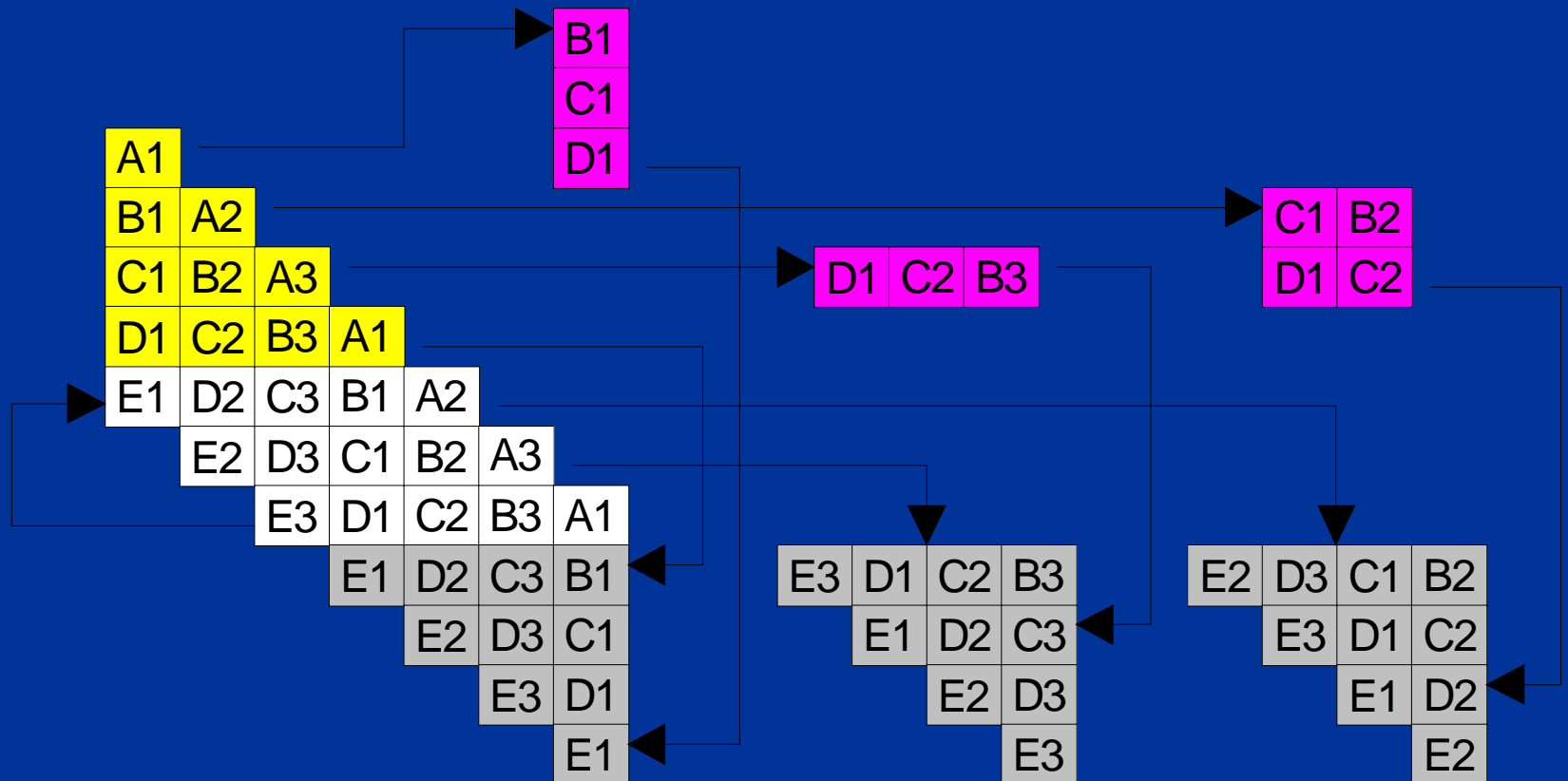

Hardware Support for Modulo Scheduling

Reducing Code Size

Multiple Epilogues



Cydra-5 Rotating Registers

- The register file is extended into a 2-D structure
 - The original registers span the vertical direction
 - Each register is extended in the horizontal direction
 - A current pointer points to the column that serve as the current registers for assignments
 - The use of registers carry an offset in the horizontal direction, into a past version of the register
 - A special loop branch is inserted at the end of each software pipeline stage

Itanium Rotating Register File

- Predicate registers Pr16-Pr63
- Floating-Point registers Fr32-Fr127
- General purpose registers any multiples of 8 starting at Gr32
- At each stage of a modulo scheduled loop, each set of rotating registers undergo implicit renaming
 - Register N from the previous stage becomes register N+1 in the current stage

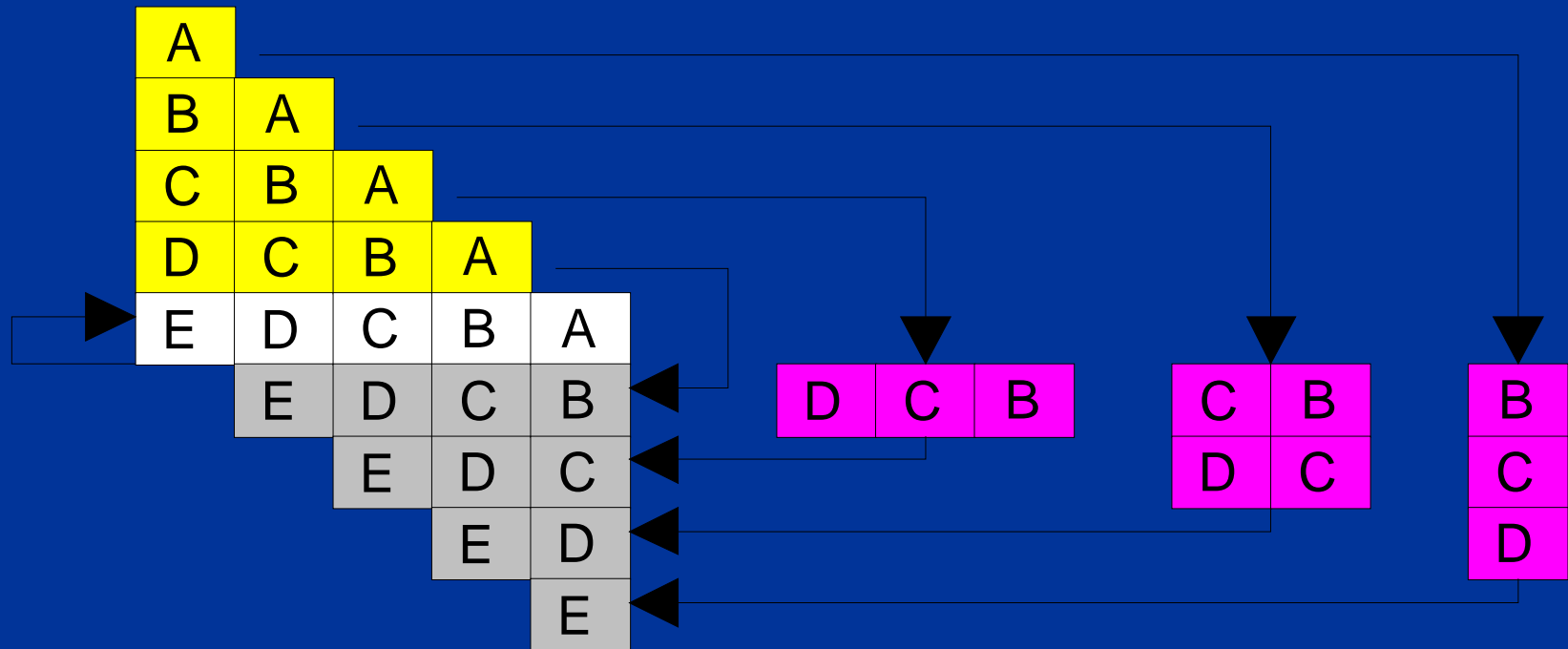
Itanium Rotation Operation

- Each rotating register set is accessed through its own Rotating Register Base (rrb)
- The physical register number used is the sum of the register number in the instruction plus rrb
- At the end of each stage, rrb is decremented by one
 - $rrb+N$ in the current stage is the same as $(rrb-1)+(N+1)$ in the next stage

Usage model

- Each live range occupies K consecutive registers if it spans K stages in the schedule
 - Say Gr32, Gr33, Gr34 if the live range spans three stages.
 - Add up the number of consecutive registers needed by all live ranges; allocate the next smallest multiple of 8
- Use a special loop branch to bump rrb at the end of every stage of pipeline

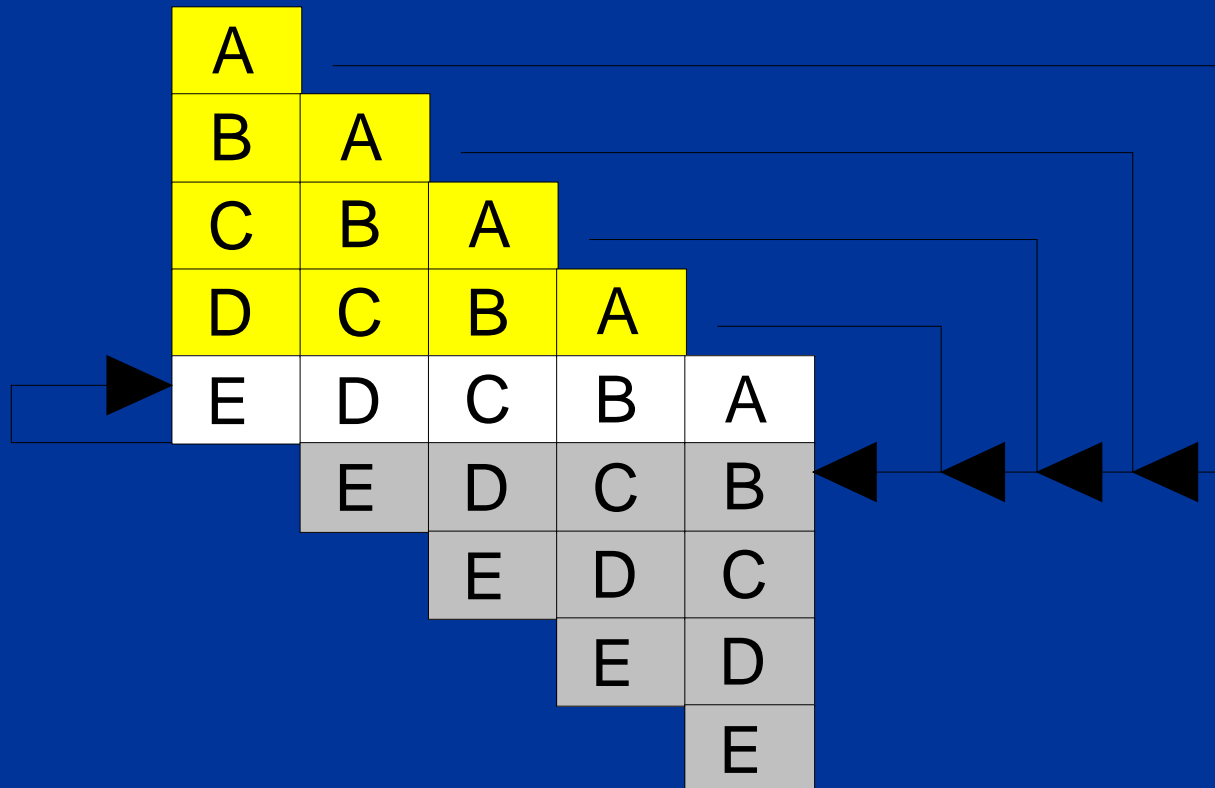
With Rotating Registers



Rotating Predicate Registers

- In Cydra-5, instructions in the software pipelined loops are guarded with a predicated register with a rotation offset
 - Instructions in stage A have offset 0, stage B offset 1, Stage C offset 2, etc.
 - Before entering the loop, initialize the current predicate register to 1 and all past versions to 0's (10000000)
 - The special loop branch instruction at the end of each stage sets the next current to 1 (0) and bump the current pointer if there are more iterations (if no more iterations).

With Rotating Registers and Predicated Execution



Itanium Rotating Predicates

- Rotating predicate registers start at p16.
 - P63 becomes p16 after rotation
- Related software pipeline support
 - ar.lc (loop count, initialized to N-1, N is total trip count)
 - ar.ec (epilog count, initialized to total number of stages in the pipeline)

Itanium rotating predicates

- Instructions at each stage uses a different predicate
 - A: p16, B: p17, C: p18, etc.
- Initialize `ar.lc`, `ar.ec`, and rotating predicates before entering loop
 - P16 to 1, all others 0

Special loop branch effect

	Ar.lc	Ar.ec	P63	rrb
Prolog	dec	same	1	dec
Kernel	dec	same	1	dec
Epilog	0	dec	0	dec

Itanium rotating predicate

operation example (5 stages, 2 iterations)

stage	ar.lc	ar.ec	p16	p17	p18	p19	p20
0	1	5	1	0	0	0	0
1	0	5	1	1	0	0	0
2	0	4	0	1	1	0	0
3	0	3	0	0	1	1	0
4	0	2	0	0	0	1	1
5	0	1	0	0	0	0	0

Kernel Only Code



Other hardware support

- See
 - M. C. Merten and W. W. Hwu ,
“Modulo Schedule Buffers,”
Proceedings of the 34th International
Symposium on Microarchitecture,
December, 2001

For a different type of modulo scheduling
hardware support for very low cost,
low power VLIW processors