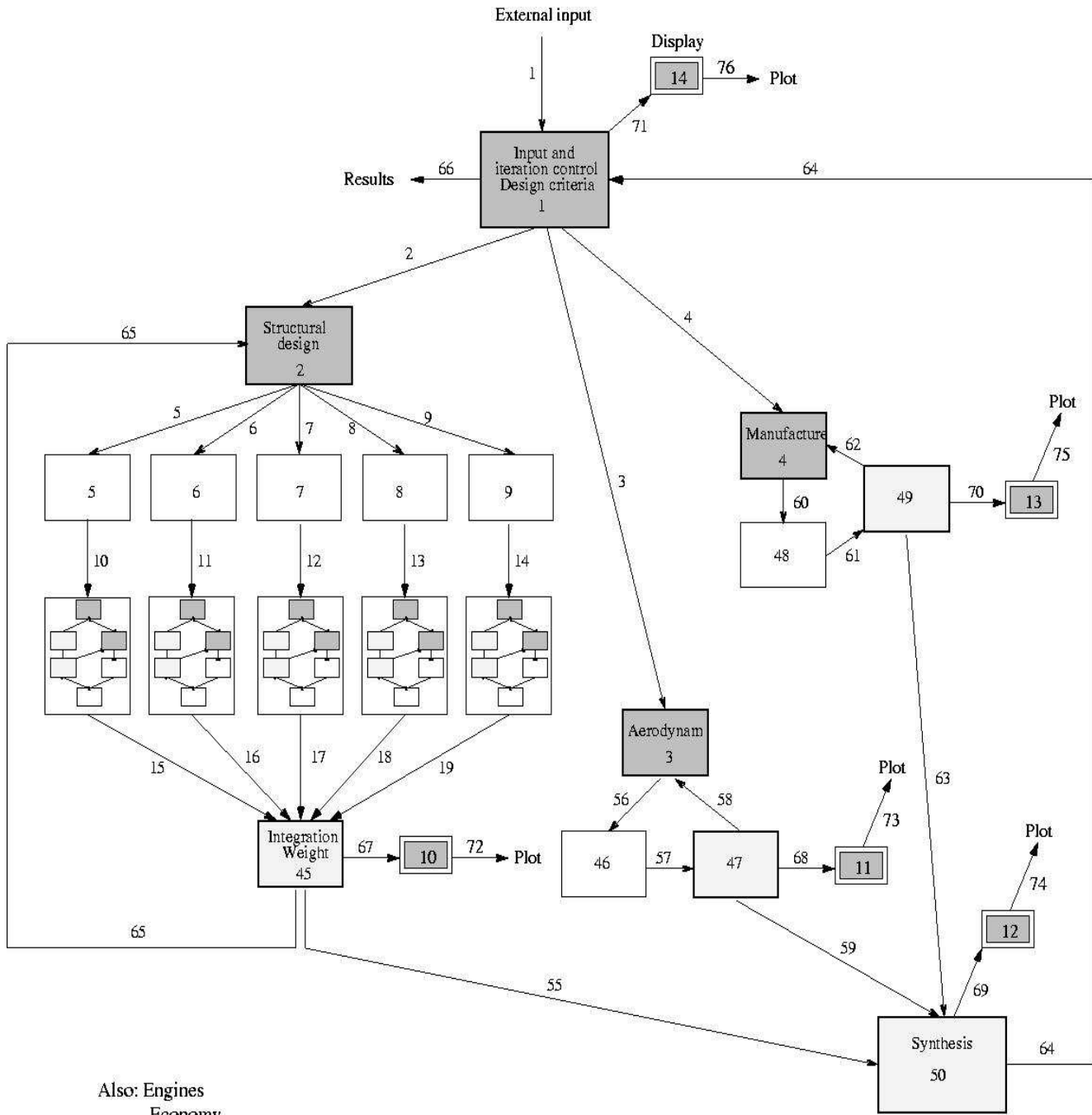


Design of an aircraft wing: Job dependency and mapping

Process



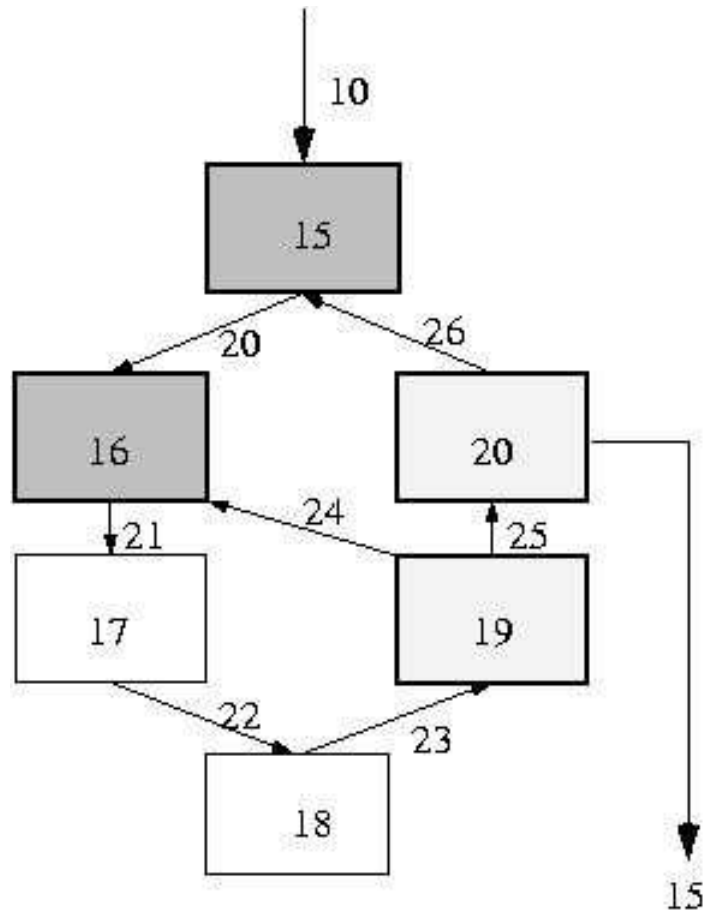
Design of an aircraft wing: The first 100 seconds



Also: Engines
 Economy
 Maintenance
 Safety
 Fatigue
 Life cycle cost
 etc.

Design of wing structure

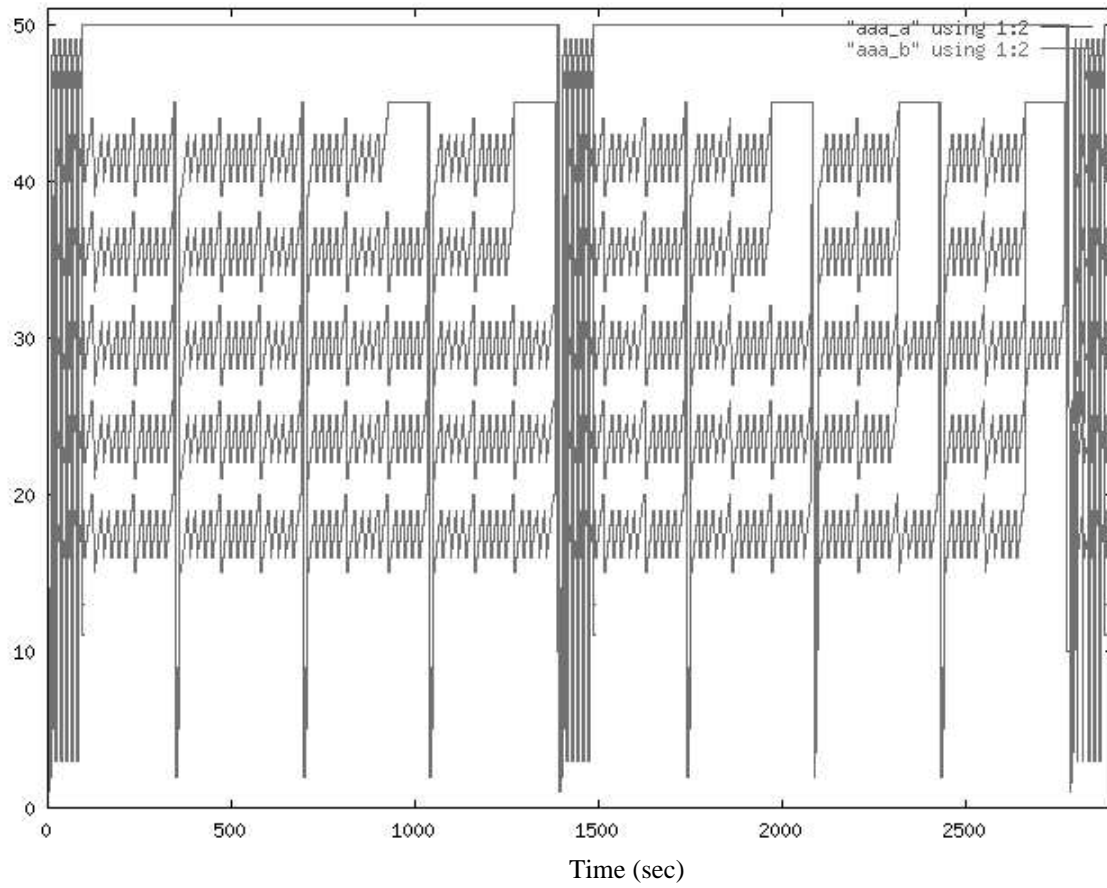
Design of an aircraft wing II: Network topology



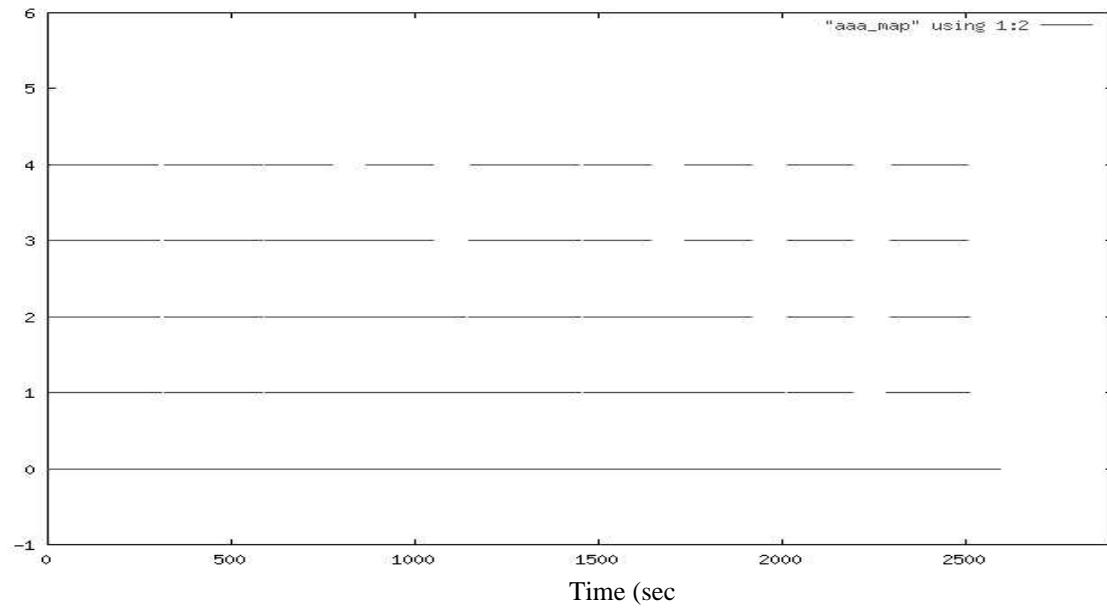
Iteration block

Figure shows the details of the iteration box between data sets 10 and 15 in the previous diagram

Process



Processor



Design of an aircraft wing II: Job sequence and process mapping

Design manager's aids:

KTOOL

Simulator for finding program dependencies

Supervisor and controller for live executions

CTMAP

Process to processor mapping

The future:

Build knowledge base

References on the subject of "Crack propagation in Composites"

- 1. Andersson B, Babuska I and Stehlin P, "Reliable Multiple-Site Damage Analysis of 3D Structures", FFA TN 1998-18.**
- 2. Andersson B, Babuska I and Stehlin P, "Reliable analysis of 3D Multiple-Site Fatigue Crack Growth", The Aeronautical Research Institute of Sweden and Texas Institute of Computational and Applied Mechanics, Univ of Texas, 1998.**
- 3. Nilsson K-F and Andersson B, "Analysis Methodology for Fatigue Crack Propagation and Residual Strength of Joints with Widespread Fatigue Damage", 1999 USAF Aircraft Structural Integrity Program (ASIP) Conference, San Antonio, Texas, USA, 30 Nov - 2 Dec 1999.**
- 4. Fawaz S and Andersson B, "Accurate Stress Intensity Factor Solutions for Unsymmetric Corner Cracks at a Hole", Proceedings of Fourth Joint DoD/FAA/NASA Conference on Aging Aircraft, 15-18 May 2000, St Louis, Missouri, U.S.**

References on concurrent design can be obtained from the author at stehlin@smr.ch.