

AHB to APB Bridge Design

Project Report

By- Sanskar Verma

College-VIT Vellore

Instructor-Sathyapriya

Table of Contents

- Introduction
- AMBA Overview
- AHB to APB Bridge Design
 - Detailed Design
- Basic Implementation Tools
 - Verilog Implementation
- Simulation and Verification
 - Conclusion
 - References

Introduction

In modern embedded systems, efficient communication between high-performance processors and low-power peripherals is essential. The Advanced Microcontroller Bus Architecture (AMBA) facilitates this through standardized bus protocols, specifically the Advanced High-performance Bus (AHB) and Advanced Peripheral Bus (APB). The AHB is designed for high-speed, high-frequency modules, while the APB is optimized for low-power peripheral functions. This project focuses on designing an AHB to APB bridge, enabling seamless communication between these two buses. The bridge translates high-speed AHB transactions into the simpler, lower-power APB protocol, ensuring efficient data transfer and synchronization in complex embedded systems.

AMBA Overview

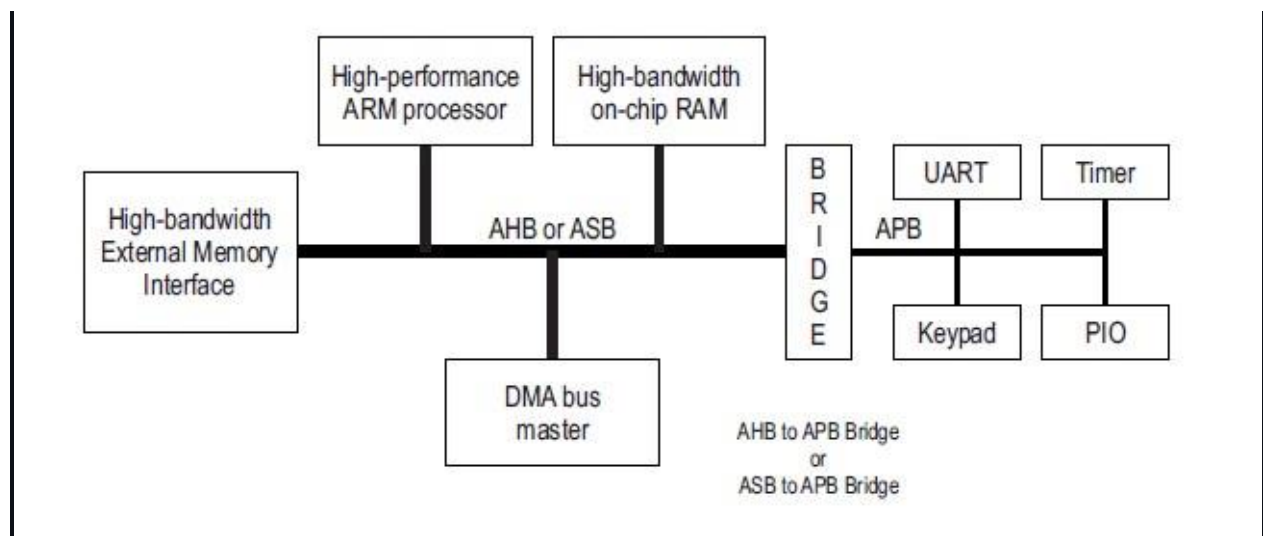
The Advanced Microcontroller Bus Architecture (AMBA) specification defines an on-chip communications standard for designing high-performance embedded microcontrollers. Three distinct buses are defined within the AMBA specification:

1. Advanced High-performance Bus (AHB)
2. Advanced System Bus (ASB)
3. Advanced Peripheral Bus (APB)

Advanced High-performance Bus (AHB) The AMBA AHB is for high-performance, high clock frequency system modules. The AHB acts as the high-performance system backbone bus. AHB supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral macrocell functions. AHB is also specified to ensure ease of use in an efficient design flow using synthesis and automated test techniques.

Advanced System Bus (ASB) The AMBA ASB is for highperformance system modules. AMBA ASB is an alternative bus system suitable for use where the high-performance features of AHB are not required. ASB also supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral macrocell functions.

Advanced Peripheral Bus (APB) The AMBA APB is for low-power peripherals. AMBA APB is optimized for minimal power consumption and reduced interface complexity to support peripheral functions. APB can be used in conjunction with either version of the system bus.



AHB to APB Bridge Design

The AHB to APB bridge design facilitates communication between high-speed AHB modules and low-power APB peripherals in embedded systems. The bridge translates AHB transactions to APB protocol, handling address decoding, data transfer, and control signal synchronization. It includes components like the AHB and APB interfaces, FIFO buffers, and a state machine to manage states such as IDLE, SETUP, and ACCESS. The design ensures efficient data transfer while maintaining low power consumption, critical for embedded applications. A Verilog implementation and thorough simulation verify the bridge's functionality and performance, making it a vital component in modern SoC designs.

Detailed Design

Components

- AHB Interface: Handles communication with the AHB.
- APB Interface: Handles communication with the APB.

- FIFO Buffers: Manages data flow between the two buses.
- State Machine: Controls the data transfer and ensures proper sequencing.

State Machine

The state machine manages the transition between different states during the data transfer process:

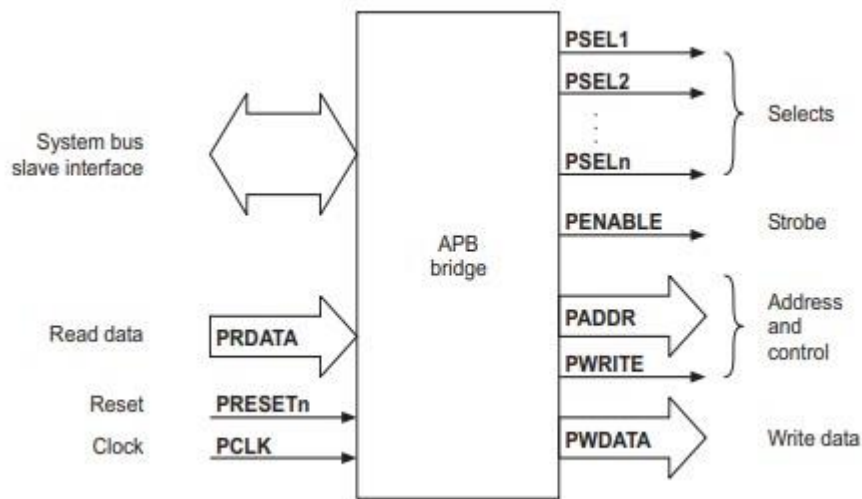
- IDLE: The default state, waiting for an AHB transaction.
- SETUP: Prepares the APB for an upcoming transaction.
- ACCESS: Performs the data transfer between AHB and APB.

To design and simulate a synthesizable AHB to APB bridge interface using Verilog and run single read and single write tests using AHB Master and APB Slave testbenches. The bridge unit converts system bus transfers into APB transfers and performs the following functions:

- Latches the address and holds it valid throughout the transfer.

- Decodes the address and generates a peripheral select, PSELx. Only one select signal can be active during a transfer. Drives the data onto the APB for a write transfer
- Drives the APB data onto the system bus for a read transfer
- Generates a timing strobe, PENABLE, for the transfer Can implement single read and write operations successfully.

The Diagram Below shows the Interface:



Basic Implementation Tools

HDL Used : Verilog

Simulator Tool Used: ModelSIM

Synthesis Tool Used: Quartus Prime

Family: Cyclone V

Device: 5CSXFC6D6F31I7ES

Verilog Implementation

Verilog codes for Bridge with it's Testbench.

```
Ln#
1 module bridge_rtl(
2     input hclk, hresetn, hselapb, hwrite,
3     input [1:0] htrans, input [31:0] haddr,
4     input [31:0] hdata,
5     input [31:0] prdata,
6     output reg [31:0] paddr, pwdata,
7     output reg psel, penable, pwrite,
8     output reg hresp, hready,
9     output reg [31:0] hrdata
10 );
11
12 parameter idle = 3'b000;
13 parameter read = 3'b001;
14 parameter wait = 3'b010;
15 parameter write = 3'b011;
16 parameter write_p = 3'b100;
17 parameter wenable_p = 3'b101;
18 parameter wenable = 3'b110;
19 parameter renable = 3'b111;
20 reg [31:0] haddr_temp, hdata_temp;
21 reg [2:0] present_state, next_state;
22 reg valid, hwrite_temp;
23
24 always@(*)
25 begin
26     if(hselapb == 1'b1 && (htrans == 2'b10 || htrans == 2'b11))
27         valid = 1'b1;
28     else
29         valid = 1'b0;
30
31     if(hresetn == 1'b0)
32         present_state = idle;
33     else
34         present_state = next_state;
35 end
36
37
38 always@(present_state)
39 begin
40     case(present_state)
41     idle:
42         begin
43             psel = 1'b0;
44             penable = 1'b0;
45             hready = 1'b1;
46
47             if(valid == 1'b0)
48                 next_state = idle;
49             else if(valid == 1'b1 && hwrite == 1'b0)
50                 next_state = read;
51             else if(valid == 1'b1 && hwrite == 1'b1)
52                 next_state = wait;
53         end
54
55     read:
56         begin
57             psel = 1'b1;
58             paddr = haddr;
59             pwrite = 1'b0;
60             penable = 1'b0;
61             hready = 1'b0;
62
63             next_state = renable;
64         end
65
66     renable:
67         begin
68             penable = 1'b1;
69             hrdata = prdata;
70             hready = 1'b1;
71             if(valid == 1'b1 && hwrite == 1'b0)
72                 next_state = read;
73             if(valid == 1'b1 && hwrite == 1'b1)
74                 next_state = wait;
75         end
76     endcase
77 end
```

```
Ln#
73     if(valid == 1'b1 && hwrite == 1'b0)
74         next_state = read;
75     else if(valid == 1'b1 && hwrite == 1'b1)
76         next_state = wait;
77     else if(valid == 1'b0)
78         next_state = idle;
79 end
80
81 wait:
82 begin
83     penable = 1'b0;
84     haddr_temp = haddr;
85     hwrite_temp = hwrite;
86     if(valid == 1'b0)
87         next_state = write;
88     else if(valid == 1'b1)
89         next_state = write_p;
90 end
91
92
93 write:
94 begin
95     psel = 1'b1;
96     paddr = haddr_temp;
97     pwdata = hdata;
98     pwrite = 1'b1;
99     penable = 1'b0;
100    hready = 1'b0;
101
102    if(valid == 1'b0)
103        next_state = wenable;
104    else if(valid == 1'b1)
105        next_state = wenable_p;
106 end
107
108 write_p:
109 begin
110     psel = 1'b1;
```

```
Ln#
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

```

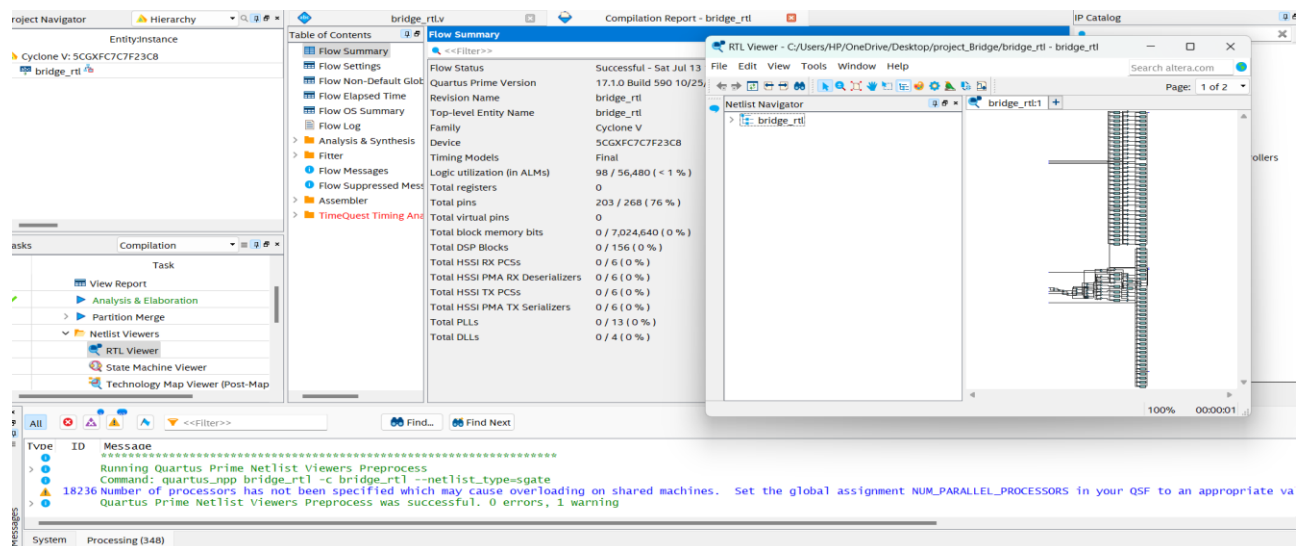
C:/Users/HP/OneDrive/Desktop/project_Bridge/bridge_tb.v (/bridge_tb) - Default
Ln#
1  module bridge_tb(
2
3
4
5
6      );
7
8      ////////////////ahb slave
9      reg hclk,hresetn,hselapb,hwrite;
10     reg [1:0] htrans;
11     reg [31:0] haddr,hwdata;
12     wire hresp;
13     wire [31:0] hrdata;
14
15     ////////////////apb output signal
16
17     reg [31:0] prdata;
18     wire psel,penable, pwrite,hready;
19     wire [31:0] paddr,pwdata;
20
21     bridge_rtl bridge(hclk,hresetn,hselapb,hwrite,htrans,
22                      haddr,hwdata,
23                      prdata,
24                      paddr,pwdata,
25                      psel,penable,pwrite,
26                      hresp,hready,
27                      hrdata);
28
29     initial
30     begin
31         hclk = 0;
32     end
33
34     always #10 hclk = ~hclk;
35
36
37     task reset_bridge();
38     begin

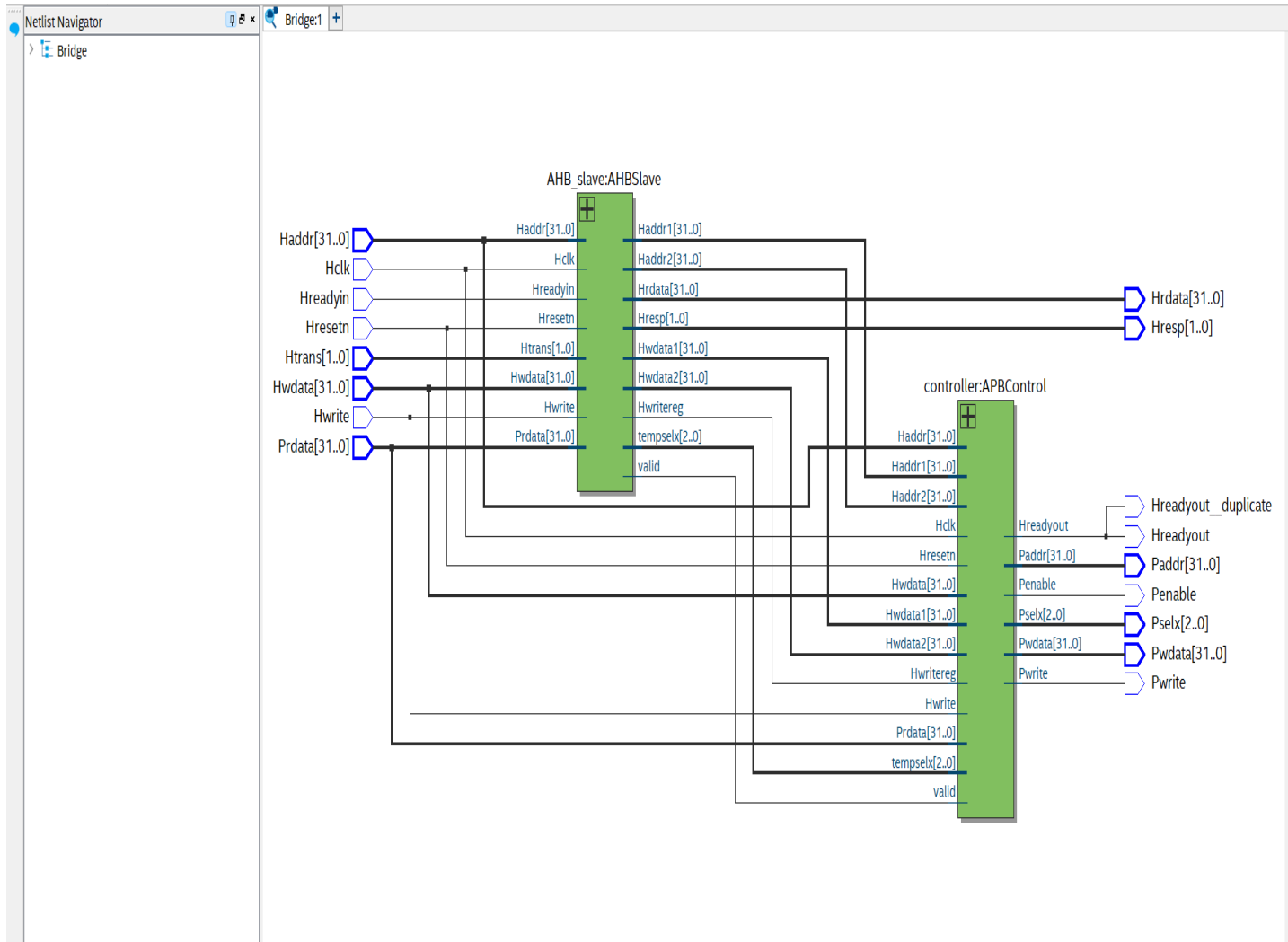
```

```

33
34     always #10 hclk = ~hclk;
35
36
37     task reset_bridge();
38     begin
39         @(negedge hclk)
40             hresetn = 1;
41         @(negedge hclk)
42             hresetn = 0;
43     end
44     endtask
45
46
47
48     initial
49     begin
50
51         reset_bridge ();
52         #10 ;
53         hwrite = 1'b0;
54         hselapb = 1'b1;
55         htrans = 2'b10;
56         haddr = 32;
57         #10 ;
58         hwrite = 1'bx;
59         hselapb = 1'b0;
60         htrans = 2'bxx;
61         haddr = 32'hxxxxx_xxxx;
62         #5 ;
63         prdata =40;
64     end
65 endmodule
66
67
68
69

```





Conclusion

During my internship, I focused on designing and simulating an AHB to APB bridge interface, a crucial component in embedded systems for facilitating communication between high-speed AHB modules and low-power APB peripherals. The project involved multiple phases, each contributing to a comprehensive understanding of hardware design and verification.

I started by understanding the requirements and specifications of the AHB and APB protocols, and the functionality needed for the bridge to translate transactions between these two buses. Using Verilog, I designed the bridge to include key components such as the AHB interface, APB interface, a state machine for managing data transfers, and an address decoder for generating peripheral select signals (PSEL).

To verify the functionality of the design, I developed detailed testbenches for both the AHB master and APB slave. These testbenches simulated single read and write operations, ensuring that the bridge could accurately latch addresses, drive data onto the APB for write transfers, retrieve data from the APB for read transfers, and generate the necessary timing strobe (PENABLE).

The project utilized ModelSIM for simulation, allowing me to thoroughly test and debug the design, and Quartus Prime for synthesis, targeting the Cyclone V family of FPGAs. This combination of tools ensured that the design not only met theoretical specifications but also practical implementation criteria.

Through this project, I gained hands-on experience in hardware description languages, simulation and synthesis tools, and the complete design flow from concept to implementation. The successful completion of this project demonstrated the functionality and efficiency of the AHB to APB bridge, highlighting its importance in system-on-chip (SoC) architectures and enhancing my skills in digital design and verification.