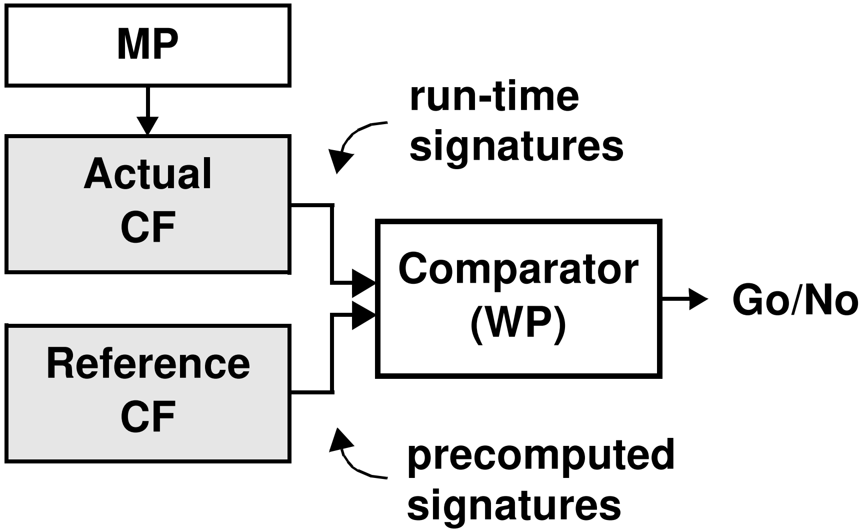
Watchdog-processors

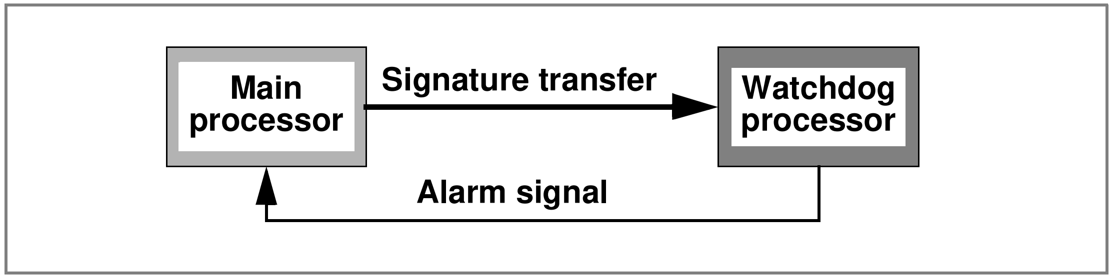
- **Simple coprocessor**
- **Concurrent detection of *control flow* errors in the main processor (50-70% of the transient errors)**
- **Other tasks of a WP?**
  - **Basic time-out checks on the signature transfer (~timer)**
  - **Credibility checks of the computed values**
  - **Check of the memory access rights (~MMU)**
- **Base of the control flow monitoring?**
  - **Program: statements + control operations**
  - **Program control graph: vertices + edges**
  - **Encoding: signatures assigned to the vertices**
  - **Control flow: signature sequence**
  - **WDP: actual and reference signature sequences**

General scheme of a watchdog-processor

- **Test scheme:**



- **Scheme of a WDP**



Watchdog-processors as control flow checkers

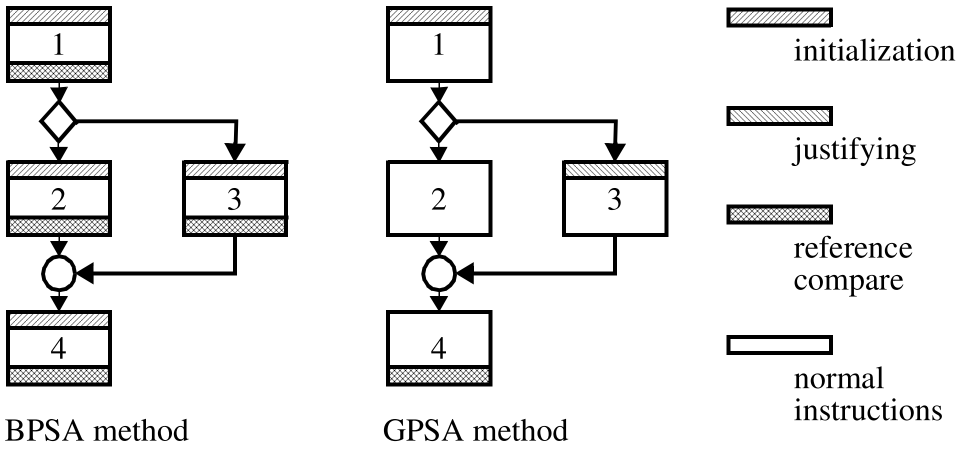
- **Types of the watchdog-processors:**

Typical methods		Reference control flow		
		Stored signatures	Extracted program	Embedded signatures
	Derived signatures	RMP	WDP	BPSA
	Assigned signatures	ESIC	SIC	SEIS

- **Derived signatures: instruction fetch sequence**
  - Instruction prefetch queue?
  - On-chip cache?
- **Assigned signatures: the MP sends the actual signatures**
  - MP overhead
  - No instruction-level checking (error latency)
  - Preprocessor program needed
  - Complex and slow HW (~ the MP)

Derived signatures

- BPSA: Basic Path Signature Analysis
  - Vertices of the CFG: branch-free instruction blocks
  - Actual signature:
    - Compaction of the run-time instruction codes
    - Generated by a HW compaction device (LFSR)
  - Reference signatures:
    - Precalculated, compaction of the instruction codes of the vertex
    - Embedded in the code at the end of instruction blocks
  - Checks:
    - Comparison at the end of the instruction block
    - Compaction device is reset (or preset the reference)
    - Checked: sequence of instructions inside of the instruction blocks



GPSA: Generalized Path Signature Analysis

- Vertices of the CFG: sequences of instruction blocks
- Justifying signatures: equilizing the different paths
- Identifying special signatures:
  - Tag bits
  - Special instruction codes
- Checks:
  - Also the sequence of instruction blocks
  - At the merge vertex (equilized by the justifying signatures)
  -
  -
  -

RMP: Roving Monitoring Processor

- Actual signatures:
  - Generated by a HW compaction device asynchronously
  - Signature pipelines (FIFO) are used to connect multiple processors
- Reference signatures:
  - Downloaded to the WP before program run

WDP: Watchdog Direct Processing

- **Actual signatures:**
  - Generated by a HW compaction device
  - Branch and target instructions are detected automatically (address bus is monitored)
- **Reference signatures:**
  - Embedded into a (shadow) watchdog program (reference value + next instruction)
- **Advantages:**
  - No program modification is required
  - Relatively simple watchdog processor (a few instructions)

Assigned signatures

- Common properties:
  - Vertices of the CFG: statements of the high-level language
  - Actual signatures: Identifying the program state
    - Assigned by a preprocessor before compilation
    - Transferred explicitly by the main program to the WP
  - Reference signature database: Downloaded to the WP before start
  - Preprocessor functions:
    - Parsing the program, source
    - Extracting the CFG
    - Assigning the signatures (encoding of the CFG)
    - Inserting signature transfer code into the program source
  - Checks:
    - Valid signatures
    - Valid sequence of signatures
  - Drawbacks
    - Performance degradation
    - (Re)compilation is needed
    - Reduced fault coverage (instruction sequence is not checked)
    - Increased error latency (signatures are assigned to high level statements)

SIC: Structural Integrity Checking

- **Assigned signatures: Incremental numbering of the vertices**
- **Reference signatures:**
  - **Embedded into a watchdog program (compiled graph tracing)**
- **WP implementation: General purpose processor + reference program**

```
main()
{
   int i,j=0,k[12];

   for (i=0; i<12; i++){

      j=i+j*(i+1);

      k[i]=j;
   }

}
```

Original C program

```
main()
{
   int i,j=0,k[12];
   SEND_SIG(S1);
   for (i=0; i<12; i++){
      SEND_SIG(S2);
      j=i+j*(i+1);
      SEND_SIG(S3);
      k[i]=j;
   }
   SEND_SIG(S4);
}
```

Preprocessed C program

Figure 1 Assigned signatures example

ESIC: Extended Structural Integrity Checking

- Assigned signatures: Numbering of the vertices
- Reference signatures:
  - Signature database (CFG in a sparse matrix format)
  - General search/compare engine
- WP implementation: General purpose processor + reference database
- Extended checks:
  - Procedure calling mechanism (reference stack)
  - Multiple processes (context switching in the search engine)
  - Interrupts (special signatures signaling the start)

- SEIS: Signature Encoded Instruction Stream
  - Assigned signatures: Redundant encoding of the vertices
  - Reference signatures: A single reference signature is enough
  - WP implementation: Simple and fast combinational logic

Comparison of the methods

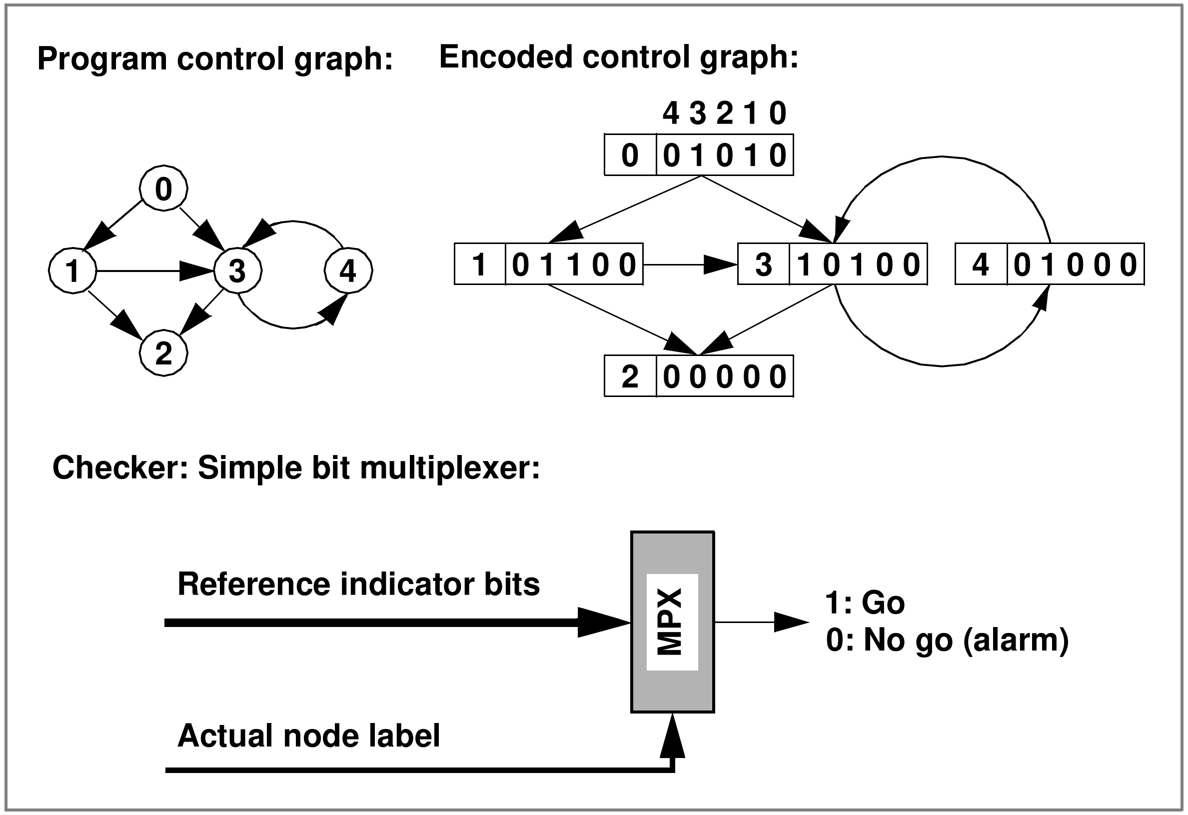
Table 1

		Reference control flow			
		Stored reference signatures	Extracted reference program	Embedded reference signatures	
Actual control flow	Derived actual signatures	-	-	+	no WP database
		+	-	+	simple, fast WP
		+	+	-	no MP overhead
	Assigned actual signatures	-	-	+	no WP database
		-	-	+	simple, fast WP
		-	-	-	no MP overhead

Signature Encoded Instruction Stream (SEIS)

- **Main idea:**
  - Redundant signatures:
    - Vertex + successor identifiers “Where I’ am and where can I go?”
- **Advantages:**
  - Actual signature is reference for the successor validation
  - Simple and fast (combinatorial) checker
- **Basic example: 1 of N codes**
  - Vertex identifier: Unique label

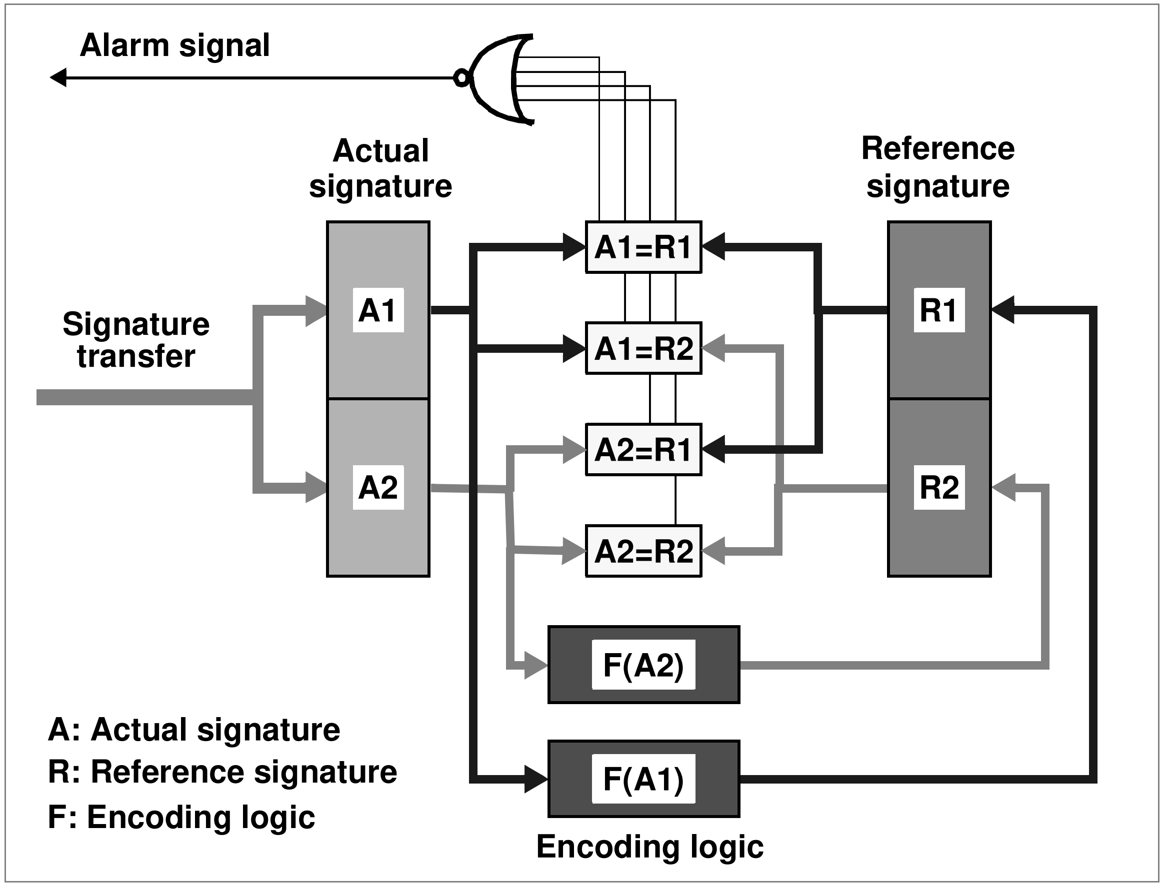
- **Successor identifiers: Indicator bits**



- **Indicator bit field width in large graphs?**

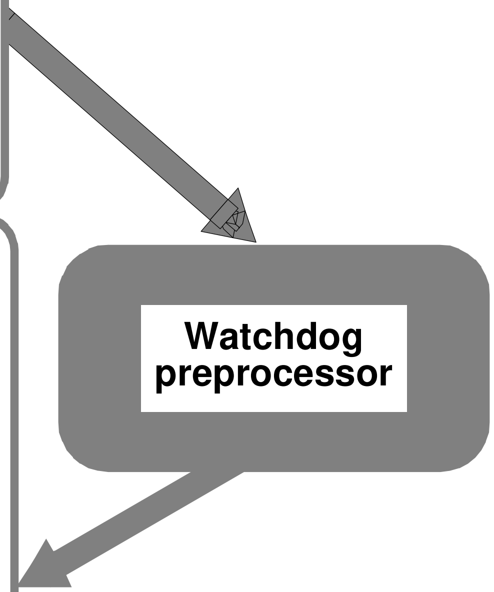
Possibility for simplification

- Very few successors of a vertex
  - Few multidirectional control transfer instruction
- Actual signature of a vertex: reference signature to be compared with the next
- Ordered signatures among paths in the CFG

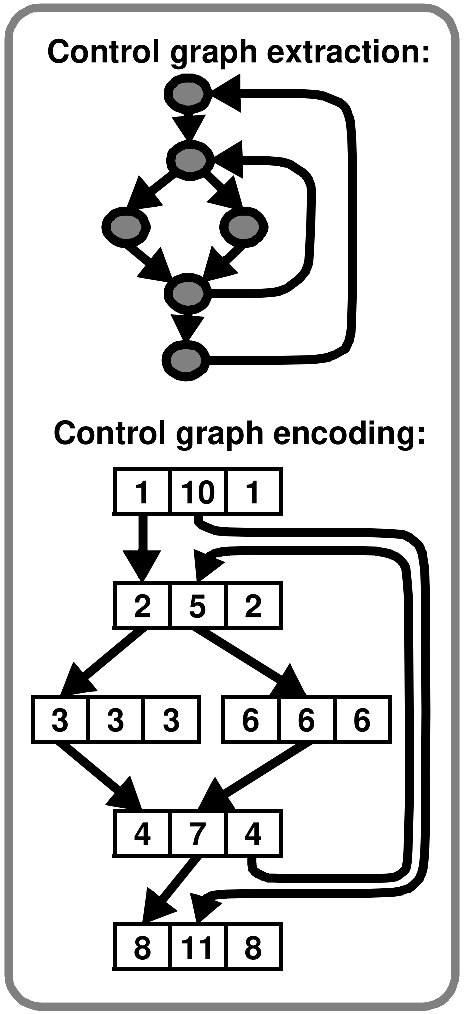


Encoding algorithm

```
procedure() {
  l for (i=0; i<MAX; i++) {
    l if (a>b) {
      l stat1;}
      else {
      l stat2;}
  l  }
l}
```




```
procedure() {
  SEND(1,10,1);
  for (i=0; i<MAX; i++) {
    SEND(2,5,2);
    if (a>b) {
      SEND(3,3,3);
      stat1;}
    else {
      SEND(6,6,6);
      stat2;}
    SEND(4,7,4);}
  SEND(8,11,8);
}
```



Properties

- **Actual signatures:**
  - assigned
  - no bus observability
  - preprocessor
- **Reference signatures**
  - precomputed
  - embedded (no reference database)
- **Checker architecture: simple (multitasking)**
- **Control flow checking of C programs**
- **Functions: Encoded control graphs (SEIS method)**
- **Function calls:**
  - Function call stack in the WP (ESIC method):
    - SOP: push the actual reference; SOP not checked
    - EOP: pop the stored reference; EOP checked
  - Whole program as a single control graph (connecting nodes)
- **Tasks: Independent references / stacks**

Hardware realization of the SEIS checker

Parts:

- Reference signature register
- Actual signature and encoding function (logic)
- Signature comparator
- Signature stack

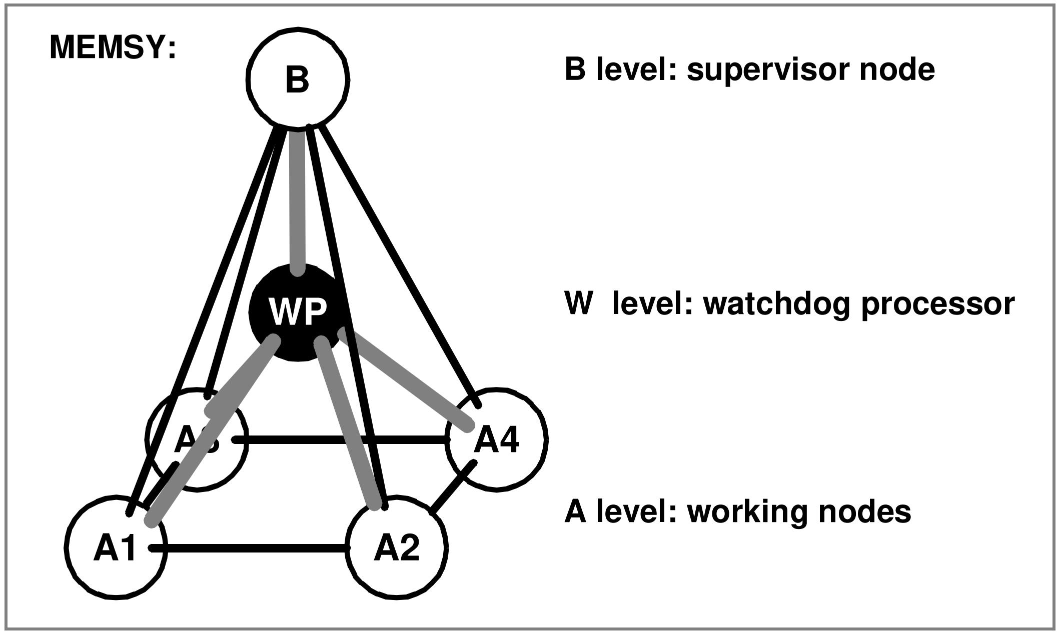
Multiprocessor watchdog

- **Multiprocessing - multitasking environment**
  - **High-speed signature validation**
  - **Elimination of the reference database**
- **WP integrated into the fault tolerance *system***
  - **Status informations for the supervisory node**
  - **Two-phase checkpoint generation and recovery scheme**
- **Hierarchical check of the applications**

Level	Checked attribute
Processor	- Actual task ID (task changes)
Task	- Function ID (function changes and returns) - Signature transfer time-out (fatal errors)
Function	- Valid signature flow (encoded control graph)

Reduction of the hardware complexity

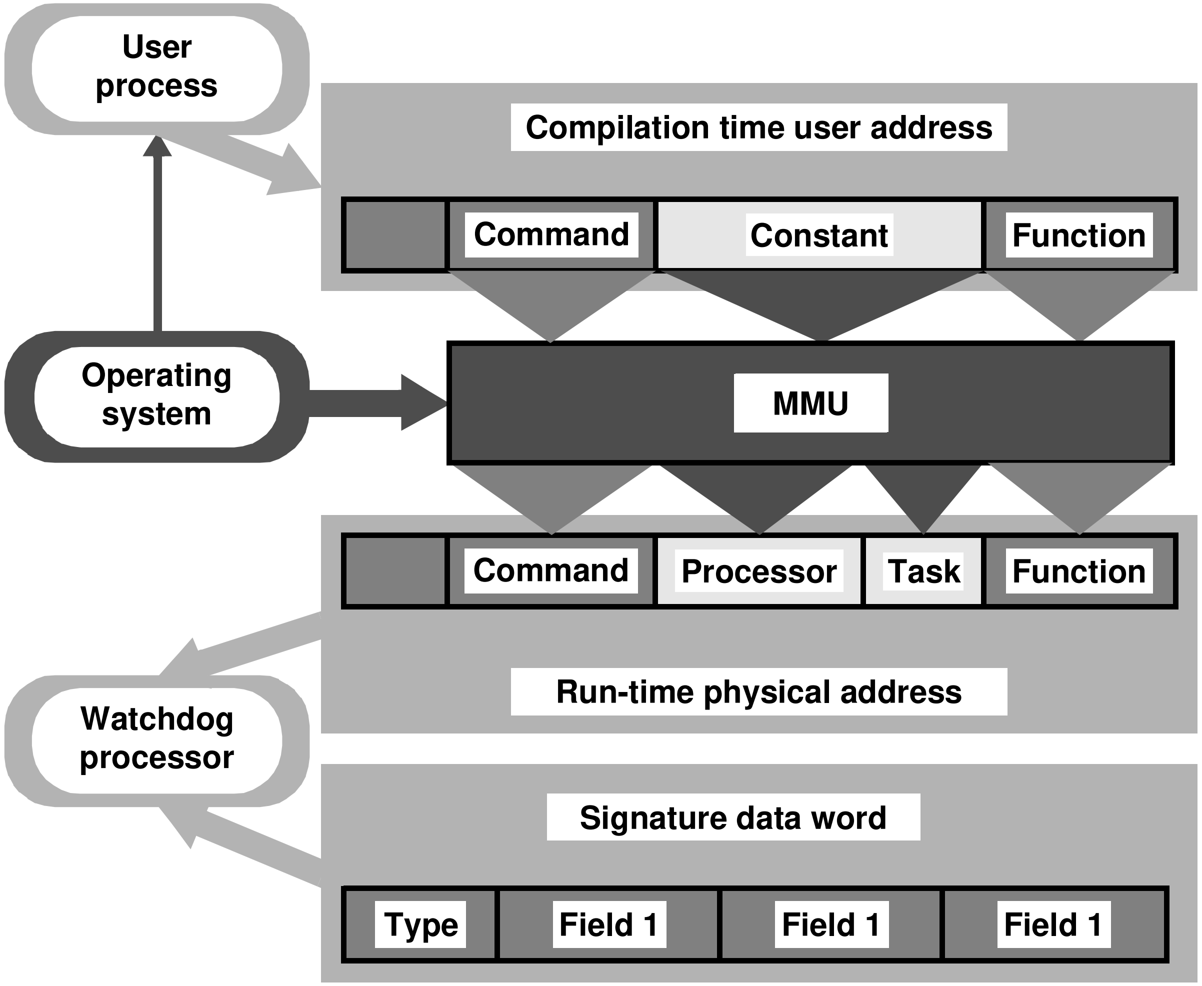
- Check multiple computing nodes by a single WP
- Internal resource sharing in the WP
- Shared resources:
  - Signature stack (one stack, split)
  - Signature comparator
  - Timer (separated task time variables)



Format of an assigned signature

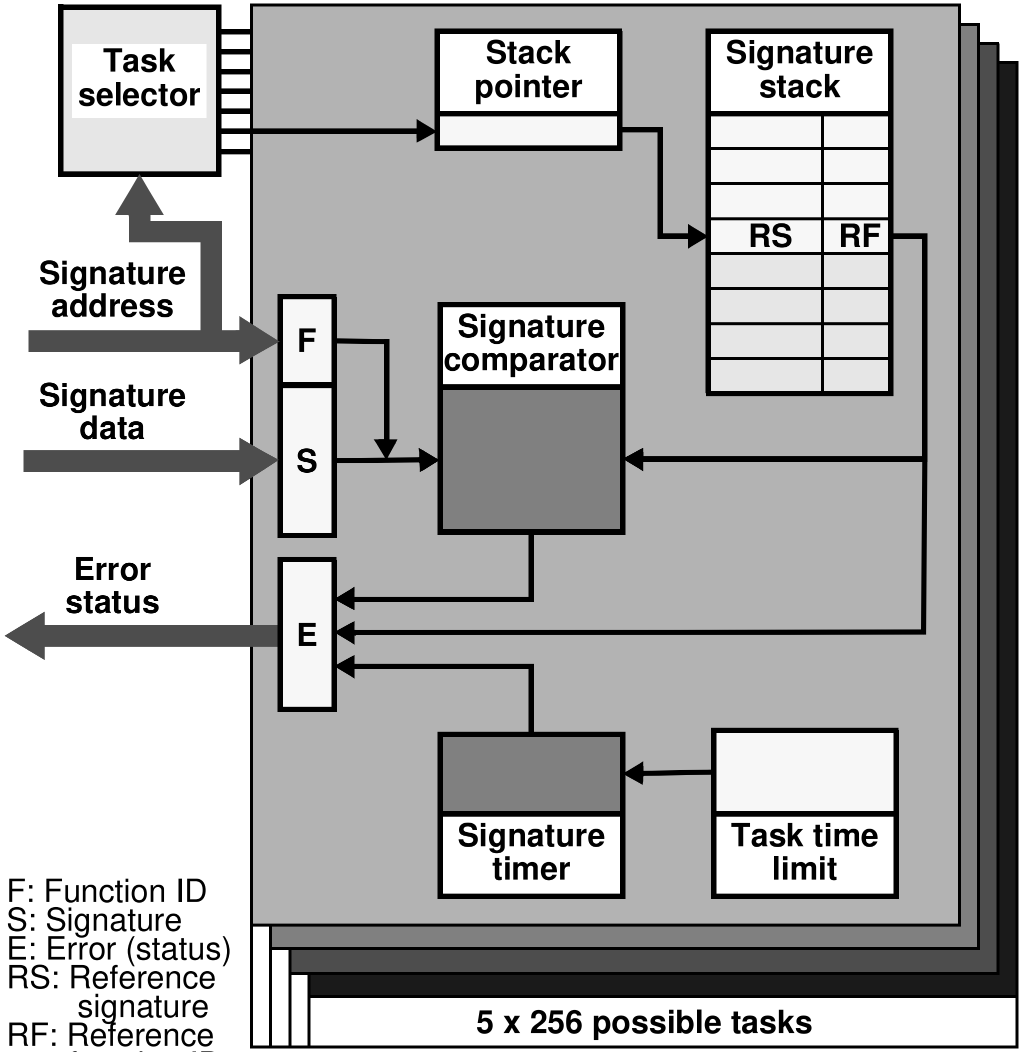
- **Physical address word:**
  - Command ID (commands: signature transfer, controlling the WP)
  - Processor ID (4 processor per computing node)
  - Task ID (256 task per computing node)
  - Function ID (1024 function per task)
- **Data word:**
  - Signature type: NRM, SOP, EOP, NTM
  - Three signature fields for the control graph encoding

Address translation by the MMU:



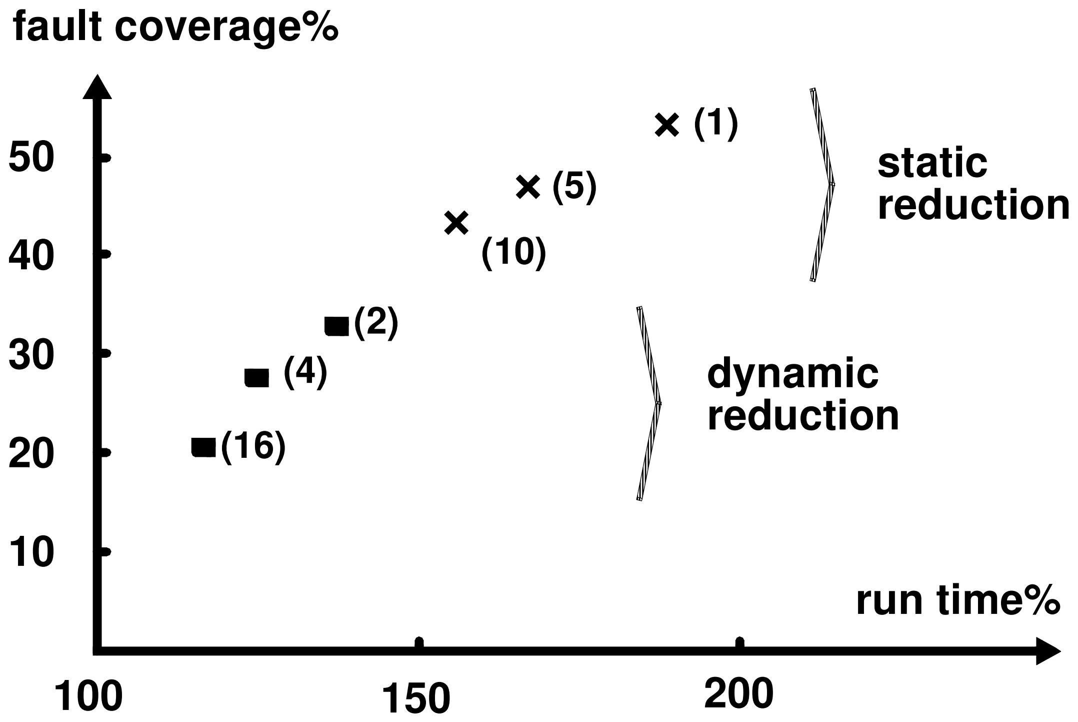
- Processor and task ID need not be known at compilation time
- Shared libraries can be used
- Access rights are automatically checked (logic structure of the task check

Goal: Independent checks for each task

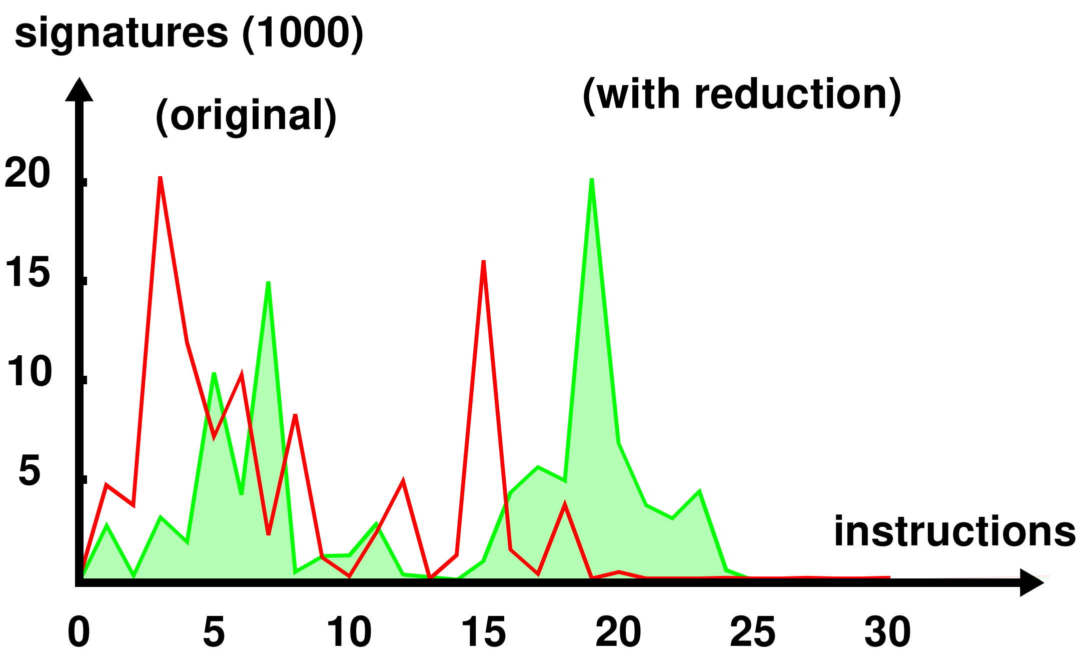


F: Function ID
S: Signature
E: Error (status)
RS: Reference signature
RF: Reference function ID

Fault coverage



Error latency



Evaluation of WDP

- 60-80% coverage for control faults, *assures the execution of data checks*
- 10-20% performance loss
- Short error latency vs. communication ba(n)dwith
- *SW implementation* possible
  but reasonable only if the WDP runs on another processor
  SMP, or distributed
- Check of *multitasking* (process synchronization) possible (CSP-like)
- No COTS support till yet, but low complexity