

Validating Physical Access Layer of WiMAX Using System Verilog

By

Albert Chiang, Wei-Hua Han
Synopsys, Inc.

Bhanu Kapoor
Mimasic

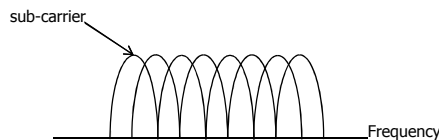
Contents

- WiMAX and OFDM/OFDMA
- Verification Challenges
- Application of SystemVerilog in WiMAX Validation
- Conclusions

WiMAX

- WiMAX
 - Worldwide Interoperability of Microwave Access
 - The “last mile” technology to replace broadband cable network
- 802.16-2004
 - Air Interface for Fixed Broadband Wireless Access Systems
- 802.16e
 - Air Interface for Fixed and Mobile Broadband Wireless Access Systems

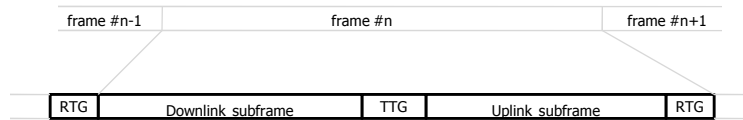
OFDM/OFDMA



- OFDM
 - Orthogonal Frequency-Division Multiplexing
 - a multi-carrier transmission technique
 - using multiple frequencies to simultaneously transmit multiple signals in parallel
- OFDMA
 - Orthogonal Frequency Division Multiple Access
 - Based on OFDM
 - Allows sub-carriers to be assigned to different users

OFDMA Frames

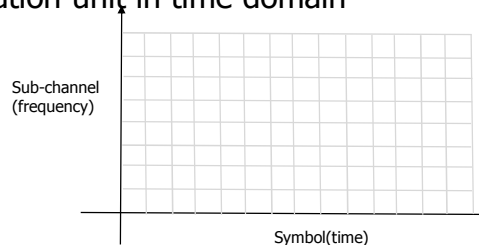
- Frame
 - One complete set of downlink and uplink transmissions
 - Frame length: 2ms to 20ms



TTG: transmit transition gap
RTG: receive transition gap

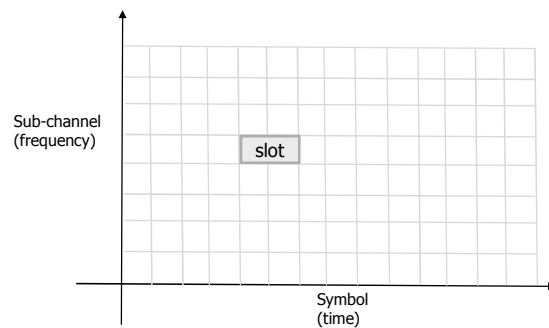
OFDMA: sub-channel and symbol

- Sub-channel
 - Smallest logical allocation unit in frequency domain
 - Containing one or more physical carriers
 - Multiple schemes are defined for mapping physical carriers into sub-channels
- Symbol
 - Smallest allocation unit in time domain



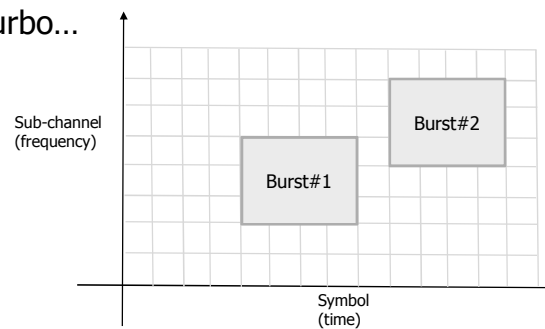
OFDMA: slot

- Minimum possible data allocation unit
- Containing one sub-channel and one to three symbols
 - Depending on the zone types



OFDMA: burst

- Containing slots
- Take a certain number of sub-channels and symbols
- Same modulation and coding scheme within one burst
 - QPSK, QAM..
 - Convolutional, Turbo...



OFDMA: zone

- A part of frame
- Different types depending on how sub-carriers are mapped into sub-channels
- Slot and Zone
 - DL-FUSC: Downlink Full Usage Sub-channels
 - One slot is one sub-channel(48 sub-carriers) by one symbol
 - DL-PUSC: Downlink Partial Usage Sub-channels
 - One slot is one sub-channel(24 sub-carriers) by two symbols
 - UL-PUSC: Uplink Partial Usage Sub-channels
 - One slot is one sub-channel(16 sub-carriers) by three symbols
 - TUSC, AMC..

OFDMA: functional verification challenge

- The OFDMA frame is really flexible
 - The length of the frame
 - How many DL zones, how many UP zones
 - Zone types
 - How many bursts in each zone
 - Modulation and coding scheme in each burst
 - the places(positions) of bursts in each zone
 -
- Constrained Random Verification

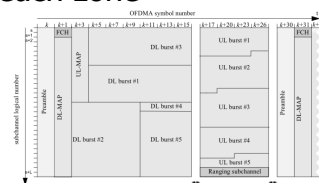


Figure 218—Time plan - one TDD time frame (with only mandatory zone)

SystemVerilog for Verification

- Verification complexity increasing rapidly for complex designs like WiMAX
- Powerful features in SystemVerilog for Verification
 - Class: OOP
 - Randomization and Constraint
 - Queue and Dynamic Array
 - Coverage

SystemVerilog: OOP

- Implemented through SystemVerilog "class"
 - Inheritance
 - Encapsulation
 - Polymorphism

```
class burst_c extends vmm_data;
    virtual function void display();
endfunction
endclass
class myburst1_c extends burst_c;
    virtual function void display();
endfunction
endclass
class myburst2_c extends burst_c;
    virtual function void display();
endfunction
endclass

burst_c b;
myburst1_c b1 = new;
myburst2_c b2 = new;

b=b1;
b.display();
b=b2;
b.display();
```

SystemVerilog: Constraint and Randomization

- Implemented through object-based randomization and constraint
 - Rand/randc data members
 - Randomize/pre_randomize/post_randomize methods

```
class burst_c extends vmm_data;
  rand crv_pkg::fec_t fec_type;
  rand int rep_factor;
  //boosting?
  rand bit is_boosting;
  rand crv_pkg::burst_t burst_type;

  constraint valid {
    rep_factor inside {1,2,4,6};
  }
  ...
endclass

burst_c b=new;
b.randomize();
b.randomize() with { fec_type==FECL; }
```

SystemVerilog: Queue

- Variable-size, ordered collection of homogeneous elements
- Dynamic insertion and removal
- Method
 - Find_first,find_last_index..
 - Sort, reverse...
- Other arrays
 - Dynamic Array
 - Assoc array

```
class zone_c extends vmm_data;
  typedef enum {pusc_d1, fusc_d1,amc_d1,
               pusc_ul,amc_ul} zone_t;
  rand zone_t zone_type;
  rand int zone_width; // number_of_slots
  rand burst_c burst[$];
  constraint cons_burst_non_overlap {
    foreach(burst[i])
      foreach(burst[j])
        if(j!=i) {
          ...
        }
  }
  ...
endclass
```

SystemVerilog: Functional Coverage

- Constructs to measure how much of the design specification (the test plan) has been exercised

```
class sb_c extends vmm_data;
  burst_c burst;
  covergroup burst_cov;
    burst_type_cov: coverpoint burst.burst_type;
  endgroup
  function new();
    burst_cov = new;
  endfunction
  function void write(burst_c burst);
    this.burst = burst;
    burst_cov.sample();
    $display("burst_type coverage is
             %f",burst_cov.get_coverage());
  endfunction
endclass
```

SystemVerilog: the frame example

```
class frame_c extends vmm_data;
  rand int frame_len;
  rand int ttg_gap_len;
  rand int rtg_gap_len;
  rand zone_c dl_zone[$];
  rand zone_c ul_zone[$];
  rand unsigned int dl_zone_width[$]
  rand unsigned int ul_zone_width[$]
  constraint cons_frame_len;
  constraint cons_gap;
  constraint cons_symbols;
  constraint valid_zone_width;
  constraint valid_zone_type;
  ...
endclass
```

```
constraint frame_c::cons_frame_len {
  frame_len inside {2000,3000};
}
constraint frame_c::cons_gap {
  ttg_gap_len inside {[20:80]};
  rtg_gap_len inside {[50:100]};
}
constraint frame_c::cons_symbols {
  num_of_symbols == (frame_len -
  ttg_gap_len - rtg_gap_len)/100;
}
constraint frame_c::valid_zone_width {
  ul_zone_width.sum() +
  dl_zone_width.sum() == num_of_symbols;
}
constraint frame_c::valid_zone_type {
  dl_zone[0].zone_type == pusc_dl;
}
...
```

Conclusion

- SystemVerilog provides a rich infrastructure and a set of useful constructs for the physical layer functional verification of WiMAX OFMDA
- The verification concepts revealed here were implemented with Synopsys VCS® at a major wireless design company
- The Verification Methodology Manual (VMM) provides clear guidelines on how to easily create a constrained random, coverage driven, self-checking, extensible, reusable, and scalable testbench