Speculative High-Level Synthesis of Instruction Set Processors

Jean-Michel Gorius
Univ Rennes

Steven Derrien, Simon Rokicki
CAIRN Team, Univ Rennes, IRISA, Inria

COLQ – M2 SIF Colloquium
February 1st, 2021

Introduction

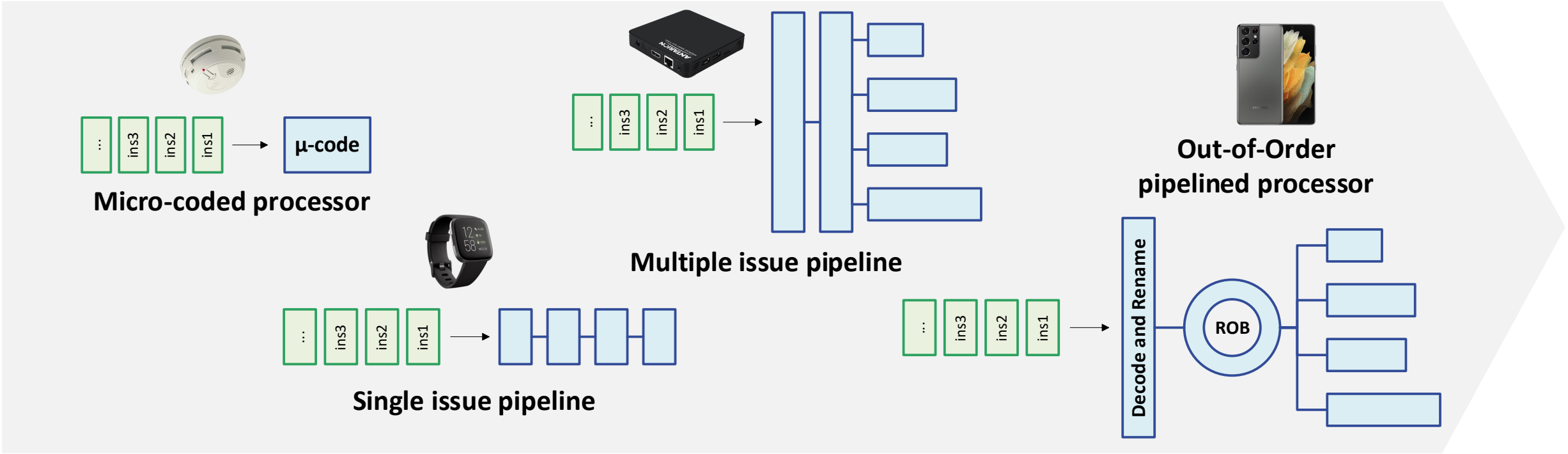
Low energy

High performance

A processor with a given Instruction Set (ISA) can be implemented in many ways

[Hennessy and Patterson, 2017] Hennessy, J. L. and Patterson, D. A. (2017). *Computer Architecture, Sixth Edition: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 6th edition.

Processor Design Landscape



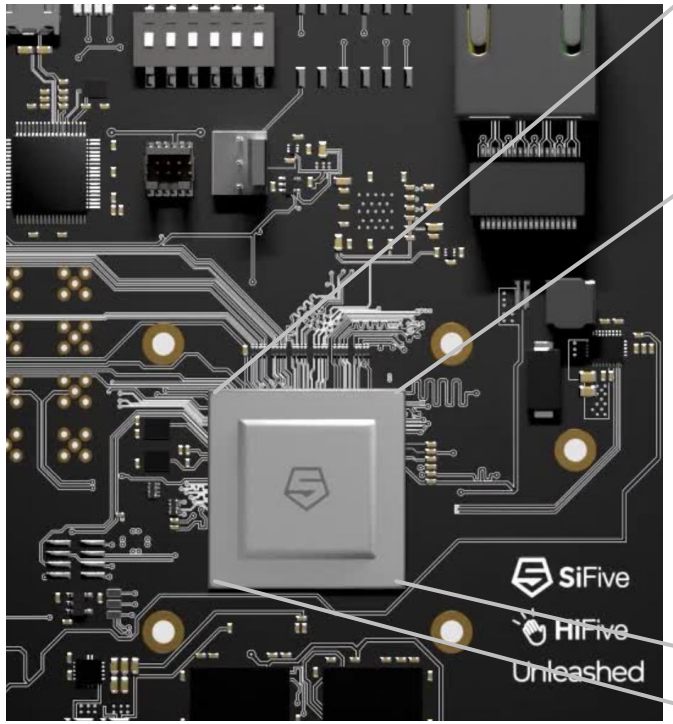
Low energy

High performance

Key Idea

Customizable architecture for heterogeneous embedded applications

Customizing Instruction Set Processors




```
module pipeline (
  input wire clk,
  input wire reset,
  output reg [`ADDR_LEN-1:0] pc,
  input wire [8*`INSN_LEN-1:0] idata,
  output wire [`DATA_LEN-1:0] dmem_wdata,
  output wire dmem_we,
  output wire [`ADDR_LEN-1:0] dmem_addr,
  input wire [`DATA_LEN-1:0] dmem_data
  );
  wire stall_IF;
  wire kill_IF;
  wire stall_ID;
  wire kill_ID;
  wire stall_DP;
  wire kill_DP
  //IF
  // Signal from pipe_if
  wire prcond;
  wire [`ADDR_LEN-1:0] npc;
  wire [`INSN_LEN-1:0] inst1;
  wire [`INSN_LEN-1:0] inst2;
  wire invalid2_pipe;
  wire [`GSH_BHR_LEN-1:0] bhr;
  //Instruction Buffer
  reg prcond_if;
  reg [`ADDR_LEN-1:0] npc_if;
  reg [`ADDR_LEN-1:0] pc_if;
  reg [`INSN_LEN-1:0] inst1_if;
  reg [`INSN_LEN-1:0] inst2_if;
  reg inv1_if;
  reg inv2_if;
  reg bhr_if;
  wire attachable;
  //ID
  //Decode Info1
  wire [`IMM_TYPE_WIDTH-1:0] imm_type_1;
  wire [`REG_SEL-1:0] rs1_1;
  wire [`REG_SEL-1:0] rs2_1;
  wire [`REG_SEL-1:0] rd_1;
  wire [`SRC_A_SEL_WIDTH-1:0] src_a_sel_1;
  wire [`SRC_B_SEL_WIDTH-1:0] src_b_sel_1;
  wire wr_reg_1;
  wire uses_rs1_1;
  wire uses_rs2_1;
  wire illegal_instruction_1;
  wire [`ALU_OP_WIDTH-1:0] alu_op_1;
  wire [`RS_ENT_SEL-1:0] rs_ent_1;
  wire [2:0] dmem_size_1;
  wire [`MEM_TYPE_WIDTH-1:0] dmem_type_1;
  wire [`MD_OP_WIDTH-1:0] md_req_op_1;
  wire md_req_in_1_signed_1;
  wire md_req_in_2_signed_1;
  wire [`MD_OUT_SEL_WIDTH-1:0]
md_req_out_sel_1;
  //Decode Info2
  wire [`IMM_TYPE_WIDTH-1:0] imm_type_2;
  wire [`REG_SEL-1:0] rs1_2;
  wire [`REG_SEL-1:0] rs2_2;
  wire [`REG_SEL-1:0] rd_2;
  wire [`SRC_A_SEL_WIDTH-1:0] src_a_sel_2;
  wire [`SRC_B_SEL_WIDTH-1:0] src_b_sel_2;
  wire wr_reg_2;
  wire uses_rs1_2;
  wire uses_rs2_2;
  wire illegal_instruction_2;
  wire [`ALU_OP_WIDTH-1:0] alu_op_2;
  wire [`RS_ENT_SEL-1:0] rs_ent_2;
  wire [2:0] dmem_size_2;
  wire [`MEM_TYPE_WIDTH-1:0] dmem_type_2;
  wire [`MD_OP_WIDTH-1:0] md_req_op_2;
  wire md_req_in_1_signed_2;
  wire md_req_in_2_signed_2;
  wire [`MD_OUT_SEL_WIDTH-1:0]
md_req_out_sel_2;
...
```

Customizing the compiler

- Retargetable compilation
- Well studied problem

Customizing hardware

Early work for Application-Specific Instruction Set Processor (ASIP) design
[Cloutier and Thomas, 1993; Huang and Despain, 1993]

[Cloutier and Thomas, 1993] Cloutier, R. J. and Thomas, D. E. (1993). Synthesis of pipelined instruction set processors. In *Proceedings of the 30th International Design Automation Conference*, DAC '93, page 583–588, New York, NY, USA. Association for Computing Machinery

[Huang and Despain, 1993] Huang, I.-J. and Despain, A. M. (1993). Hardware/software resolution of pipeline hazards in pipeline synthesis of instruction set processors. In *Proceedings of the 1993 IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '93, page 594–599, Washington, DC, USA. IEEE Computer Society Press

[Patterson and Hennessy, 2017] Patterson, D. A. and Hennessy, J. L. (2017). *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition.

High-Level Synthesis

Synthesizing circuits from an algorithmic specification

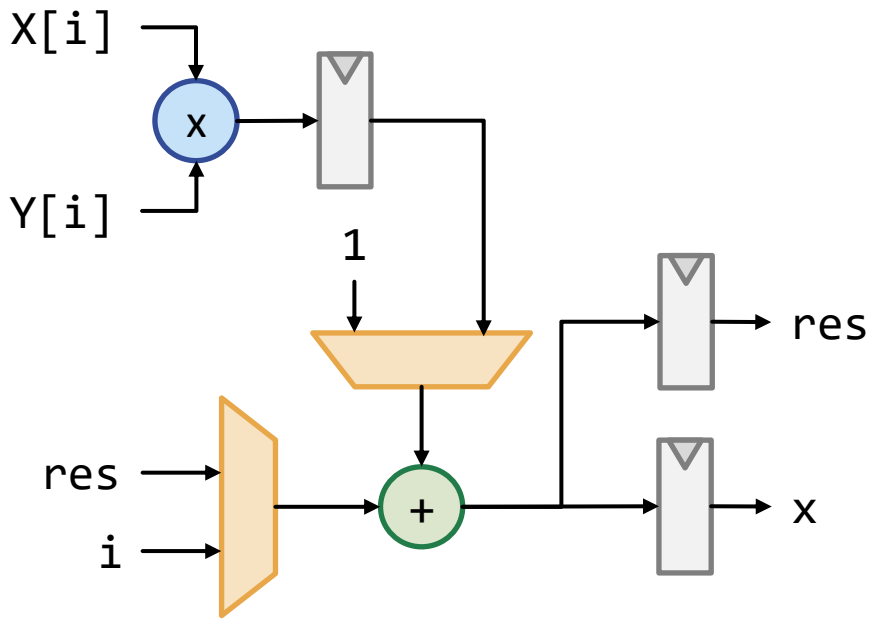
Efficient design synthesis for regular access patterns and computations

```
int X[N], Y[N];
int tmp, res = 0;
#pragma HLS mult=1, adder=1
for(int i = 0; i < N; ++i) {
    tmp = X[i] * Y[i];
    res += tmp;
}
```

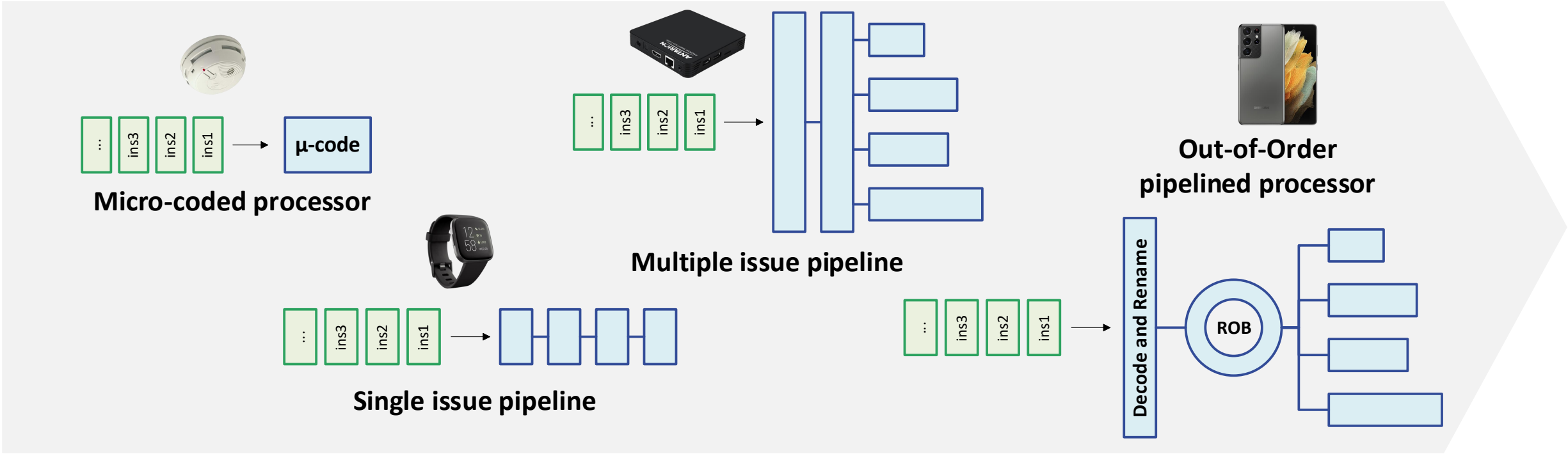
XILINX

HLS

Mentor
A Siemens Business



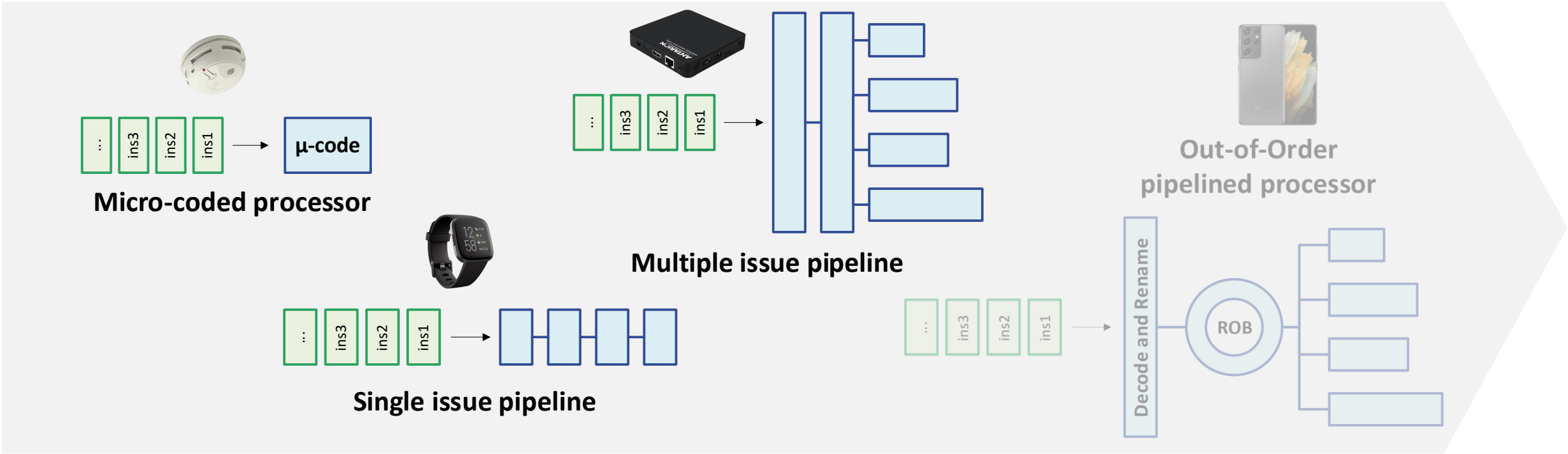
Objective



Research Question

How to make **pipelined** processor design amenable to HLS design flows?

Objective



Research Question

How to make **pipelined** processor design amenable to HLS design flows?

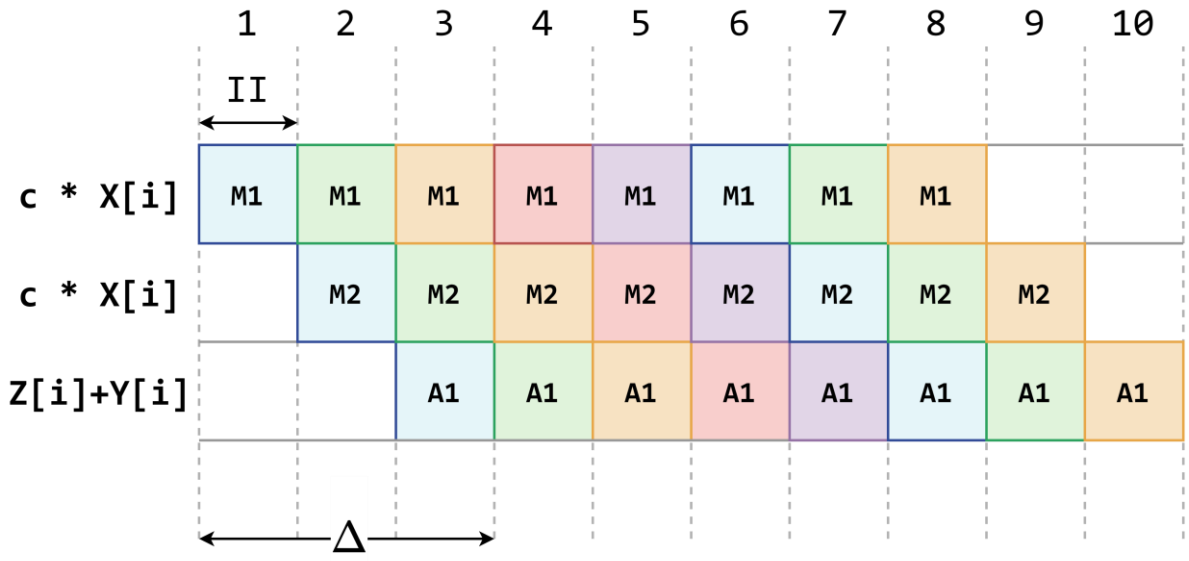
Loop Pipelining and Static Scheduling

Loop Pipelining

- Goal: Expose Instruction-Level Parallelism (ILP)
- Similar to Modulo Scheduling [Lam, 1988; Rau, 1994]

```
int X[N], Y[N];
int Z[N];
int c;
for(int i = 0; i < N; ++i) {
    Z[i] = c * X[i];
    Z[i] += Y[i];
}
```

Z = c*X + Y



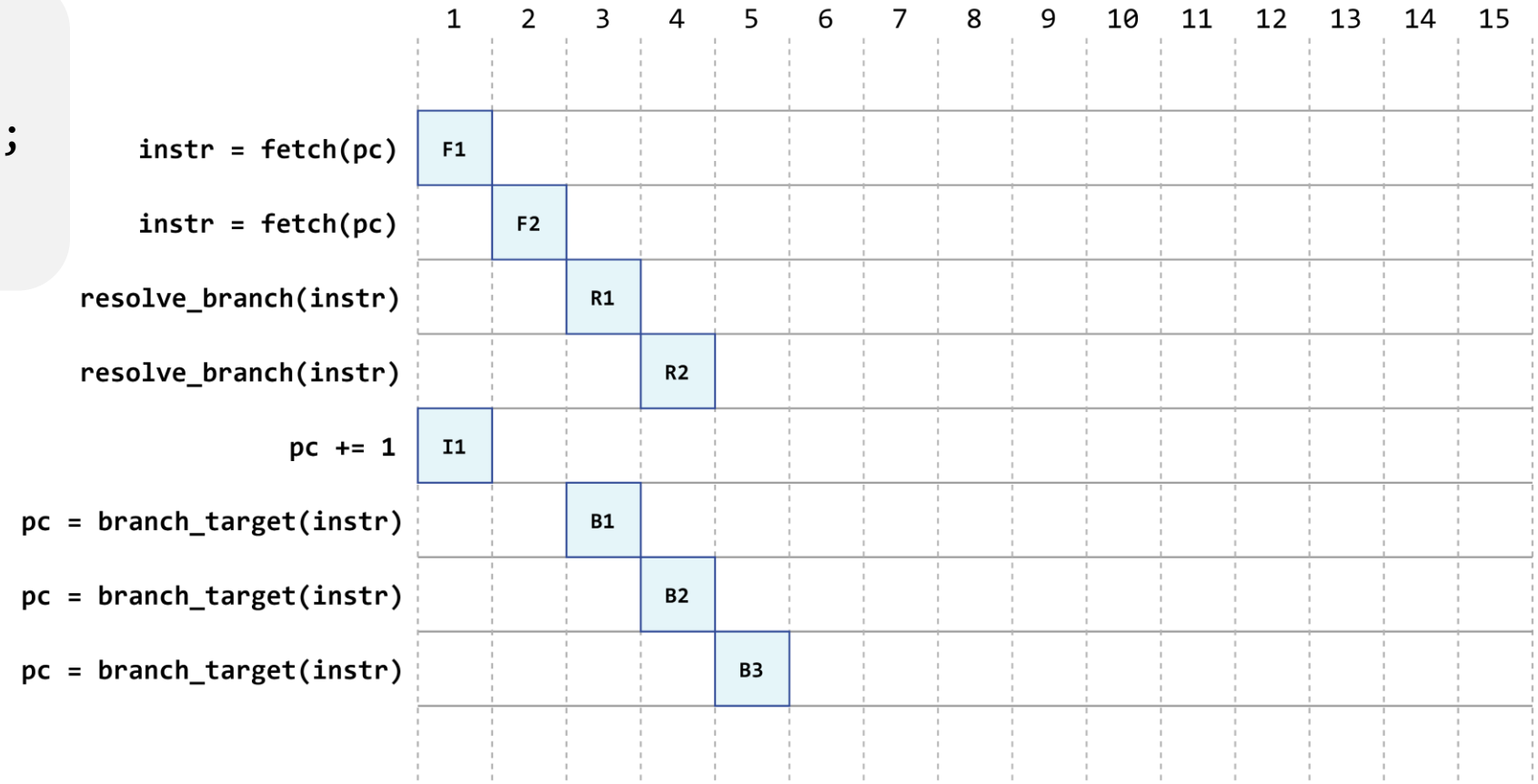
[Rau, 1994] Rau, B. R. (1994). Iterative modulo scheduling: An algorithm for software pipelining loops. In *Proceedings of the 27th Annual International Symposium on Microarchitecture*, MICRO 27, page 63–74, New York, NY, USA. Association for Computing Machinery

[Lam, 1988] Lam, M. (1988). Software pipelining: An effective scheduling technique for VLIW machines. In *Proceedings of the ACM SIGPLAN 1988 Conference on Programming Language Design and Implementation*, PLDI '88, page 318–328, New York, NY, USA. Association for Computing Machinery

Loop Pipelining and Static Scheduling

```
instr = fetch(pc);
if(resolve_branch(instr))
  pc = branch_target(instr);
else
  pc += 1;
```

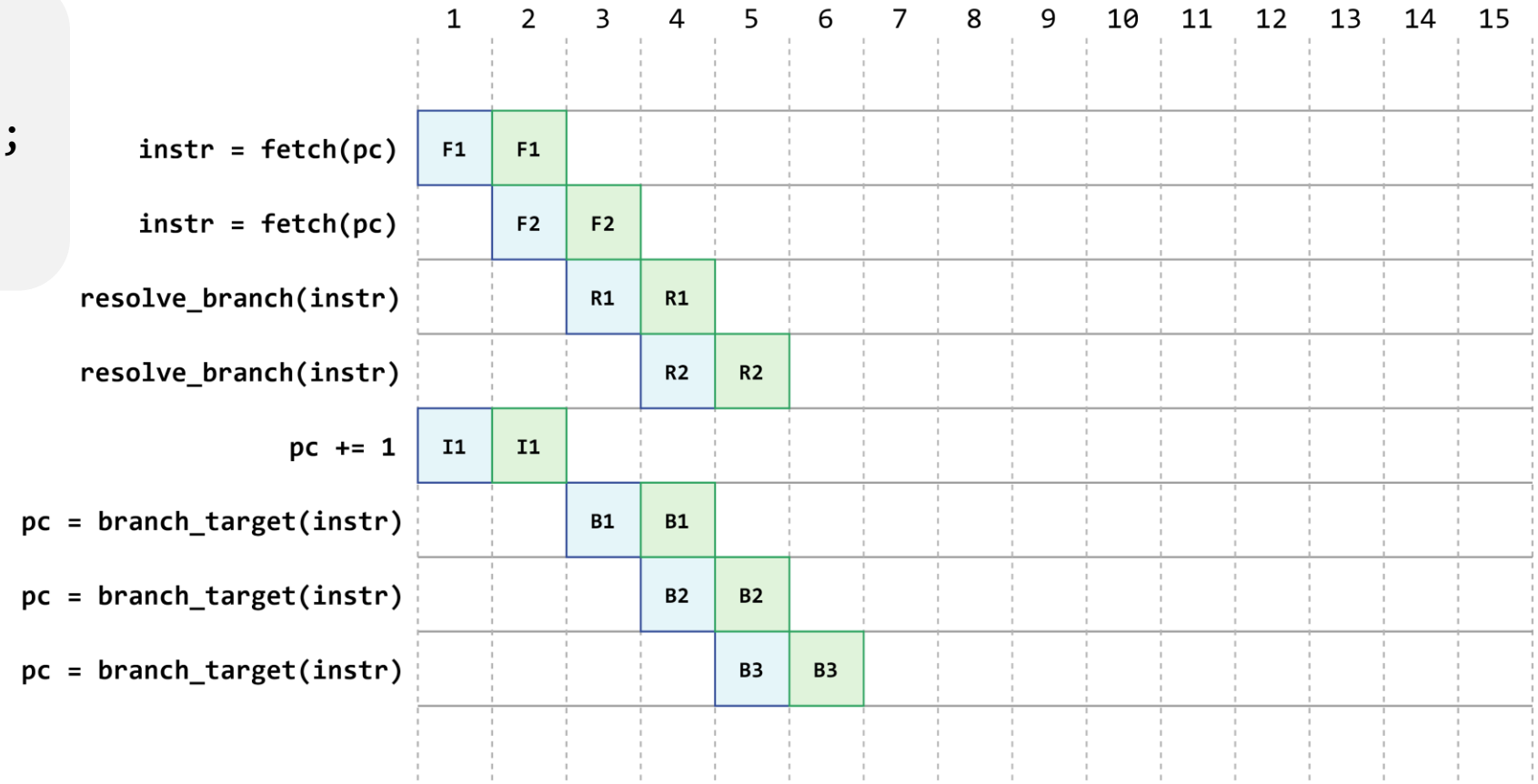
Simplified ISS



Loop Pipelining and Static Scheduling

```
instr = fetch(pc);
if(resolve_branch(instr))
  pc = branch_target(instr);
else
  pc += 1;
```

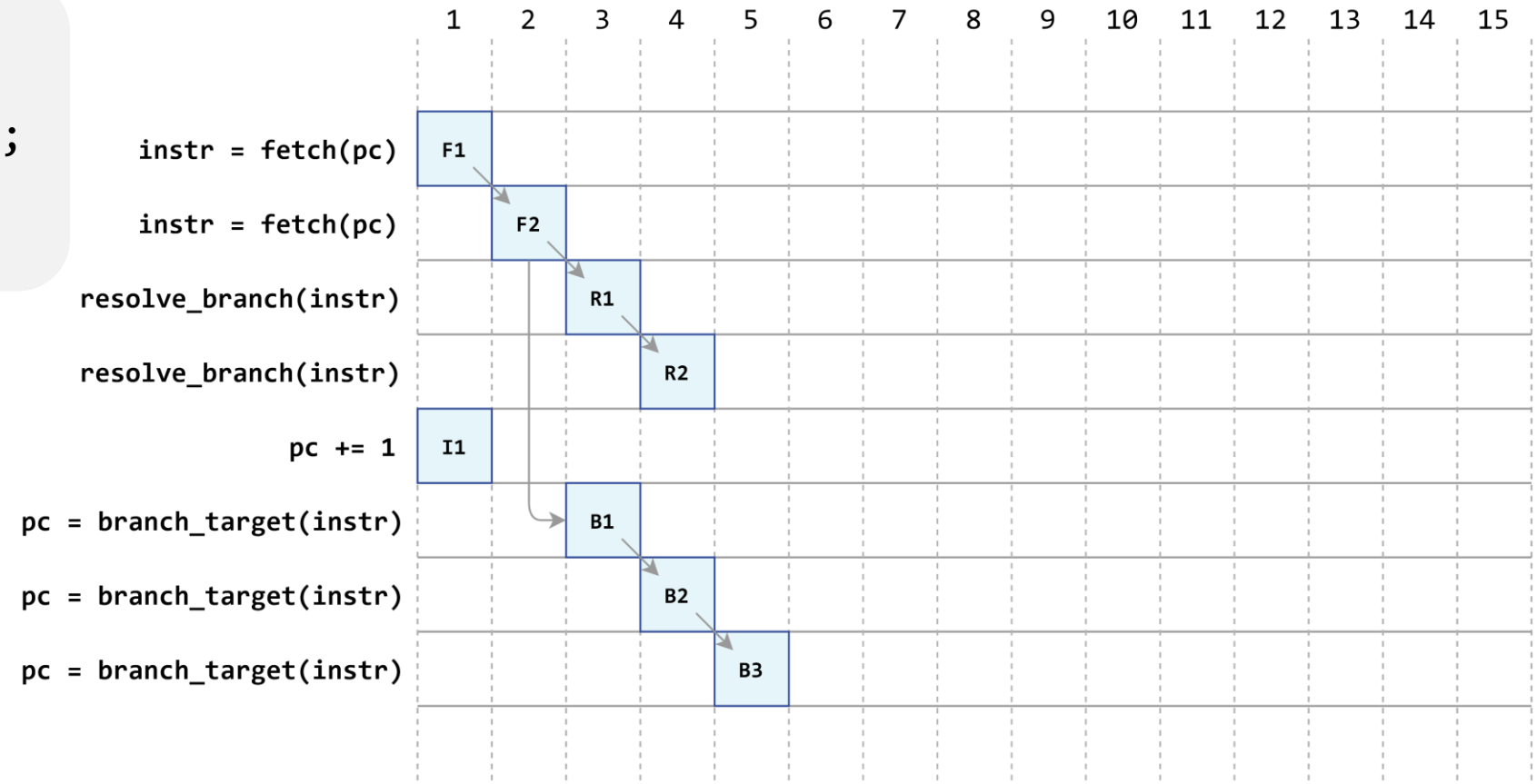
Simplified ISS



Loop Pipelining and Static Scheduling

```
instr = fetch(pc);
if(resolve_branch(instr))
  pc = branch_target(instr);
else
  pc += 1;
```

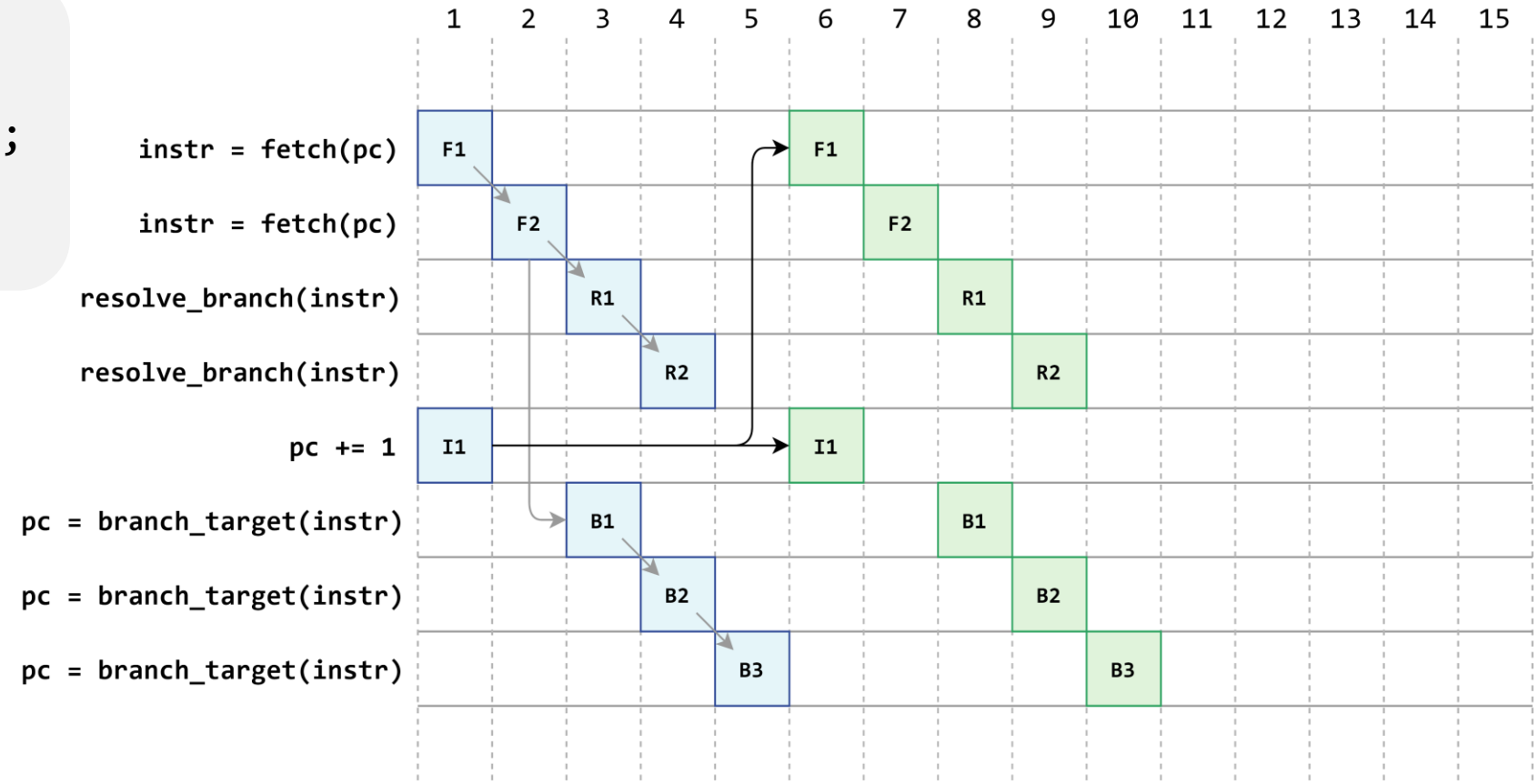
Simplified ISS



Loop Pipelining and Static Scheduling

```
instr = fetch(pc);
if(resolve_branch(instr))
  pc = branch_target(instr);
else
  pc += 1;
```

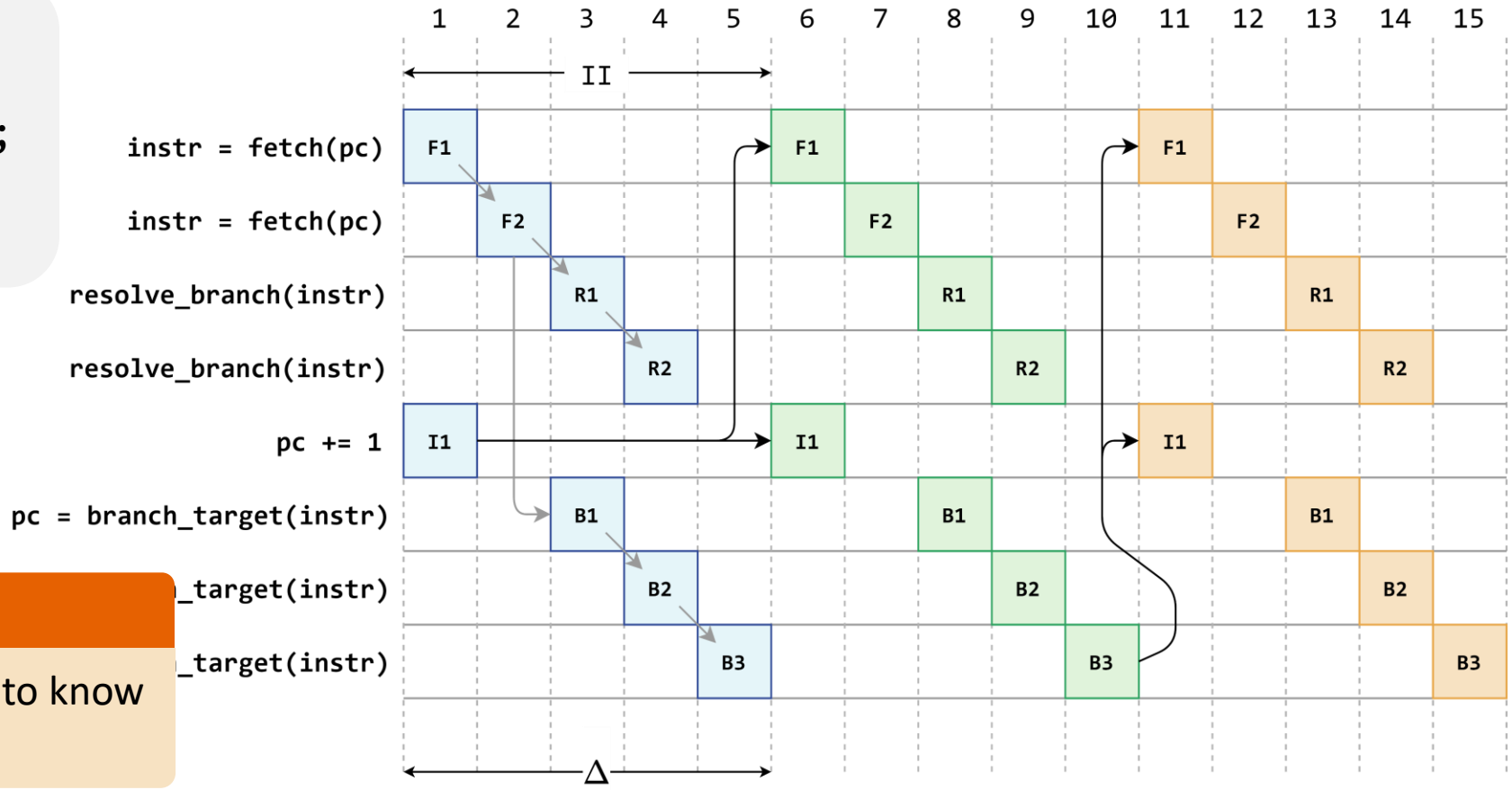
Simplified ISS



Loop Pipelining and Static Scheduling

```
instr = fetch(pc);
if(resolve_branch(instr))
  pc = branch_target(instr);
else
  pc += 1;
```

Simplified ISS



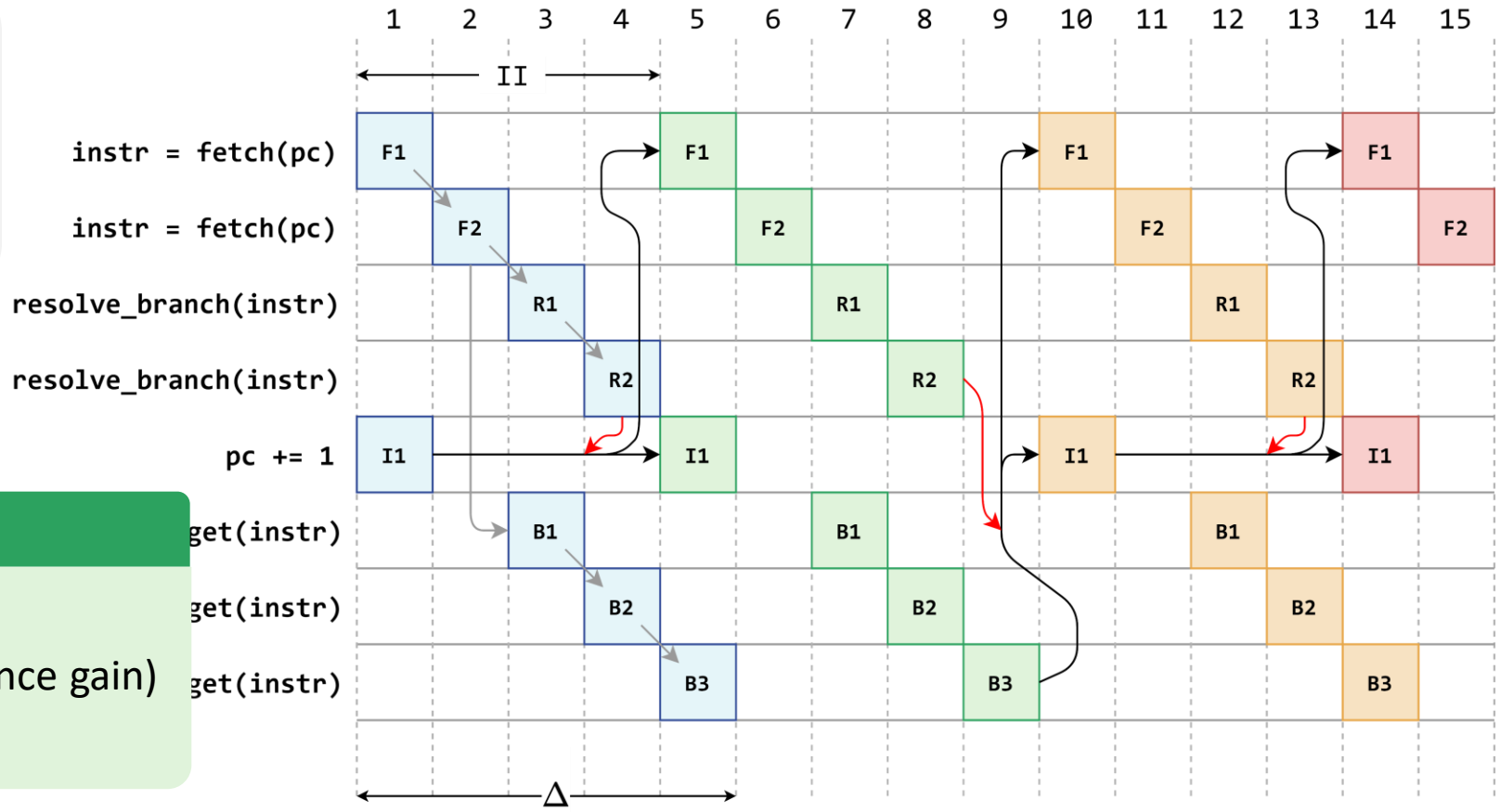
Inefficient design

Need to complete an entire iteration to know the next value of PC

Dynamic Scheduling

```
instr = fetch(pc);
if(resolve_branch(instr))
  pc = branch_target(instr);
else
  pc += 1;
```

Simplified ISS



Dynamically Scheduled HLS

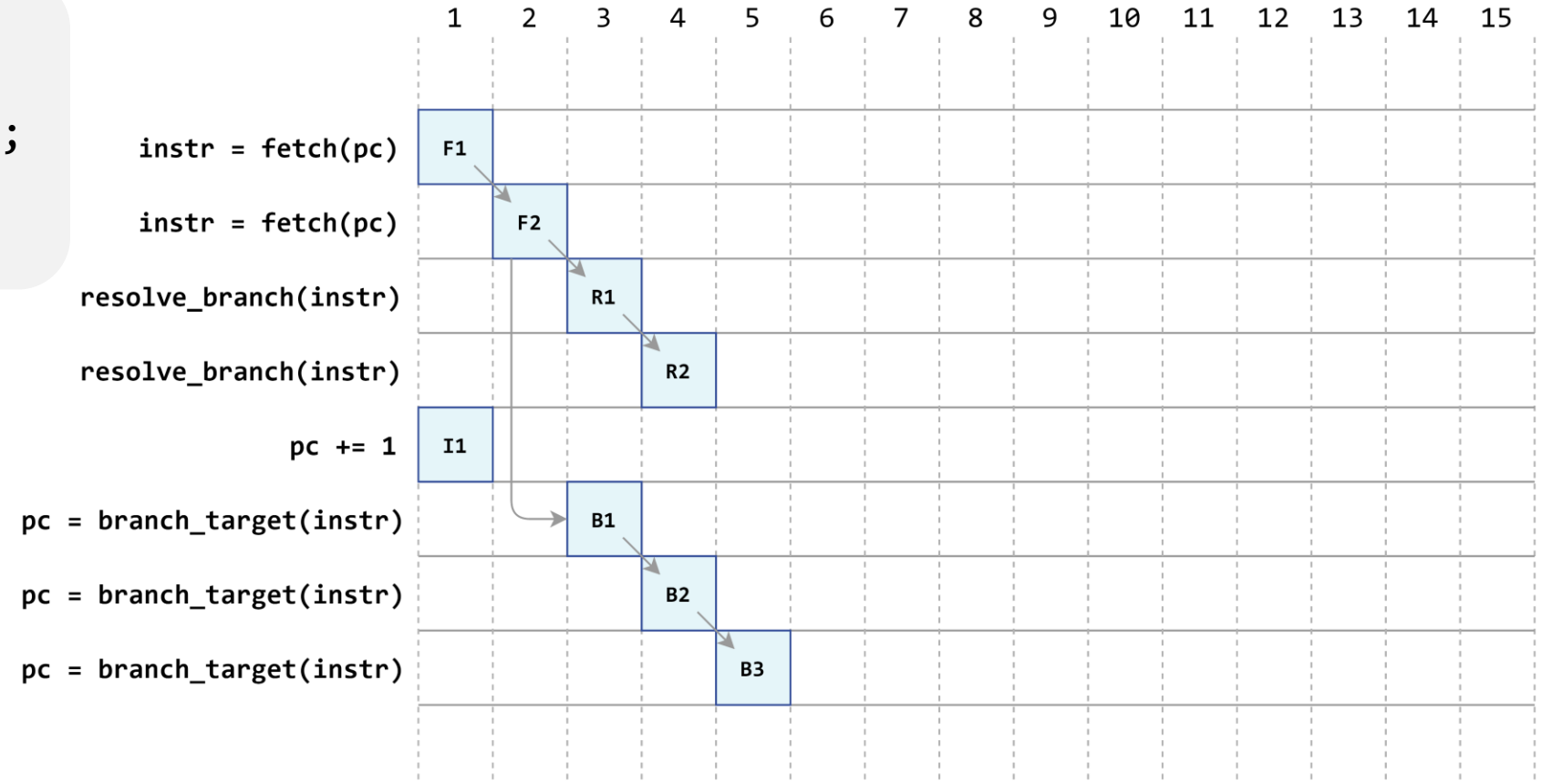
- Exposes some ILP
- Reduces II from 5 to 4 (20% performance gain)
- Still far away from II = 1

[Josipovic et al., 2018] Josipovic, L., Ghosal, R., and Ienne, P. (2018). Dynamically scheduled high-level synthesis. In *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '18, page 127–136, New York, NY, USA. Association for Computing Machinery

Speculative High-Level Synthesis

```
instr = fetch(pc);
if(resolve_branch(instr))
  pc = branch_target(instr);
else
  pc += 1;
```

Simplified ISS



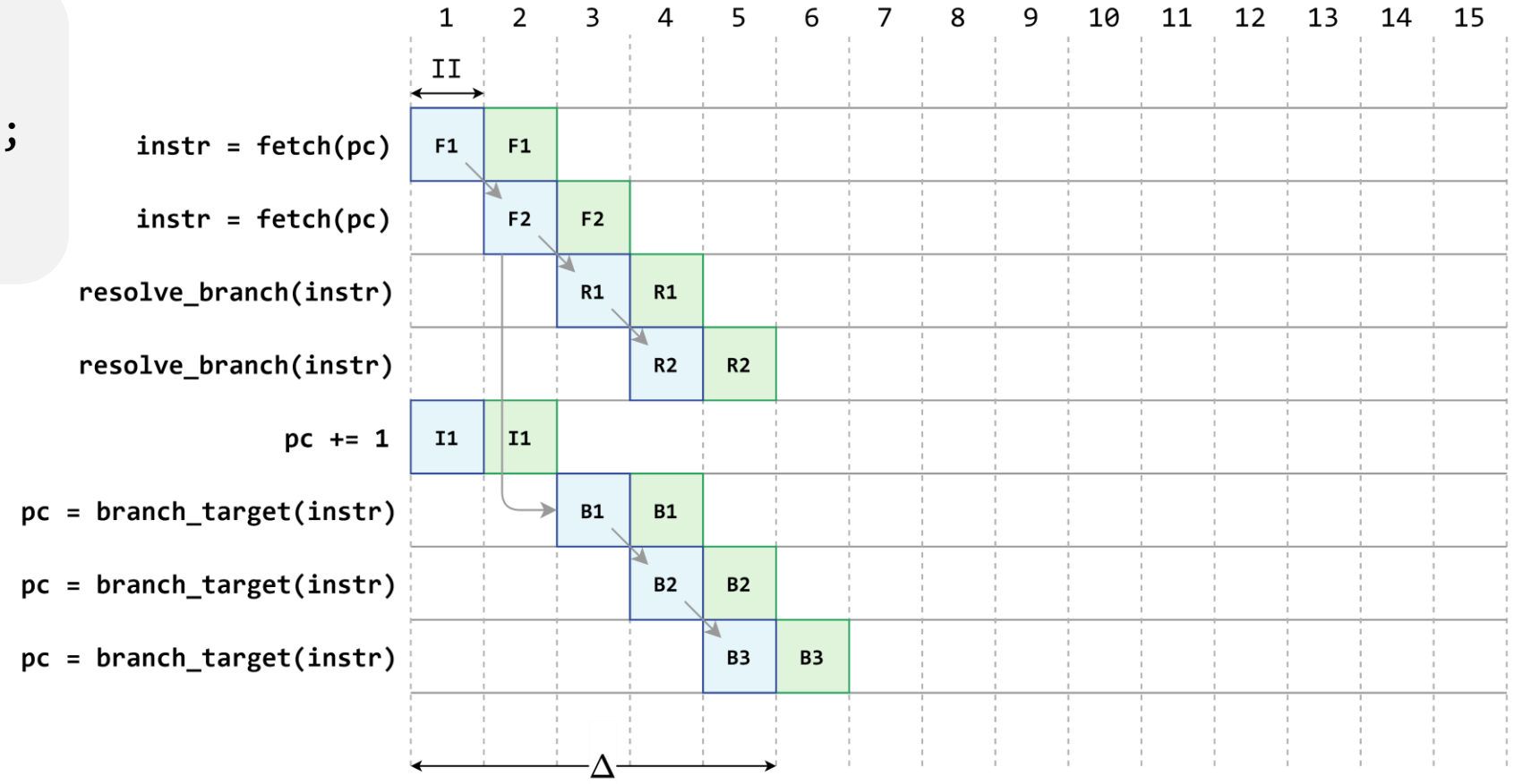
[Derrien et al., 2020] Derrien, S., Marty, T., Rokicki, S., and Yuki, T. (2020). Toward speculative loop pipelining for high-level synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):4229–4239.

[Josipovic et al., 2019] Josipovic, L., Guerrieri, A., and Ienne, P. (2019). Speculative dataflow circuits. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '19, page 162–171, New York, NY, USA. Association for Computing Machinery

Speculative High-Level Synthesis

```
instr = fetch(pc);
if(resolve_branch(instr))
  pc = branch_target(instr);
else
  pc += 1;
```

Simplified ISS



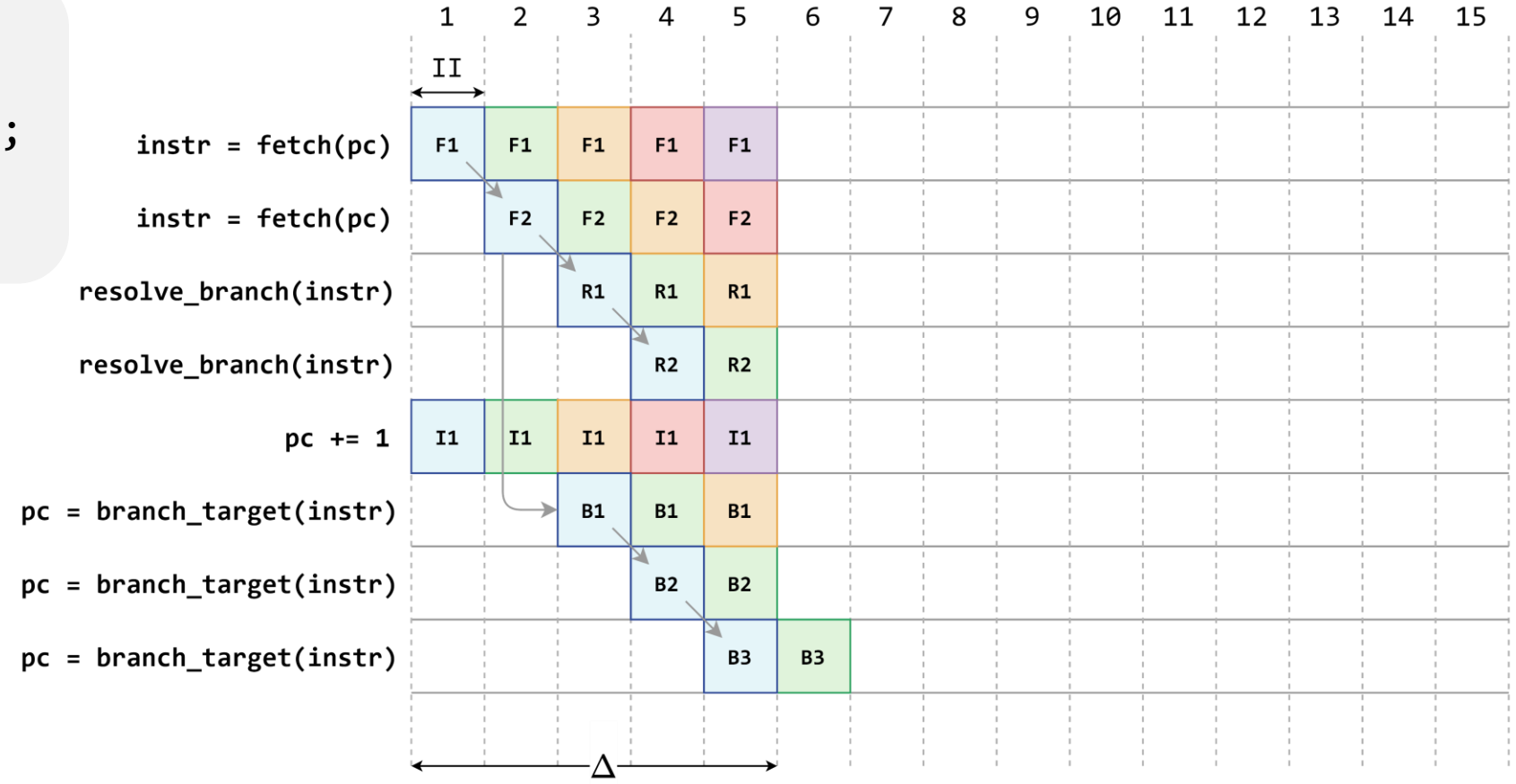
[Derrien et al., 2020] Derrien, S., Marty, T., Rokicki, S., and Yuki, T. (2020). Toward speculative loop pipelining for high-level synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):4229–4239.

[Josipovic et al., 2019] Josipovic, L., Guerrieri, A., and Ienne, P. (2019). Speculative dataflow circuits. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '19, page 162–171, New York, NY, USA. Association for Computing Machinery

Speculative High-Level Synthesis

```
instr = fetch(pc);
if(resolve_branch(instr))
  pc = branch_target(instr);
else
  pc += 1;
```

Simplified ISS



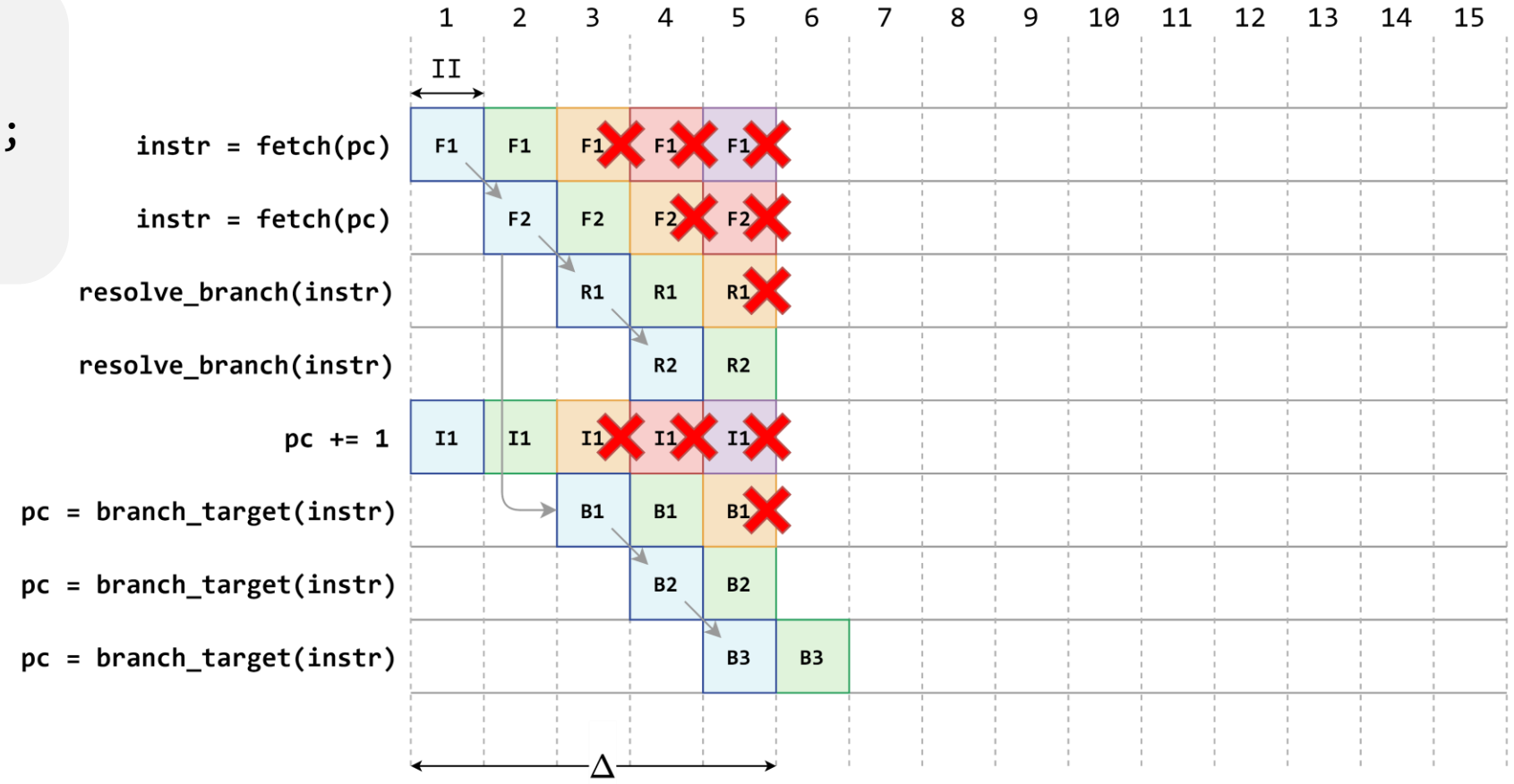
[Derrien et al., 2020] Derrien, S., Marty, T., Rokicki, S., and Yuki, T. (2020). Toward speculative loop pipelining for high-level synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):4229–4239.

[Josipovic et al., 2019] Josipovic, L., Guerrieri, A., and Ienne, P. (2019). Speculative dataflow circuits. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '19, page 162–171, New York, NY, USA. Association for Computing Machinery

Speculative High-Level Synthesis

```
instr = fetch(pc);
if(resolve_branch(instr))
  pc = branch_target(instr);
else
  pc += 1;
```

Simplified ISS



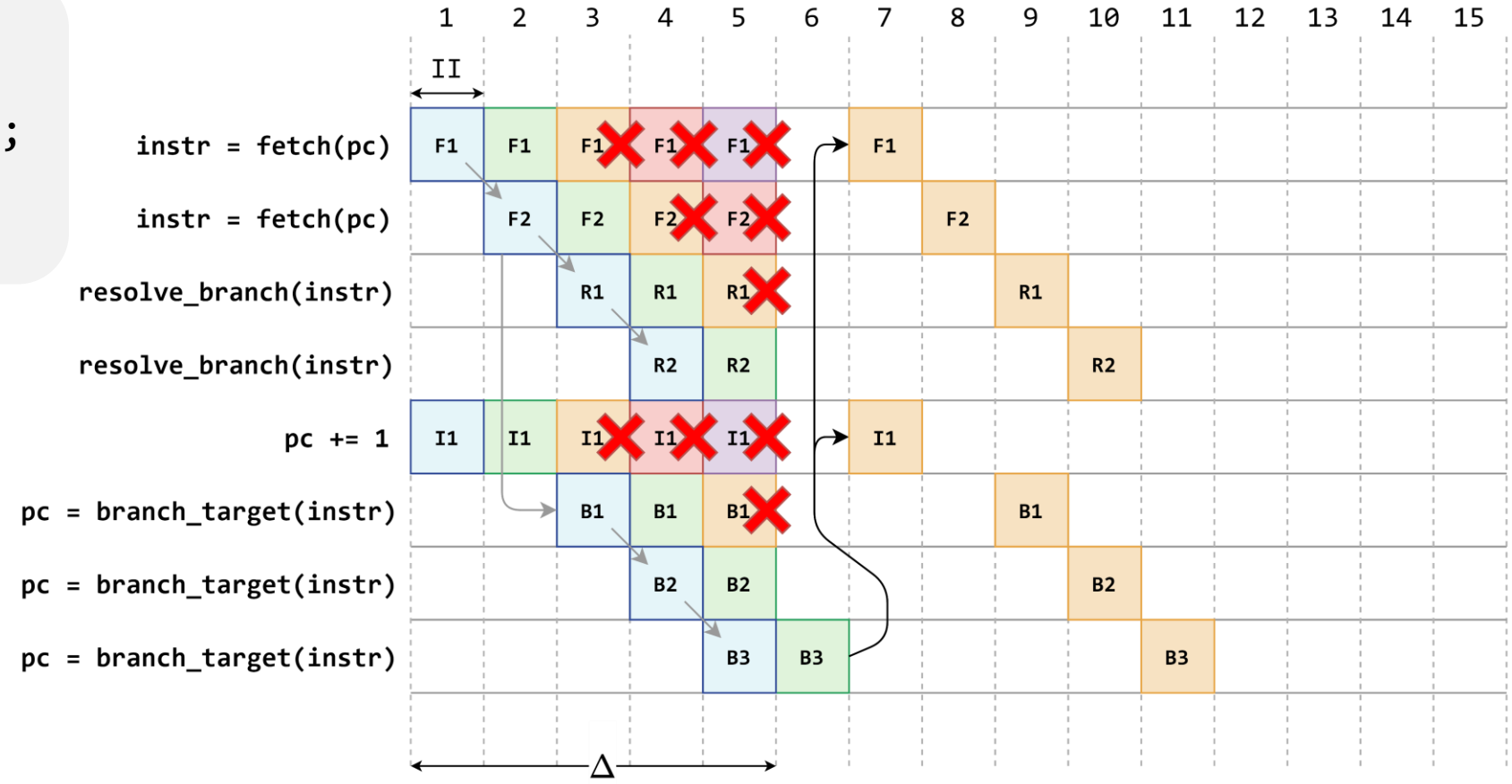
[Derrien et al., 2020] Derrien, S., Marty, T., Rokicki, S., and Yuki, T. (2020). Toward speculative loop pipelining for high-level synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):4229–4239.

[Josipovic et al., 2019] Josipovic, L., Guerrieri, A., and Ienne, P. (2019). Speculative dataflow circuits. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '19, page 162–171, New York, NY, USA. Association for Computing Machinery

Speculative High-Level Synthesis

```
instr = fetch(pc);
if(resolve_branch(instr))
  pc = branch_target(instr);
else
  pc += 1;
```

Simplified ISS



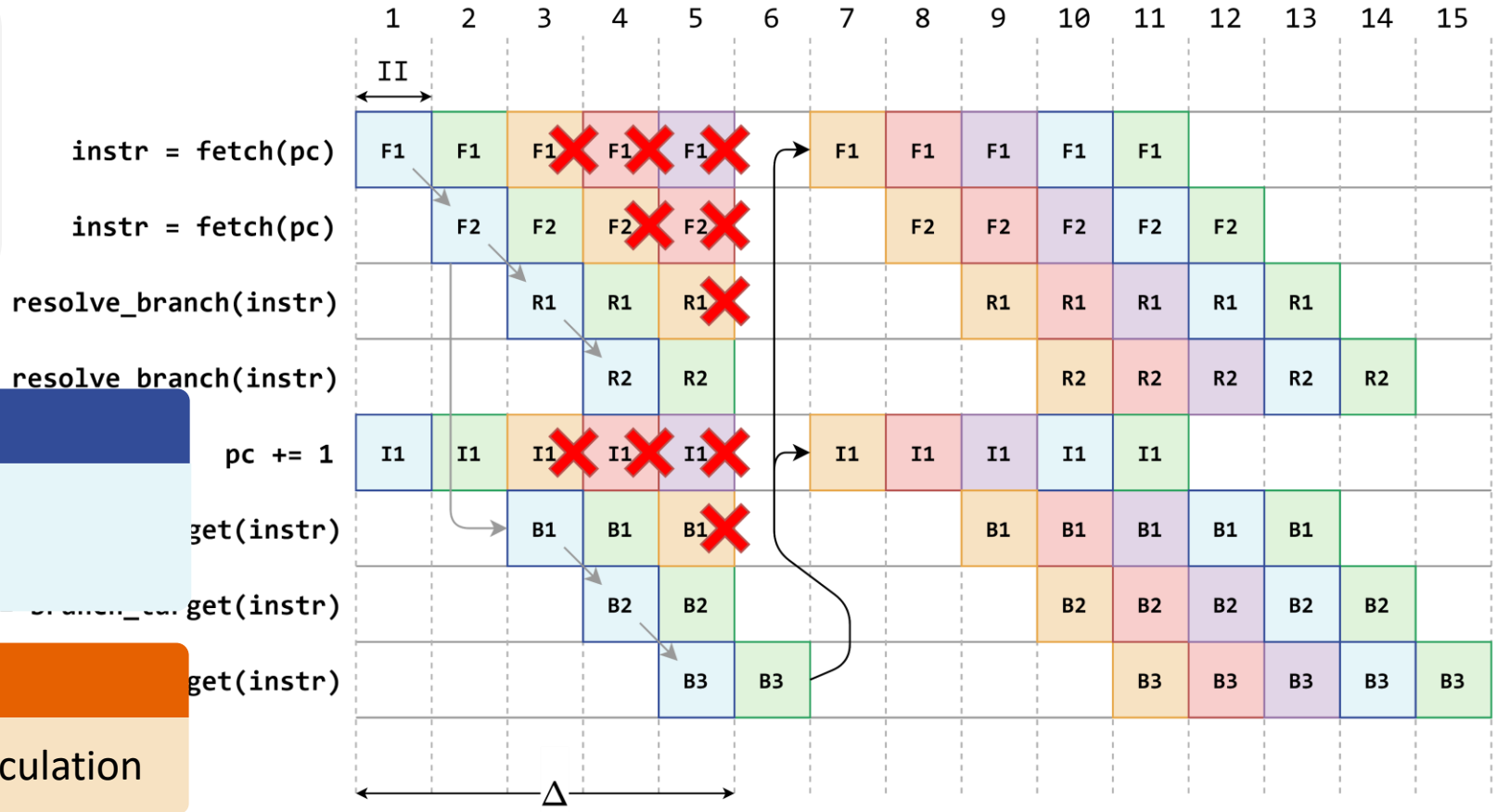
[Derrien et al., 2020] Derrien, S., Marty, T., Rokicki, S., and Yuki, T. (2020). Toward speculative loop pipelining for high-level synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):4229–4239.

[Josipovic et al., 2019] Josipovic, L., Guerrieri, A., and Ienne, P. (2019). Speculative dataflow circuits. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '19, page 162–171, New York, NY, USA. Association for Computing Machinery

Speculative High-Level Synthesis

```
instr = fetch(pc);
if(resolve_branch(instr))
  pc = branch_target(instr);
else
  pc += 1;
```

Simplified ISS



Speculative HLS

- Achieves II = 1 in most cases
- Maximal ILP and resource utilization

Takeaway

Pipelined processor design requires speculation

[Derrien et al., 2020] Derrien, S., Marty, T., Rokicki, S., and Yuki, T. (2020). Toward speculative loop pipelining for high-level synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):4229–4239.

[Josipovic et al., 2019] Josipovic, L., Guerrieri, A., and Ienne, P. (2019). Speculative dataflow circuits. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '19, page 162–171, New York, NY, USA. Association for Computing Machinery

Speculative Dataflow Circuits

Dataflow circuits

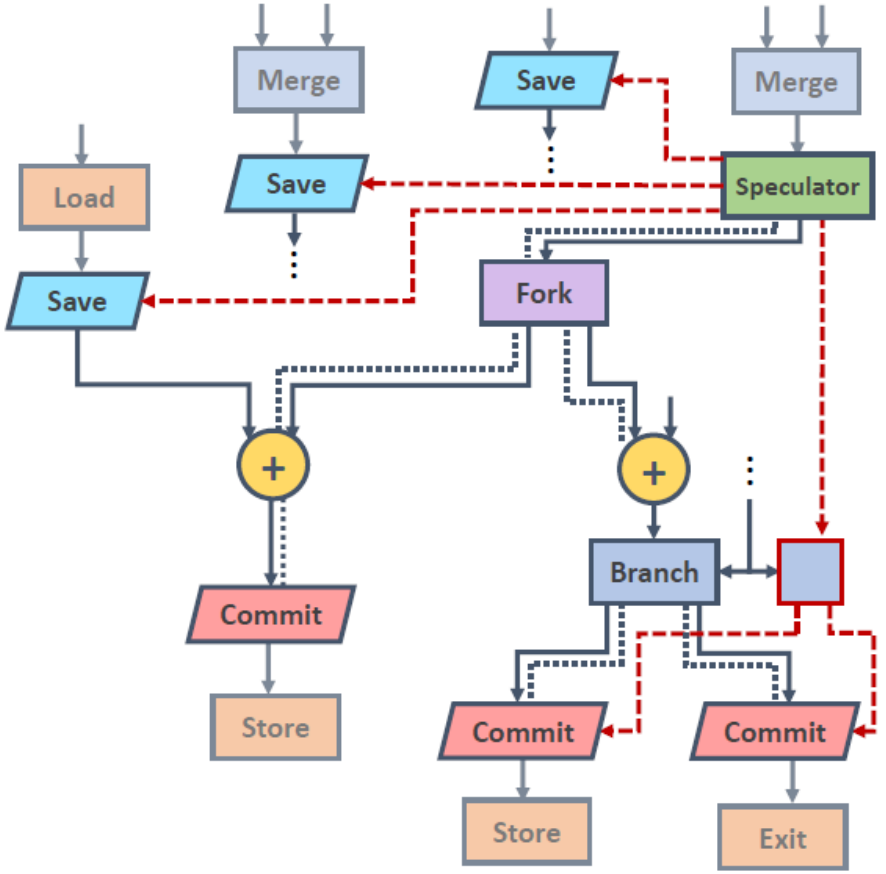
- Regular datapath components coupled to a handshaking protocol
- Compositional computational model

Speculation support

- Insertion of speculative components in the circuit
- *Speculator*, *Commit* and *Save* units: transfer of *speculative tokens*

Limitations

- Requires a complete backend re-design
- Limited support for intertwined speculations



[Josipovic et al., 2019] Josipovic, L., Guerrieri, A., and Ienne, P. (2019). Speculative dataflow circuits. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '19, page 162–171, New York, NY, USA. Association for Computing Machinery

[Nurvitadhi et al., 2011] Nurvitadhi, E., Hoe, J. C., Kam, T., and Lu, S.-L. L. (2011). Automatic pipelining from transactional datapath specifications. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(3):441–454.

Speculative Loop Pipelining

Source-to-source transformations

- Transform the input code to expose potential speculations

Speculation support

- Decouple control logic from datapath
- HLS toolchain infers the speculative structure

Limitations

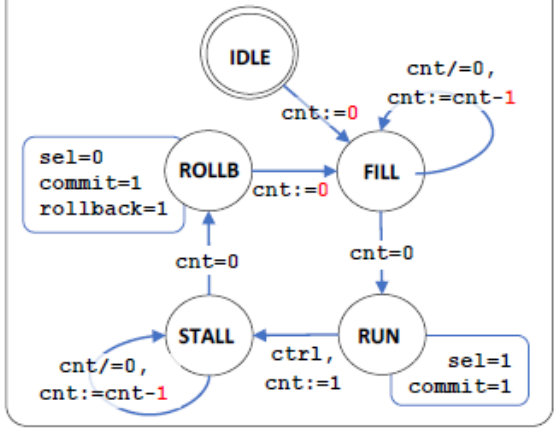
- Lack of support for intertwined speculations

```
do {
  tmp=x;
  if(C(x)) {
    // slow
    x = S(tmp);
  } else {
    // fast
    x = F(tmp,y);
  }
  y = H(tmp,y)
} while (!x)
```

Latency estimates	
C	2
S	5
F	1
H	1

```
2 #pragma hls distance mis_x=5
  do {
  #pragma hls pipeline II=1
1   ctrl[t]  = C (s_x[t-2]);
    mis_x[t] = S (s_x[t-5]);
    s_x[t] = F (s_x[t-1],s_y[t-1]);
    s_y[t] = H (s_x[t-1],s_y[t-1]);

    cs = nextState(cs,ctrl[t]);   3

4   if (cs.rollback) {
      s_y[t] = s_y[t-4];
      s_x[t] = mis_x[t]
    }
5   if (cs.commit) {
      x = cs.sel? s_x[t-1]:mis_x[t];
      y = s_y[t-1];
    }
    t++;
  } while(!(x && cs.commit));   6
```

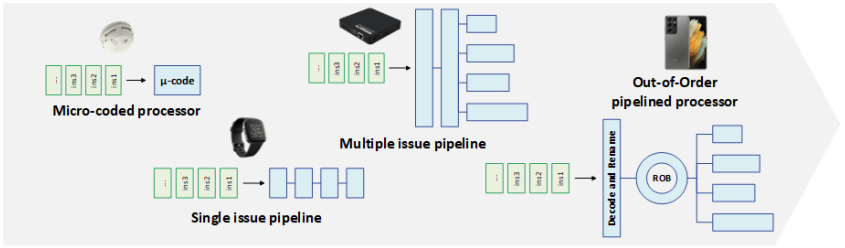
```
enum tstate = {IDLE, FILL, … }

struct fsm {
   int3 cnt;
   tstate cs;
   bool commit,rollback,sel;
} cs;
```



[Derrien et al., 2020] Derrien, S., Marty, T., Rokicki, S., and Yuki, T. (2020). Toward speculative loop pipelining for high-level synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):4229–4239.

Conclusion

Processor Design Landscape



Key Idea

Customizable architecture for heterogeneous embedded applications

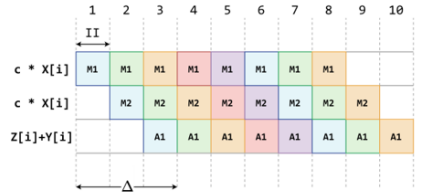
Loop Pipelining and Static Scheduling

Loop Pipelining

- Goal: Expose Instruction-Level Parallelism (ILP)
- Similar to Modulo Scheduling [Lam, 1988; Rau, 1994]

```
int X[N], Y[N];
int Z[N];
int c;
for(int i = 0; i < N; ++i) {
    Z[i] = c * X[i];
    Z[i] += Y[i];
}
```

Z = c*X + Y



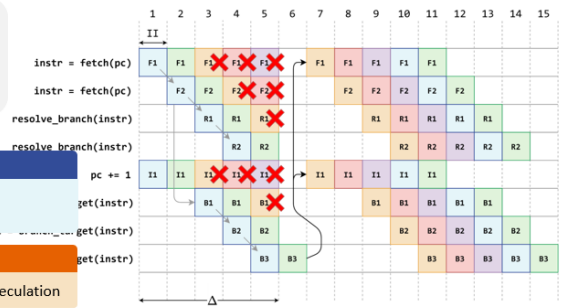
[Rau, 1994] Rau, B. R. (1994). Iterative modulo scheduling: An algorithm for software pipelining loops. In *Proceedings of the 27th Annual International Symposium on Microarchitecture*, MICRO 27, page 63–74, New York, NY, USA. Association for Computing Machinery

[Lam, 1988] Lam, M. (1988). Software pipelining: An effective scheduling technique for VLIW machines. In *Proceedings of the ACM SIGPLAN 1988 Conference on Programming Language Design and Implementation*, PLDI '88, page 318–328, New York, NY, USA. Association for Computing Machinery

Speculative High-Level Synthesis

```
instr = fetch(pc);
if(resolve_branch(instr))
  pc = branch_target(instr);
else
  pc += 1;
```

Simplified ISS



Speculative HLS

- Achieves II = 1 in most cases
- Maximal ILP and resource utilization

Takeaway

Pipelined processor design requires speculation

[Derrien et al., 2020] Derrien, S., Marty, T., Rokicki, S., and Yuki, T. (2020). Toward speculative loop pipelining for high-level synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):4229–4239.

[Josipovic et al., 2019] Josipovic, L., Guerrieri, A., and Ienne, P. (2019). Speculative dataflow circuits. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '19, page 162–171, New York, NY, USA. Association for Computing Machinery

Speculative Loop Pipelining

Source-to-source transformations

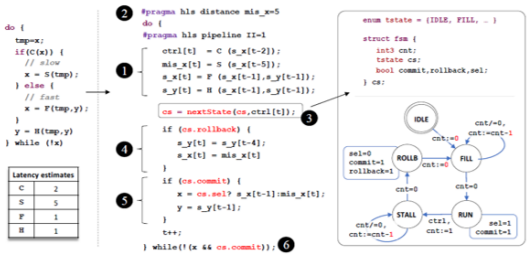
- Transform the input code to expose potential speculations

Speculation support

- Decouple control logic from datapath
- HLS toolchain infers the speculative structure

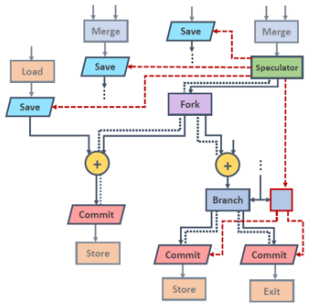
Limitations

- Lack of support for intertwined speculations



[Derrien et al., 2020] Derrien, S., Marty, T., Rokicki, S., and Yuki, T. (2020). Toward speculative loop pipelining for high-level synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):4229–4239.

Speculative Dataflow Circuits



…ngs of the 2019 ACM/SIGDA International Symposium on Field-

…sactional datapath specifications. *IEEE Transactions on Computer-Aided*

Upcoming work

Final Goal

Synthesize in-order pipelined instruction set processors from an instruction set simulator

Speculative Loop Pipelining

Focus on source-to-source transformations as introduced by [Derrien et al., 2020]

Multiple speculation support

Study the interwinding of multiple speculations

Fine-grain misspeculation recovery

Improve existing techniques to support finer-grain roll-back and misspeculation recovery

[Derrien et al., 2020] Derrien, S., Marty, T., Rokicki, S., and Yuki, T. (2020). Toward speculative loop pipelining for high-level synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):4229–4239.