Rapid Cycle-Accurate Simulator for High-Level Synthesis (FLASH)

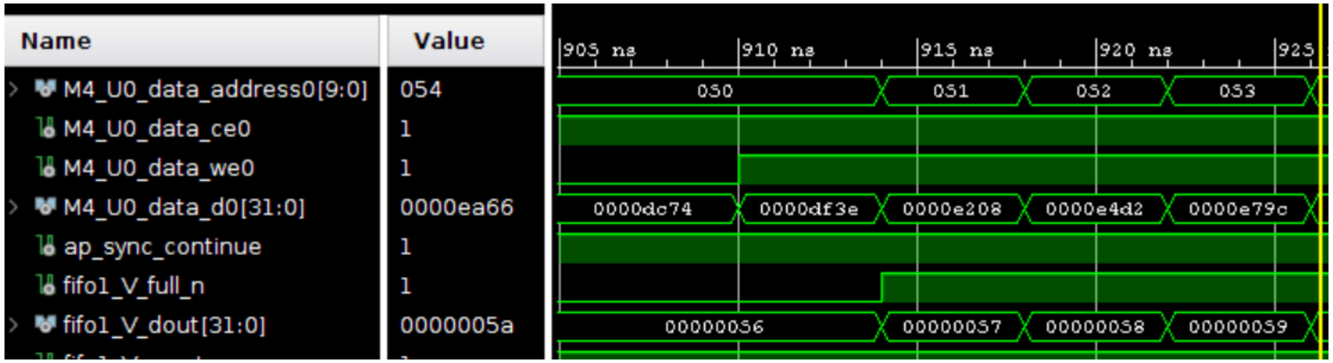
Yuze Chi, Young-kyu Choi, Jason Cong, and Jie Wang

University of California, Los Angeles

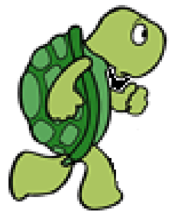
Supported by Intel and NSF Joint Research Center on
Computer Assisted Programming for Heterogeneous Architectures (CAPA)

Motivation

- RTL co-simulation for HLS



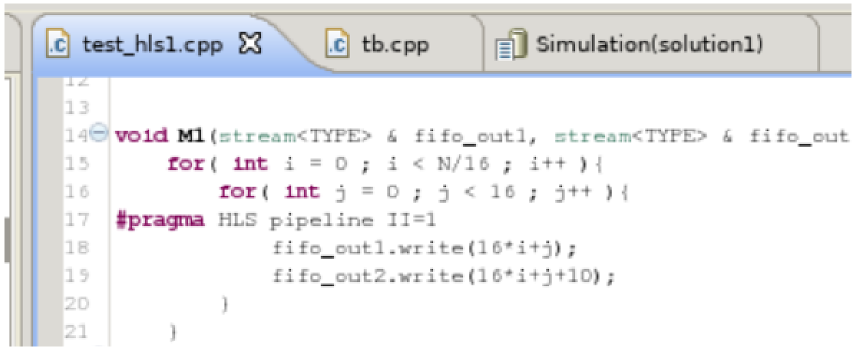
```
Name                        Value
M4_U0_data_address0[9:0]    054        050 | 051 | 052 | 053
M4_U0_data_ce0              1
M4_U0_data_we0              1
M4_U0_data_d0[31:0]         0000ea66   0000dc74 | 0000df3e | 0000e208 | 0000e4d2 | 0000e79c
ap_sync_continue            1
fifo1_V_full_n              1
fifo1_V_dout[31:0]          0000005a   00000056 | 00000057 | 00000058 | 00000059
```



Too **slow**...
(ex matmul: 192s)

Difficult to understand

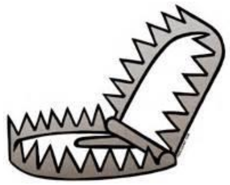
- SW simulation for HLS



```
void M1(stream<TYPE> & fifo_out1, stream<TYPE> & fifo_out
    for( int i = 0 ; i < N/16 ; i++ ){
        for( int j = 0 ; j < 16 ; j++ ){
#pragma HLS pipeline II=1
            fifo_out1.write(16*i+j);
            fifo_out2.write(16*i+j+10);
        }
    }
```

100X to 1000X faster than RTL co-sim
(ex matmul: 0.05s)

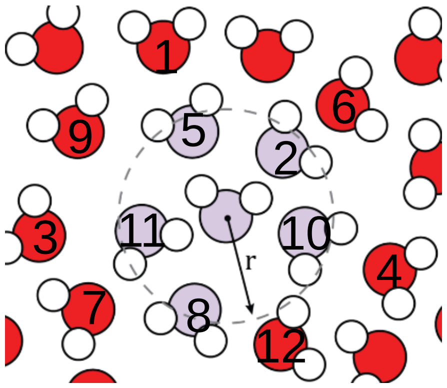
Easy to understand

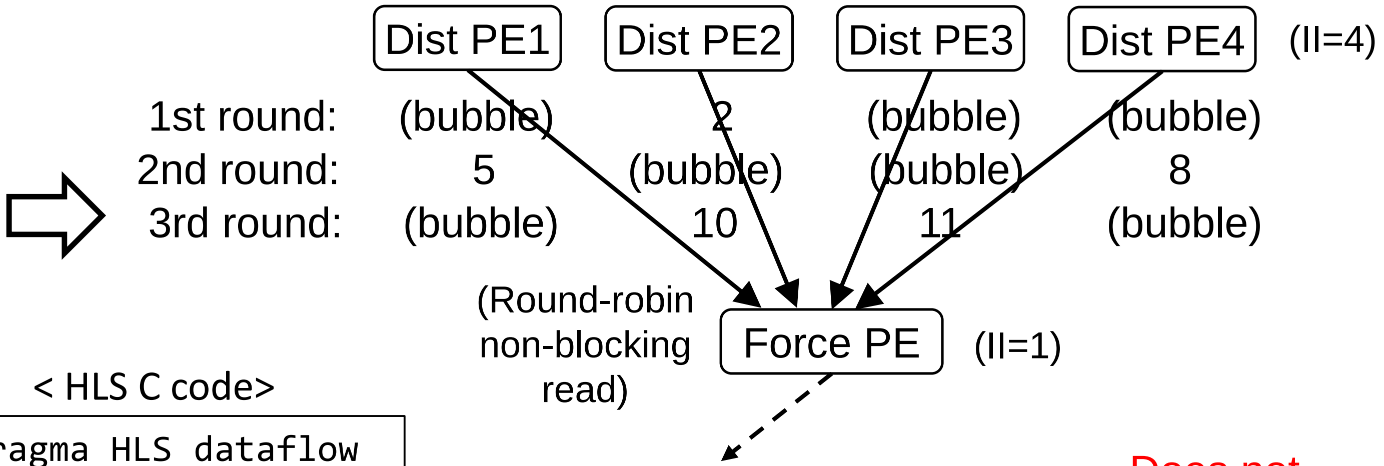


- But can it measure the execution time?
- Is it producing the ***correct*** result?

https://www.goodfreephotos.com/cache/vector-images/confused-idea-lightbulb.png
http://clipart-library.com/clipart/133840.htm
https://pixabay.com/en/light-bulb-idea-enlightenment-plan-1926533/

- HLS simulation of molecular dynamics





< HLS C code>

```
#pragma HLS dataflow
Dist_PE1();
Dist_PE2();
Dist_PE3();
Dist_PE4();
Force_PE();
```

Simulated in instantiation order → Missing bubbles

RTL sim output: 2 5 8 10 11

SW sim output: 5 2 11 8 10

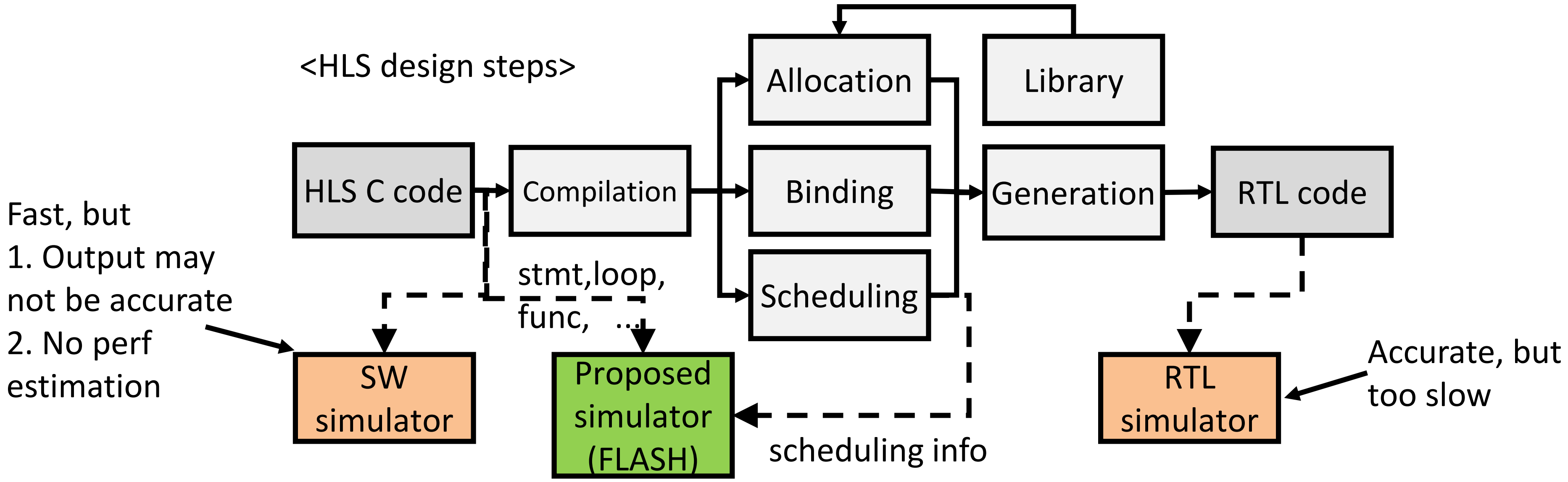
Does not match!

– Reason

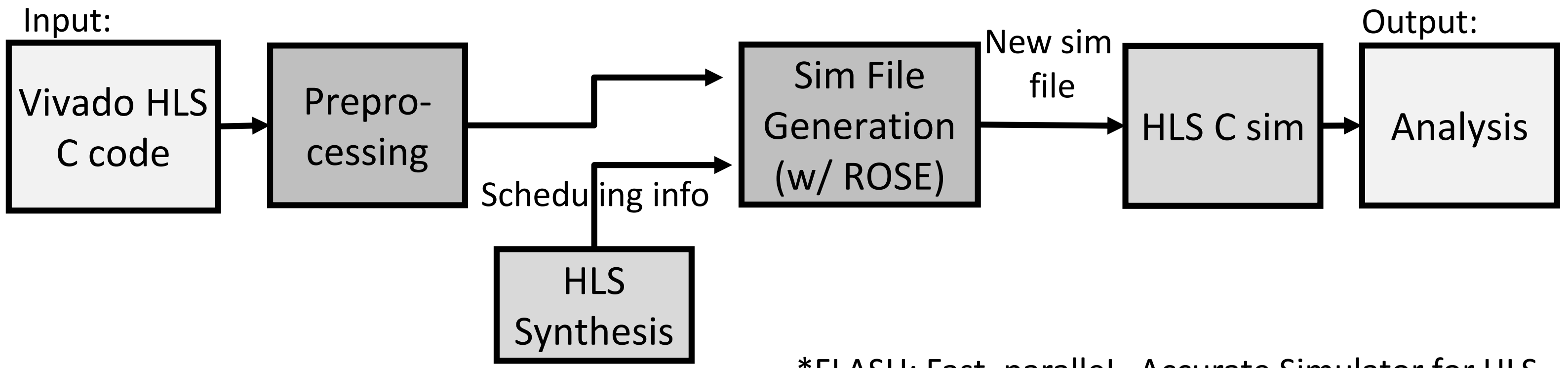
	Xilinx Viv HLS C Sim	Intel OpenCL HLS Sim
FIFO depth	**Unlimited**	Exact
Exec model	**Sequential**	Concurrent
Feedback	**Not supported**	Supported
Sim speed	~5 Mcycle/s	**~1 Mcycle/s**
Sim order	Deterministic	**Non-deterministic**
Cycle-acc	**Not cycle-accurate**	**Not cycle-accurate**

Christophe Rowley, https://en.wikibooks.org/wiki/Molecular_Simulation/Radial_Distribution_Functions

- Conventional simulation flows & proposed approach

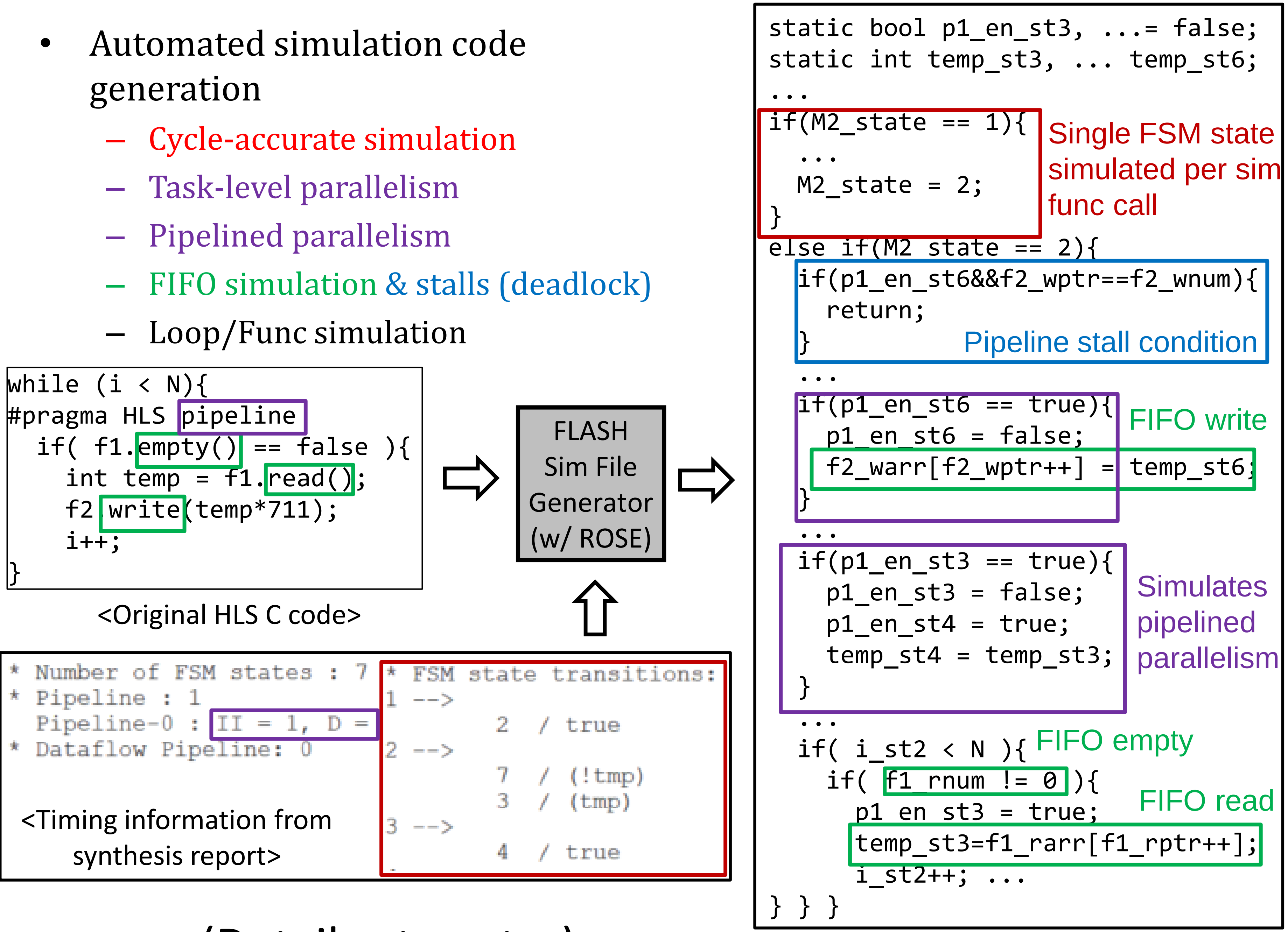


- Overall simulation framework of FLASH*



*FLASH: Fast, paralleL, Accurate Simulator for HLS

- Automated simulation code generation
  - Cycle-accurate simulation
  - Task-level parallelism
  - Pipelined parallelism
  - FIFO simulation & stalls (deadlock)
  - Loop/Func simulation



```
while (i < N){
#pragma HLS pipeline
  if( f1.empty() == false ){
    int temp = f1.read();
    f2.write(temp*711);
    i++;
}
```

<Original HLS C code>

```
* Number of FSM states : 7
* Pipeline : 1
  Pipeline-0 : II = 1, D =
* Dataflow Pipeline: 0
```

```
* FSM state transitions:
1 -->
        2  / true
2 -->
        7  / (!tmp)
        3  / (tmp)
3 -->
        4  / true
```

<Timing information from synthesis report>

```
static bool p1_en_st3, ...= false;
static int temp_st3, ... temp_st6;
...
if(M2_state == 1){
  ...
  M2_state = 2;
}
else if(M2_state == 2){
  if(p1_en_st6&&f2_wptr==f2_wnum){
    return;
  }
  ...
  if(p1_en_st6 == true){
    p1_en_st6 = false;
    f2_warr[f2_wptr++] = temp_st6;
  }
  ...
  if(p1_en_st3 == true){
    p1_en_st3 = false;
    p1_en_st4 = true;
    temp_st4 = temp_st3;
  }
  ...
  if( i_st2 < N ){
    if( f1_rnum != 0 ){
      p1_en_st3 = true;
      temp_st3=f1_rarr[f1_rptr++];
      i_st2++; ...
} } }
```

Single FSM state simulated per sim func call
Pipeline stall condition
FIFO write
Simulates pipelined parallelism
FIFO empty
FIFO read

<Transformed C code for simulation>

(Details at poster)

- Simulation time comparison

Benchmark	V C Sim	V RTL Sim	I OCL Sim	FLASH
Toy_mpath	0.602s (1.00X)	492s (817X)	4.60s (7.64X)	0.570s (0.947X)
Stencil	1.46s (1.00X)	113s (77.4X)	2.63s (1.80X)	1.25s (0.856X)
MD_sim	0.0547s (1.00X)	100s (1,830X)	0.0921s (1.68X)	0.0677s (1.24X)
Mat_mul	0.0539s (1.00X)	192s (3,560X)	0.201s (3.73X)	0.0810s (1.50X)
AVG	(1.00X)	(1,570X)	(3.71X)	(1.13X)

Deep (55) pipeline

Frequent FIFO stall
(FIFO depth=1)

The proposed simulator (FLASH):

– runs at a comparable speed with SW simulation (= 1.00X / 1.13X)

– is faster than RTL simulation by 3 orders of magnitude (=1570X/1.13X)

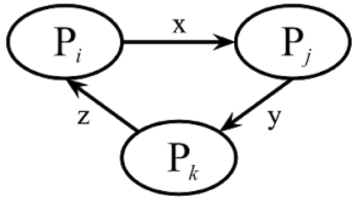
– in some cases, is faster than SW simulation (reason discussed in posters)

– has more overhead with deep pipelines or with frequent FIFO stalls

- Key take-away
  - HLS SW simulation based on the **scheduling information**
    - Can help solve the **correctness** issue and **rapidly** provide **accurate performance estimation**
  - This could substantially **decrease the validation time** of HLS tool customers

Cycle-accurate performance estimation

Correct output data



Detect deadlock situation

- We hope the presented result could motivate vendors to adopt similar approach in their HLS tools

- Thank you!

https://pixabay.com/en/dart-board-arrow-bull-s-eye-25780/
https://pixabay.com/en/correct-mark-green-continue-right-2214020/
http://www.bhanage.com/2017/02/linux-difference-deadlocks-livelocks.html