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
June 30th, 2021

High-Level Synthesis of Instruction Set Processors

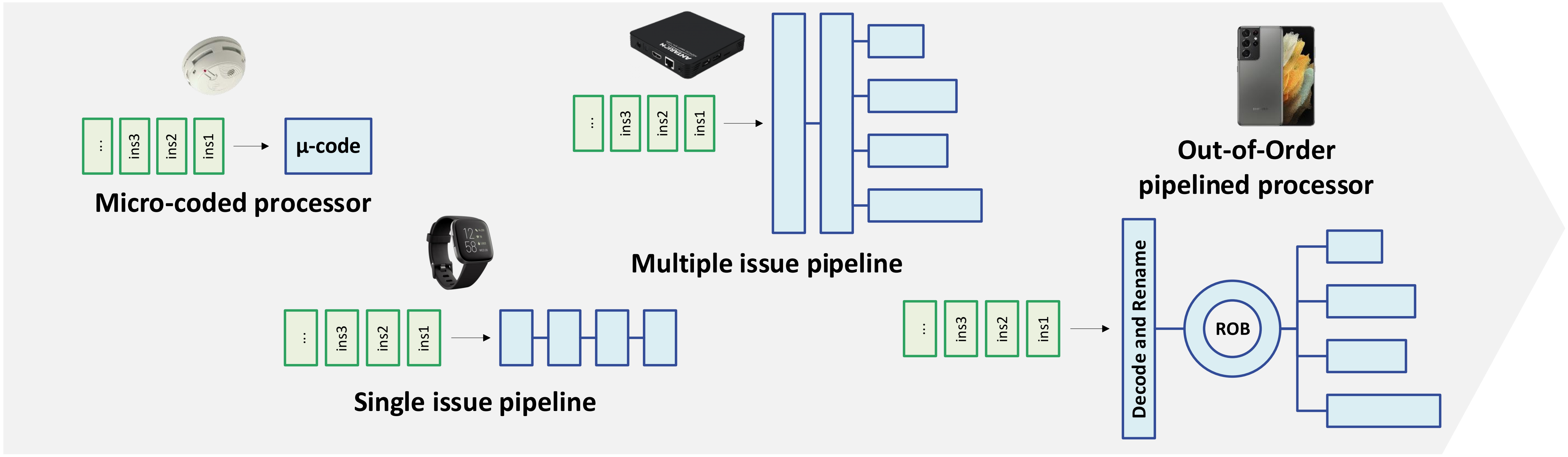
Low energy

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



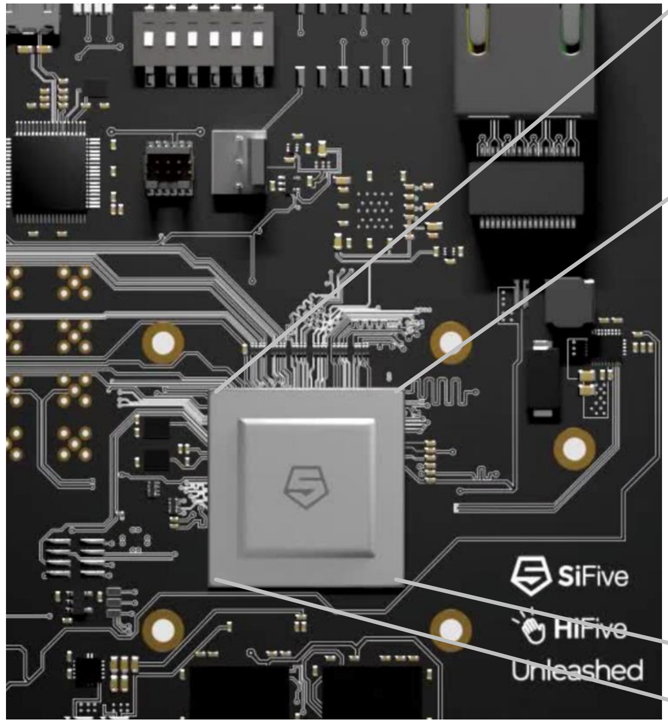
Low energy

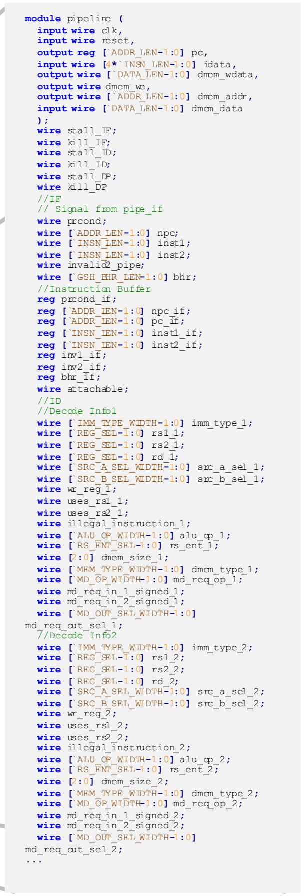
High performance

Takeaway

Need for customizable architecture for heterogeneous embedded applications

Customizing Instruction Set Processors





```verilog
module pipeline (
  input wire clk,
  input wire reset,
  output reg [`ADDR_LEN-1:0] pc,
  input wire [4*`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 hardware

Early work for Application-Specific Instruction Set Processor design using Domain-Specific Languages [Cloutier and Thomas, 1993; Huang and Despain, 1993]

Previous approaches still rely on low-level specifications of the hardware

Our work

Automatic synthesis of **pipelined processors** from a **behavioral description**

[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

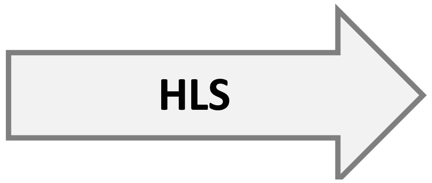
[Patterson and Hennessy, 2017] Patterson, D. A. and Hennessy, J. L. (2017). *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition.

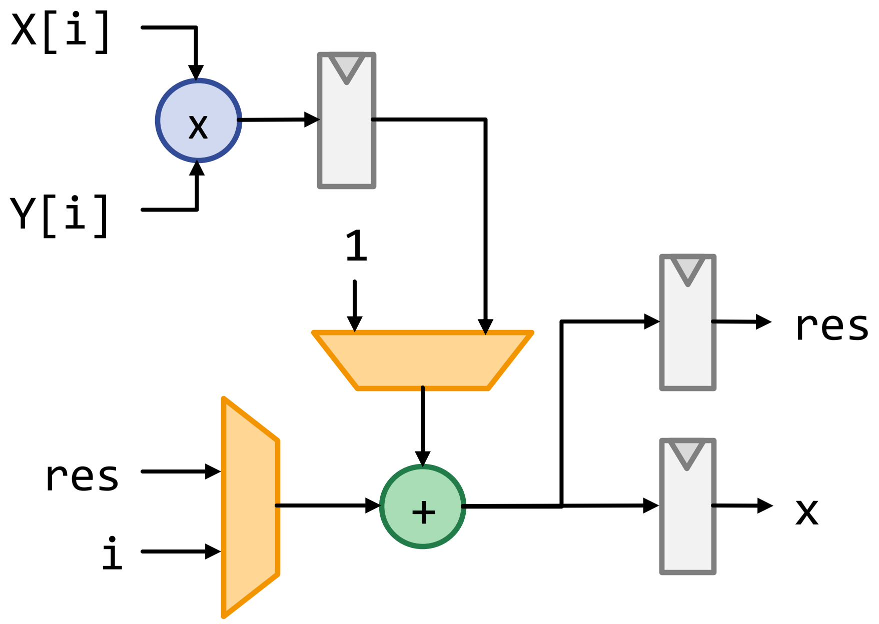
High-Level Synthesis

Synthesizing circuits from an algorithmic specification

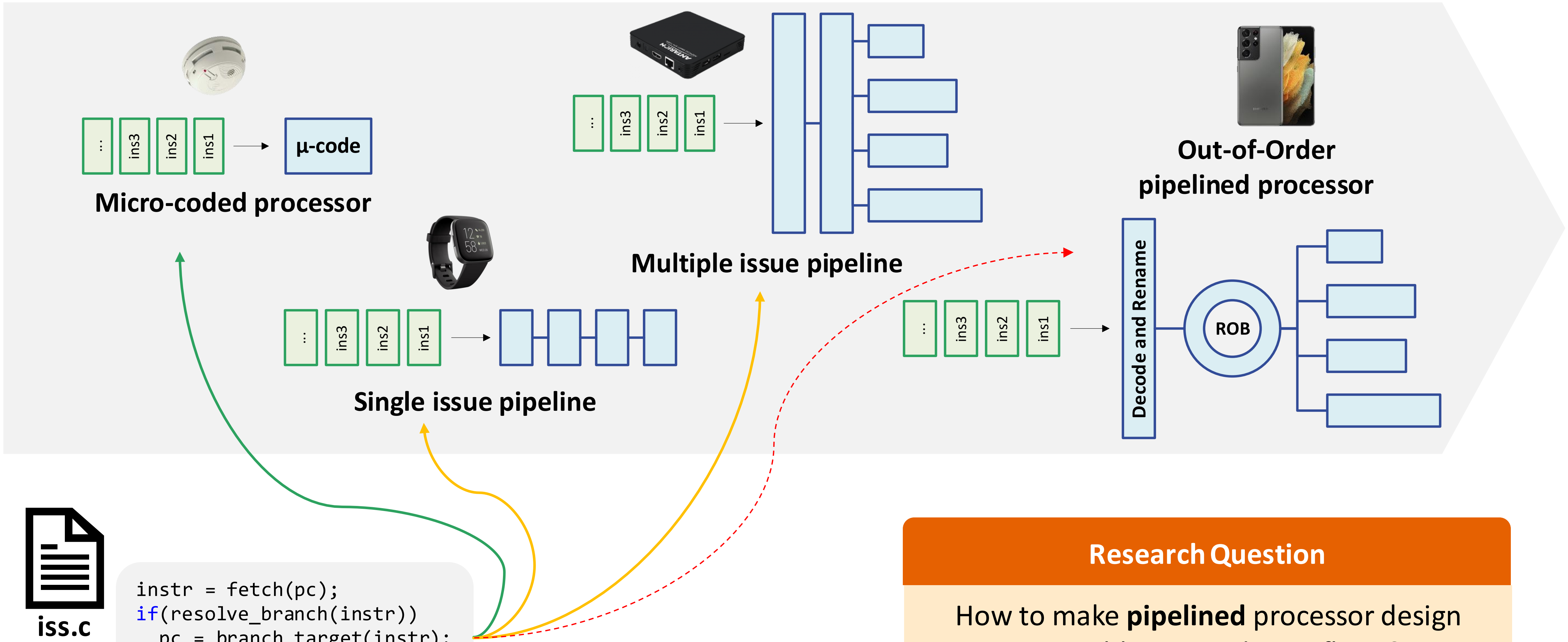
Efficient design synthesis for regular access patterns and computations

```
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;
}
```





Problem Statement



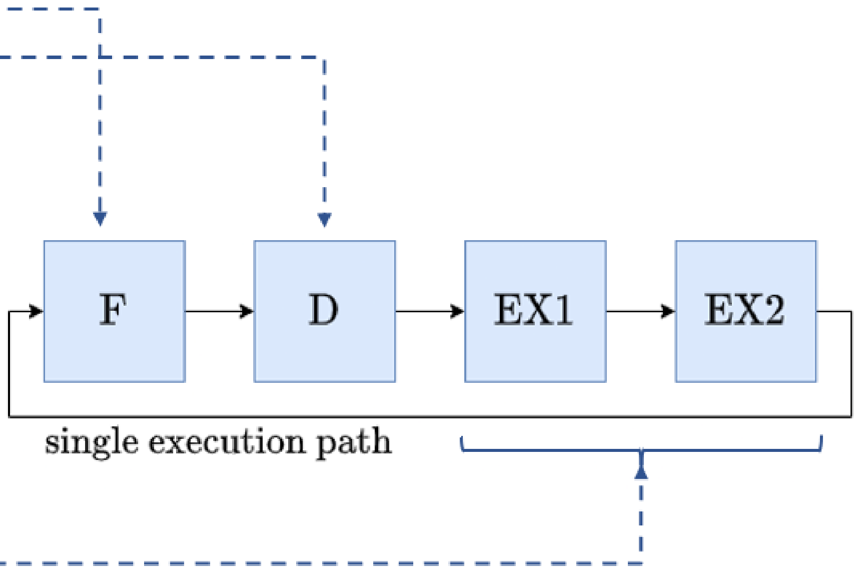
```
instr = fetch(pc);
if(resolve_branch(instr))
  pc = branch_target(instr);
else
  pc += 1;
```

Research Question

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

Mapping an ISS to Hardware

```
while (1) {
  ir = fetch(pc);
  op, rd, wr = decode(ir);
  if (op == BR) {
    pc = branch_target(op);
  } else {
    pc = pc + 4;
    switch(op) {
    case NEG:
      arg = regs[rd];
      regs[wr] = neg(arg);
      break;
    case ADDPC: // ...
    case LOAD:  // ...
    // ...
    }
  }
}
```

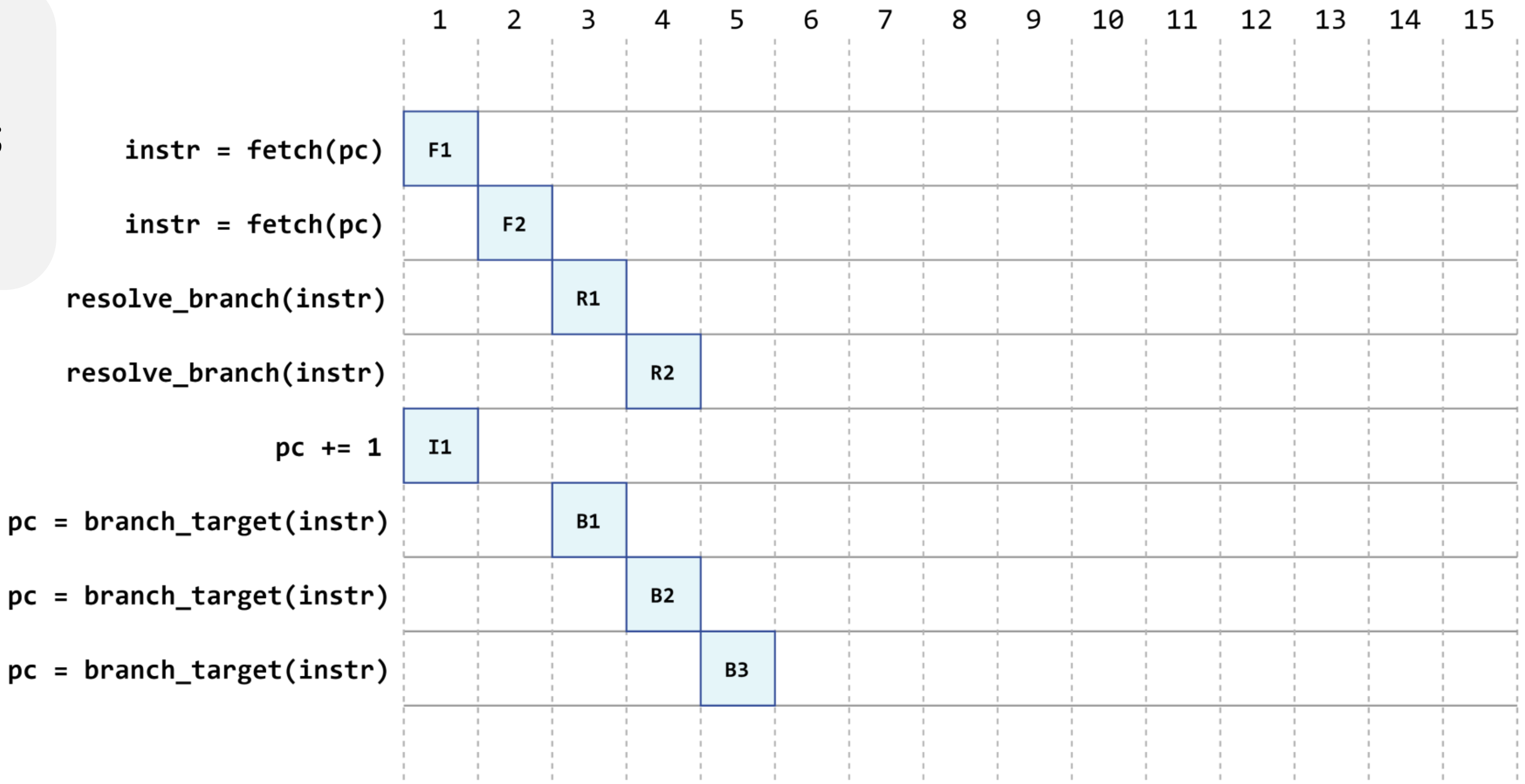


1	2	3	4	5	6	7	8	9
F	D	EX1	EX2					
	F	D	EX1	EX2				
		F	D	D	EX1	EX2		
			F	F	D	EX1	EX2	
					F	D	EX1	EX2

Loop Pipelining and Static Scheduling

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

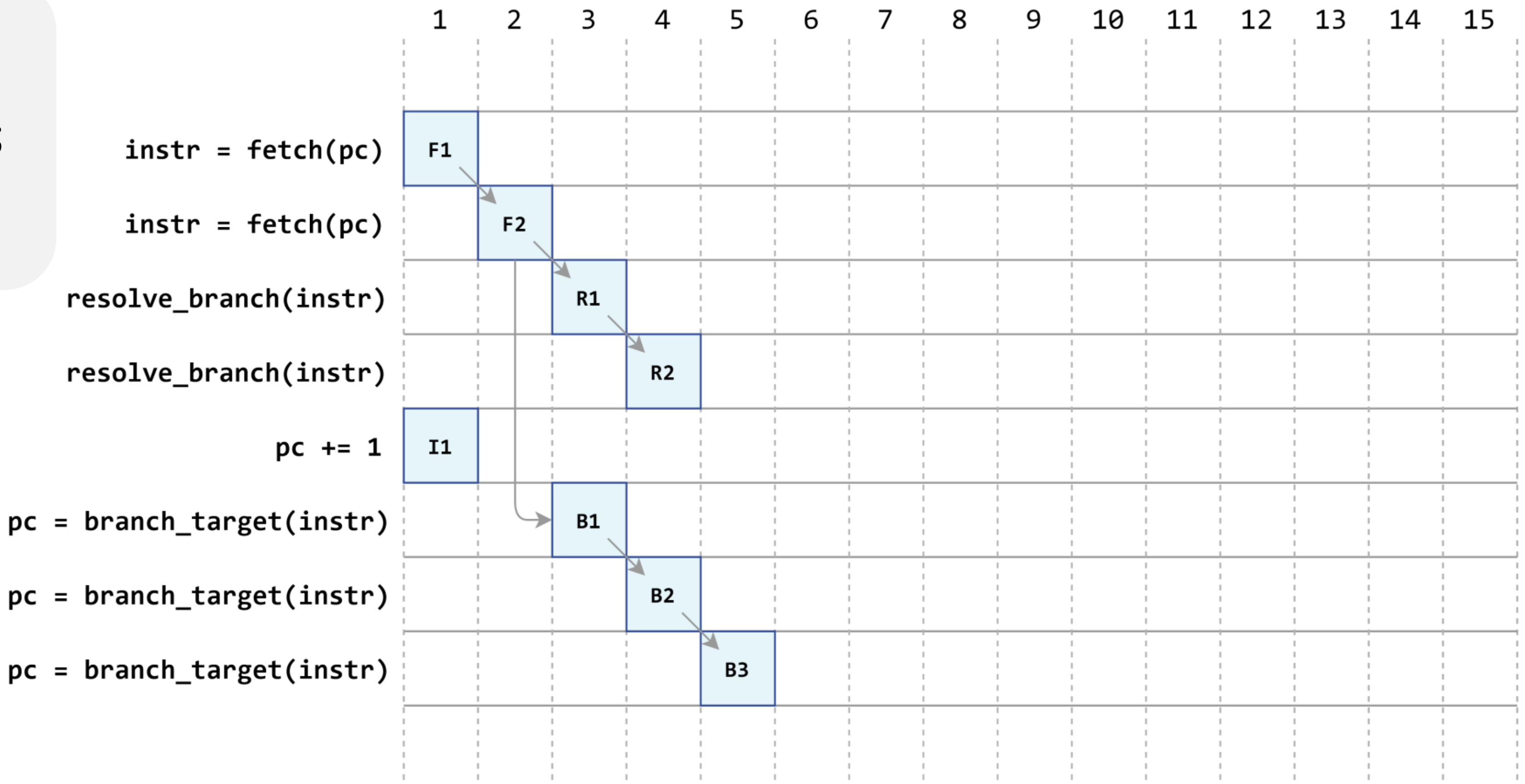
Simplified ISS



Loop Pipelining and Static Scheduling

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

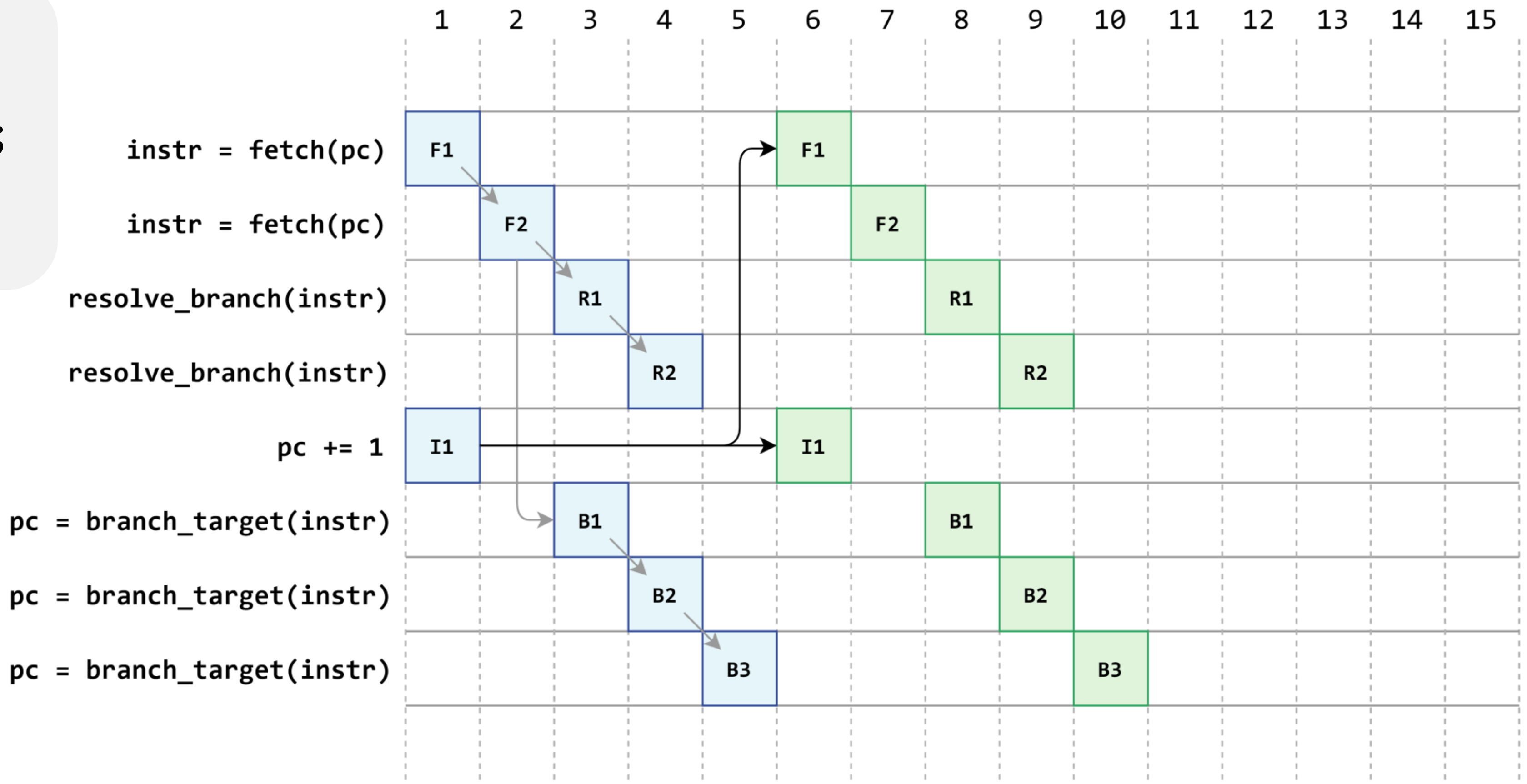
Simplified ISS



Loop Pipelining and Static Scheduling

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

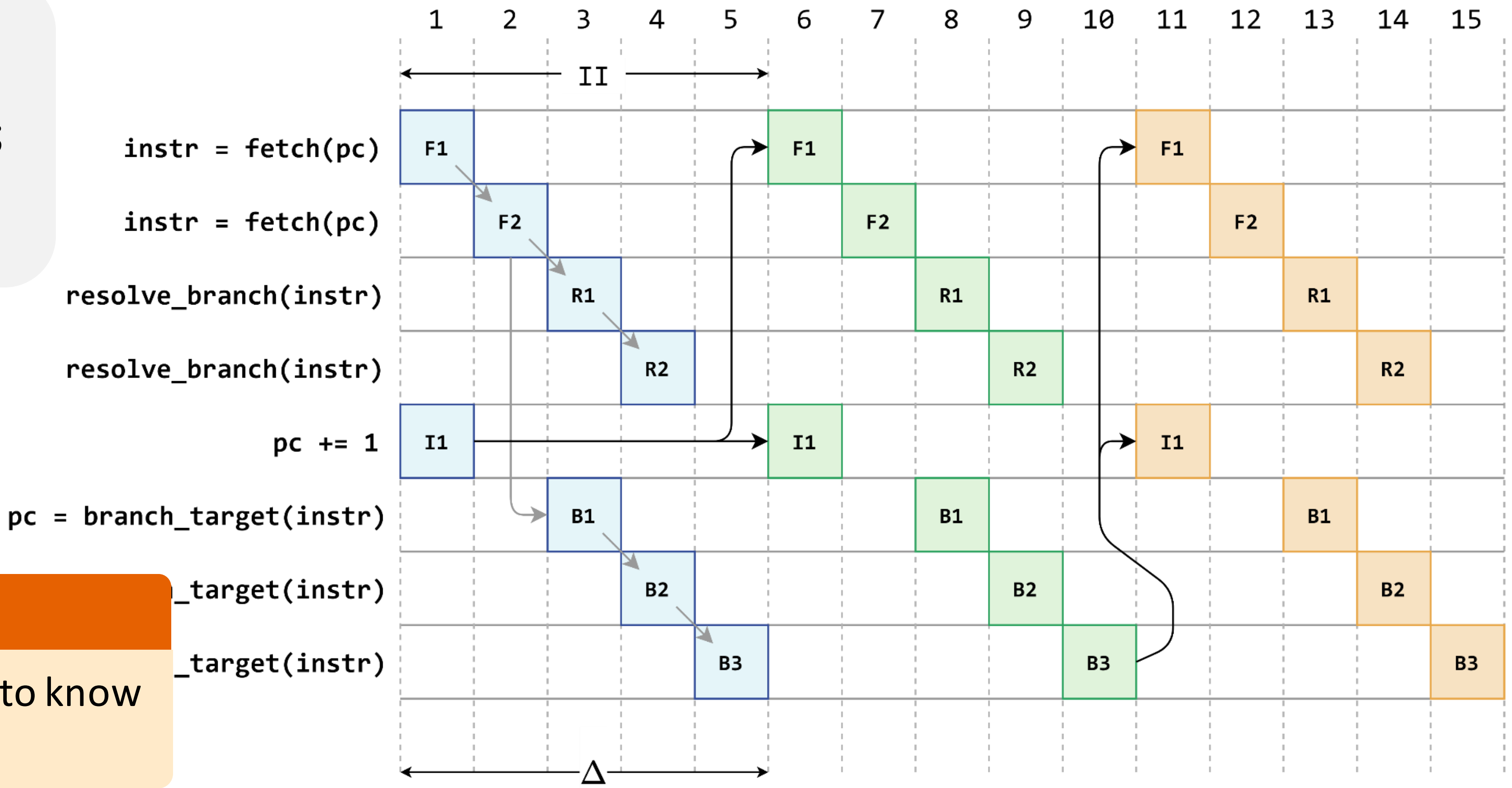
Simplified ISS



Loop Pipelining and Static Scheduling

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

Simplified ISS



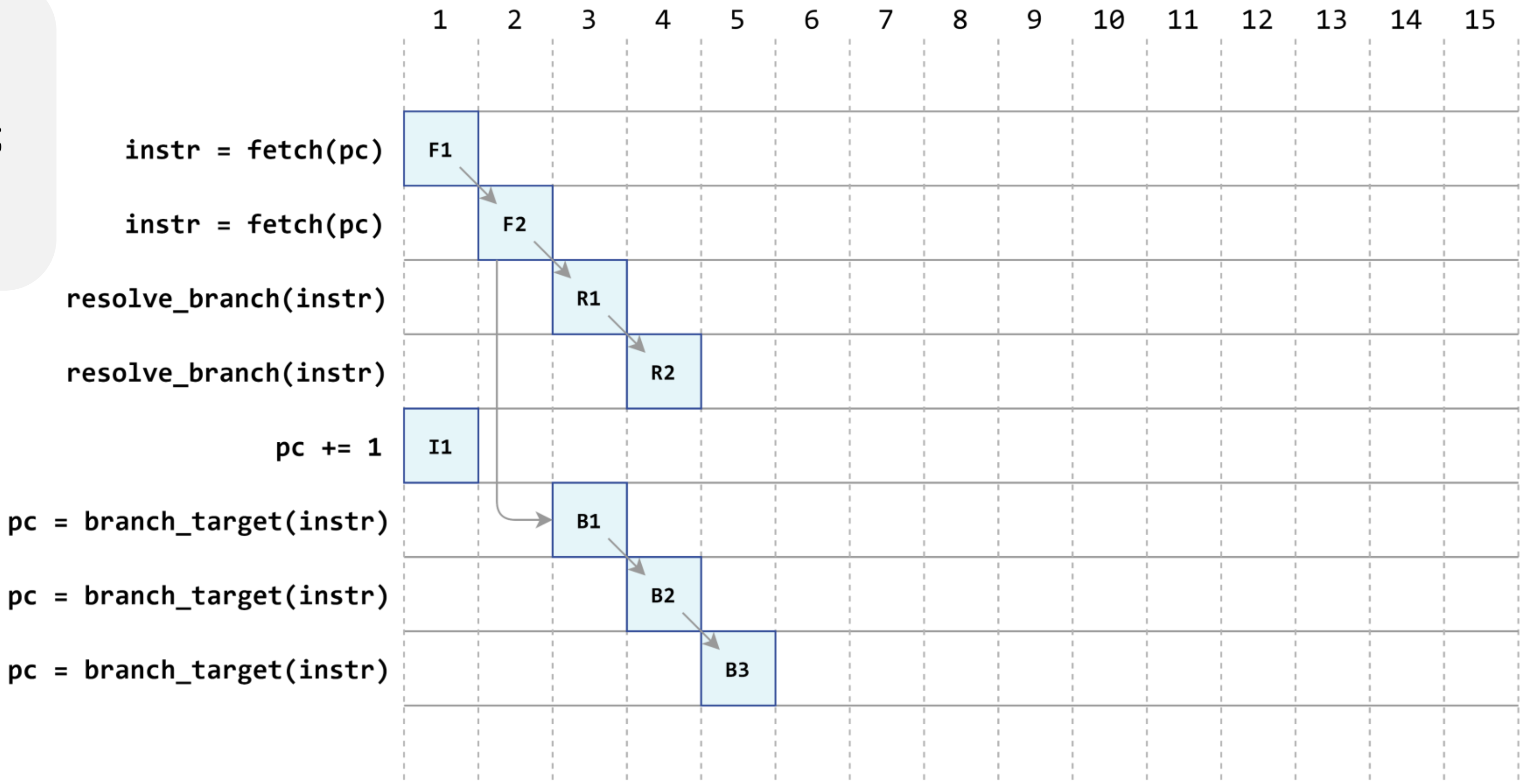
Inefficient design

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

Speculative High-Level Synthesis

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

Simplified ISS



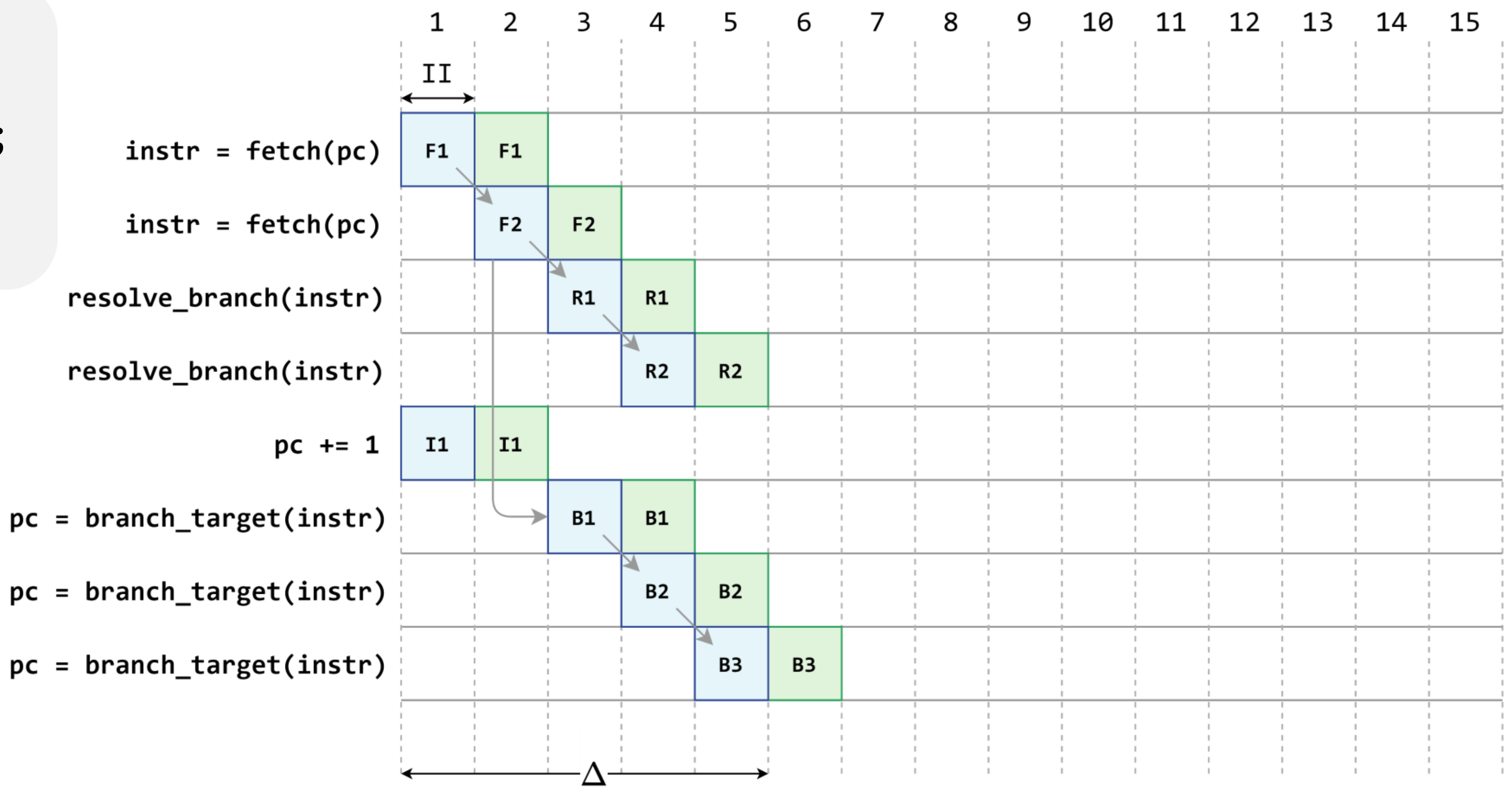
[Derrien et al., 2020] Derrien, S., Marty, T., Rokicki, S., and Yuki, T. (2020). Toward speculative loop pipelining for high-level synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):4229–4239.

[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



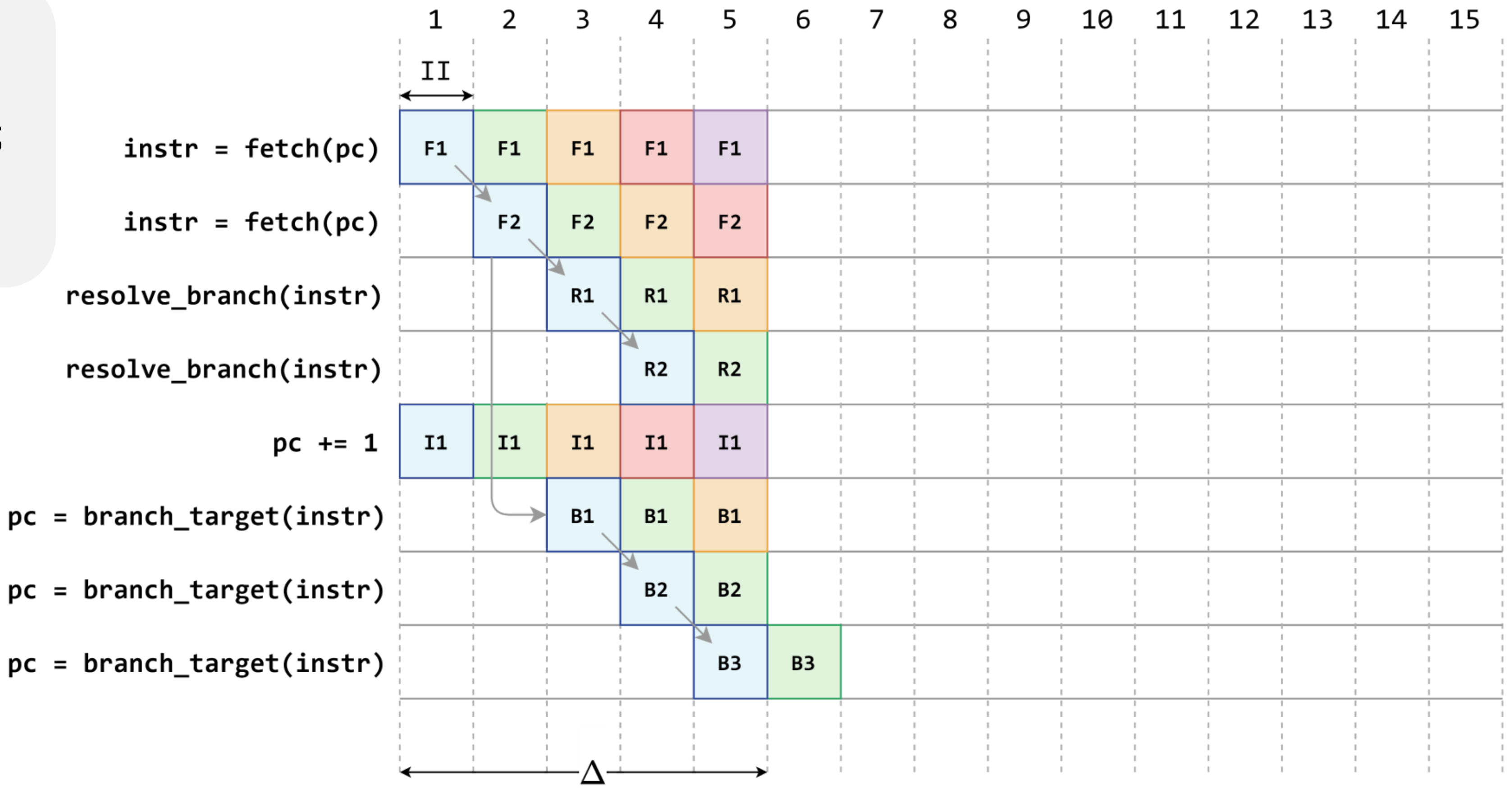
[Derrien et al., 2020] Derrien, S., Marty, T., Rokicki, S., and Yuki, T. (2020). Toward speculative loop pipelining for high-level synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):4229–4239.

[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



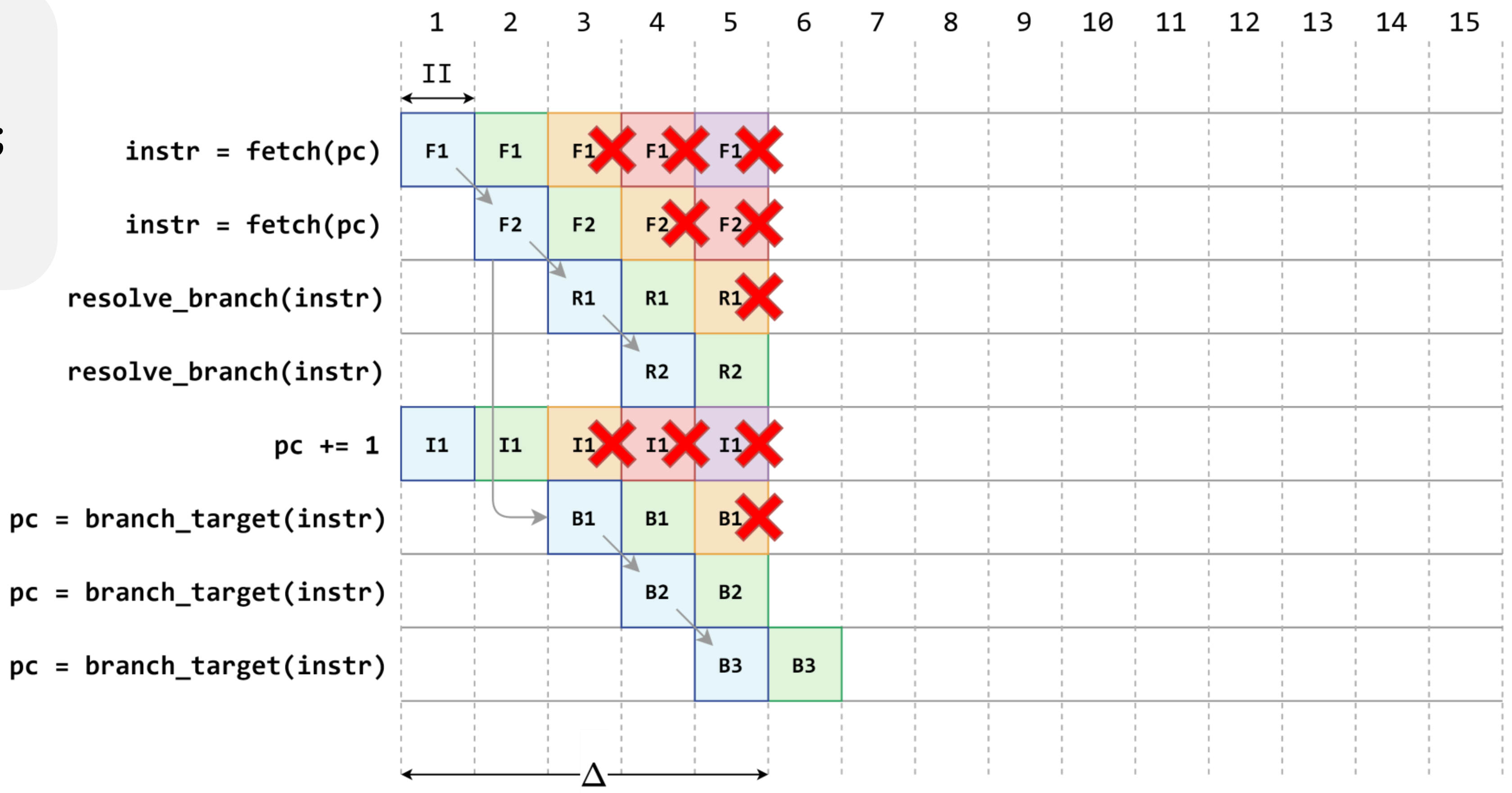
[Derrien et al., 2020] Derrien, S., Marty, T., Rokicki, S., and Yuki, T. (2020). Toward speculative loop pipelining for high-level synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):4229–4239.

[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



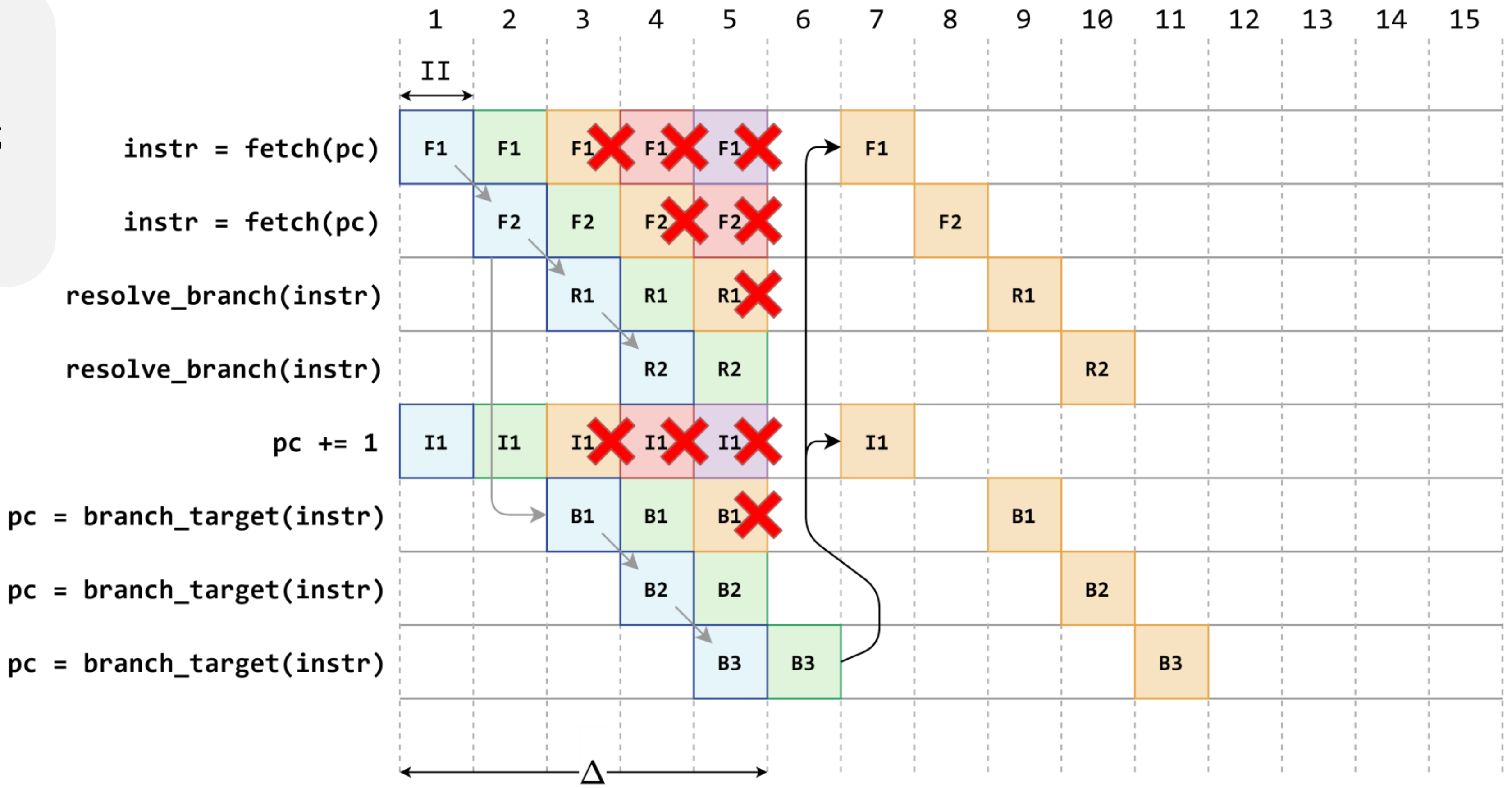
[Derrien et al., 2020] Derrien, S., Marty, T., Rokicki, S., and Yuki, T. (2020). Toward speculative loop pipelining for high-level synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):4229–4239.

[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



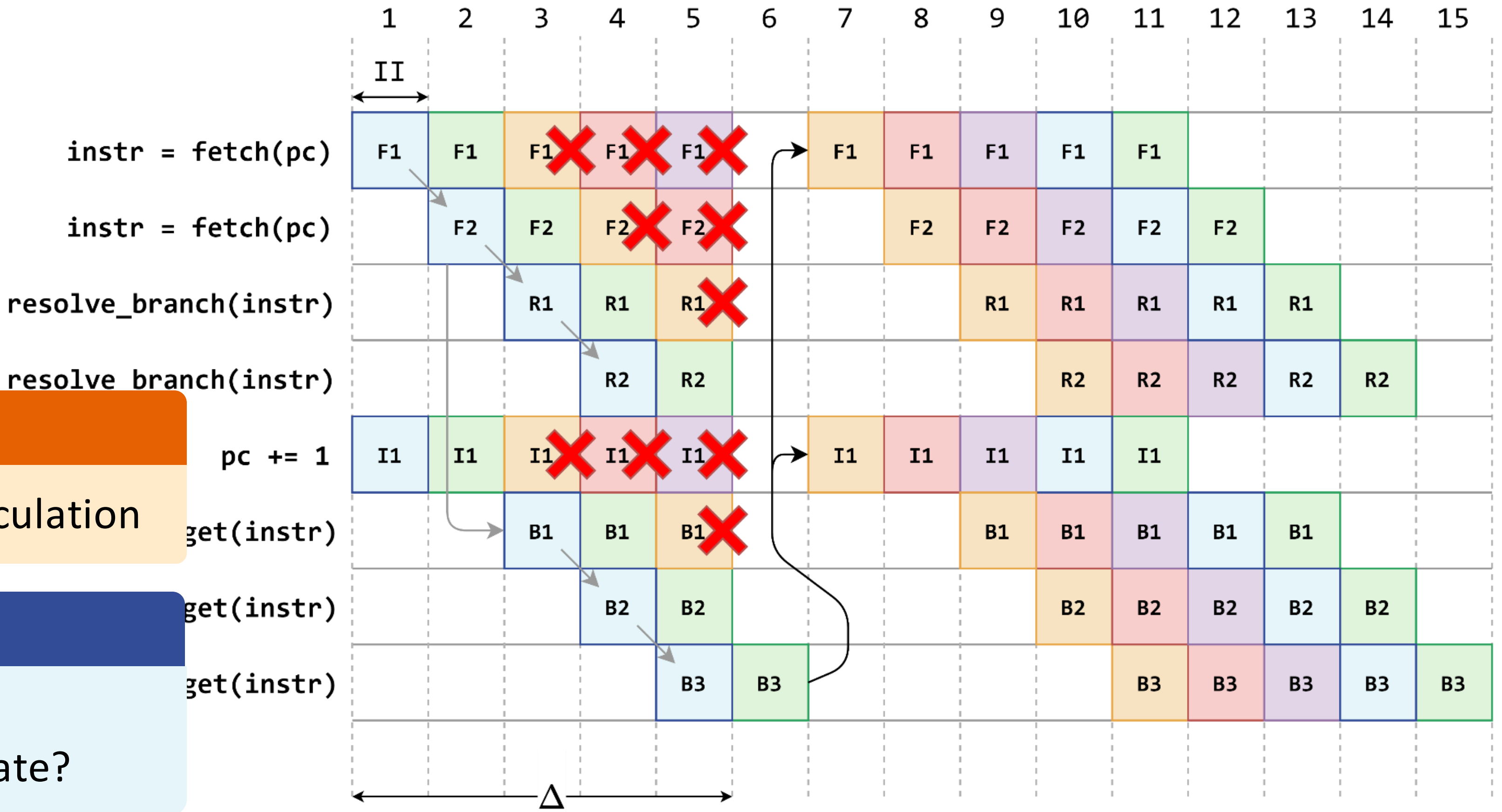
[Derrien et al., 2020] Derrien, S., Marty, T., Rokicki, S., and Yuki, T. (2020). Toward speculative loop pipelining for high-level synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):4229–4239.

[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 [Josipovic'2019, Derrien'2020]

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

Simplified ISS



Takeaway

Pipelined processor design requires speculation

Speculative HLS

- Maximal ILP and resource utilization
- What to speculate on? How to speculate?

[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

```
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

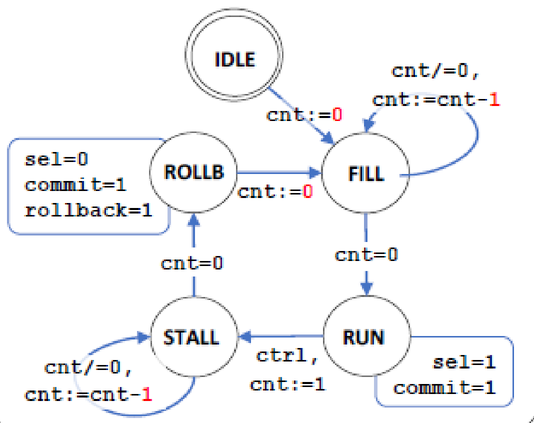
```
2  #pragma hls distance mis_x=5
   do {
   #pragma hls pipeline II=1
1      ctrl[t]   = C (s_x[t-2]);
       mis_x[t] = S (s_x[t-5]);
       s_x[t]  = F (s_x[t-1],s_y[t-1]);
       s_y[t]  = H (s_x[t-1],s_y[t-1]);

       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
```

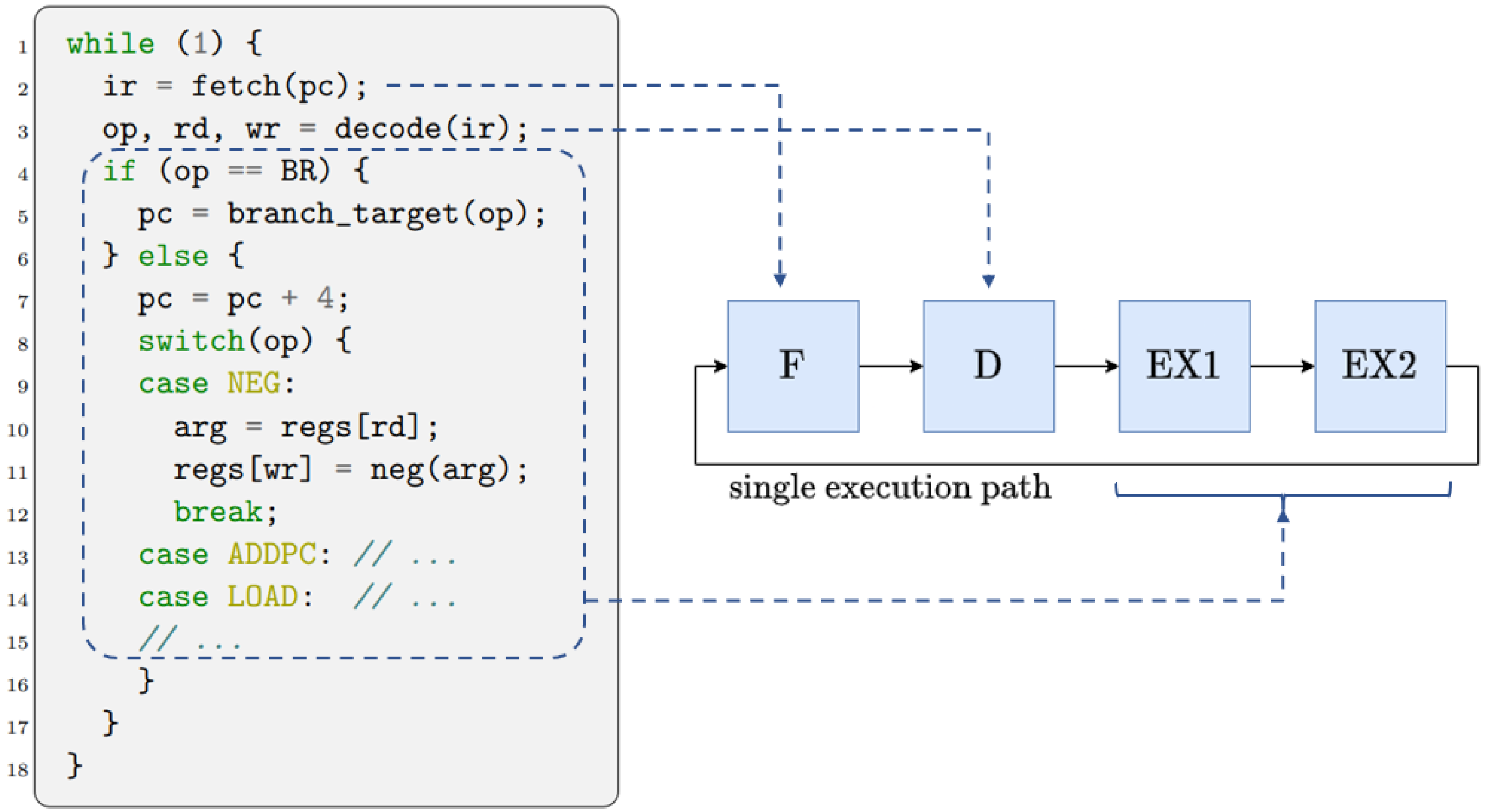
```
enum tstate = {IDLE, FILL, … }

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



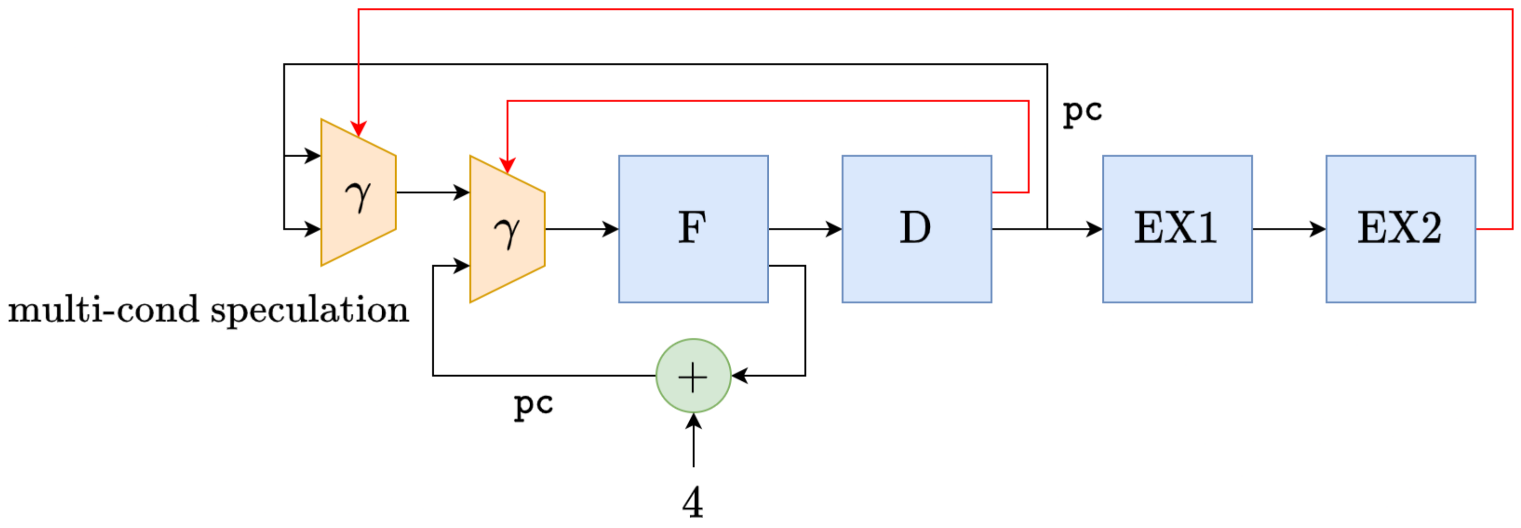
[Derrien et al., 2020] Derrien, S., Marty, T., Rokicki, S., and Yuki, T. (2020). Toward speculative loop pipelining for high-level synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):4229–4239.

Reminder: Mapping an ISS to Hardware

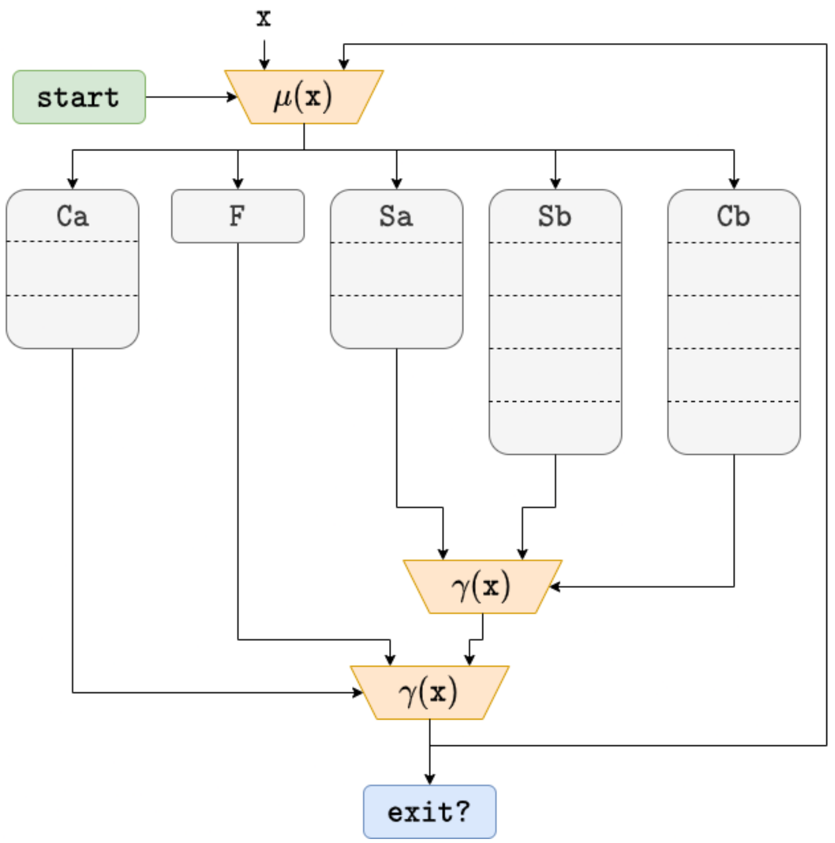

```
while (1) {
  ir = fetch(pc);
  op, rd, wr = decode(ir);
  if (op == BR) {
    pc = branch_target(op);
  } else {
    pc = pc + 4;
    switch(op) {
    case NEG:
      arg = regs[rd];
      regs[wr] = neg(arg);
      break;
    case ADDPC: // ...
    case LOAD:  // ...
    // ...
    }
  }
}
```

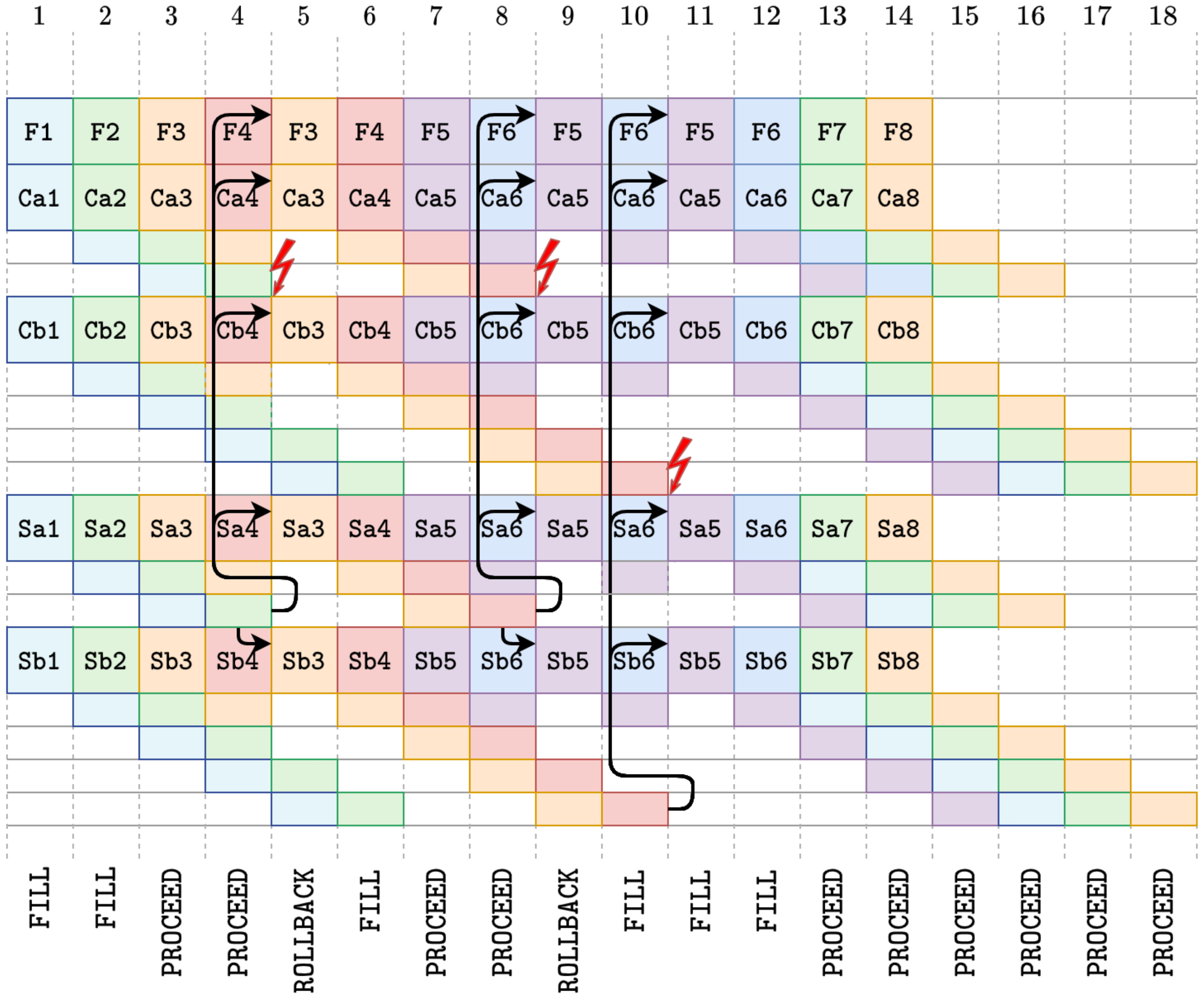
Speculating on the Value of PC

```
while(1) {
  ir = fetch(pc);
  op, op0, imm = decode(ir);
  if(op == BR)
    pc = imm;
  else if(op == CONDBR)
    pc = exec(op, op0, imm);
  else
    pc = pc + 4;
}
```



Multi-Conditional Speculation





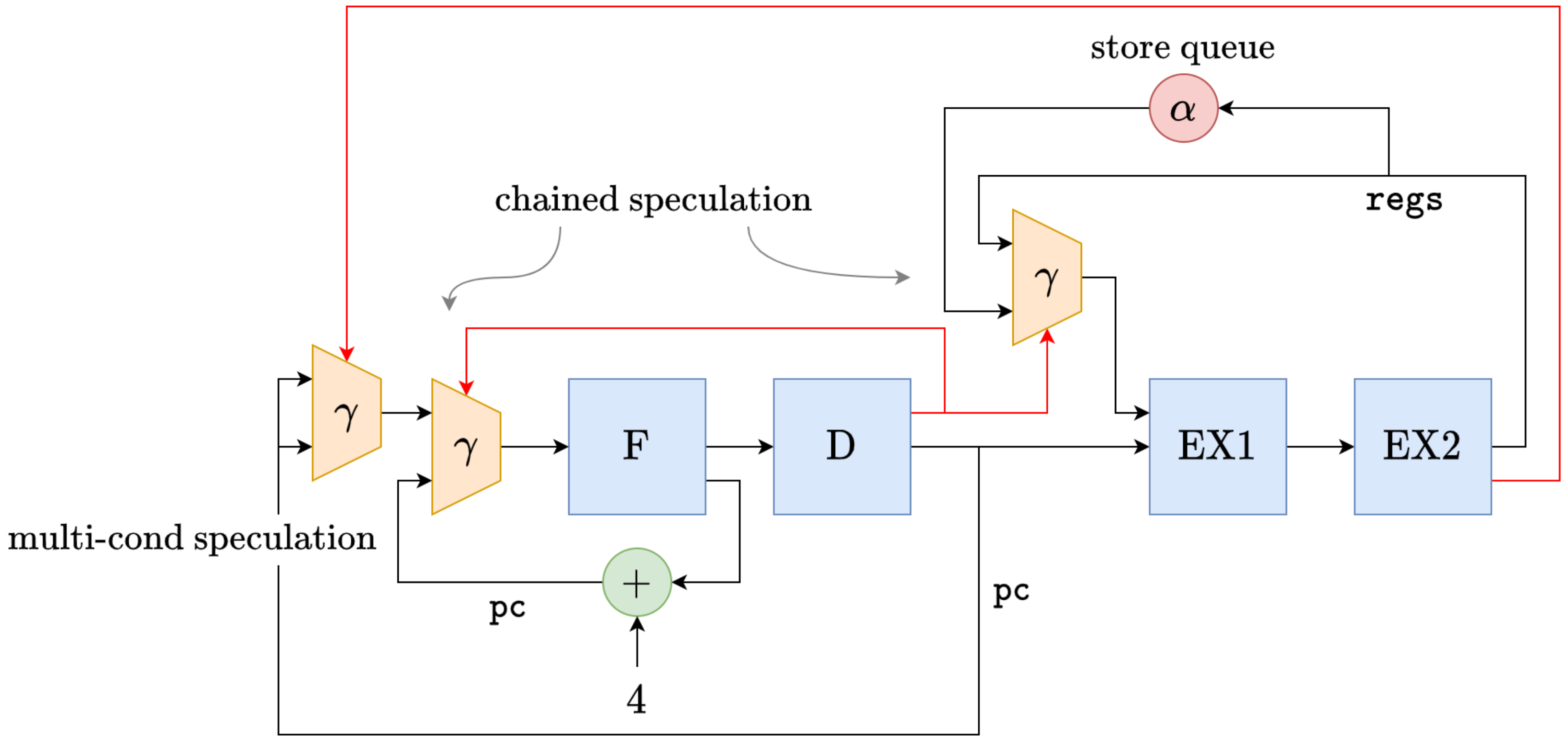
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
F1 F2 F3 F4 F3 F4 F5 F6 F5 F6 F5 F6 F7 F8
Ca1 Ca2 Ca3 Ca4 Ca3 Ca4 Ca5 Ca6 Ca5 Ca6 Ca5 Ca6 Ca7 Ca8
Cb1 Cb2 Cb3 Cb4 Cb3 Cb4 Cb5 Cb6 Cb5 Cb6 Cb5 Cb6 Cb7 Cb8
Sa1 Sa2 Sa3 Sa4 Sa3 Sa4 Sa5 Sa6 Sa5 Sa6 Sa5 Sa6 Sa7 Sa8
Sb1 Sb2 Sb3 Sb4 Sb3 Sb4 Sb5 Sb6 Sb5 Sb6 Sb5 Sb6 Sb7 Sb8
FILL FILL PROCEED PROCEED ROLLBACK FILL PROCEED PROCEED ROLLBACK FILL FILL FILL PROCEED PROCEED PROCEED PROCEED PROCEED PROCEED

Key Idea

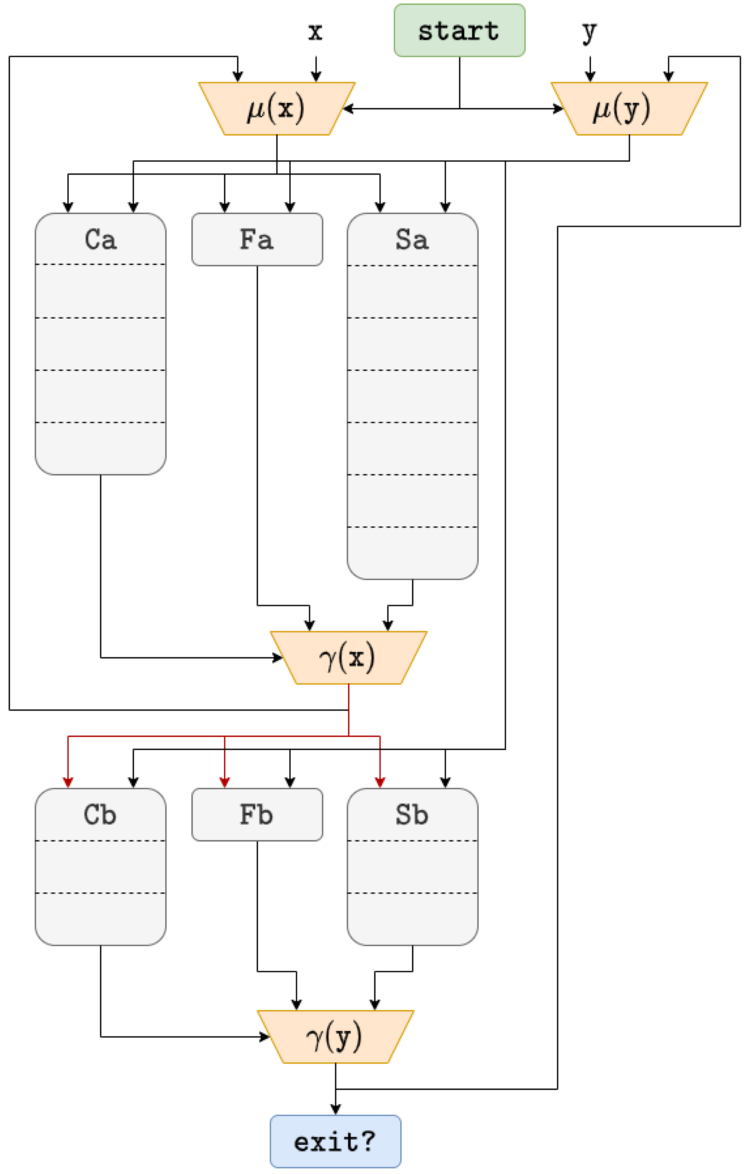
Speculatively select paths of increasing length

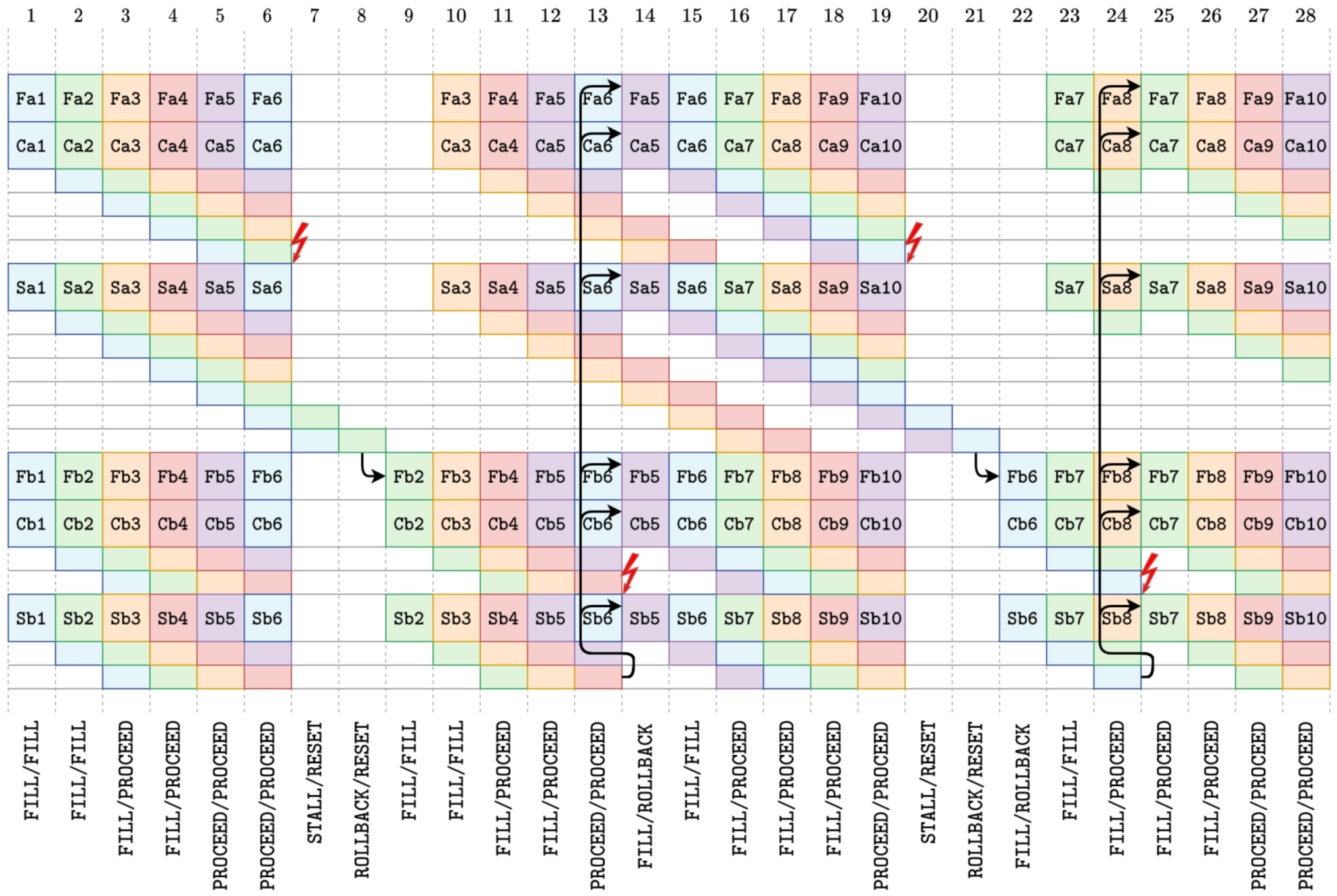
Adding Register Alias Speculation

```
while (1) {
  // ...
  case NEG:
    if (rd == prev_wr)
      arg = ex2;
    else
      arg = regs[rd];
    regs[wr] = ex2
             = neg(arg);
    prev_wr = wr;
    break;
  // ...
}
```



Chained Speculation





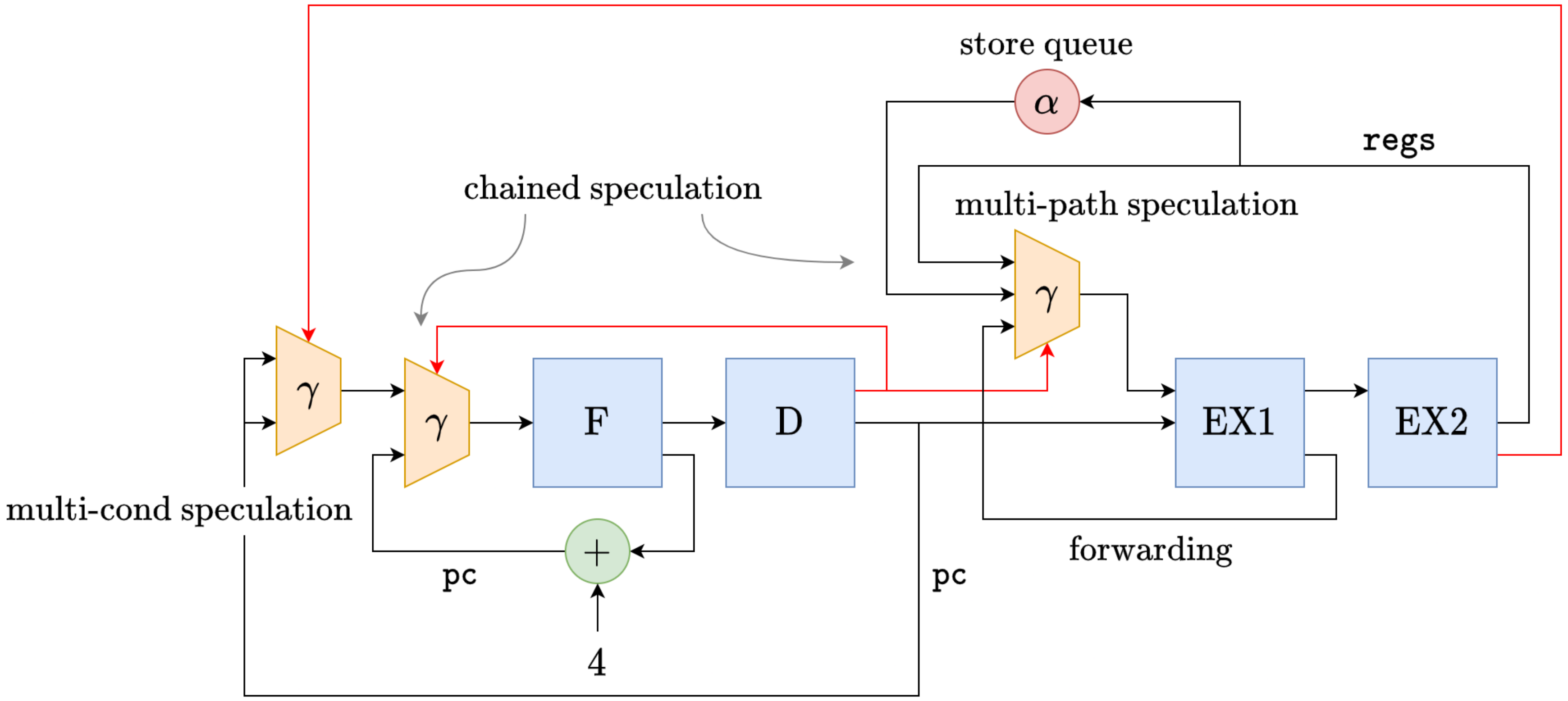
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
Fa1 Fa2 Fa3 Fa4 Fa5 Fa6 Fa3 Fa4 Fa5 Fa6 Fa5 Fa6 Fa7 Fa8 Fa9 Fa10 Fa7 Fa8 Fa7 Fa8 Fa9 Fa10
Ca1 Ca2 Ca3 Ca4 Ca5 Ca6 Ca3 Ca4 Ca5 Ca6 Ca5 Ca6 Ca7 Ca8 Ca9 Ca10 Ca7 Ca8 Ca7 Ca8 Ca9 Ca10
Sa1 Sa2 Sa3 Sa4 Sa5 Sa6 Sa3 Sa4 Sa5 Sa6 Sa5 Sa6 Sa7 Sa8 Sa9 Sa10 Sa7 Sa8 Sa7 Sa8 Sa9 Sa10
Fb1 Fb2 Fb3 Fb4 Fb5 Fb6 Fb2 Fb3 Fb4 Fb5 Fb6 Fb5 Fb6 Fb7 Fb8 Fb9 Fb10 Fb6 Fb7 Fb8 Fb7 Fb8 Fb9 Fb10
Cb1 Cb2 Cb3 Cb4 Cb5 Cb6 Cb2 Cb3 Cb4 Cb5 Cb6 Cb5 Cb6 Cb7 Cb8 Cb9 Cb10 Cb6 Cb7 Cb8 Cb7 Cb8 Cb9 Cb10
Sb1 Sb2 Sb3 Sb4 Sb5 Sb6 Sb2 Sb3 Sb4 Sb5 Sb6 Sb5 Sb6 Sb7 Sb8 Sb9 Sb10 Sb6 Sb7 Sb8 Sb7 Sb8 Sb9 Sb10
FILL/FILL
FILL/FILL
FILL/PROCEED
FILL/PROCEED
PROCEED/PROCEED
PROCEED/PROCEED
STALL/RESET
ROLLBACK/RESET
FILL/FILL
FILL/FILL
FILL/PROCEED
FILL/PROCEED
PROCEED/PROCEED
FILL/ROLLBACK
FILL/FILL
FILL/PROCEED
FILL/PROCEED
FILL/PROCEED
PROCEED/PROCEED
STALL/RESET
ROLLBACK/RESET
FILL/ROLLBACK
FILL/FILL
FILL/PROCEED
FILL/PROCEED
FILL/PROCEED
PROCEED/PROCEED
PROCEED/PROCEED

Key Idea

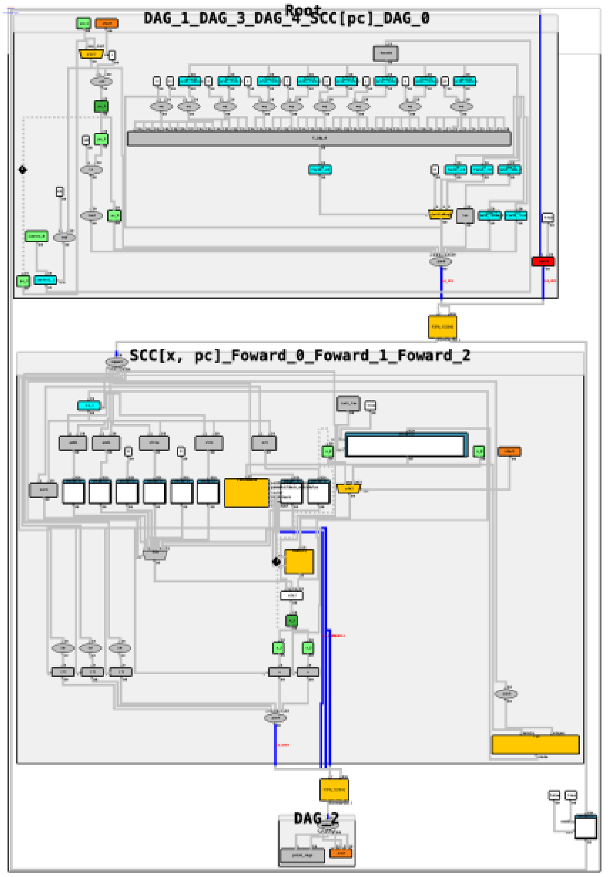
Restart everything if Ca mispeculates, but restart only y if Cb mispeculates

Pipelined Processor with Forwarding

```
while (1) {
  // ...
  case NEG:
    if (rd == prev_wr &&
          ex1_data_avail(rd))
        arg = ex1;
    else {
      if (rd == prev_wr)
        arg = ex2;
      else
        arg = regs[rd];
    }
    ex1 = exec1(arg);
    set_ex1_data_avail(rd);
    ex2 = exec2(ex1);
    regs[wr] = ex2;
    prev_wr = wr;
    break;
  // ...
}
```



Current Status


Root
DAG_1_DAG_3_DAG_4_SCC[pc]_DAG_0
SCC[x, pc]_Foward_0_Foward_1_Foward_2
DAG_2

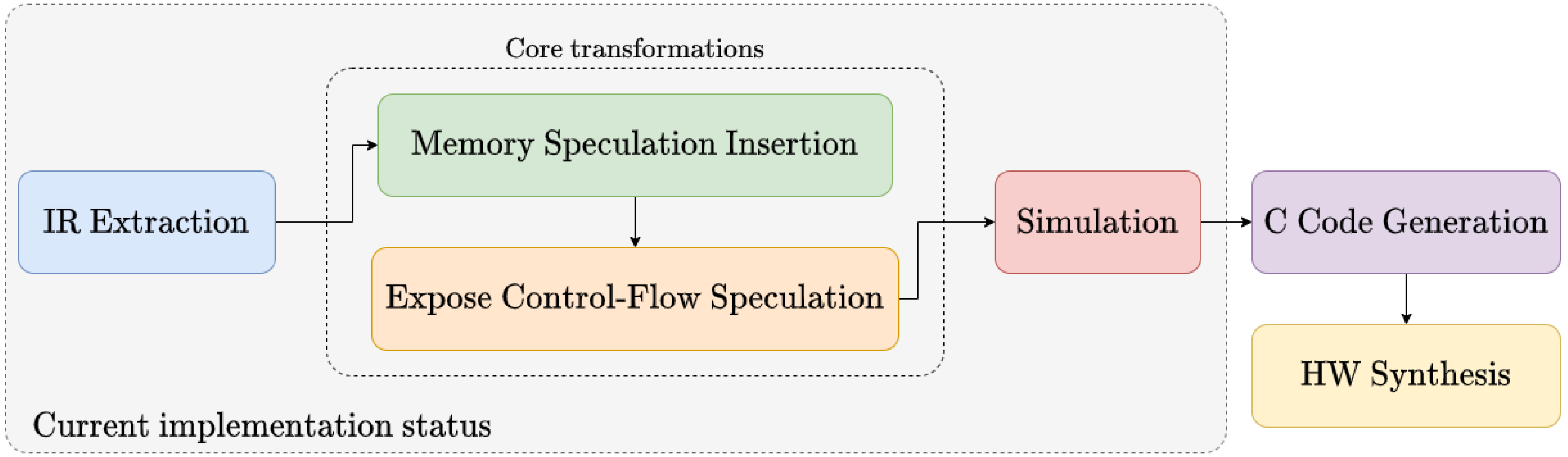
Problem Characterization

Taxonomy of speculation patterns

GeCoS
Generic Compiler Suite

Implementation Status

- Transformation passes to make speculative patterns emerge
- Initial validation infrastructure for multiple speculations


Core transformations
IR Extraction
Memory Speculation Insertion
Expose Control-Flow Speculation
Simulation
C Code Generation
HW Synthesis
Current implementation status

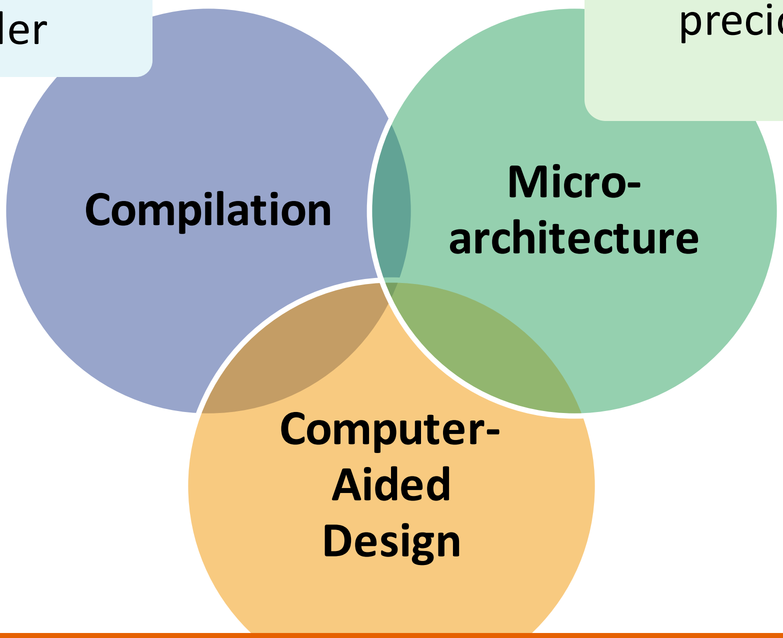
Conclusion

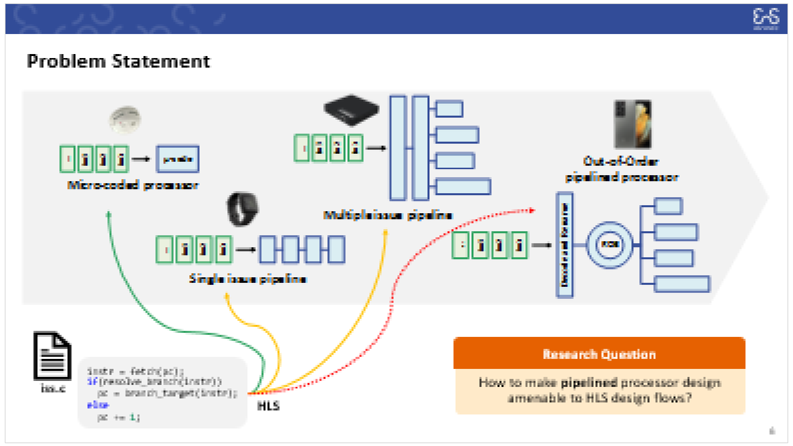
Compiling Speculative Hardware

Represent speculation at the compiler level, manipulate hardware in the compiler

Need for New Tools

We cannot afford to continue wasting precious transistors and need fast and reliable prototyping tools

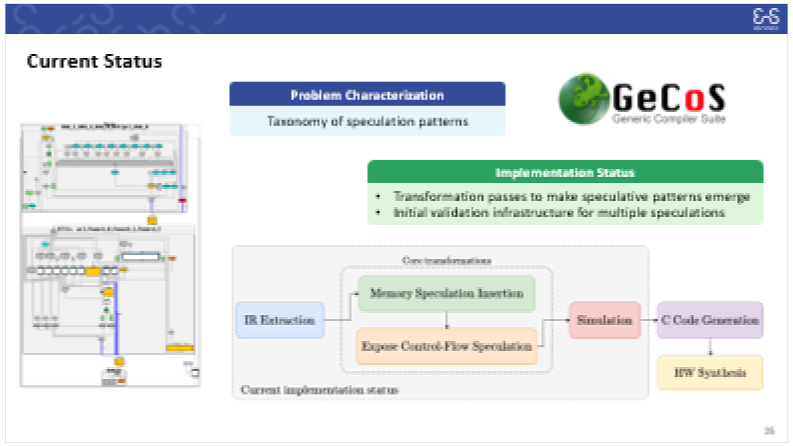



Problem Statement

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

Research Question

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


Current Status

Problem Characterization

Taxonomy of speculation patterns

Implementation Status

- Transformation passes to make speculative patterns emerge
- Initial validation infrastructure for multiple speculations

Design Space Exploration

Easily explore tradeoffs between different pipeline depths, speculation configurations

Backup

Application-Specific Instruction Set Processors

```
if((IN.pRn == WB.IN.pRn) && (WB.IN.writeback_code == WRITEBACK_WRITERN)) {
  // bypass the value from WB
  OUT.op_2 = WB.IN.result;
} else {
  // read the value from the register file
  TpRn = IN.pRn;
  OUT.op_2 = R[TpRn.ExtractToLong(0,4)];
}
```

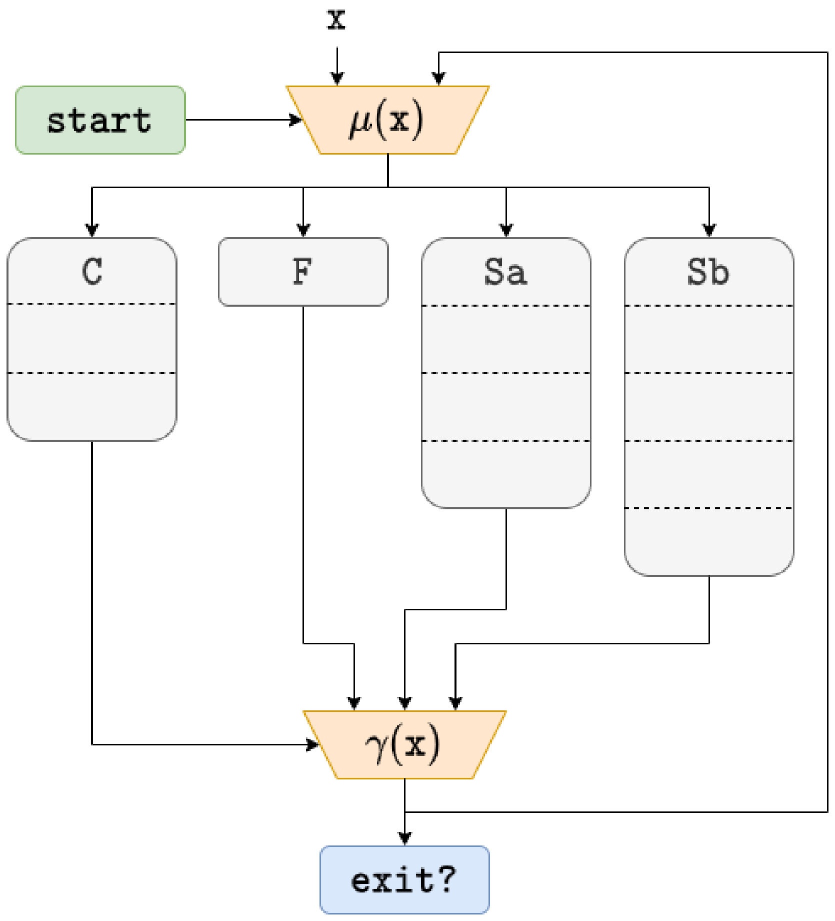
Example LISATek code [Klemm et al., 2007]

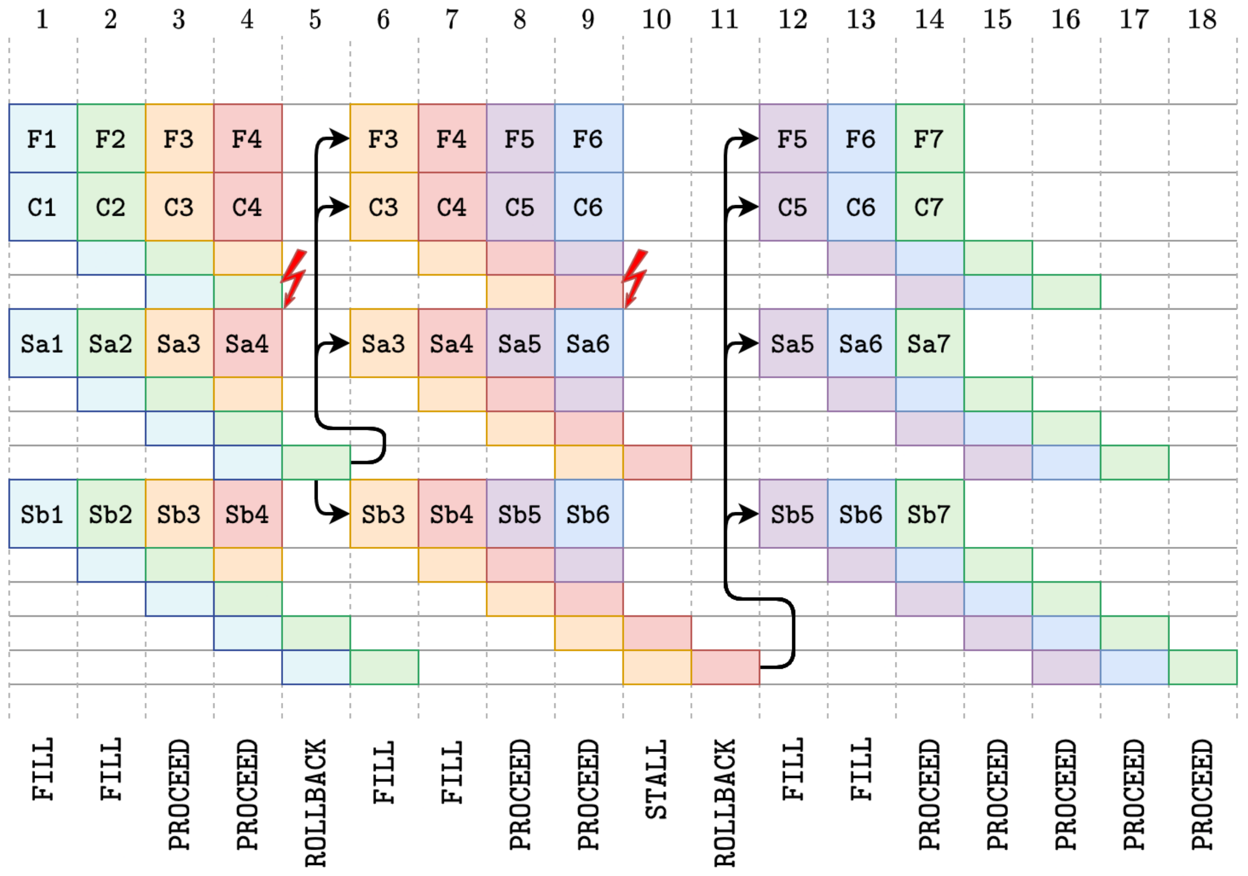
[Klemm et al., 2007] Klemm, R., Sabugo, J.P., Ahlendorf, H., & Fettweis, G. (2007). Using LISATek for the Design of an ASIP Core including Floating Point Operations. *MBMV*.

[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

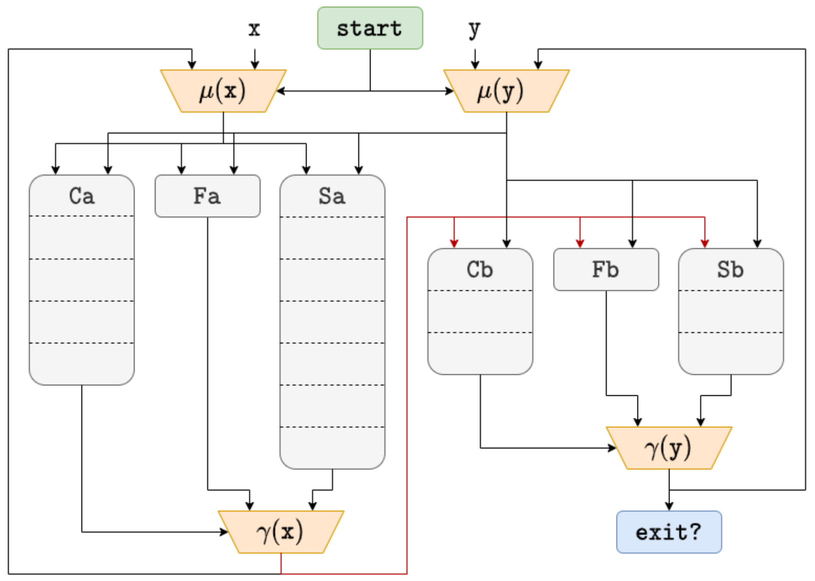
[Huang and Despain, 1993] Huang, I.-J. and Despain, A. M. (1993). Hardware/software resolution of pipeline hazards in pipeline synthesis of instruction set processors. In *Proceedings of the 1993 IEEE/ACM International Conference on Computer-Aided Design*, ICCAD'93, page 594–599, Washington, DC, USA.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

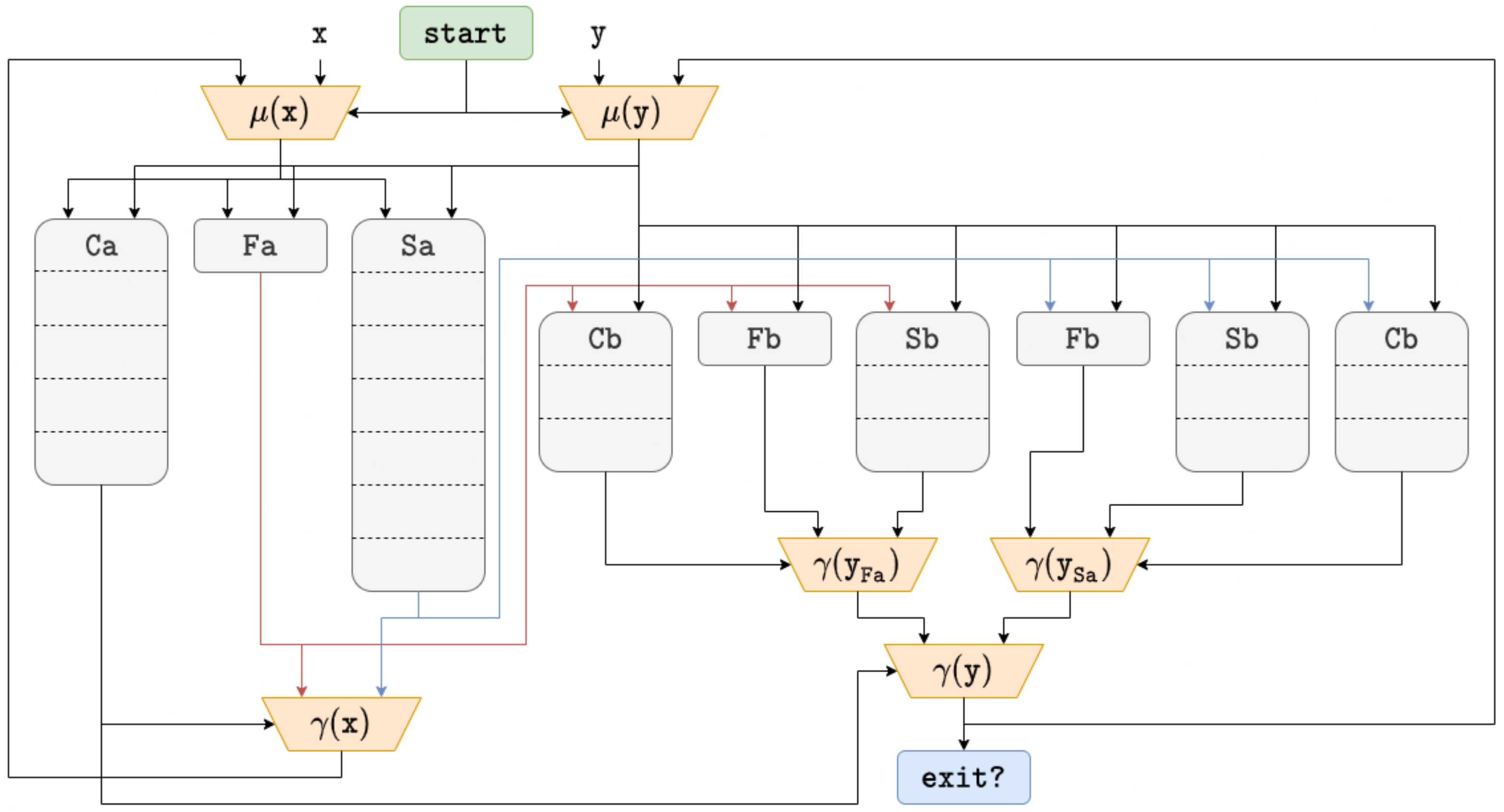
Multi-Path Speculation



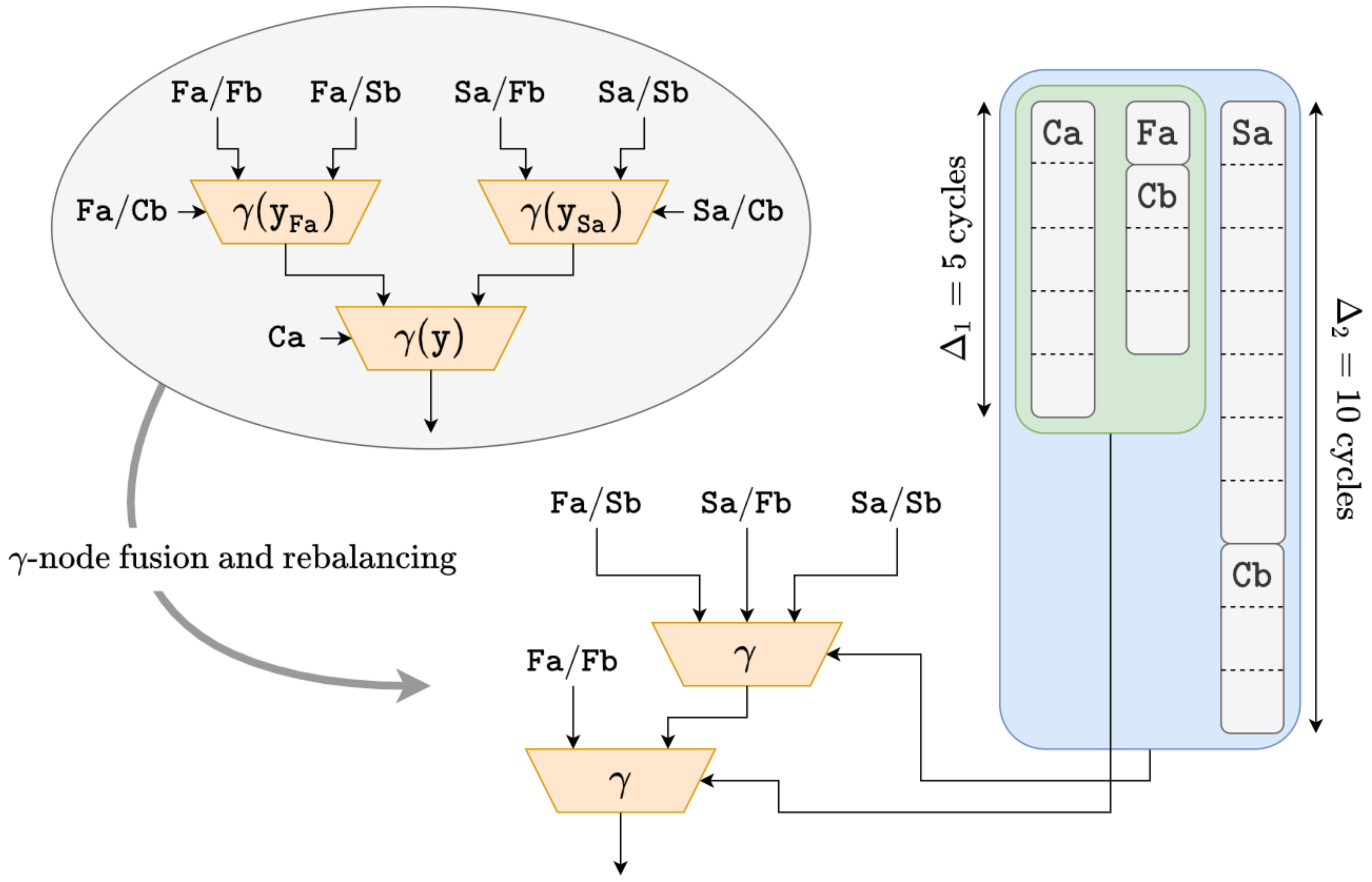


Exploring Speculation Patterns





Transforming Speculation Patterns


Fa/Fb
Fa/Sb
Sa/Fb
Sa/Sb
Fa/Cb
$\gamma(y_{Fa})$
$\gamma(y_{Sa})$
Sa/Cb
Ca
$\gamma(y)$
γ-node fusion and rebalancing
Fa/Sb
Sa/Fb
Sa/Sb
Fa/Fb
γ
γ
$\Delta_1 = 5$ cycles
Ca
Fa
Cb
Sa
Cb
$\Delta_2 = 10$ cycles