A Scalable Multi-Path Microarchitecture for Efficient GPU Control Flow

HPCA 2014

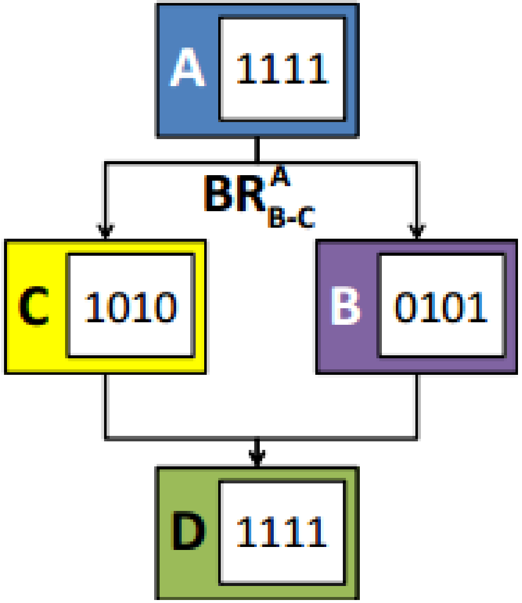
GPU for non-graphics computing

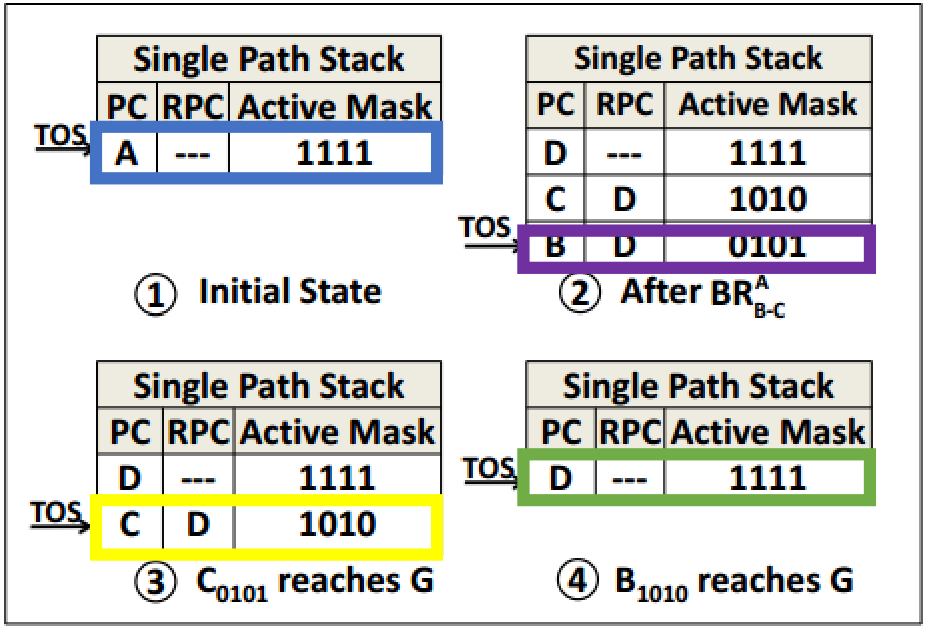
- Single Instruction Multiple Thread(SIMT) execution model
  - Use a **single** instruction sequencer to operate on **a group of** threads

- **Improve efficiency** by amortizing instruction fetch and decode cost.

- Reduce **thread level parallelism(TLP)** with **divergent control flow**
  - Current GPUs serialize the execution of divergent paths

Single-Path Stack Execution Model





	Single Path Stack		
	PC	RPC	Active Mask
TOS	A	---	1111

① Initial State

	Single Path Stack		
	PC	RPC	Active Mask
	D	---	1111
	C	D	1010
TOS	B	D	0101

② After $BR^{A}_{B\text{-}C}$

	Single Path Stack		
	PC	RPC	Active Mask
	D	---	1111
TOS	C	D	1010

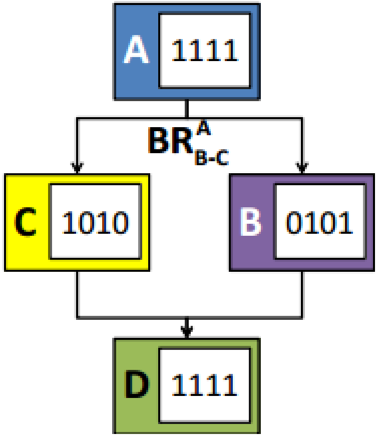
③ C_{0101} reaches G

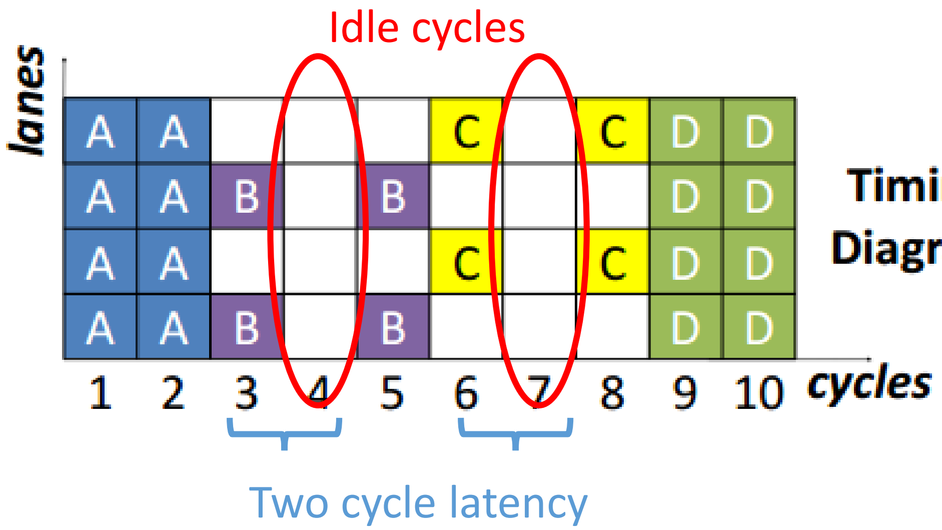
	Single Path Stack		
	PC	RPC	Active Mask
TOS	D	---	1111

④ B_{1010} reaches G

- Per warp (4 threads) stack is used to manage divergence
- Serialize the execution of diff control flow paths (B & C)
- *Immediate postdominator(IPDOM)* is set as the RPC
  - The earliest point where all divergent threads are guaranteed to execute

Stack-Based Reconvergence Limitations





1	2	3	4	5	6	7	8	9	10
A	A				C		C	D	D
A	A	B		B				D	D
A	A				C		C	D	D
A	A	B		B				D	D

Idle cycles

Two cycle latency

Timing Diagram

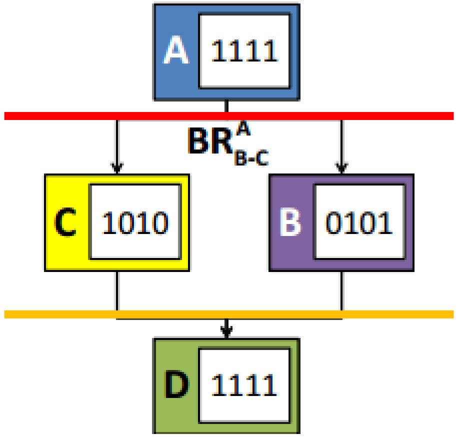
- Assume two instructions per block

- Allow only *a single* control flow path to execute at a time -> reduce the number of running threads
- Potential: during the idle cycles due to long latency, alternative paths could make progress.

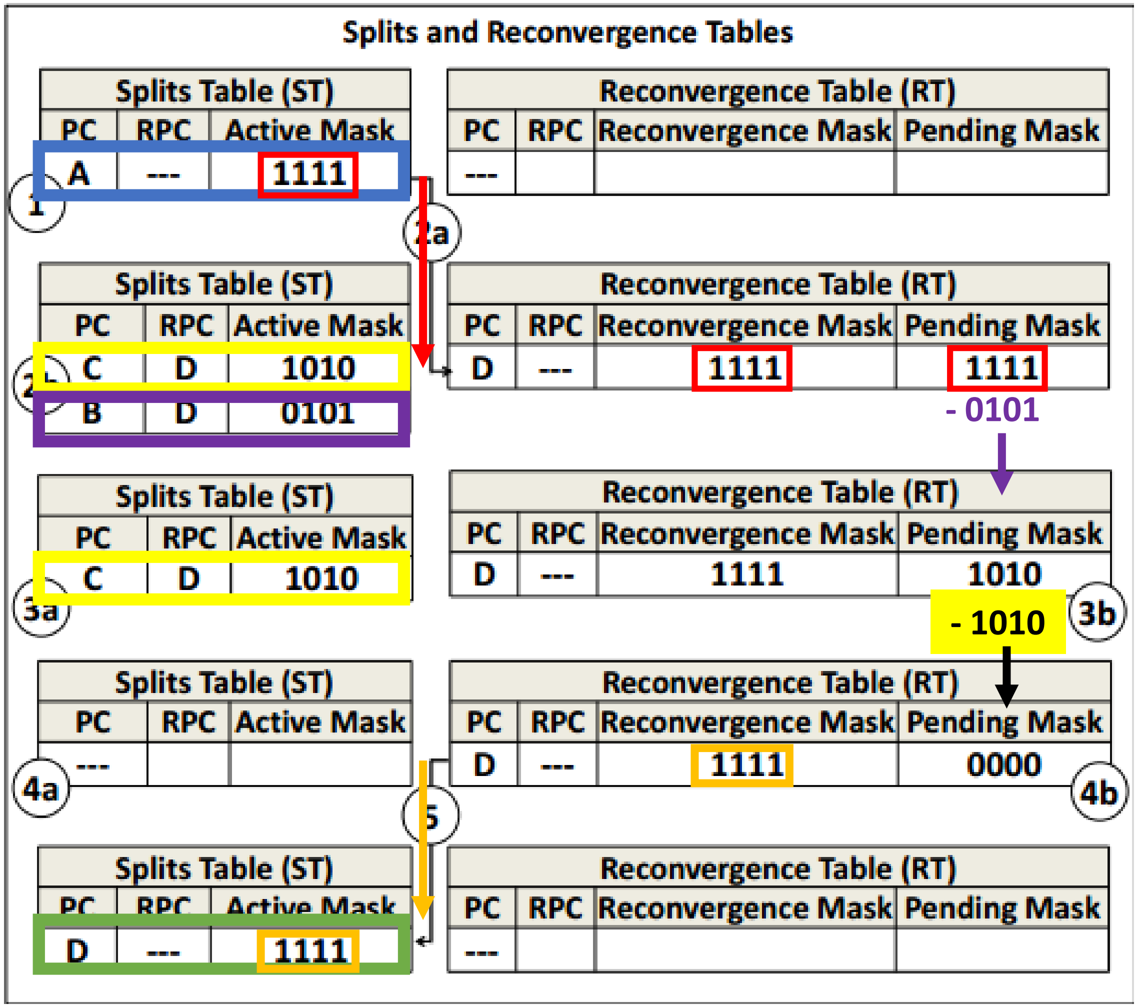
Contribution

- Goal
  - Allow concurrent scheduling of **any number** of warp splits while maintaining IPDOM reconvergence
  - Enable early reconvergence opportunistically at run-time
- Scheme
  - Warp split table (ST)
    - the state of warp splits executing in parallel basic blocks
  - Reconvergence table (RT)
    - reconvergence points for the splits

Operation of MP IPDOM



Pending mask: threads that have not yet reached the reconvergence point



Splits and Reconvergence Tables

Splits Table (ST)

PC	RPC	Active Mask
A	---	1111

Reconvergence Table (RT)

PC	RPC	Reconvergence Mask	Pending Mask

Splits Table (ST)

PC	RPC	Active Mask
C	D	1010
B	D	0101

Reconvergence Table (RT)

PC	RPC	Reconvergence Mask	Pending Mask
D	---	1111	1111

- 0101

Splits Table (ST)

PC	RPC	Active Mask
C	D	1010

Reconvergence Table (RT)

PC	RPC	Reconvergence Mask	Pending Mask
D	---	1111	1010

- 1010

Splits Table (ST)

PC	RPC	Active Mask

Reconvergence Table (RT)

PC	RPC	Reconvergence Mask	Pending Mask
D	---	1111	0000

Splits Table (ST)

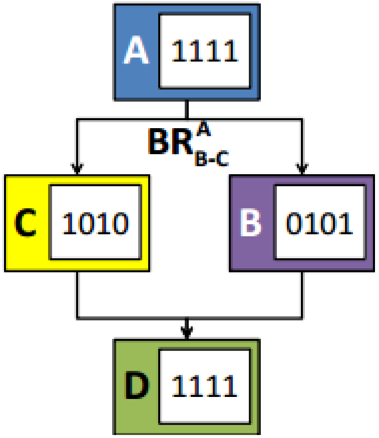
PC	RPC	Active Mask
D	---	1111

Reconvergence Table (RT)

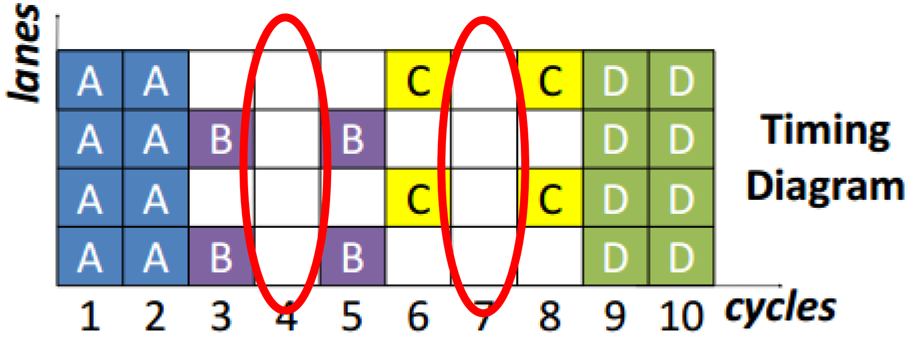
PC	RPC	Reconvergence Mask	Pending Mask

- Upon each update to the pending active mask in the RT table, the Pending Mask is checked if it is all zeros.

Timing Diagram of MP IPDOM

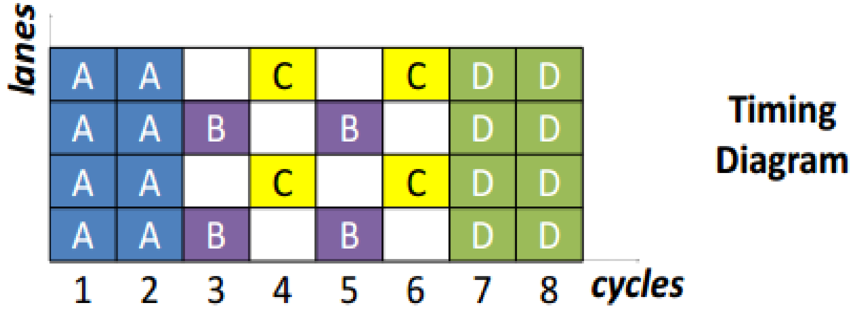


Single-Path Stack



	1	2	3	4	5	6	7	8	9	10
lanes	A	A				C		C	D	D
	A	A	B		B				D	D
	A	A				C		C	D	D
	A	A	B		B				D	D

Multi-Path Stack



	1	2	3	4	5	6	7	8
lanes	A	A		C		C	D	D
	A	A	B		B		D	D
	A	A		C		C	D	D
	A	A	B		B		D	D

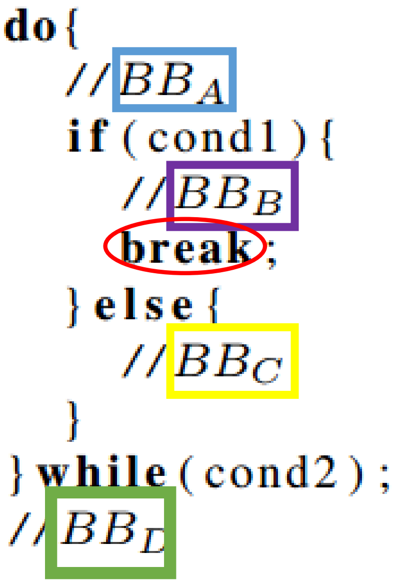
- Occupy the idle cycles with the alternative control path

Opportunistic Early Reconvergence

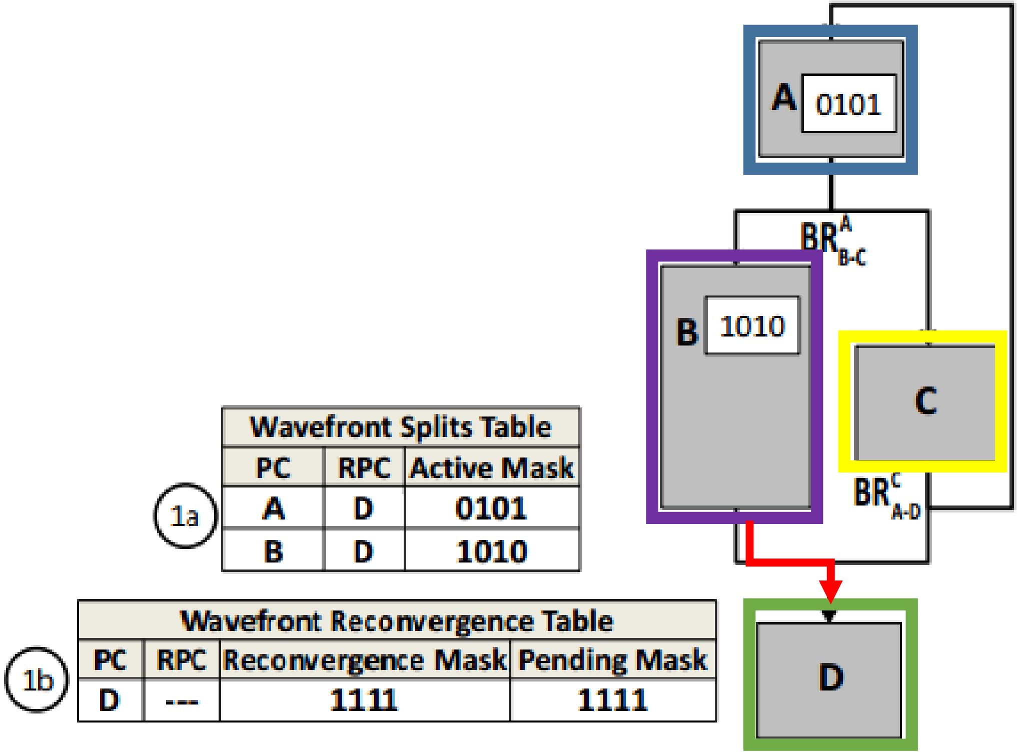
- IPDOM reconvergence point: the **earliest guaranteed** reconvergence point.

- In certain situations, there are **opportunities** to reconverge at **earlier points** than the IPDOM point.
- **Not guaranteed** for all executions.

Opportunistic Early Reconvergence



```
do{
  //BB_A
  if(cond1){
    //BB_B
    break;
  }else{
    //BB_C
  }
}while(cond2);
//BB_D
```



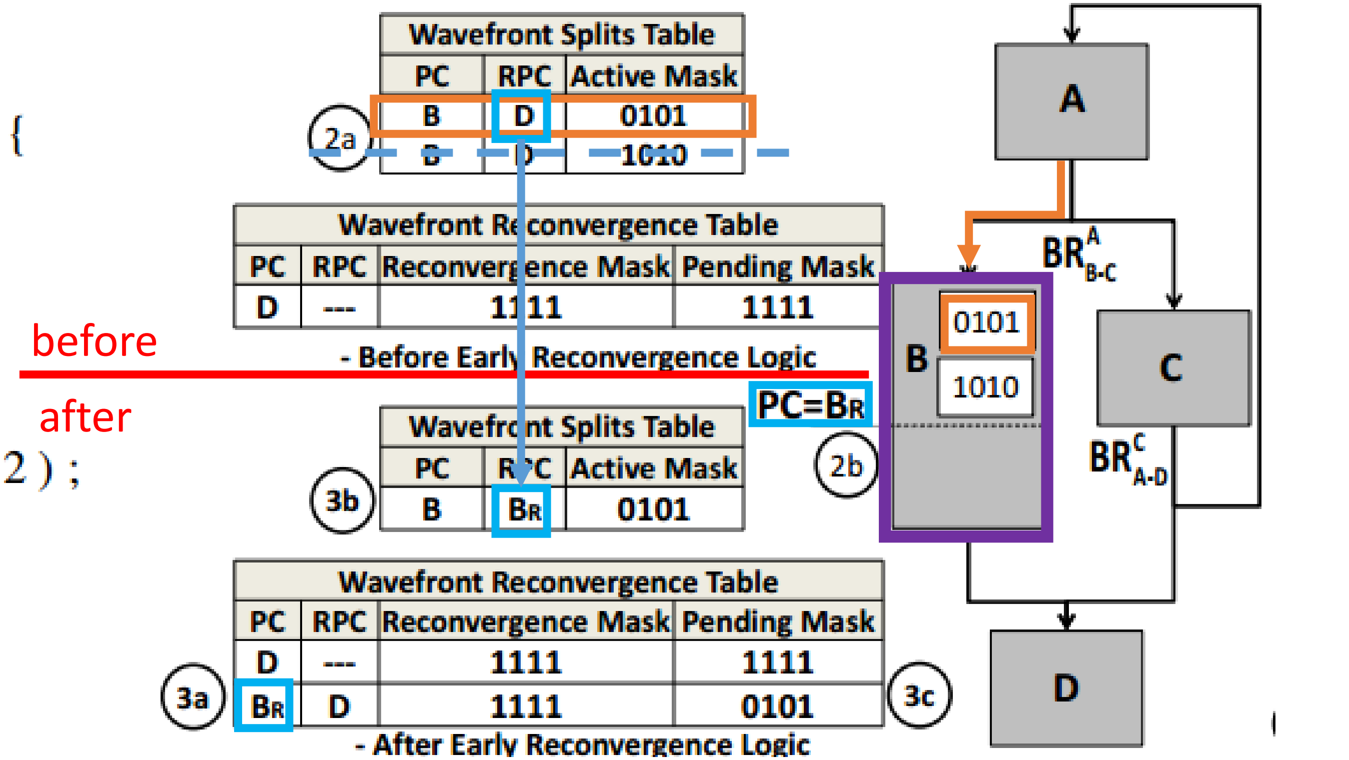
Wavefront Splits Table		
PC	RPC	Active Mask
A	D	0101
B	D	1010

Wavefront Reconvergence Table			
PC	RPC	Reconvergence Mask	Pending Mask
D	---	1111	1111

- A divergence at BR^{C}_{A-D} results in two splits B_{1010} and C_{0101}, and split C_{0101} reaches BR^{C}_{A-D} **before** split B_{1010} finishes executing basic block B

Opportunistic Early Reconvergence

```
do{
   //BB_A
   if(cond1){
      //BB_B
      break;
   }else{
      //BB_C
   }
}while(cond2);
//BB_D
```



Wavefront Splits Table		
PC	RPC	Active Mask
B	D	0101
~~D~~	~~D~~	~~1010~~

Wavefront Reconvergence Table			
PC	RPC	Reconvergence Mask	Pending Mask
D	---	1111	1111

- Before Early Reconvergence Logic

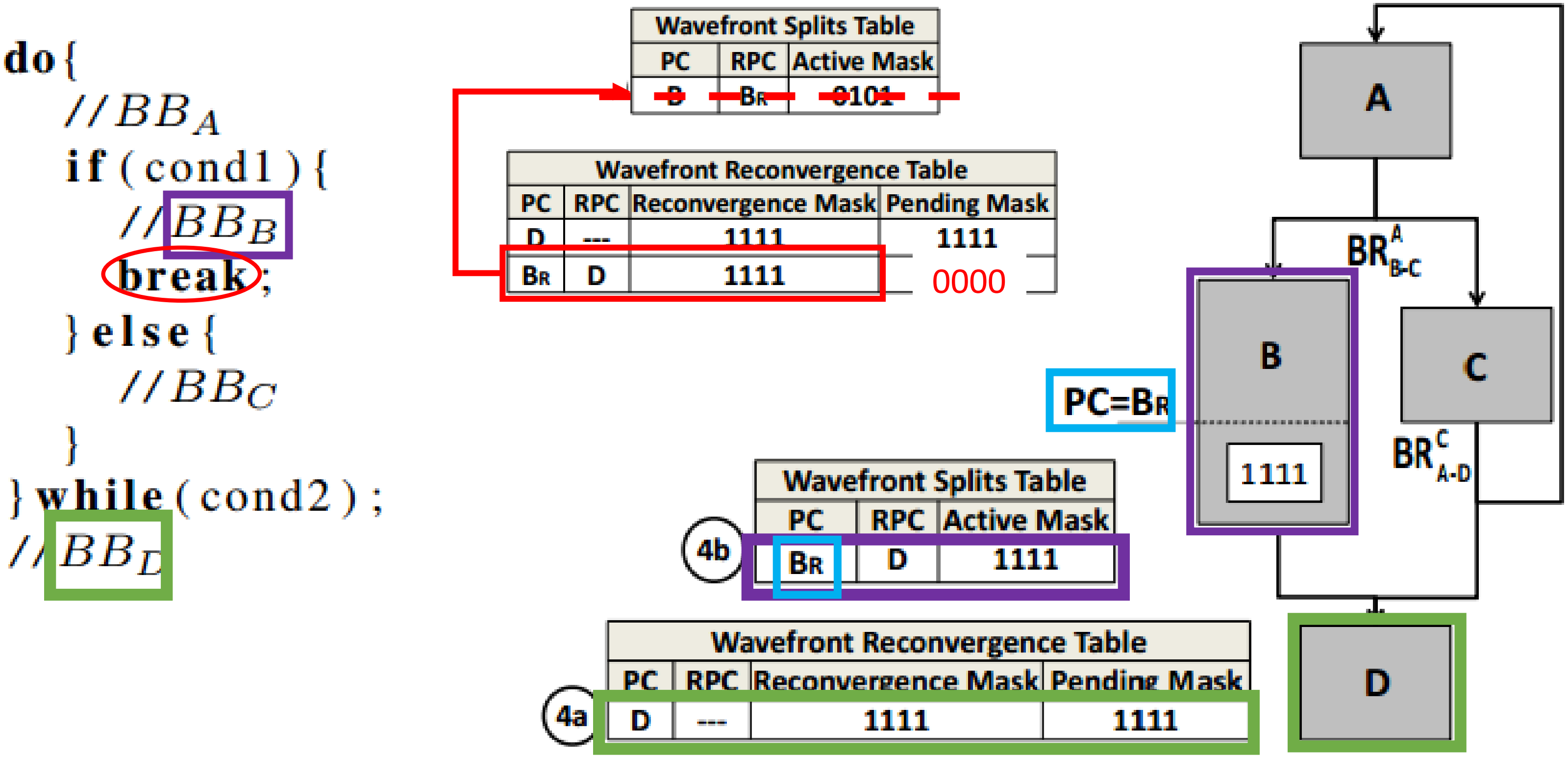
Wavefront Splits Table		
PC	RPC	Active Mask
B	BR	0101

Wavefront Reconvergence Table			
PC	RPC	Reconvergence Mask	Pending Mask
D	---	1111	1111
BR	D	1111	0101

- After Early Reconvergence Logic

- An early reconvergence opportunity: two splits (B_{0101} and B_{1010}) of the same warp executing the same basic block B
- The early reconvergence point is the program counter of the next instruction of the leading warp split (B_{1010}).

Opportunistic Early Reconvergence



```
do{
    //BB_A
    if(cond1){
        //BB_B
        break;
    }else{
        //BB_C
    }
}while(cond2);
//BB_D
```

Wavefront Splits Table		
PC	RPC	Active Mask
~~B~~	~~BR~~	~~0101~~

Wavefront Reconvergence Table			
PC	RPC	Reconvergence Mask	Pending Mask
D	---	1111	1111
BR	D	1111	0000

4b

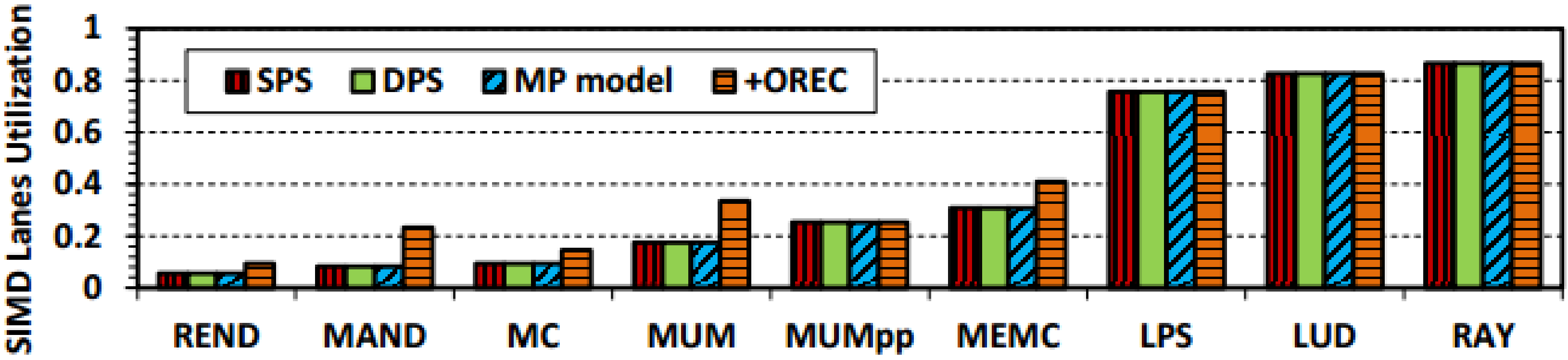
Wavefront Splits Table		
PC	RPC	Active Mask
BR	D	1111

4a

Wavefront Reconvergence Table			
PC	RPC	Reconvergence Mask	Pending Mask
D	---	1111	1111

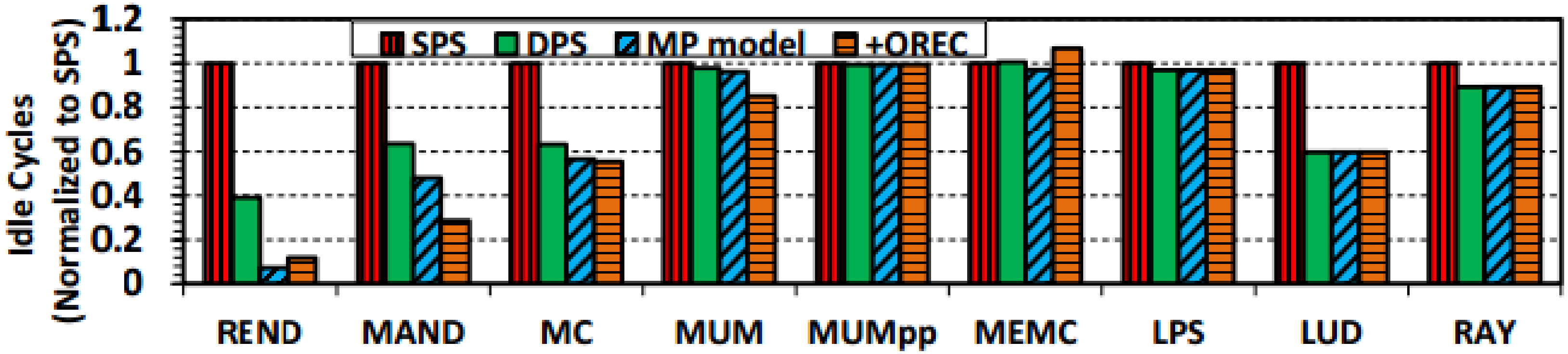
- Warp split B_{0101} reaches the early reconvergence point B_R
- B_{0101} ‘s entry in the ST is invalidated
- Pending mask of the reconvergence entry B_{R1111} is updated
- The reconvergence entry B_{R1111} moves from the RT to the ST

SIMD Unit Utilization

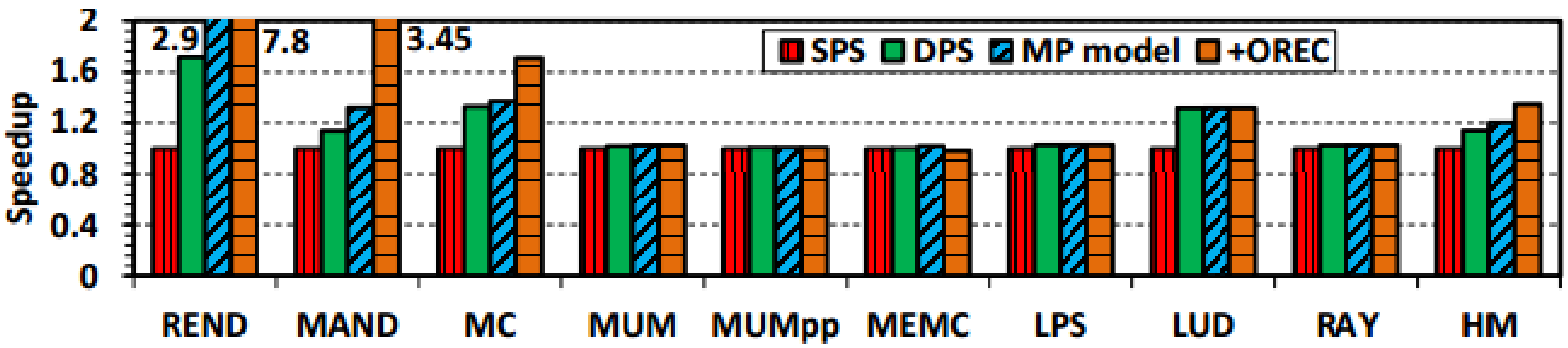


- Same SIMD unit utilization for SPS, DPS and basic MP: all reconverge at the IPDOM reconvergence points
- Opportunistic rec. opt: an average of 48% and up to 182%

Idle Cycles



Overall Performance



- The speedup comes mainly from:
    - Reduced idle cycles (i.e., more warp split instructions per cycle)
    - Improved SIMD units' utilization (i.e., more throughput per warp split instructtion).
- MP with opportunistic reconvergece has harmonic mean speedup of:
    - 32% over the SPS model
    - 18.6% over the basic MP
    - 12.5% over DPS models

Conclusions

- Propose a novel mechanism that enables multi-path execution in GPUs. Achieved by two tables:
    - One tracks the concurrent executable paths upon every branch
    - The other tracks the reconvergence points of these branches
- Modify the proposed model to enable opportunistic early reconvergence at run time
- Evaluations show 32% speedup over conventional single-path SIMT execution