For $I=0$ To 2 Loop
Wait until $clk$'s event and $clk$='1';
If (rgb[I] < 248) Then
  $P$=rgb[I] mod 8;
...
For I=0 To 2 Loop
Wait until clk’event and clk=’1’;
If (rgb[I] < 248) Then
P=rgb[I] mod 8;
...
Person-Month for 20k gates

### System Design
- Simulation: 40
- Schematic Entry: 60

### Logic Design
- Hierarchy, Generators: 50
- Logic Synthesis: 40
- High-Level Synthesis: 30

### Physical Design
- Placement & Routing: 70
- ≤ 1979
- ~1983
- ~1986
- ~1990
- ~1994
Methodology selection — the selection of the overall test strategies, structures, and design constraints.

Testability-driven synthesis and test structure insertion — the selection and insertion of testability-enhancement structure.

Design verification — the verification that the inserted test structure and the CUT operate as intended.
Papachristou et al. (CWR) - use binding techniques to eliminate self-loops and thus minimize test time.

Lee (Princeton) - perform scheduling and binding to minimize cycles and sequential depth.

Avra (Stanf.) - use register allocation and binding techniques to eliminate self-loops and maximize logic sharing.

Flottes et al. (LIRMM) - testability improvement based on testability analysis to provide test paths for each module.

Vishakantaiah et al. (Texas) - modify behavioral specification to increase controllability/observability of modules.
- Intermediate representation capturing the design during the whole synthesis process.
- Systematic testability analysis procedure.
- Transformational approach to design.
- Formalization of the whole synthesis problem as a global optimization problem.
- Use efficient optimization heuristics.
Performance
Cost
Power
Testability
...

Traditional Approach

Allocation
Scheduling
Binding
Test insertion

Transformation-Based Approach

Optimization
Testability Analysis

- Quantitative testability (controllability/observability) measure reflects:
  - test generation cost,
  - difficulty of achieving high fault coverage, and
  - test application cost.

- An algorithm should accept the design representation and generate the controllability and observability of the basic units, taking into account:
  - the structure of a design,
  - the depth from I/O ports for access, and
  - the characteristics of the functional blocks used.
Testability Analysis

Example

\[
\begin{align*}
\text{In} & \quad \text{Out} \\
L_1, S_1 & \quad L_8, S_4 \\
L_4, S_5 & \quad \geq \\
L_9, S_3 & \quad L_{10}, S_3 \\
L_{11}, S_2 & \quad \text{C}_1 \\
\end{align*}
\]
- Low-impact testability enhancement.

- High-impact testability enhancement.

- Selection of testability improvement techniques is based on:
  - Testability analysis results.
  - Basic test strategy.
  - Test requirement.
  - Global optimization.
Merger transformation -> data path allocation.

\[
\begin{align*}
S_{11} & \quad S_{11} & \quad S_8 & \quad S_8 & \quad S_{11} & \quad S_8 \\
> & \quad + & > & \quad + & \quad > & \quad + \\
S_{11} & \quad S_{11} & \quad S_8 & \quad S_8 & \quad S_{11} & \quad S_8 & \quad S_8
\end{align*}
\]

Selection of nodes to merge was traditionally based on connectivity/ closeness, resulting in difficult-to-test designs:

- Nodes with good controllability and bad observability are usually merged together, since they are close to the primary inputs.
- Nodes with good observability and bad controllability are also merged together.
- Many loops are generated.
Testability analysis results are used to guide the selection of merged nodes.

The objective is to generate a data path with good controllability and observability for all of the nodes.

**Controllability/Observability-Balance Allocation:**

- Merge nodes with good controllability and bad observability to node with good observability and bad controllability.

  => Low-impact testability enhancement.
The most difficult-to-test register is selected.

Module binding is done to change it into a scan register.

The implication in terms of hardware overhead and performance degradation will immediately be checked.

If the testability requirement still cannot be satisfied, repeat the above process.

To speed up the algorithm, several registers can be selected at each iteration. They should be in different loops.
Fault-coverage 91.72%

FC scanning C: 92.83%

FC scanning M: 90.83%

FC scanning C & S: 97.08%

control part

data path
VHDL Specification

Compiler Front-end

Testability-Improvement Transformations

Intermediate Representation Testability Measure

High-Level Synthesis Testability Analysis

DFT Architecture Primitives

RTL Implementation with built-in DFT
Integration of design and test is the only solution to the test problem.

Testability must be considered at all stages of the design process, including the synthesis process.

Transformation-based approach to HLTS is very efficient.

- Quick exploration of design space.
- Accurate feedback to designers.
- Enhance automation.
- Support the integration with even more functionalities.

HLTS leads usually to low overhead for the testable designs.