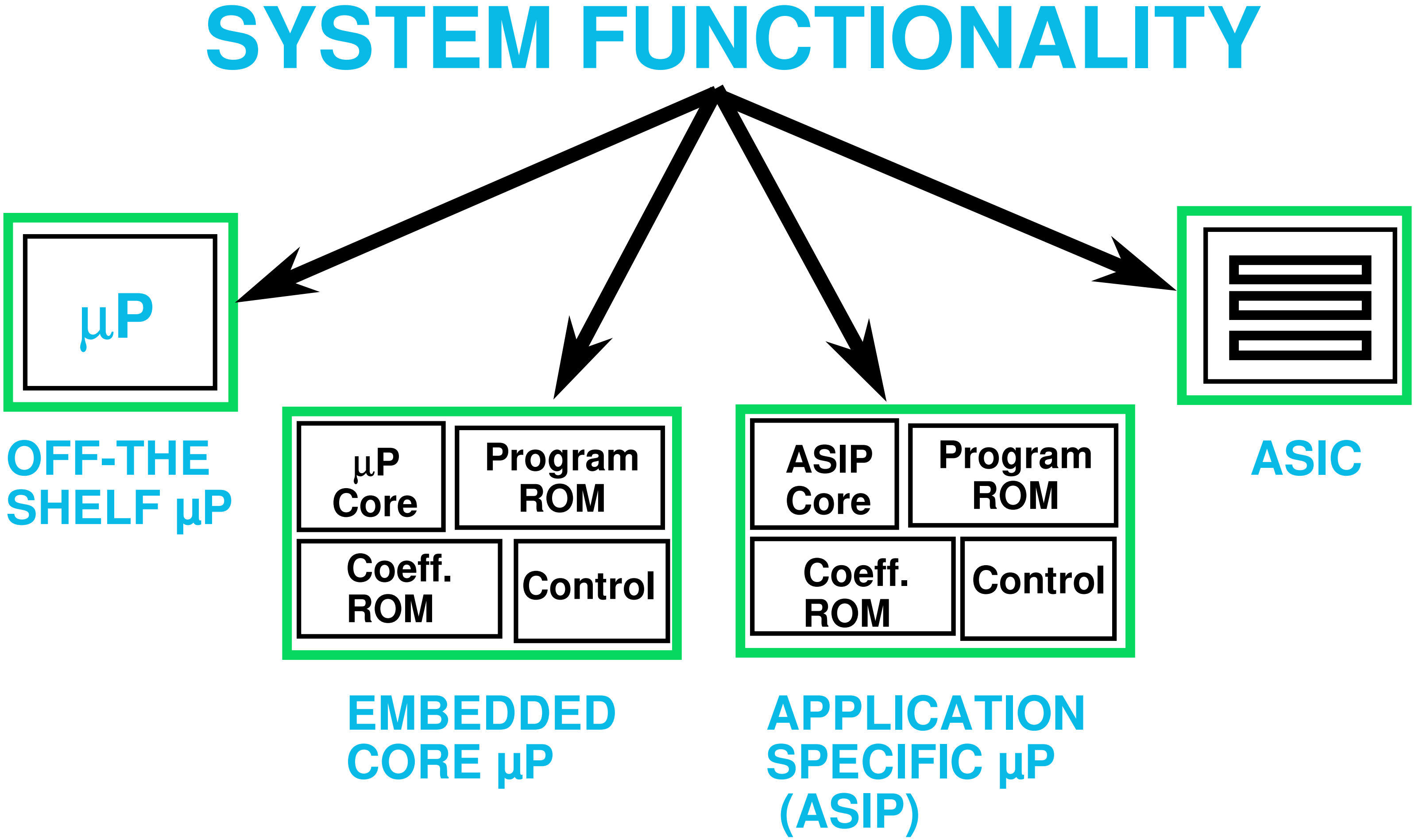
Implementing Digital Systems



SYSTEM FUNCTIONALITY

OFF-THE SHELF µP

EMBEDDED CORE µP

APPLICATION SPECIFIC µP (ASIP)

ASIC

What is an ASIP?

Application-Specific Instruction Processor

Processor architecture tailored not just for application domain (e.g., DSP, microcontrollers), but for specific sets of applications (e.g., audio, engine control)

ASIP characteristics

– Greater design cost (processor + compiler)

+ Higher performance, lower power than commercial cores, more flexibility than ASIC

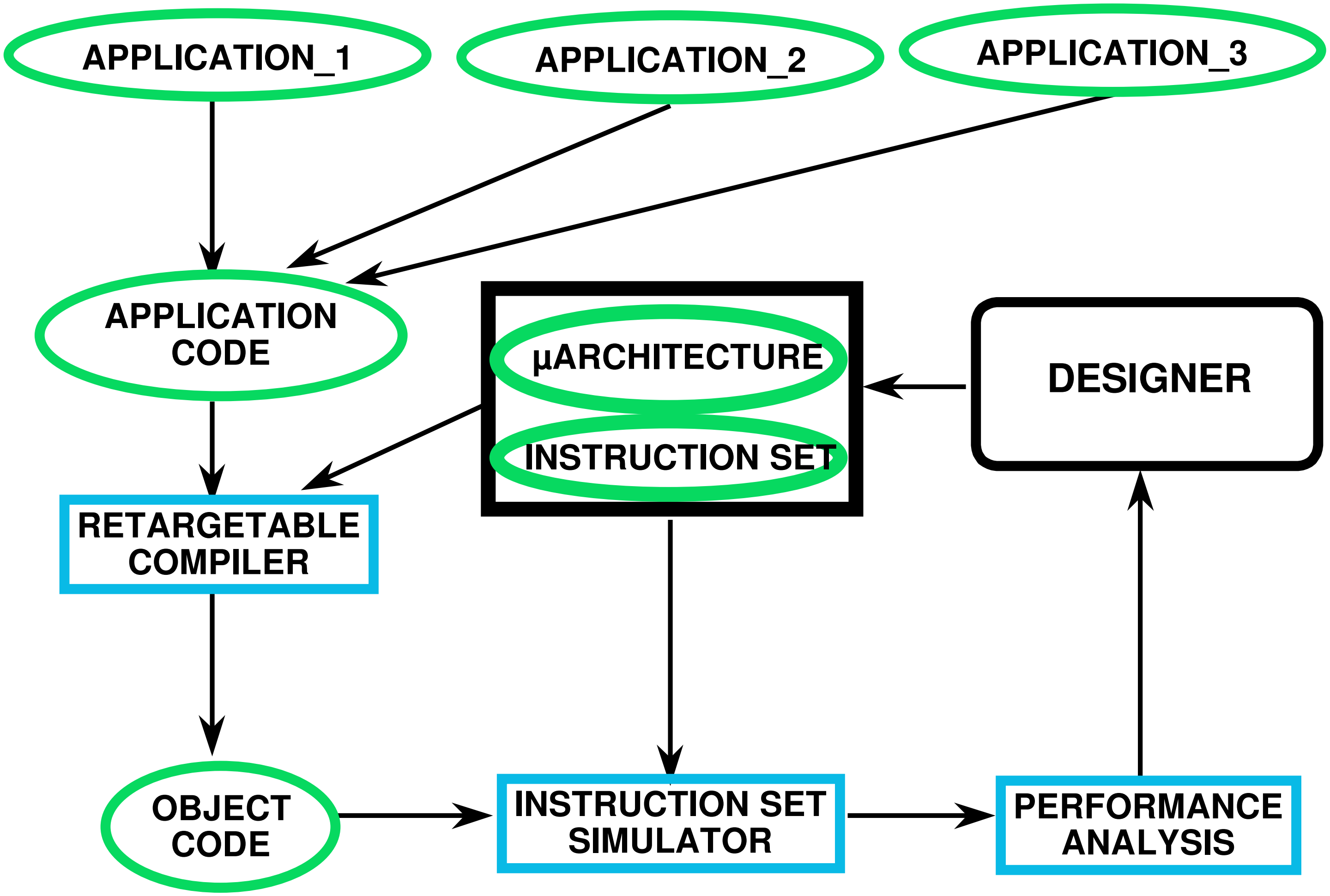
ASIP Design

Given a set of applications, determine μarchitecture of ASIP (i.e., configuration of functional units in datapaths, instruction set)

To accurately evaluate performance of processor on a given application need to compile the application program onto the processor datapath and simulate object code

However, the μarchitecture of the processor is a design parameter!

Processor Design Flow


APPLICATION_1

APPLICATION_2

APPLICATION_3

APPLICATION CODE

µARCHITECTURE

INSTRUCTION SET

DESIGNER

RETARGETABLE COMPILER

OBJECT CODE

INSTRUCTION SET SIMULATOR

PERFORMANCE ANALYSIS

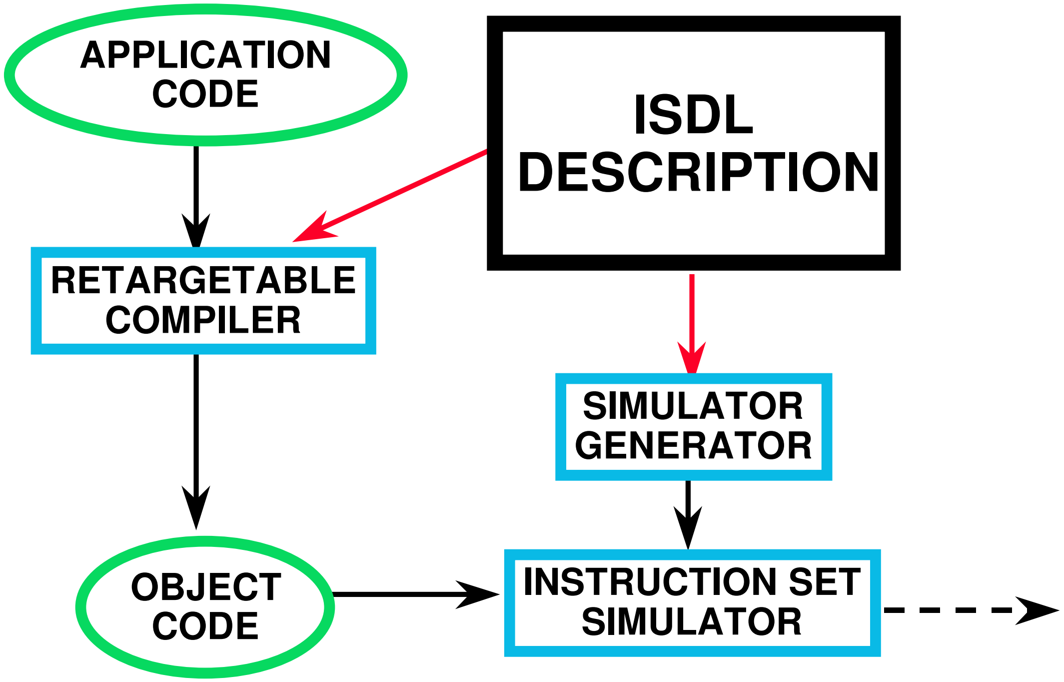
Instruction Set Description Language

Describes processors with instruction-level parallelism, i.e., multiple, possibly irregular, execution units, and arbitrary interconnect topology

Multiple fields for multiple execution units and constraints to disallow illegal combinations of operations

Describes VLIW architectures, can be generalized to superscalar

Using ISDL


APPLICATION CODE

ISDL DESCRIPTION

RETARGETABLE COMPILER

SIMULATOR GENERATOR

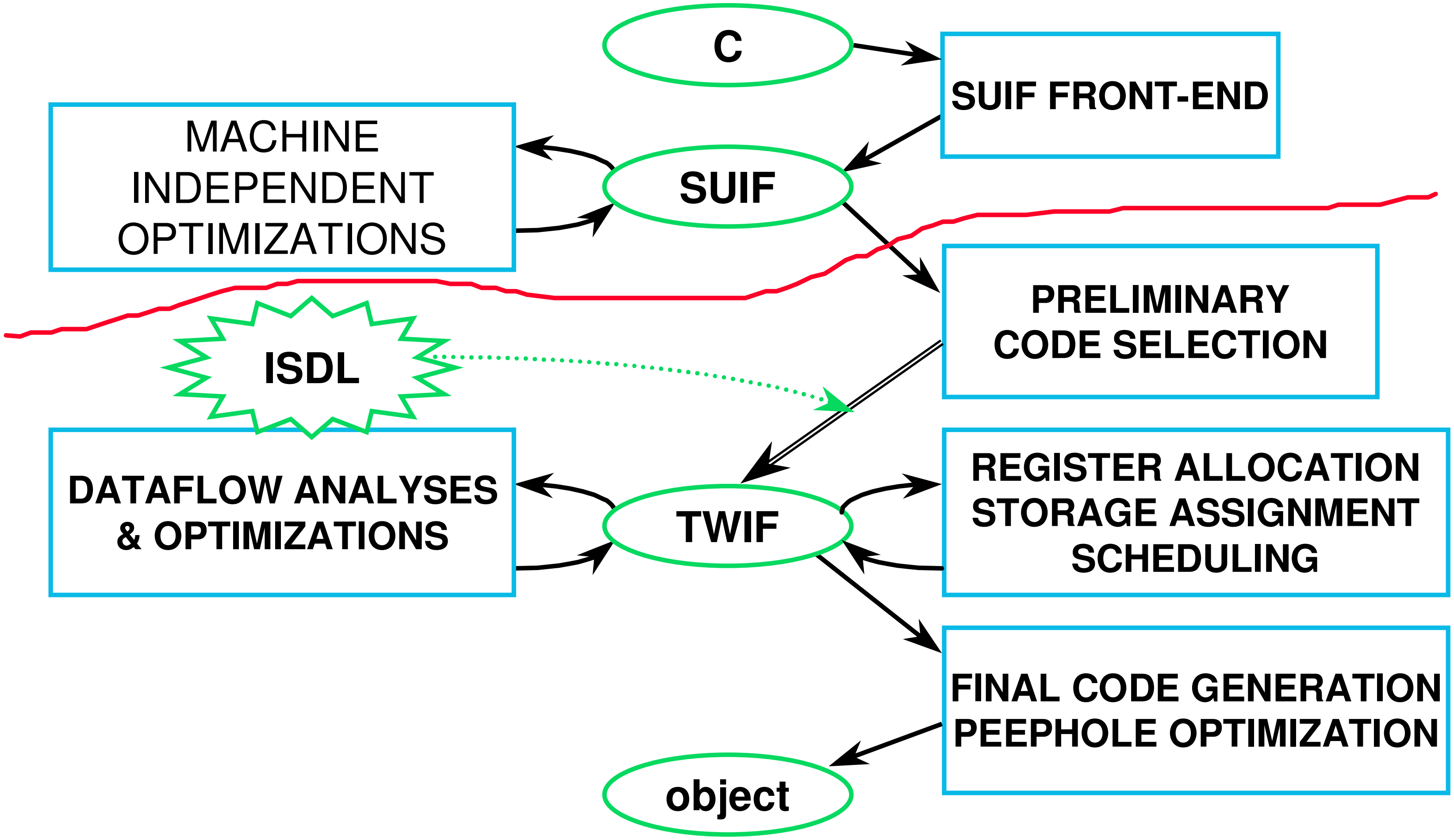
OBJECT CODE

INSTRUCTION SET SIMULATOR

Automatic generation of assembly code, assembler, and instruction set simulator

Code <u>optimization</u> is the most challenging task

Compiler Framework



C

SUIF FRONT-END

MACHINE INDEPENDENT OPTIMIZATIONS

SUIF

PRELIMINARY CODE SELECTION

ISDL

DATAFLOW ANALYSES & OPTIMIZATIONS

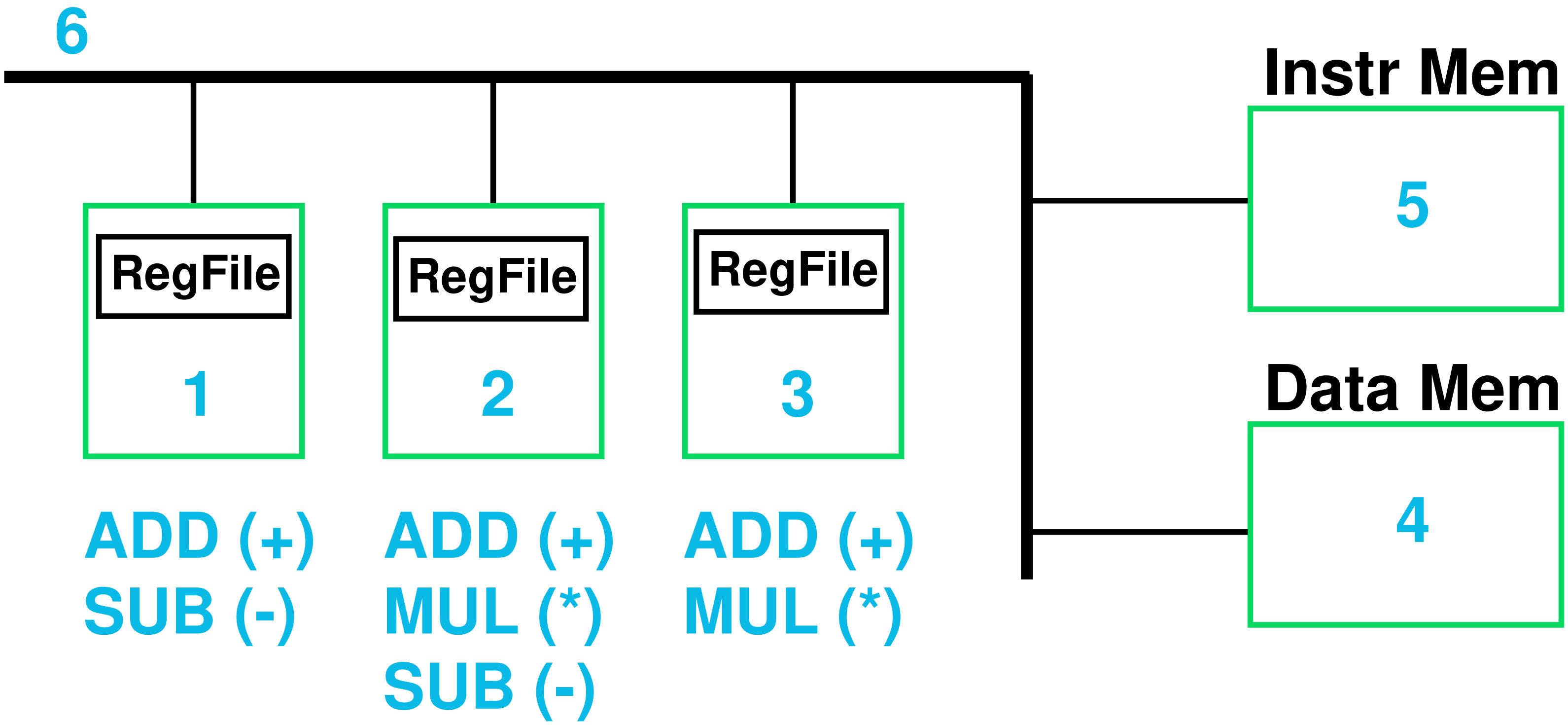
TWIF

REGISTER ALLOCATION STORAGE ASSIGNMENT SCHEDULING

FINAL CODE GENERATION PEEPHOLE OPTIMIZATION

object

Simple VLIW Architecture



ISDL Description Fragment

Storage

Memory 4	SIZE
Memory 5	SIZE
RegFile 1	SIZE
RegFile 2	SIZE
RegFile 3	SIZE
Wire 6	SIZE

Field 1

1.ADD ...
1.SUB ...

Field 2

2.ADD ...
2.MUL ...
2.SUB ...

Field 3

3.ADD ...
3.MUL ...

Field 4

4.LD ...
4.ST ...

Field 5

5.LD ...
5.ST …

Field 6

6.MOVE ...

Required Compiler Optimizations

Machine independent optimizations

- Parallelizing transformations (lots of them!)
- Common subexpression elimination
- Strength reduction
- Code motion

Machine dependent optimizations

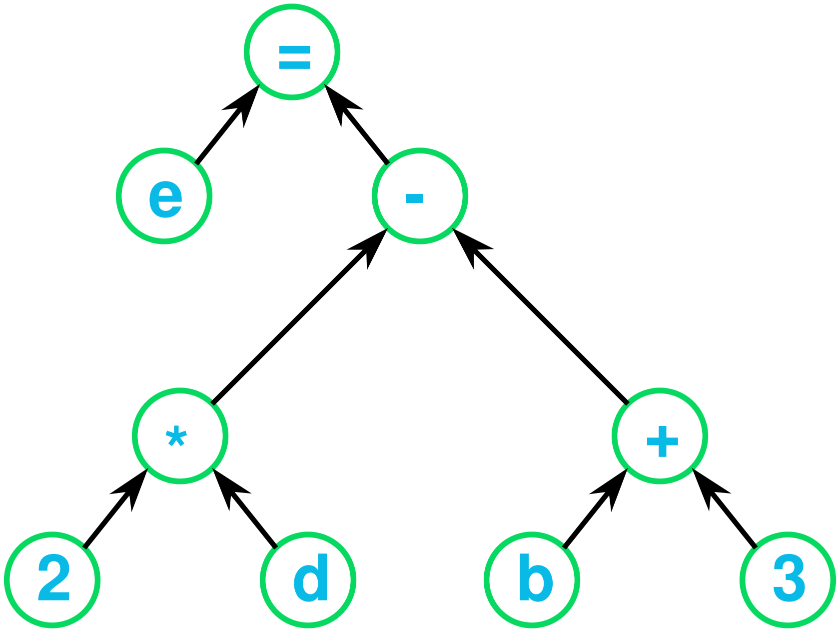
- Loop unrolling and software pipelining
- Static allocation (non-recursive procedure calls)
- Storage layout (arrays, scalars)
- Optimization of mode setting instructions

→ Instruction selection, scheduling, and register allocation

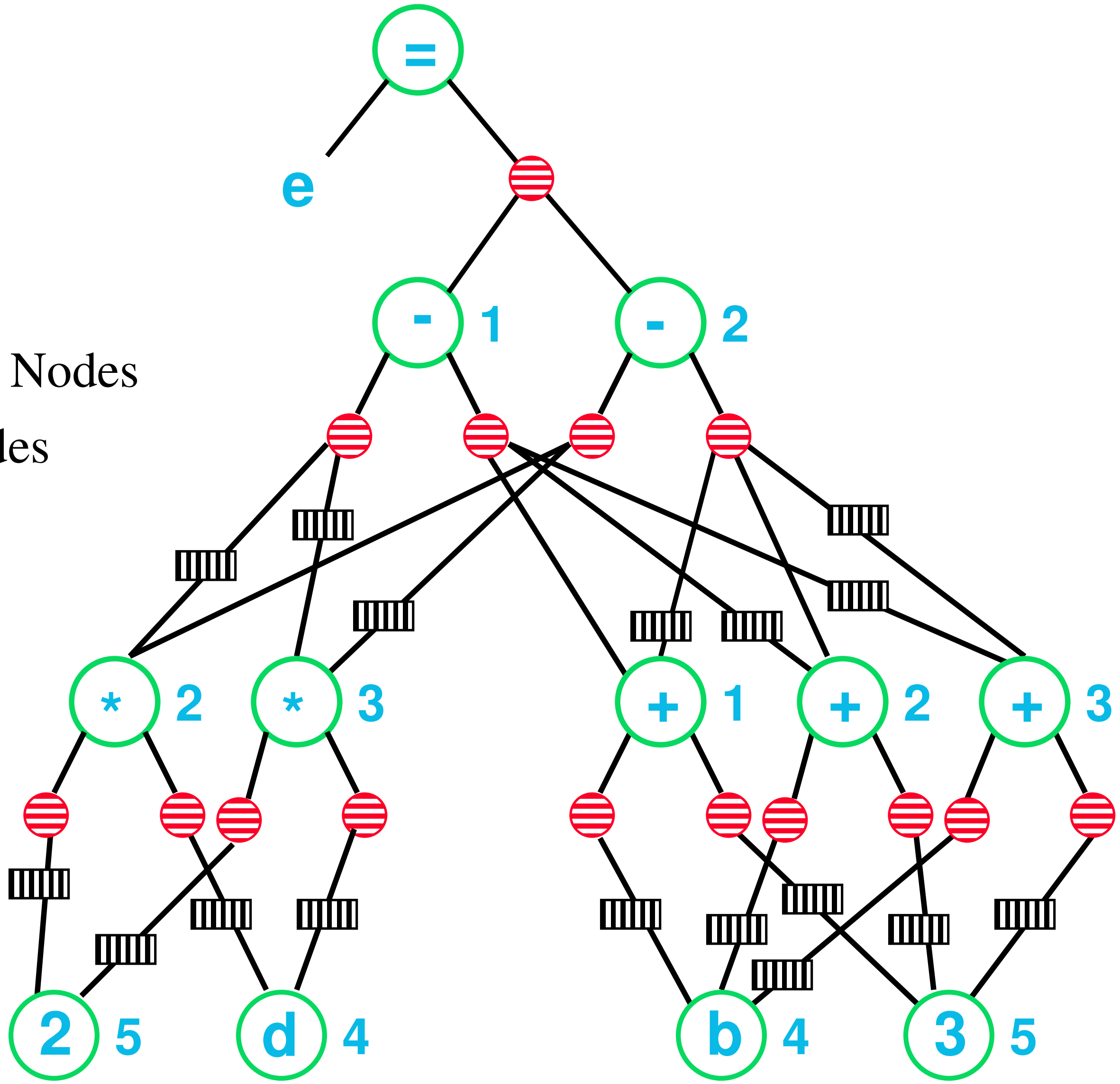
Basic Block DAG

Basic Block

```
a = b + 3
c = 2 * d
e = c - a
```



Split-Node DAG



▥ Transfer Nodes

⊜ Split-Nodes

Split-Node DAG - 2

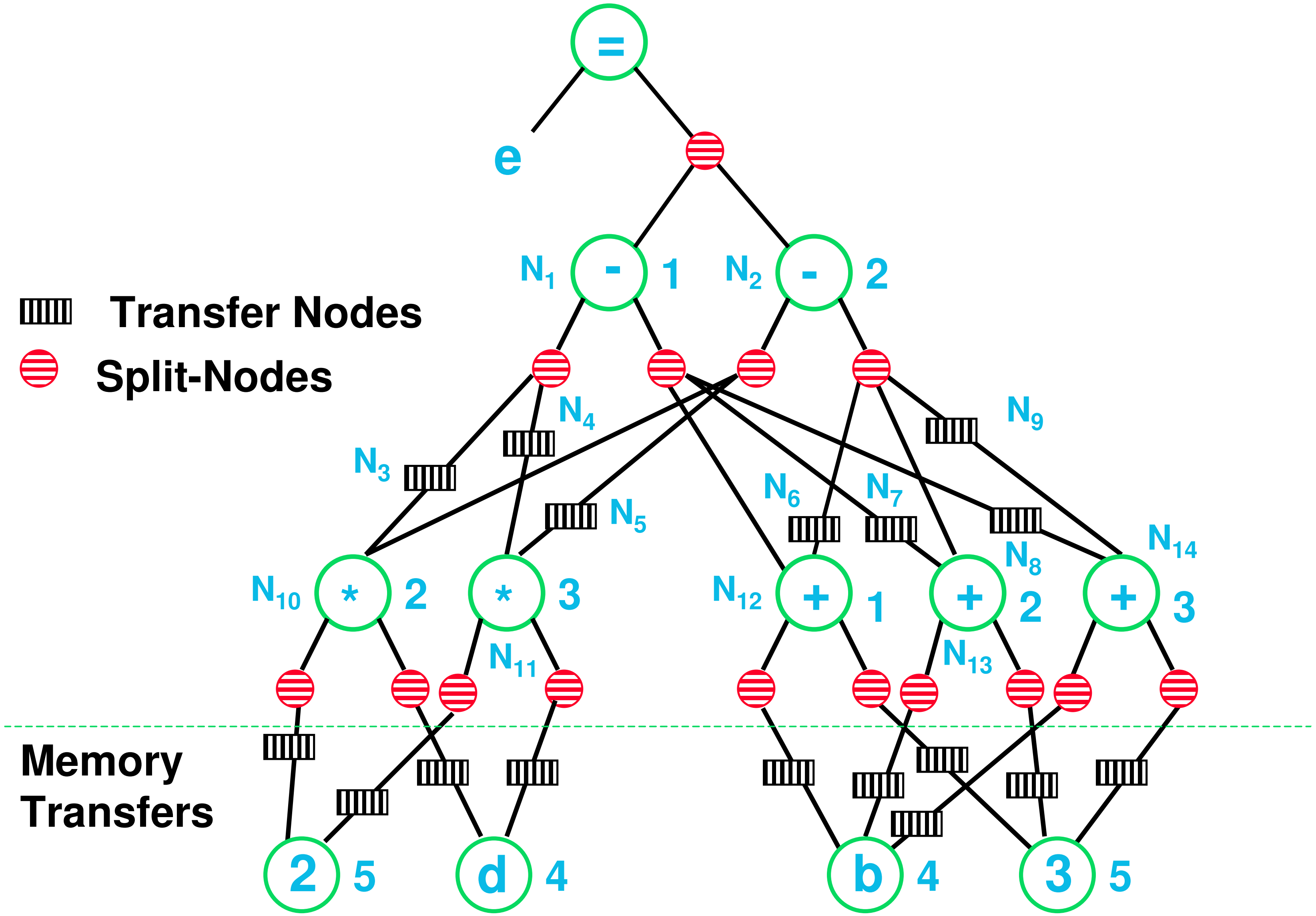
Split-Node DAG represents:

- All the legal assignments of basic block nodes to functional units
- The data transfers implied by pairs of assignments of basic block nodes connected by an edge

Goal is to find parallelism in the basic block, which can be exploited by the target architecture

- Implies grouping operator nodes and data transfer nodes into VLIW instructions
- Constraints may disallow certain groupings

Split-Node DAG - 3



▥ Transfer Nodes

● Split-Nodes

Memory Transfers

Parallelism Matrix

	N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	N14
N1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
N2	1	0	1	1	1	1	1	1	1	1	1	1	1	1
N3	1	1	0	1	1	1	1	1	1	1	1	0	0	0
N4	1	1	1	0	1	1	1	1	1	1	1	0	0	0
N5	1	1	1	1	0	1	1	1	1	1	1	0	0	0
N6	1	1	1	1	1	0	1	1	1	0	0	1	1	1
N7	1	1	1	1	1	1	0	1	1	0	0	1	1	1
N8	1	1	1	1	1	1	1	0	1	0	0	1	1	1
N9	1	1	1	1	1	1	1	1	0	0	0	1	1	1
N10	1	1	1	1	1	0	0	0	0	0	1	0	1	0
N11	1	1	1	1	1	0	0	0	0	1	0	0	0	1
N12	1	1	0	0	0	1	1	1	1	0	0	0	1	1
N13	1	1	0	0	0	1	1	1	1	1	0	1	0	1
N14	1	1	0	0	0	1	1	1	1	0	1	1	1	0

0 - can be done in parallel

1 - cannot be done in parallel

	N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	N14
N1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
N2	1	0	1	1	1	1	1	1	1	1	1	1	1	1
N3	1	1	0	1	1	1	1	1	1	1	1	0	0	0
N4	1	1	1	0	1	1	1	1	1	1	1	0	0	0
N5	1	1	1	1	0	1	1	1	1	1	1	0	0	0
N6	1	1	1	1	1	0	1	1	1	0	0	1	1	1
N7	1	1	1	1	1	1	0	1	1	0	0	1	1	1
N8	1	1	1	1	1	1	1	0	1	0	0	1	1	1
N9														