EE-577B Project - Phase 2
Gold Variable-Width Processor

Andrew Adam
Harsh Bhardwaj

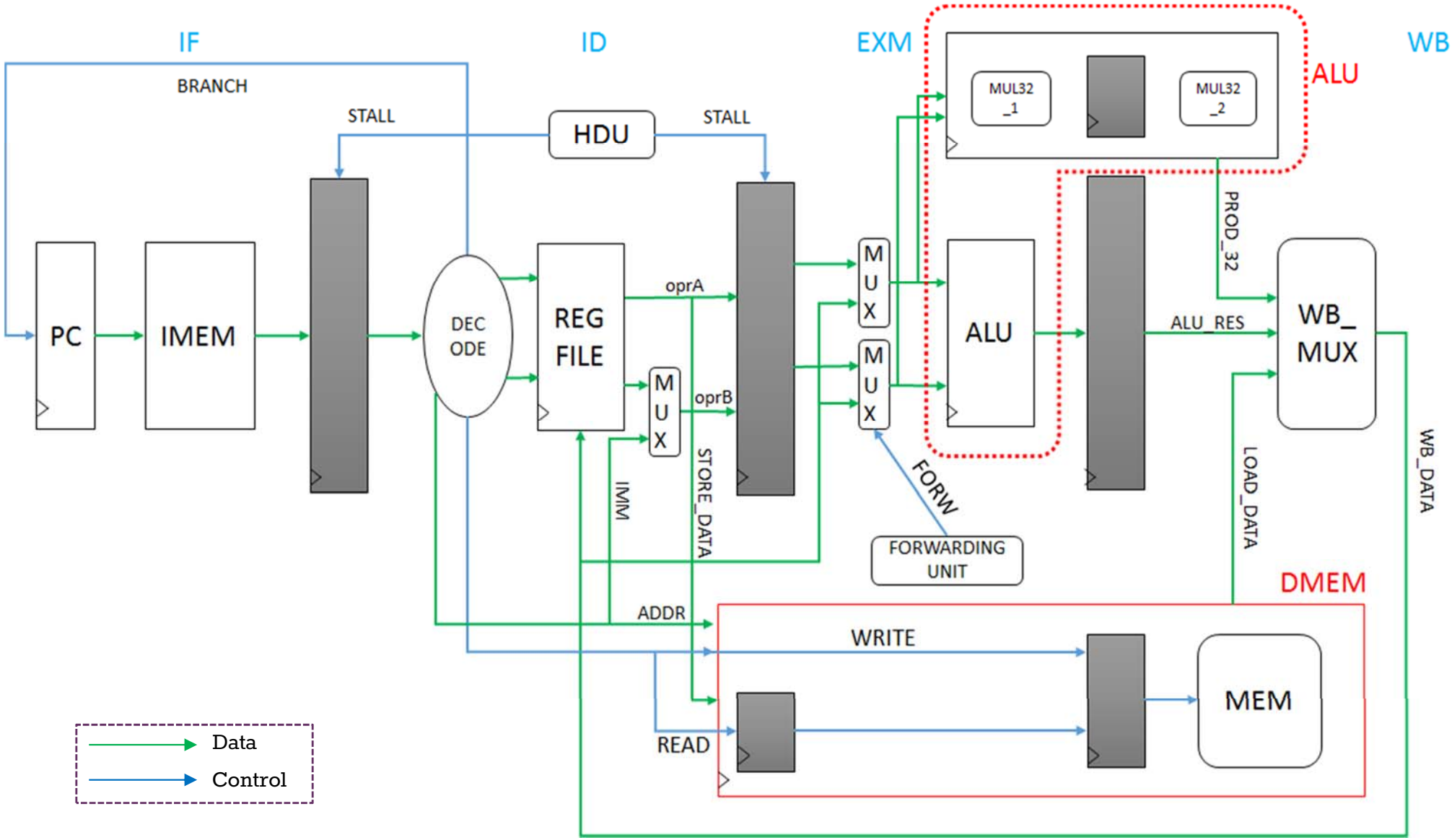
Agenda

- Overview of design
  - Division of work
- Processor Pipeline
  - Register File
  - Stalling
  - Forwarding
  - Branches
- ALU
  - Shifters
  - Multipliers
- Testing & Synthesis
- Q & A

Overview

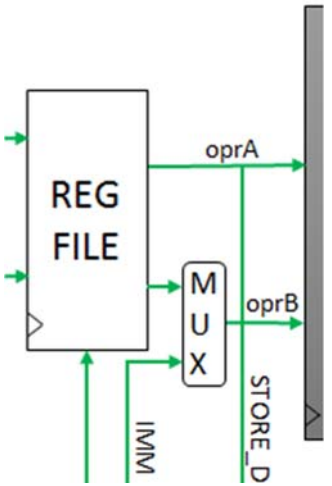
- Four-stage variable-width processor
- Flow
  - Design → RTL → Testing → Synthesis
- Division of work
  - Andrew
    - Logical & Shift instructions
    - Memory instructions
    - Register file
    - Program counter
  - Harsh
    - Arithmetic instructions
    - Branch instructions
    - Perl compiler script
  - Both
    - Pipeline structure and components
    - ALU structure

High-level Block Diagram



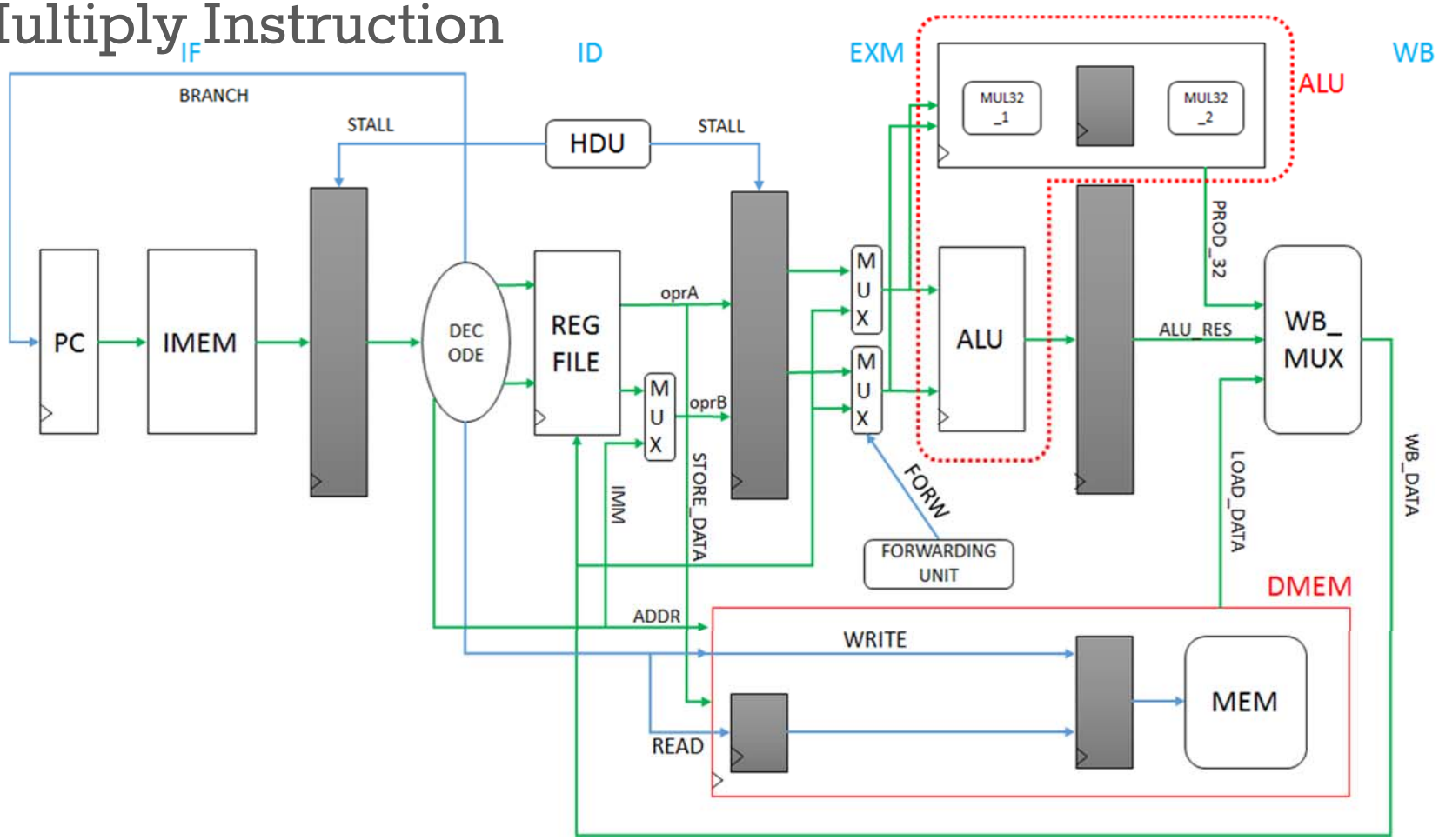
Processor Pipeline - Register File

- Tradeoff between 1-D [width*depth] vs 2-D reg file [width][depth]
- Key Features:
  - Internally forward if write address = read address
  - Participation fields used in reg file
    - 1) Ex stage calculation assumes all fields participate
    - 2) Reg file - get value of current reg file
    - 3) Calculate what to write back based on new write data, current reg, and PPP
    - Each instruction therefore takes 1 cycle for execution (2 cycles for multiply)



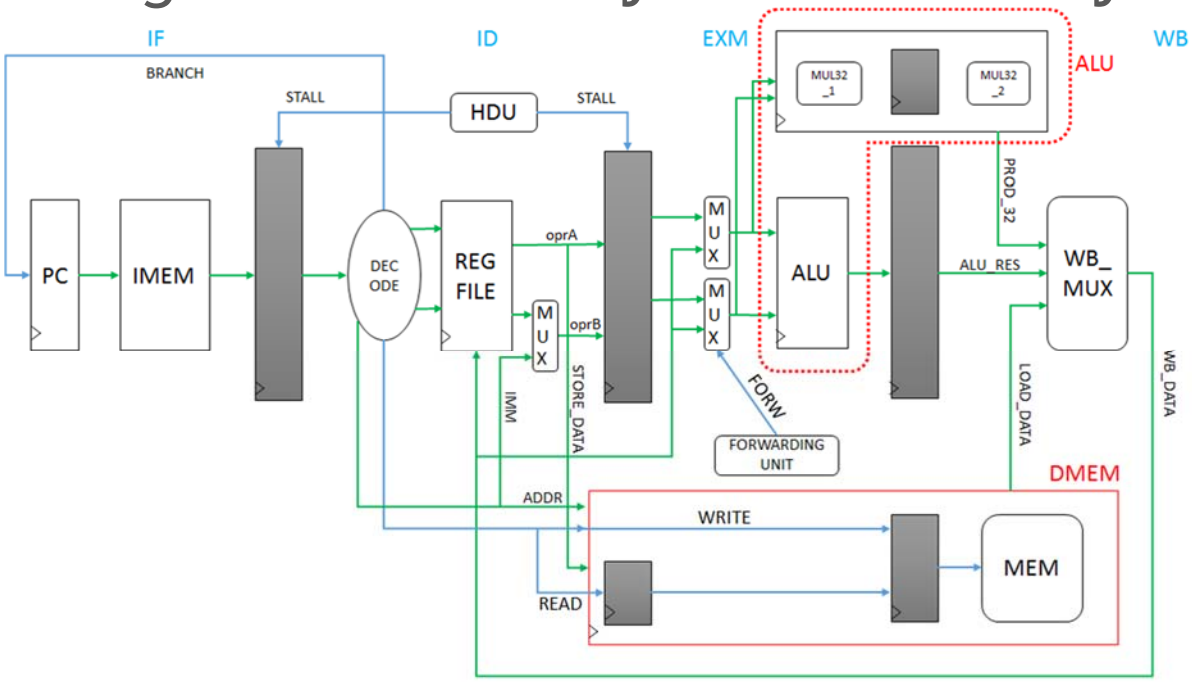
Processor Pipeline - Stalling

- Store Instruction
- Branch Instruction
- Multiply Instruction



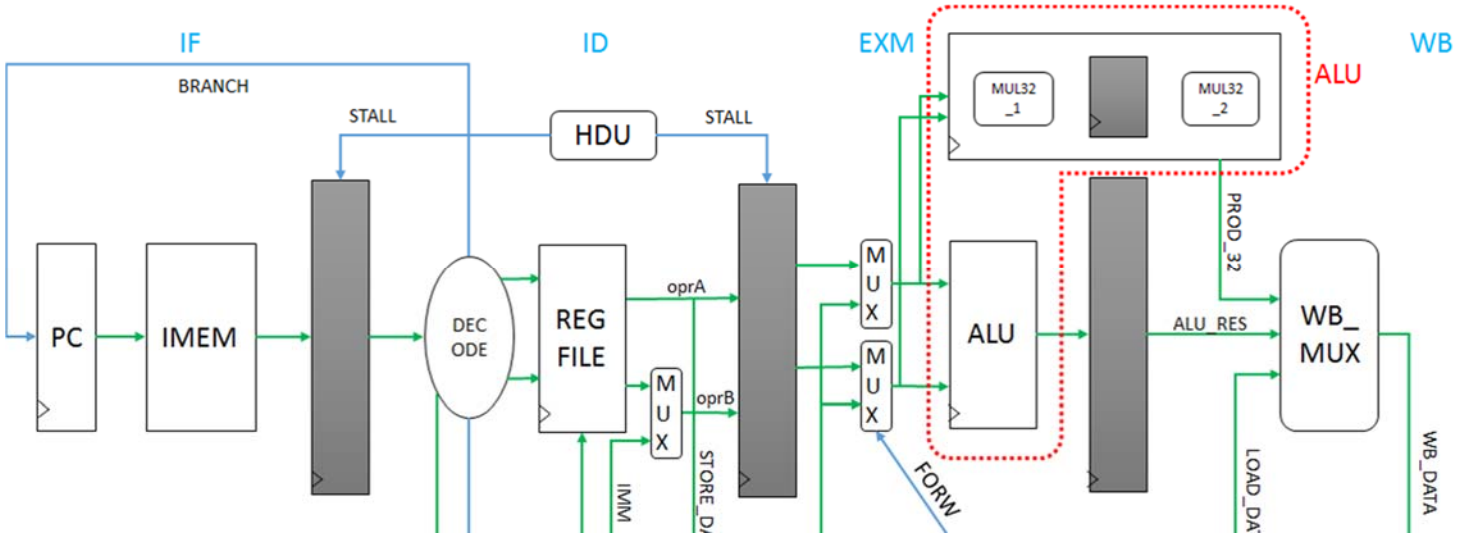
Processor Pipeline - Forwarding

- Forward according to PPP value when:
  - Write back stage destination register file = EX stage stage destination register file
  - Register write is enabled in both EX and WB stages
  - Applies for both R-type and Load instructions
- When stalling - forward only on lst clock cycle of stall



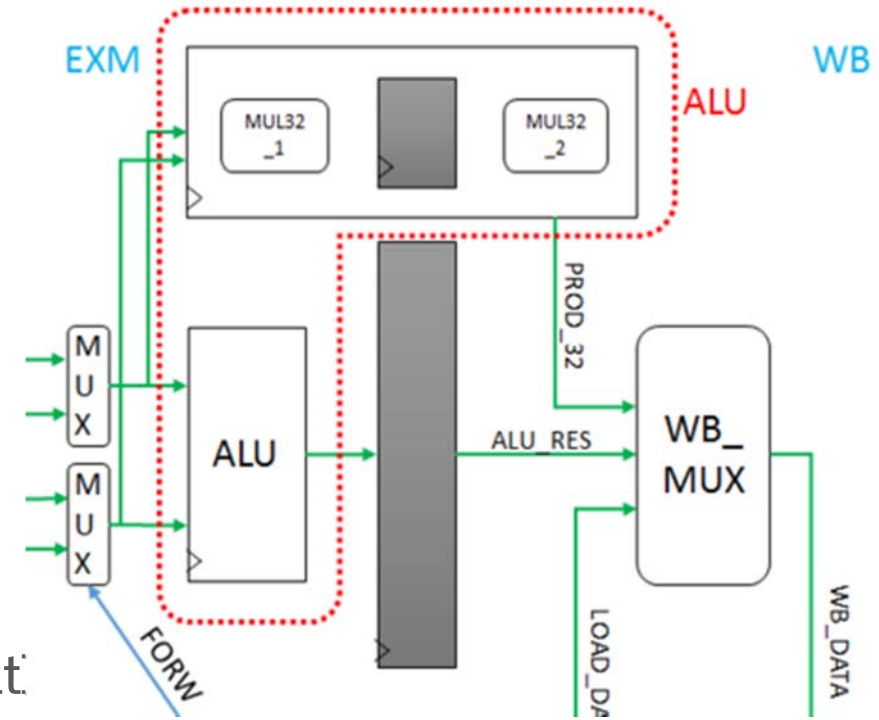
Processor Pipeline - Branches

- ID stages includes a comparator
  - Determines if the register file address read is zero or not zero
- The result of this ID branch signal is used to load the PC if stall if not active
- If current branch in ID stage is successful, tag the instruction in IF stage to be flushed



ALU

- Logical Operations
  - AND, OR, XOR, NOT, MOV
- Arithmetic Operations
  - ADD, SUB
  - Multiplication
- Shift Operations
  - Rotate
  - Logical Shift
  - Arithmetic Shift
- WW field determines operand widt
  - Not considered for logical operations
- Two-cycles for 32-bit multiplication
  - One-cycle operation for all other instructions



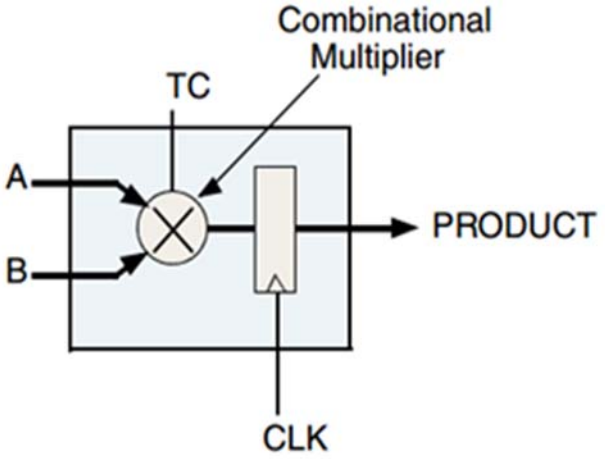
ALU - Shifters

- Logical Shifting
  - Left- and right-shift
  - Eight 8-bit shifters
  - Four 16-bit shifters
  - Two 32-bit shifters
  - One 64-bit shifter
- Arithmetic Shifting
  - Only right-shift
- Shift amount - Immediate or register-value

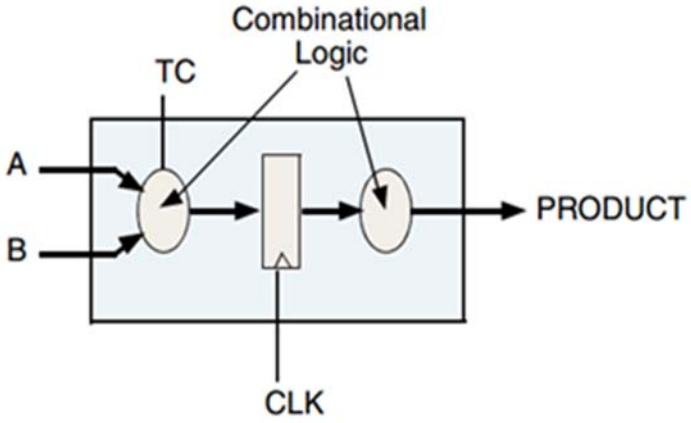
- Coding style for arithmetic-shifting
  - $signed
    - Type-cast function
    - Sign-extension
    - Synthesizable
  - “>>>”
    - Verilog arithmetic right-shift operator
  - result = $signed(oprA) >>> oprB[58:63];

ALU - Multipliers

- 8-bit Multipliers
  - Four 8x8 multipliers
  - Byte multiplication
- 16-bit Multipliers
  - Two 16x16 multipliers
  - Halfword multiplication
- 32-bit Multiplier
  - Word multiplication
  - One 2-stage pipelined DesignWare multiplier instance
    - DW02_mult_2_stage
    - Implementation - Booth-recoded Wallace-tree
- Even and Odd multiplication
- Square Operations



(a) Before pipeline retiming
(before compile)



(b) After pipeline retiming
(after compile)

Fig. - DW02_mult_2_stage block diagram
(source - Synopsys DesignWare document library)

Testing & Synthesis

- Test methodology
  - Write assembly code
  - Perl script to convert to binary
  - Observe waveforms and DMEM dump file
  - Separate test-benches for register-file, ALU
  - Test-cases
    - ALU operations
    - Load and Store
    - Branches
    - Stalling & Forwarding
- Synthesis
  - Meets 4 ns clock-period requirement
  - 2.4 ns cycle time possible too!
  - report_resources command

Q & A