ASD Assignment – constraints

Rohan M.

1602-22-735-093

1. Randomized Meal Generator

Question: Generate random meals with the following constraints:

The meal must have exactly 1 main dish, 1 side dish, and 1 dessert.

A vegetarian main dish cannot be paired with a non-vegetarian side dish.

The total calorie count must be less than 1500.

Code :

```
class meal_gen;
 typedef enum {Veg = 1, Non_Veg = 2} food_type;

 rand int main_dish_no;
 rand int side_dish_no;
 rand int dessert_no;
 rand int main_dish_calories;
 rand int side_dish_calories;
 rand int dessert_calories;
 rand food_type main_dish_type;
 rand food_type side_dish_type;
 int total_calories;

 constraint dish_count {
  main_dish_no == 1;
  side_dish_no == 1;
  dessert_no == 1;
}

constraint calorie_ranges {
  main_dish_calories inside {[200:800]};
  side_dish_calories inside {[100:500]};
  dessert_calories inside {[150:400]};
}

constraint total_calories_constraint {
  main_dish_calories + side_dish_calories + dessert_calories < 1500;
}

constraint veg_compatibility {
  (main_dish_type == Veg) -> (side_dish_type == Veg);
}

function void total_calories_sum();
  total_calories = main_dish_calories + side_dish_calories + dessert_calories;
  $display(" Total calories: %0d", total_calories);
endfunction

function void debug();
  if (!this.randomize()) begin
    $display("Randomization failed due to conflicting constraints.");
  end else begin
    $display("Randomization succeeded!");
  end
  endfunction
endclass


module meal_check;
  initial begin
    meal_gen meal = new();
    for (int i = 0; i < 5; i++) begin
      meal.debug(); // Debug function to handle randomization
      $display("\nMeal %0d:", i + 1);
      $display(" Number of main dishes: %0d", meal.main_dish_no);
      $display(" Number of side dishes: %0d", meal.side_dish_no);
      $display(" Number of desserts: %0d", meal.dessert_no);
      $display(" Type of main dish (1 = veg; 2 = Non-Veg): %0d", meal.main_dish_type);
      $display(" Type of side dish (1 = veg; 2 = Non-Veg): %0d", meal.side_dish_type);
      $display(" Main dish calories: %0d", meal.main_dish_calories);
      $display(" Side dish calories: %0d", meal.side_dish_calories);
      $display(" Dessert calories: %0d", meal.dessert_calories);
      meal.total_calories_sum();
    end
  end
endmodule
```

Output :

```
CPU time: .249 seconds to compile + .246 seconds to elab + .243 seconds to link
Chronologic VCS simulator copyright 1991-2023
Contains Synopsys proprietary information.
Compiler version U-2023.03-SP2_Full64; Runtime version U-2023.03-SP2_Full64;  Apr  7
11:10 2025
```

```
Randomization succeeded!

Meal 1:
 Number of main dishes: 1
 Number of side dishes: 1
 Number of desserts: 1
 Type of main dish (1 = veg; 2 = Non-Veg): 2
 Type of side dish (1 = veg; 2 = Non-Veg): 2
 Main dish calories: 508
 Side dish calories: 467
 Dessert calories: 332
 Total calories: 1307
Randomization succeeded!

Meal 2:
 Number of main dishes: 1
 Number of side dishes: 1
 Number of desserts: 1
 Type of main dish (1 = veg; 2 = Non-Veg): 1
 Type of side dish (1 = veg; 2 = Non-Veg): 1
 Main dish calories: 635
 Side dish calories: 136
 Dessert calories: 241
 Total calories: 1012
Randomization succeeded!

Meal 3:
 Number of main dishes: 1
 Number of side dishes: 1
 Number of desserts: 1
 Type of main dish (1 = veg; 2 = Non-Veg): 2
 Type of side dish (1 = veg; 2 = Non-Veg): 1
 Main dish calories: 633
 Side dish calories: 234
 Dessert calories: 217
 Total calories: 1084
Randomization succeeded!

Meal 4:
 Number of main dishes: 1
 Number of side dishes: 1
 Number of desserts: 1
 Type of main dish (1 = veg; 2 = Non-Veg): 2
 Type of side dish (1 = veg; 2 = Non-Veg): 2
 Main dish calories: 242
 Side dish calories: 399
 Dessert calories: 359
 Total calories: 1000
Randomization succeeded!

Meal 5:
 Number of main dishes: 1
 Number of side dishes: 1
 Number of desserts: 1
 Type of main dish (1 = veg; 2 = Non-Veg): 2
 Type of side dish (1 = veg; 2 = Non-Veg): 2
 Main dish calories: 477
 Side dish calories: 399
 Dessert calories: 348
 Total calories: 1224
           V C S   S i m u l a t i o n   R e p o r t
Time: 0 ns
CPU Time:      0.260 seconds;       Data structure size:   0.0Mb
```

Movie Schedule Randomizer

Question: Generate a random movie screening schedule with the following constraints:

No two movies should overlap in the same screening room.

Each movie must have at least one evening slot (after 6 PM).

Comedy movies must not be scheduled before 12 PM.

Code:

```
class movie_screening;

  rand int screen[5];

  rand int start_time[5];

  rand int duration[5];

  rand bit comedy[5];
  constraint room_no_overlap {
    foreach (screen[i]) {
      screen[i] inside {[1:3]};
      start_time[i] inside {[9:22]};
      duration[i] inside {[1:3]};

      foreach (screen[j]) {
        if (i != j && screen[i] == screen[j]) {
          (start_time[i] + duration[i] <= start_time[j]) ||
          (start_time[j] + duration[j] <= start_time[i]);
        }
      }
    }
  }

  constraint evening_slot {
    (start_time[0] >= 18) || (start_time[1] >= 18) ||
    (start_time[2] >= 18) || (start_time[3] >= 18) ||
    (start_time[4] >= 18);
}


  constraint comedy_time {
    foreach (comedy[i]) {
      comedy[i] -> start_time[i] >= 12;
    }
  }
  function void debug();
    if (!this.randomize()) begin
      $display("Randomization failed due to conflicting constraints.");
    end else begin
      $display("Randomization succeeded!");
    end
  endfunction
endclass

module movie_schedule_check;
  int i,j,evening_count;
  initial begin
    movie_screening schedule = new();
    for (i = 0; i < 3; i++) begin
      schedule.debug();
      $display("\nSchedule %0d:", i + 1);
      for (j = 0; j < 5; j++) begin
        $display(" Movie %0d:", j + 1);
        $display("  Screen: %0d", schedule.screen[j]);
        $display("  Start Time: %0d:00", schedule.start_time[j]);
        $display("  Duration: %0d hours", schedule.duration[j]);
        $display("Comedy Mvoie: %s", schedule.comedy[j] ? "Yes" : "No");
        $display("  End Time: %0d:00", schedule.start_time[j] + schedule.duration[j]);
      end

      evening_count = 0;
      foreach (schedule.start_time[j]) begin
      if (schedule.start_time[j] >= 18) evening_count++;
     end
     $display(" Movies in evening slots (after 6 PM): %0d", evening_count);
   end
  end
endmodule
```

Output :

```
CPU time: .248 seconds to compile + .239 seconds to elab + .222 seconds to link
Chronologic VCS simulator copyright 1991-2023
Contains Synopsys proprietary information.
Compiler version U-2023.03-SP2_Full64; Runtime version U-2023.03-SP2_Full64;  Apr  7
11:17 2025
Randomization succeeded!

Schedule 1:
 Movie 1:
  Screen: 3
  Start Time: 14:00
  Duration: 1 hours
Comedy Mvoie: Yes
  End Time: 15:00
 Movie 2:
  Screen: 2
  Start Time: 16:00
  Duration: 2 hours
Comedy Mvoie:  No
  End Time: 18:00
 Movie 3:
  Screen: 2
  Start Time: 10:00
  Duration: 1 hours
Comedy Mvoie:  No
  End Time: 11:00
 Movie 4:
  Screen: 1
  Start Time: 20:00
  Duration: 2 hours
Comedy Mvoie:  No
  End Time: 22:00
 Movie 5:
  Screen: 2
  Start Time: 12:00
  Duration: 2 hours
Comedy Mvoie: Yes
  End Time: 14:00
 Movies in evening slots (after 6 PM): 1
Randomization succeeded!

Schedule 2:
 Movie 1:
  Screen: 1
  Start Time: 16:00
  Duration: 1 hours
Comedy Mvoie: Yes
  End Time: 17:00
 Movie 2:
  Screen: 1
  Start Time: 22:00
  Duration: 3 hours
Comedy Mvoie:  No
  End Time: 25:00
 Movie 3:
  Screen: 2
  Start Time: 16:00
  Duration: 1 hours
Comedy Mvoie:  No
  End Time: 17:00
 Movie 4:
  Screen: 2
  Start Time: 17:00
  Duration: 1 hours
Comedy Mvoie:  No
  End Time: 18:00
 Movie 5:
  Screen: 2
  Start Time: 19:00
  Duration: 3 hours
```

```
Comedy Mvoie:  No
  End Time: 22:00
 Movies in evening slots (after 6 PM): 2
Randomization succeeded!

Schedule 3:
 Movie 1:
  Screen: 3
  Start Time: 14:00
  Duration: 2 hours
Comedy Mvoie:  No
  End Time: 16:00
 Movie 2:
  Screen: 1
  Start Time: 21:00
  Duration: 3 hours
Comedy Mvoie: Yes
  End Time: 24:00
 Movie 3:
  Screen: 1
  Start Time: 15:00
  Duration: 3 hours
Comedy Mvoie: Yes
  End Time: 18:00
 Movie 4:
  Screen: 3
  Start Time: 22:00
  Duration: 2 hours
Comedy Mvoie:  No
  End Time: 24:00
 Movie 5:
  Screen: 3
  Start Time: 9:00
  Duration: 3 hours
Comedy Mvoie:  No
  End Time: 12:00
 Movies in evening slots (after 6 PM): 2
           V C S   S i m u l a t i o n   R e p o r t
Time: 0 ns
CPU Time:      0.500 seconds;       Data structure size:   0.0Mb
```

}

3. Gift Distribution

Question: Simulate a random gift distribution with the following constraints:

No person can receive the same gift twice.

At least 20% of the gifts must be under the "premium" category.

The distribution must ensure that everyone receives at least one gift.

Code :

```
class Gift_Distrib;
 rand int gift_to_person[10];
 rand bit  premium[10];
  int num_premium;

 function void pre_randomize();
   num_premium = 0;
 endfunction




 constraint no_duplicate_gifts {
   foreach (gift_to_person[i]) {
     gift_to_person[i] inside {[0:9]};
     foreach (gift_to_person[j]) {
       if (i != j) {
         gift_to_person[i] != gift_to_person[j];
       }
     }
   }
 }
```

```
  constraint everyone_gets_gift {
    unique {gift_to_person};
  }

  function void debug();
    if (!this.randomize()) begin
      $display("Randomization failed due to conflicting constraints.");
    end else begin
      $display("Randomization succeeded!");
    end
  endfunction


endclass

module gift_distrib_check;
  int received_gifts[10];
  int duplicates;
  int not_gifted = 0;
  initial begin
    Gift_Distrib gifts = new();
    for (int i = 0; i < 5; i++) begin
      gifts.debug();
      $display("\nGift Distribution %0d:", i + 1);
      foreach (received_gifts[i]) received_gifts[i] = 0;

      for (int j = 0; j < 10; j++) begin
        $display(" Person %0d : received gift %0d (Premium: %s)",
                 j, gifts.gift_to_person[j], gifts.premium[j] ? "Yes" : "No");
        received_gifts[gifts.gift_to_person[j]]++;
      end

      duplicates = 0;
      foreach (received_gifts[j]) begin
        if (received_gifts[j] > 1) duplicates++;
      end
      $display(" Duplicate gifts found: %0d ", duplicates);

      foreach (received_gifts[j]) begin
        if (received_gifts[j] == 0) not_gifted = not_gifted + 1;
      end
      $display(" People without gifts: %0d ", not_gifted);
    end
  end
endmodule
```

Output :

```
CPU time: .256 seconds to compile + .254 seconds to elab + .222 seconds to link
Chronologic VCS simulator copyright 1991-2023
Contains Synopsys proprietary information.
Compiler version U-2023.03-SP2_Full64; Runtime version U-2023.03-SP2_Full64;  Apr  7
11:38 2025
```

```
Randomization succeeded!

Gift Distribution 1:
 Person 0 : received gift 2 (Premium: Yes)
 Person 1 : received gift 6 (Premium: Yes)
 Person 2 : received gift 3 (Premium: Yes)
 Person 3 : received gift 4 (Premium:  No)
 Person 4 : received gift 7 (Premium:  No)
 Person 5 : received gift 9 (Premium: Yes)
 Person 6 : received gift 0 (Premium:  No)
 Person 7 : received gift 8 (Premium: Yes)
 Person 8 : received gift 1 (Premium:  No)
 Person 9 : received gift 5 (Premium:  No)
 Duplicate gifts found: 0
 People without gifts: 0
Randomization succeeded!

Gift Distribution 2:
 Person 0 : received gift 8 (Premium:  No)
 Person 1 : received gift 0 (Premium: Yes)
 Person 2 : received gift 2 (Premium:  No)
 Person 3 : received gift 3 (Premium: Yes)
 Person 4 : received gift 5 (Premium: Yes)
 Person 5 : received gift 9 (Premium: Yes)
 Person 6 : received gift 7 (Premium: Yes)
 Person 7 : received gift 6 (Premium:  No)
 Person 8 : received gift 4 (Premium:  No)
 Person 9 : received gift 1 (Premium:  No)
 Duplicate gifts found: 0
 People without gifts: 0
Randomization succeeded!

Gift Distribution 3:
 Person 0 : received gift 2 (Premium: Yes)
 Person 1 : received gift 0 (Premium:  No)
 Person 2 : received gift 6 (Premium: Yes)
 Person 3 : received gift 7 (Premium:  No)
 Person 4 : received gift 1 (Premium: Yes)
 Person 5 : received gift 8 (Premium:  No)
 Person 6 : received gift 3 (Premium:  No)
 Person 7 : received gift 4 (Premium:  No)
 Person 8 : received gift 9 (Premium: Yes)
 Person 9 : received gift 5 (Premium:  No)
 Duplicate gifts found: 0
 People without gifts: 0
Randomization succeeded!

Gift Distribution 4:
 Person 0 : received gift 3 (Premium: Yes)
 Person 1 : received gift 0 (Premium: Yes)
 Person 2 : received gift 8 (Premium: Yes)
 Person 3 : received gift 9 (Premium: Yes)
 Person 4 : received gift 7 (Premium: Yes)
 Person 5 : received gift 2 (Premium:  No)
 Person 6 : received gift 4 (Premium: Yes)
 Person 7 : received gift 1 (Premium:  No)
 Person 8 : received gift 5 (Premium: Yes)
 Person 9 : received gift 6 (Premium: Yes)
 Duplicate gifts found: 0
 People without gifts: 0
Randomization succeeded!

Gift Distribution 5:
 Person 0 : received gift 8 (Premium: Yes)
 Person 1 : received gift 7 (Premium: Yes)
 Person 2 : received gift 1 (Premium:  No)
 Person 3 : received gift 9 (Premium: Yes)
 Person 4 : received gift 4 (Premium:  No)
 Person 5 : received gift 2 (Premium: Yes)
 Person 6 : received gift 0 (Premium: Yes)
 Person 7 : received gift 6 (Premium:  No)
 Person 8 : received gift 5 (Premium: Yes)
 Person 9 : received gift 3 (Premium:  No)
 Duplicate gifts found: 0
 People without gifts: 0
           V C S   S i m u l a t i o n   R e p o r t
Time: 0 ns
CPU Time:      0.290 seconds;       Data structure size:   0.0Mb
```

4. 1. AXI Transaction Generator.

Create a class AXI_Transaction to represent transactions on an AXI bus. Include fields: addr, data, burst_type, len (length of burst), and id.

Constraints:

The addr should be aligned to the burst size (len * 4 bytes).

Allow burst_type values: INCR (1) and WRAP (2) only.

The len must be between 1 and 16.

Randomize a unique id for each transaction.

Challenge: Generate 10 AXI transactions and verify that all generated addresses are aligned based on the burst length.

Code:

```
class AXI_gen;
 rand bit [31:0] addr;
 rand bit [31:0] data;
 rand bit [1:0] burst_type;
 rand bit [3:0] len;
 rand bit [3:0] id;

 localparam INCR = 1;
 localparam WRAP = 2;

 constraint addr_alignment {
  addr % (len * 4) == 0;
 }

 constraint burst_type_values {
  burst_type inside {INCR, WRAP};
 }
```

```
  constraint len_range {
    len inside {[1:16]};
  }

   function void debug();
    if (!this.randomize()) begin
      $display("Randomization failed due to conflicting constraints.");
    end else begin
      $display("Randomization succeeded!");
    end
   endfunction



  function void display();
    $display("AXI Transaction:");
    $display("  ID: %0d", id);
    $display("  Address: 0x%0h", addr);
    $display("  Data: 0x%0h", data);
    $display("  Burst Type: %0s", burst_type == INCR ? "INCR" : "WRAP");
    $display("  Length: %0d", len);
  endfunction
endclass

module axi_transactionr;
  initial begin
    AXI_gen transactions[10];
    bit [3:0] used_ids[16];
    foreach (used_ids[i]) used_ids[i] = 0;


    for (int i = 0; i < 5; i++) begin
      transactions[i] = new();


      transactions[i].randomize() with {
        foreach (used_ids[j])
          if (used_ids[j] == 1)
            id != j;
      };


      used_ids[transactions[i].id] = 1;


      $display("\nTransaction %0d:", i);
      transactions[i].display();



      if (transactions[i].addr % (transactions[i].len * 4) != 0) begin
        $display("error");
      end
    end
  end
endmodule
```

Output :

```
CPU time: .303 seconds to compile + .278 seconds to elab + .239 seconds to link
Chronologic VCS simulator copyright 1991-2023
Contains Synopsys proprietary information.
```

```
Compiler version U-2023.03-SP2_Full64; Runtime version U-2023.03-SP2_Full64;  Apr  7
11:44 2025

Transaction 0:
AXI Transaction:
  ID: 5
  Address: 0x23cdf378
  Data: 0xe91ad7b9
  Burst Type: INCR
  Length: 6

Transaction 1:
AXI Transaction:
  ID: 7
  Address: 0x51b40a18
  Data: 0xa311a307
  Burst Type: WRAP
  Length: 5

Transaction 2:
AXI Transaction:
  ID: 9
  Address: 0x8cdbbb30
  Data: 0xdc1c6b6b
  Burst Type: WRAP
  Length: 4

Transaction 3:
AXI Transaction:
  ID: 8
  Address: 0x22a0c85c
  Data: 0xca8e4e56
  Burst Type: WRAP
  Length: 7

Transaction 4:
AXI Transaction:
  ID: 2
  Address: 0x2da04878
  Data: 0xcd944349
  Burst Type: WRAP
```

```
 Length: 2
      V C S   S i m u l a t i o n   R e p o r t
Time: 0 ns
CPU Time:      0.650 seconds;       Data structure size:   0.0Mb
```

5. Cache Line Generator for a CPU

Design a CacheLine class with fields: tag, index, data[8] (8 words), and valid_bit.

Constraints:

Randomize the index value within [0:127].

Assign the tag a unique value for each index.

Ensure the data array is randomized such that no two words have the same value.

Set valid_bit = 1 for all generated lines.

Challenge: Generate 128 cache lines with unique tag-index pairs, ensuring the data uniqueness constraint is met.

Code :

```
class CacheLine;
  rand bit [15:0] tag;
  rand bit [6:0] index;
  rand bit [31:0] data[8];
  bit valid_bit;

  function new();
    valid_bit = 1;
  endfunction

  constraint data_unique {
    foreach (data[i]) {
      foreach (data[j]) {
        if (i != j) {
```

```
        data[i] != data[j];
      }
    }
  }
 }

 constraint index_range {
  index inside {[0:127]};
 }

 function void display();
  $display("Cache Line:");
  $display("  Index: %0d", index);
  $display("  Tag: 0x%0h", tag);
  $display("  Valid: %0d", valid_bit);
  $display("  Data:");
  foreach (data[i]) begin
    $display("    Word[%0d]: 0x%0h", i, data[i]);
  end
 endfunction
endclass

module cache_line_generator;
 int indices_used[128];

 initial begin
  CacheLine cache_lines[128];
  bit [15:0] tags_used[128];
```

```
foreach (tags_used[i]) tags_used[i] = 16'hFFFF;

for (int i = 0; i < 128; i++) begin
  cache_lines[i] = new();

  cache_lines[i].randomize() with {
    foreach (tags_used[j]) {
      if (index == j && tags_used[j] != 16'hFFFF) {
        tag != tags_used[j];
      }
    }
  };

  tags_used[cache_lines[i].index] = cache_lines[i].tag;

  if (i < 5 || i >= 123) begin
    $display("\nCache Line %0d:", i);
    cache_lines[i].display();
  end  // Missing end for if block
end  // Missing end for for loop

foreach (indices_used[i]) indices_used[i] = 0;

foreach (cache_lines[i]) begin
  indices_used[cache_lines[i].index]++;
end
```

```
    for (int i = 0; i < 5; i++) begin
      bit data_unique = 1;
      for (int j = 0; j < 8; j++) begin
        for (int k = j+1; k < 8; k++) begin
          if (cache_lines[i].data[j] == cache_lines[i].data[k]) begin
            data_unique = 0;
            $display("  Cache Line %0d: Data words %0d and %0d have the same value!",
                    i, j, k);
          end
        end
      end
    end

  end  // Missing end for initial block
endmodule
```

Output :

```
CPU time: .234 seconds to compile + .252 seconds to elab + .221 seconds to link
Chronologic VCS simulator copyright 1991-2023
Contains Synopsys proprietary information.
Compiler version U-2023.03-SP2_Full64; Runtime version U-2023.03-SP2_Full64;  Apr  7
11:51 2025

Cache Line 0:
Cache Line:
  Index: 45
  Tag: 0xf5a2
  Valid: 1
  Data:
    Word[0]: 0x2f3e6819
    Word[1]: 0x423a0c3a
    Word[2]: 0xfa58cf3f
    Word[3]: 0x8744ca05
    Word[4]: 0xf01d4978
    Word[5]: 0x699f5f34
    Word[6]: 0x4133dc9f
    Word[7]: 0xb36931a9

Cache Line 1:
Cache Line:
  Index: 87
  Tag: 0xfec9
  Valid: 1
  Data:
    Word[0]: 0xfd6a2d36
    Word[1]: 0xe855792
    Word[2]: 0xeac2318d
    Word[3]: 0x964f2d92
    Word[4]: 0x7fbe5d16
    Word[5]: 0xd5616d52
    Word[6]: 0xd66f267a
    Word[7]: 0xc7fdf03f

Cache Line 2:
Cache Line:
  Index: 89
  Tag: 0xadd4
  Valid: 1
  Data:
    Word[0]: 0xc3865323
    Word[1]: 0x56d2621e
    Word[2]: 0xd93138d0
    Word[3]: 0xe8bc36a1
    Word[4]: 0x2cd3bf3c
    Word[5]: 0xdd2e4e57
    Word[6]: 0x933b10e0
    Word[7]: 0xddd9ea3c

Cache Line 3:
Cache Line:
  Index: 63
  Tag: 0x114b
  Valid: 1
  Data:
    Word[0]: 0xd0bc41c8
    Word[1]: 0xddfef09b
    Word[2]: 0xa8372b36
    Word[3]: 0x68cc2e36
    Word[4]: 0xc1469dc3
    Word[5]: 0x9cb8d32b
    Word[6]: 0xaf19aa89
    Word[7]: 0xba8f3cef

Cache Line 4:
Cache Line:
  Index: 1
  Tag: 0xf370
  Valid: 1
  Data:
    Word[0]: 0xa109ce5a
    Word[1]: 0xb6562c54
    Word[2]: 0x99b2ce34
    Word[3]: 0xde6dfd3f
    Word[4]: 0x102f0232
    Word[5]: 0x9d8be610
    Word[6]: 0x93c9b9d5
    Word[7]: 0x5ab6e33f

Cache Line 123:
Cache Line:
  Index: 60
  Tag: 0xc9f2
  Valid: 1
  Data:
    Word[0]: 0xb4234d0
    Word[1]: 0xa8e0e514
    Word[2]: 0x970bb50
    Word[3]: 0xc73bef0e
    Word[4]: 0x6cc1ba61
    Word[5]: 0xd5e6c091
    Word[6]: 0x633fe60e
    Word[7]: 0x6b8c6bdd

Cache Line 124:
Cache Line:
  Index: 0
  Tag: 0xd751
  Valid: 1
  Data:
    Word[0]: 0xdab83439
    Word[1]: 0x757f8adb
    Word[2]: 0x481a7976
    Word[3]: 0x5ce4ee93
    Word[4]: 0xcf561cf2
    Word[5]: 0x27135dce
    Word[6]: 0xf17b01e9
    Word[7]: 0x8c72f5ec

Cache Line 125:
Cache Line:
  Index: 13
  Tag: 0xd604
  Valid: 1
  Data:
    Word[0]: 0xc1ae757a
    Word[1]: 0xa77b5a08
    Word[2]: 0x85e7afc8
    Word[3]: 0x38e4b322
    Word[4]: 0x2c00256f
    Word[5]: 0xfb495d6
    Word[6]: 0xe2f3304a
    Word[7]: 0xcae43e75

Cache Line 126:
Cache Line:
  Index: 113
  Tag: 0x655
  Valid: 1
  Data:
    Word[0]: 0x52049c1c
    Word[1]: 0xaf92c5cb
    Word[2]: 0xc3852bd6
    Word[3]: 0x496edbdc
    Word[4]: 0x5df9067f
    Word[5]: 0xefde06c
    Word[6]: 0xdcbd8be5
    Word[7]: 0x786e9d82

Cache Line 127:
Cache Line:
  Index: 59
  Tag: 0xa07e
  Valid: 1
  Data:
    Word[0]: 0xc7b44fd0
    Word[1]: 0x1ec02525
    Word[2]: 0x741cf29e
    Word[3]: 0xc745091a
    Word[4]: 0x647c692a
    Word[5]: 0x61617476
    Word[6]: 0x585f9968
    Word[7]: 0xa1ff0c82
           V C S   S i m u l a t i o n   R e p o r t
Time: 0 ns
CPU Time:      1.600 seconds;       Data structure size:   0.0Mb
```