

✓ 运行说明

修改cp.sh运行脚本generated的相对路径, 运行./cp.sh文件以及文件夹自动拷贝到 `/chipyard/sims/vcs/generated-src/chipyard.TestHarness.LargeBoomConfig` 文件夹下



如果需要替换函数以及文件, 将需要替换的模块从 `chipyard.TestHarness.LargeBoomConfig.top.v`中删除, 然后运行以下代码即可

```
make debug CONFIG=LargeBoomConfig
```

✓ 工作说明

Dcache完成模块

Dcache中除了mshrs模块未完成测试, 其他模块均已完成测试

 dcache (BoomNonBlockingDCache)	Module
  wb_io_req (DecoupledIF)	Interface
  wb_io_release (DecoupledIF)	Interface
  wb_io_meta_read (DecoupledIF)	Interface
  wb_io_lsu_release (DecoupledIF)	Interface
  wb_io_idx (ValidIF)	Interface
  wb_io_data_req (DecoupledIF)	Interface
  prober_io_wb_req (DecoupledIF)	Interface
  prober_io_state (ValidIF)	Interface
  prober_io_req (DecoupledIF)	Interface
  prober_io_rep (DecoupledIF)	Interface
  prober_io_meta_write (DecoupledIF)	Interface
  prober_io_meta_read (DecoupledIF)	Interface
  prober_io_lsu_release (DecoupledIF)	Interface
 wbArb (Arbiter_13)	Module
  wb (BoomWriteBackUnit)	Module
 prober (BoomProbeUnit)	Module
  mshrs (BoomMSHRFile)	Module
  meta_0 (L1MetadataArray)	Module
 metaWriteArb (Arbiter_9)	Module
 metaReadArb (Arbiter_10)	Module
 lsu_release_arb (Arbiter_14)	Module
 lfsr_prng (MaxPeriodFibonacciLFSR_1)	Module
 dataWriteArb (Arbiter_11)	Module
 dataReadArb (Arbiter_12)	Module
  data (BoomDuplicatedDataArray)	Module
 amoalu (AMOALU)	Module

未测试

未测试

Icache完成模块

Icache中所有模块均已完成

- [-] icache (ICache)
 - [+] {io_resp (ValidSTIF)
 - [+] {io_req (DecoupledIF)
 - [+] {auto_d (DecoupledIF)
 - [+] {auto_a (DecoupledIF)
 - [-] cache (ICacheModule)
 - unnamed\$\$_0
 - b1Row
 - b0Row
 - [+] tag_array (SyncReadMem)
 - [+] lfsr16_icache (LFSR16)
 - [+] genblk3[7].data_array1 (SyncReadMemNo...
 - [+] genblk3[7].data_array0 (SyncReadMemNo...
 - [+] genblk3[6].data_array1 (SyncReadMemNo...
 - [+] genblk3[6].data_array0 (SyncReadMemNo...
 - [+] genblk3[5].data_array1 (SyncReadMemNo...
 - [+] genblk3[5].data_array0 (SyncReadMemNo...
 - [+] genblk3[4].data_array1 (SyncReadMemNo...
 - [+] genblk3[4].data_array0 (SyncReadMemNo...
 - [+] genblk3[3].data_array1 (SyncReadMemNo...
 - [+] genblk3[3].data_array0 (SyncReadMemNo...
 - [+] genblk3[2].data_array1 (SyncReadMemNo...
 - [+] genblk3[2].data_array0 (SyncReadMemNo...
 - [+] genblk3[1].data_array1 (SyncReadMemNo...
 - [+] genblk3[1].data_array0 (SyncReadMemNo...
 - [+] genblk3[0].data_array1 (SyncReadMemNo...
 - [+] genblk3[0].data_array0 (SyncReadMemNo...

未实现TileLink函数

boomnonblockingdcache未实现函数

```
1280
1281 //need to compile
1282
1283 // TLArbiter.lowest(edge, tl_out.c, wb.io.release, prober.io.rep)
1284
1285 // io.lsu.perf.release := edge.done(tl_out.c)
1286 // io.lsu.perf.acquire := edge.done(tl_out.a)
1287
1288 //-----
1289 // load data gen
1290
1291 // need to comp
1292 // val loadgen = (0 until memWidth).map { w =>
1293 //   new LoadGen(s2_req(w).uop.mem_size, s2_req(w).uop.mem_signed, s2_req(w).addr,
1294 //     s2_data_word(w), s2_sc && (w == 0).B, wordBytes)
1295 // }
1296
1297
1298 // amoalu.io.mask := new StoreGen(s2_req(0).uop.mem_size, s2_req(0).addr, 0.U, xLen/8).mask
1299 // amoalu.io.cmd := s2_req(0).uop.mem_cmd
1300 // amoalu.io.lhs := s2_data_word(0)
1301 // amoalu.io.rhs := s2_req(0).data
1302 logic [Has11CacheParameters::nWays-1:0] s3_way;
```

boommshrfile模块未实现函数

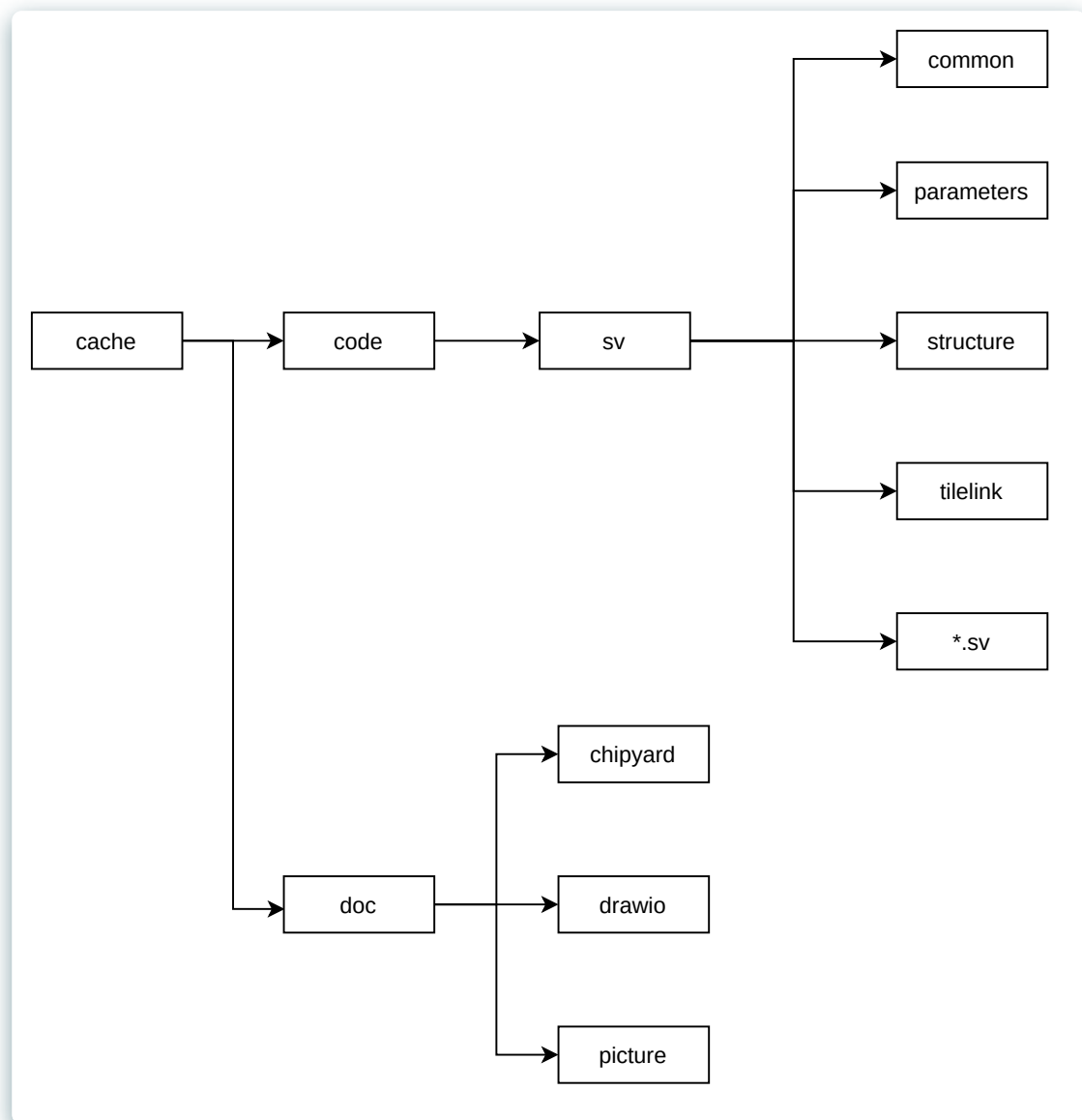
```
515
516 // mshr_alloc_idx := RegNext(AgePriorityEncoder(mshrs.map(m=>m.io.req_pri_rdy), mshr_head))
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
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
```

boommshr模块未实现函数

```
342
343 s_refill_resp: begin
344 // if (hasDataD(io_mem_grant.bits)) begin //need to imp hasdata
345 io_mem_grant.ready = io_lb_write.ready;
346 io_lb_write.valid = io_mem_grant.valid;
```

文件说明

cache的文件结构图如下所示



文件夹说明

- common文件文件夹保存常用的接口
 - decouple.sv为DecoupledIF接口
 - LSFR16.sv保存LSFR随机数发生器代码
 - mux.sv保存多路选择器代码
 - priorityencoder.sv保存优先编码器代码
 - syncreadmem.sv保存同步读写存储器模型, 需要编译成sram
 - valid.sv保存ValidIF接口
- parameters文件夹保存常用的常量参数文件
 - bundleparam.sv保存TLBundle使用的参数
 - cacheparam.sv cache中使用的参数
 - constparam.sv memory中使用的常量参数
 - param.sv使用的全局产量参数
- structure文件夹, 主要包含各种结构体

- bundleST.sv TLBUndle结构体, A, B, C, D, E通道结构体信息
 - executionST.sv指令执行相关结构体, 主要包含是否分支预测错误打断信息
 - hellacacheST.sv权限读写结构体
 - icacheST.sv ICache读写结构体
 - lsuST.sv LSU传递来的结构体
 - microopST.sv 微操作指令结构体
 - mshrST.sv mshr单元使用到的结构体
 - nbdcacheST.sv 数据读写结构体
 - tilelinkST.sv TileLink中函数使用到的结构体
- TileLink文件夹, 包含使用到的TileLink函数
 - arbiter.sv 仲裁器
 - edge.sv TileLink抽象边中使用到的函数
 - metadata.sv 权限转换函数

文件说明

- BoomDuplicatedDataArray.sv Dcache中使用的Data sram数组
- boommshr.sv mshr的状态机模块
- boommshrfile mshr模块与Dcache模块的连接模块
- boomnonblockingdcahe.sv Dcache的主文件
- hellacache.sv 权限数据存储模块
- icache.sv icache模块实现
- prefetch.sv 预取模块实现
- proberunit.sv 一致性模块维护模块
- until.sv 队列模块实现
- writebackunit.sv 写回模块实现

设计文档 矢量图源文件