CSE312: Electronic Design Automation
Fall 2022

Project (2): Verilog Linting
(DetErr the Linter)

Presented to:

Dr. Eman Mohamed

Eng. Adham

Submitted by:

Abdulrahman Maged Abdulrahman: 20P7467

Hazem Zaki Aboukhalil : 20P7516

Mariam Hany Fouad: 20P8966

Mina George Fawzy Girgis: 20P4198

Shady Emad Sabry: 20P7239

Matthew sherif shalaby : 20p6785

Karim Bassel Samir: 20P6794

fady fady fouad : 20p7341

Table of Contents

Parser

```
static String[] ParseVerilogCode(String filename) throws FileNotFoundException {
    File file = new File(filename);
    String[] line = new String[100];
    int i = 0;
    Scanner scanner = new Scanner(file);
    while (scanner.hasNextLine()) {
        line[i] = scanner.nextLine();
        i++;
    }
    return line;
}
```

Static Checker Engine

Main

```java
public class JavaApplication2 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args)throws FileNotFoundException, IOException, ClassNotFoundException {
        NewJFrame1 start = new NewJFrame1();
        start.setVisible(true);

//        File file = new File( "/Users/fady/Desktop/data.txt");
```

Note: We have added GUI to make the experience even better! Check the GUI code after the static engine checker code.

Arithmetic overflow function

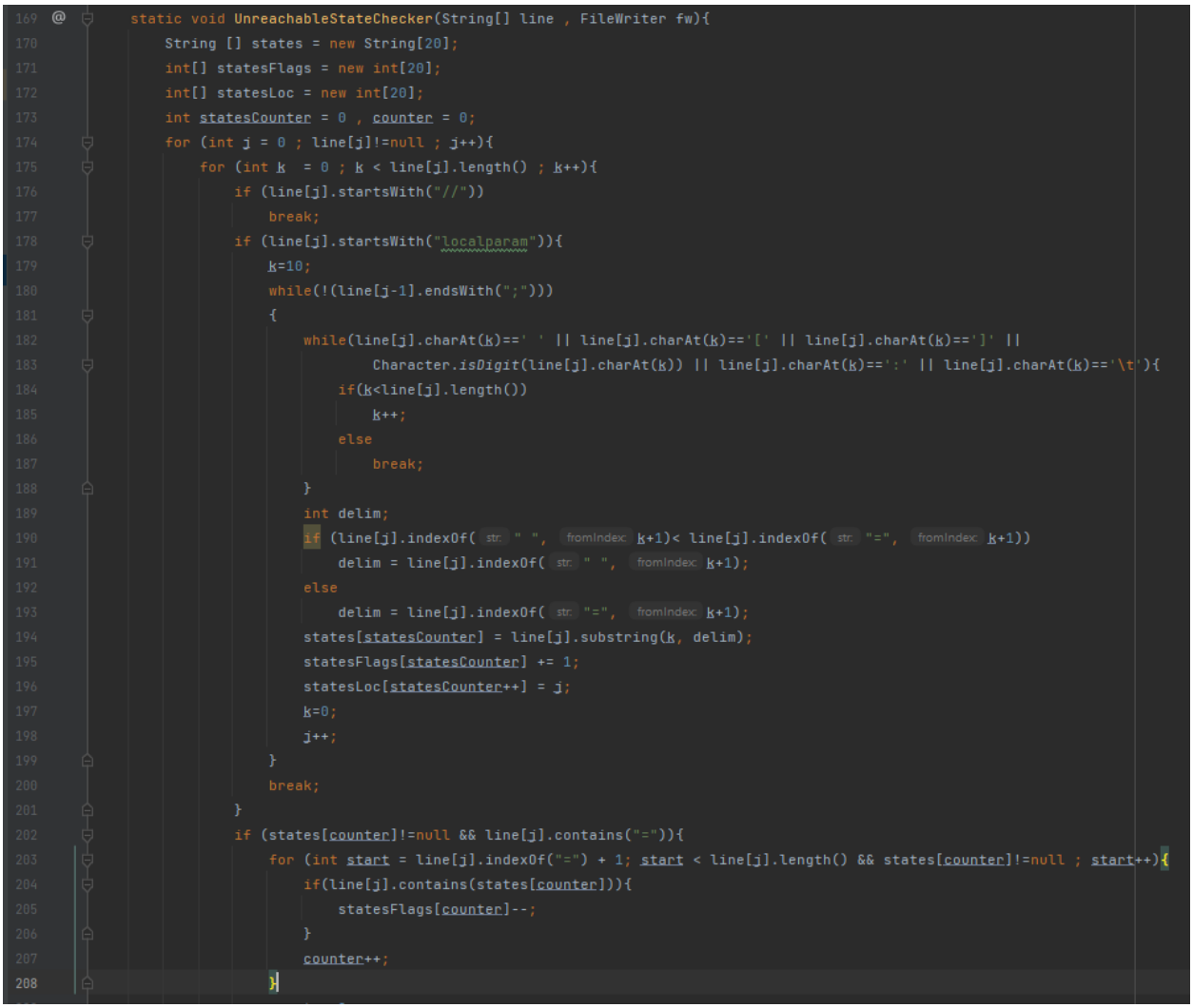
```java
@   static void arithmeticOperation(String[] line, FileWriter fw) throws IOException {

        String[] variablesArray = new String[100];
        String[] numbersArray = new String[10000];
        char[] chararray1 = new char[10];
        char[] chararray2 = new char[10];
        int indexVariablesArray = 0;
        int indexNumbersArray = 0;
        String[] tempt2 = new String[line.length];
        String[] temp = new String[line.length];

        for (int i = 1; i < line.length; i++) {
            tempt2[0] += line[i];
        }

        temp = tempt2[0].split( regex: " ");

        for (int i = 0; i < temp.length; i++) {
            temp[i] = temp[i].replaceAll( regex: "[\\[\\](){},;]", replacement: "");
            //            temp[i].split(" ");
        }

        for (int m = 0; m < temp.length; m++) {
            if (temp[m].contains("'b")) {

                temp[m] = temp[m].substring(3, temp[m].length());

                variablesArray[indexVariablesArray] = temp[m - 2];
                numbersArray[indexNumbersArray] = temp[m];

                //                chararray = numbersArray[indexVariablesArray].toCharArray();

                indexVariablesArray++;
```

```
352                 indexNumbersArray++;
353
354             }
355         }
356         int counterrros = 0;
357         fw.append("Arithmetic overflow:\n");
358
359         int indexoffirst = 0, indexofsecond = 0;
360         int r1 = 0, r1Max = 0, r2Max = 0, r2 = 0, carry = 0;
361         for (int overflow = 0; overflow < temp.length; overflow++) {
362             if (temp[overflow].contains("+")) {
363                 for (int k = 0; k < variablesArray.length; k++) {
364                     if (temp[overflow - 1].equals(variablesArray[k])) {
365                         indexoffirst = k;
366                         chararray1 = numbersArray[indexoffirst].toCharArray();
367                     }
368                     if (temp[overflow + 1].equals(variablesArray[k])) {
369                         indexofsecond = k;
370                         chararray2 = numbersArray[indexofsecond].toCharArray();
371                     }
372                 }
373
374                 String result = null;
375                 for (int i = 0; i < chararray1.length; i++) {
376                     //                    System.out.println(chararray1[i]);
377
378                     if (chararray1[chararray1.length - 1 - i] == '1') {
379                         r1 += Math.pow(2, i);
380                     }
381                     r1Max += Math.pow(2, i);
382
383                 }
384                 for (int j = 0; j < chararray2.length; j++) {
```

```
            for (int j = 0; j < chararray2.length; j++) {
                if (chararray2[chararray2.length - 1 - j] == '1') {
                    r2 += Math.pow(2, j);
                }
                r2Max += Math.pow(2, j);
            }
            if (((r1 + r2) > Math.max(r1Max, r2Max))) {
                int of = 0;
                while (line[of] != null) {
                    if (line[of].contains("+") && line[of].contains(variablesArray[indexoffirst]) &&
                            line[of].contains(variablesArray[indexofsecond])) {
                        try {
                            int lineno = of + 1;
                            fw.append("\tArithmetic overflow at line " + (lineno));
                            fw.append(" between " + variablesArray[indexoffirst] + " and " + variablesArray[indexofsecond] + "\n");
                            //fw.close();
                            counterrros++;
                        } catch (IOException e) {
                            System.out.println("IO Error!");
                        }
                        break;
                    }
                    of ++;
                }
            }
        }
    }
if(counterrros==0)
{
    fw.append("\tNo arithmetic overflow found");
    fw.append("\n");
}
}
```

```
    static void CheckParallelFullCase(String [] lines, FileWriter fw)throws IOException{
        String [] words = new String[100000];
        String [] linesplit = new String[100000];
        String [] Errors = new String[100000];
        int [] caselinenum = new int[100000];
        int i=0,j=0,k=0,z=0,e=0;
        while(lines[i] != null){
            lines[i] = lines[i].trim();
            //System.out.println(lines[i]);
            linesplit = lines[i].split( regex: " ");
            for(String a:linesplit){
                if(a.equals("case")){
                    caselinenum[z]=i;
                    z++;
                }
                words[j] = a;
                j++;
            }
            i++;
        }//split words
        int [] caseindex = new int[100000];
        i=0;
        while(words[i]!=null){
            if(words[i].equals("case")){
                //System.out.println(words[i]);

                caseindex[k]=i;
                k++;
            }
```

Unreachable FSM State function



```
static void UnreachableStateChecker(String[] line , FileWriter fw){
    String [] states = new String[20];
    int[] statesFlags = new int[20];
    int[] statesLoc = new int[20];
    int statesCounter = 0 , counter = 0;
    for (int j = 0 ; line[j]!=null ; j++){
        for (int k  = 0 ; k < line[j].length() ; k++){
            if (line[j].startsWith("//"))
                break;
            if (line[j].startsWith("localparam")){
                k=10;
                while(!(line[j-1].endsWith(";")))
                {
                    while(line[j].charAt(k)==' ' || line[j].charAt(k)=='[' || line[j].charAt(k)==']' ||
                            Character.isDigit(line[j].charAt(k)) || line[j].charAt(k)==':' || line[j].charAt(k)=='\t'){
                        if(k<line[j].length())
                            k++;
                        else
                            break;
                    }
                    int delim;
                    if (line[j].indexOf( str: " ",  fromIndex: k+1)< line[j].indexOf( str: "=",  fromIndex: k+1))
                        delim = line[j].indexOf( str: " ",  fromIndex: k+1);
                    else
                        delim = line[j].indexOf( str: "=",  fromIndex: k+1);
                    states[statesCounter] = line[j].substring(k, delim);
                    statesFlags[statesCounter] += 1;
                    statesLoc[statesCounter++] = j;
                    k=0;
                    j++;
                }
                break;
            }
            if (states[counter]!=null && line[j].contains("=")){
                for (int start = line[j].indexOf("=") + 1; start < line[j].length() && states[counter]!=null ; start++){
                    if(line[j].contains(states[counter])){
                        statesFlags[counter]--;
                    }
                    counter++;
                }
```

```
208                     }
209                     counter=0;
210                     break;
211                 }
212             }
213         }
214         try{
215             //FileWriter fw = new FileWriter(directory);
216             fw.append("FSM States: \n");
217             int flag = 0;
218             for (int s = 0 ; states[s]!=null ; s++){
219                 if(statesFlags[s]==1){
220                     fw.append("\t");
221                     fw.append(states[s]);
222                     fw.append(" initialized in line ");
223                     fw.append((char) statesLoc[s]);
224                     fw.append(" is never reached.\n");
225                     flag++;
226                 }
227             }
228             if (flag == 0)
229                 fw.append("\t No unreachable states.\n");
230             //fw.close();
231             fw.append("\n");
232         }
233         catch (IOException e){
234             System.out.println("IO Error!");
235         }
236 //        System.out.println(statesCounter);
237 //        for (int f  = 0 ; states[f]!=null ; f++){
238 //            System.out.println(states[f]);
239 //            System.out.println(statesFlags[f]);
240 //        }
241     }
```

Un-initialized Register function

```
242 @   static void unitiliazedRegs(String []line, FileWriter fw) throws IOException {
243         String[] regs = new String[100];
244         String[] plusRegs = new String[100];
245         int indexRegs = 0;
246         String[] tempt2 = new String[line.length];
247         String[] temp = new String[line.length];
248         for (int i = 1; i < line.length; i++) {
249             tempt2[0] += line[i];
250         }
251         temp = tempt2[0].split( regex: " ");
252 
253         for (int i = 0; i < temp.length; i++) {
254             temp[i] = temp[i].replaceAll( regex: "[\\[\\](){},;]",  replacement: "");
255             //            temp[i].split(" ");
256         }
257         int vb = 0;
258         for (int t = 1; t < temp.length; t++) {
259             if (temp[t].equals("=") || temp[t].equals("<=")) {
260                 plusRegs[vb] = temp[t - 1];
261                 vb++;
262             }
263         }
264         for (int m = 0; m < temp.length; m++) {
265 
266             if (temp[m].contains("reg")) {
267                 regs[indexRegs] = temp[m + 1];
268                 indexRegs++;
269 
270             }
271         }
272         boolean found = false;
273         fw.append("Uninitialized registers: \n");
274         boolean flag = false;
275         for (int i = 0; i < regs.length; i++) {
276             found = false;
277             for (int j = 0; j < plusRegs.length; j++) {
278 
279                 try {
280                     if (regs[i].equals(plusRegs[j]) || line[i].contains(plusRegs[j])) {
281                         found = true;
282                     }
```

```
                }
            } catch (Exception e) {
                continue;
            }

        }
        if (regs[i] == null) {
            continue;
        }
        if (!found) {
            int of = 0;
            while (line[of] != null) {
                if ((line[of].contains(regs[i]))) {
                    try {
                        int lineno = of + 1;
                        // System.out.print("arithmetic overflow at line " + (lineno));
                        //System.out.println(regs[i] + " is not initialized at line " + (lineno));

                        fw.append("\t" + regs[i] + " is not initialized at line " + (lineno) + "\n");
                        //fw.close();
                        flag = true;

                    } catch (IOException e) {
                        System.out.println("IO Error!");
                    }

                }
                of++;
            }
        }
    }
    if(!flag)fw.append("\tNo uninitialized registers found.\n");
    fw.append("\n");
}
1 usage
```

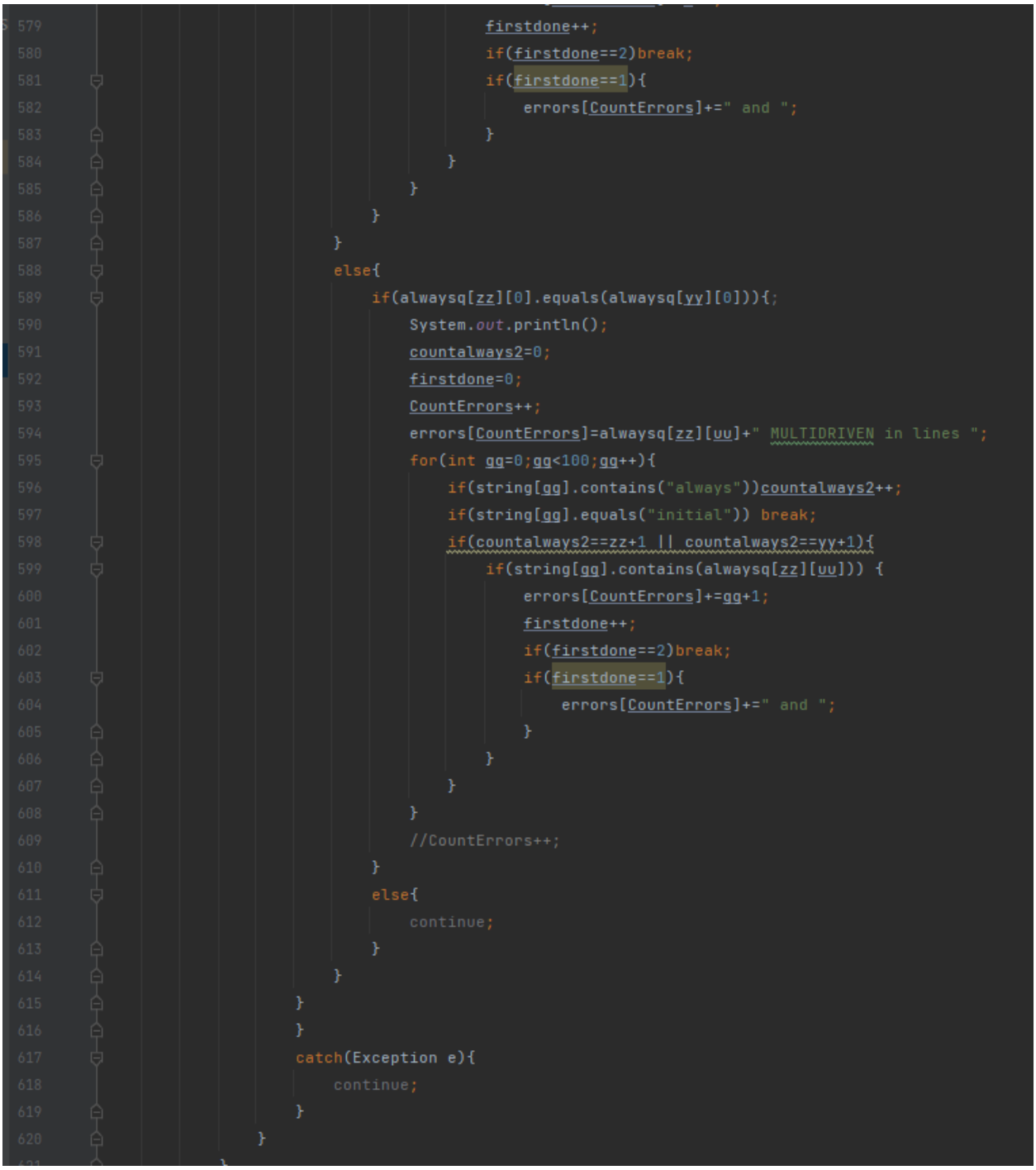
Multi-Driven Bus/Register function

```
415 @   static void CheckMultipleBusses(String [] string, FileWriter fw) throws IOException{
416     String [][]intialw= new String[2][100];
417     String [][]alwaysq= new String[2][100];
418     int firstdone=0;
419     int ww=0;
420     int w=0;
421     int al=0;
422     int all=0;
423     int qq=0;
424     int q=0;
425     String []words=new String[10000];
426     String []linesplits=new String[10000];
427     String []errors=new String[10000];
428     int CountErrors=-1;
429     int i=0;
430     int j=0;
431     int countintial=0;
432     int countintial2=0;
433     int countalways=0;
434     int countalways2=0;
435     while(string[i]!=null) {
436         string[i]=string[i].trim();
437         i++;
438     }
439     i=0;
440     while(string[i]!=null){
441         string[i]=string[i].trim();
442        // System.out.println(string[i]);
443         linesplits=string[i].split( regex: " ");
444         for(String a: linesplits){
445             words[j]=a;
446             j++;
447         }
448         i++;
449     }
450     for(int z=0;z<words.length;z++){
451         if(words[z]==null) break;
452         words[z]=words[z].trim();
453         if((words[z].length()==0) || (words[z].length()==1)){
454             for(int f=z;f<words.length;f++){
455                 if(words[f]==null) break;
456                 words[f]=words[f+1];
```

```
456                 words[f]=words[f+1];
457             }
458         }
459     }
460     for(int z=0;z<words.length;z++){
461         try{
462         words[z]=words[z].replace( target: "(", replacement: "");
463         words[z]=words[z].replace( target: ")", replacement: "");
464         words[z]=words[z].replace( target: "@", replacement: "");
465         words[z]=words[z].replace( target: "{", replacement: "");
466         words[z]=words[z].replace( target: "}", replacement: "");
467         words[z]=words[z].replace( target: ",", replacement: "");
468         if(words[z]==null) break;}
469         catch(Exception e){
470             continue;
471         }
472     }
473     for(int y=0;y<1000;y++){
474         try{
475         if((words[y]).equals("initial")){
476             for(int g=y+2;g<1000;g++){
477                 if(words[g].equals("if")) countintial++;
478                 if(words[g].equals("end")) {
479                     if(countintial==0) break;
480                     countintial--;
481                 }
482                 if(words[g].equals("case"))break;
483                 if(words[g].equals("if"))continue;
484                 if(words[g].contains("'")) continue;
485                 if(words[g].contains("#")) continue;
486                 if(words[g-1].equals("<="))continue;
487                 if(words[g-1].equals("="))continue;
488                 if(words[g+1].equals("<="))continue;
489                 if(words[g].equals("end"))continue;
490                 if(words[g].equals("begin"))continue;
491                 if(words[g].equals("="))continue;
492                 if(words[g].equals("<="))continue;
493                 if(words[g].equals("else"))continue;
494                 if(words[g].equals("&&"))continue;
495
496                 intialw[ww][w]=words[g];
497                 w++;
```

```
498             }
499             ww++;
500             w=0;
501         }
502         if((words[y]).equals("always")){
503             for(int g=y+2;g<1000;g++){
504 
505                 if(words[g].equals("if")) countalways++;
506                 if(words[g].equals("end")) {
507                     if(countalways==0) break;
508                     countalways--;
509                 }
510                 if(words[g].equals("case"))break;
511                 if(words[g].equals("if"))continue;
512             //    if(words[g].contains("1")) continue;
513                 if(words[g].contains("'")) continue;
514                 if(words[g-1].equals("<="))continue;
515                 if(words[g-1].equals("="))continue;
516                 if(words[g+1].equals("<="))continue;
517                 if(words[g].equals("end"))continue;
518                 if(words[g].equals("begin"))continue;
519                 if(words[g].equals("="))continue;
520                 if(words[g].equals("<="))continue;
521                 if(words[g].equals("else"))continue;
522                 if(words[g].equals("&&"))continue;
523 
524                 alwaysq[all][al]=words[g];
525                 al++;
526             }
527             all++;
528             al=0;
529         }
530         }
531         catch(Exception e){
532             continue;
533         }
534     }
535     /* System.out.println("****");
536      for(int rr=0;rr<2;rr++){
537         for(int kkk=0;kkk<100;kkk++){
538             if(intialw[rr][kkk]==null)continue;
539             try{
```

```
539                 try{
540                 System.out.println(intialw[rr][kkk]);
541                 }
542                 catch(Exception e){
543                 continue;
544             }
545             }
546         }
547         System.out.println("****");
548 
549         for(int rr=0;rr<2;rr++){
550             //System.out.println("be5" +alwaysq[rr][3]+" be5 ");
551             for(int kkk=0;kkk<100;kkk++){
552                 if(alwaysq[rr][kkk]==null)continue;
553                 try{
554                 System.out.println(alwaysq[rr][kkk]);
555                 }
556                 catch(Exception e){
557                 continue;
558                 }
559             }
560         }*/
561         for(int yy=0;yy<2;yy++){
562             for(int zz=yy;zz<2;zz++){
563                 for(int uu =1;uu<50;uu++){
564                     for(int ii=uu;ii<50;ii++){
565                         if(alwaysq[yy][ii]==null)continue;
566                         if(ii==uu && yy==zz)continue;
567                         try{
568                         if(alwaysq[yy][ii].equals(alwaysq[zz][uu])) {
569                             if(zz==yy){CountErrors++;
570                                 countalways=0;
571                                 firstdone=0;
572                                 errors[CountErrors]=alwaysq[zz][uu]+" MULTIDRIVEN in lines ";
573                                 for(int x=0;x<50;x++){
574                                     if(string[x].equals("initial")) break;
575                                     if(string[x].contains("always")) ++countalways;
576                                     if(countalways==zz+1){
577                                         if(string[x].contains(alwaysq[zz][uu])) {
578                                             errors[CountErrors]+=x+1;
579                                             firstdone++;
```



```
                                    firstdone++;
                                    if(firstdone==2)break;
                                    if(firstdone==1){
                                        errors[CountErrors]+=" and ";
                                    }
                                }
                            }
                        }
                    else{
                        if(alwaysq[zz][0].equals(alwaysq[yy][0])){;
                            System.out.println();
                            countalways2=0;
                            firstdone=0;
                            CountErrors++;
                            errors[CountErrors]=alwaysq[zz][uu]+" MULTIDRIVEN in lines ";
                            for(int gg=0;gg<100;gg++){
                                if(string[gg].contains("always"))countalways2++;
                                if(string[gg].equals("initial")) break;
                                if(countalways2==zz+1 || countalways2==yy+1){
                                    if(string[gg].contains(alwaysq[zz][uu])) {
                                        errors[CountErrors]+=gg+1;
                                        firstdone++;
                                        if(firstdone==2)break;
                                        if(firstdone==1){
                                            errors[CountErrors]+=" and ";
                                        }
                                    }
                                }
                            }
                            //CountErrors++;
                        }
                        else{
                            continue;
                        }
                    }
                }
                }
                catch(Exception e){
                    continue;
                }
            }
```

```
621             }
622         }
623     }
624     for(int yy=0;yy<2;yy++){
625         for(int zz=yy;zz<2;zz++){
626             for(int uu =0;uu<40;uu++){
627                 for(int ii=uu;ii<40;ii++){
628                     if(intialw[yy][ii]==null)continue;
629                     if(ii==uu && yy==zz)continue;
630                     try{
631                     if(intialw[yy][ii].equals(intialw[zz][uu])) {
632                         System.out.println();
633                         if(zz==yy){
634                             CountErrors++;
635                             countintial=0;
636                             firstdone=0;
637                             errors[CountErrors]=intialw[zz][uu]+" MULTIDRIVEN in lines ";
638                             for(int x=0;x<50;x++){
639                                 if(string[x].equals("always"))break;
640                                 if(string[x].contains("initial")) ++countintial;
641                                 if(countintial==zz+1){
642                                     if(string[x].contains(intialw[zz][uu])) {
643                                         errors[CountErrors]+=x+1;
644                                         firstdone++;
645                                         if(firstdone==2)break;
646                                         if(firstdone==1){
647                                             errors[CountErrors]+=" and ";
648                                         }
649                                     }
650                                 }
651                             }
652                         }
653                         else{
654                                 System.out.println();
655                                 countintial2=0;
656                                 firstdone=0;
657                                 CountErrors++;
658                                 errors[CountErrors]=intialw[zz][uu]+" MULTIDRIVEN in lines ";
659                                 for(int gg=0;gg<10000;gg++){
660                                     if(string[gg].equals("always"))break;
661                                     if(string[gg].contains("initial"))countintial2++;
662                                     if(countintial2==zz+1 || countintial2==yy+1){
663                                         if(string[gg].contains(intialw[zz][uu])) {
```

```
                                    if(string[gg].contains(intialw[zz][uu])) {
                                        errors[CountErrors]+=gg+1;
                                        firstdone++;
                                        if(firstdone==2)break;
                                        if(firstdone==1){
                                            errors[CountErrors]+=" and ";
                                        }
                                    }
                                }
                            }
                        }
                        }
                        catch(Exception e){
                            continue;
                        }
                    }
                }
            }
        CountErrors=0;
            fw.append("Multi-driven buses/registers:\n");
        while(errors[CountErrors]!=null){
                fw.append("\t" + errors[CountErrors++] + " \n");
            }
            if (CountErrors==0){
                fw.append("\tNo multi-driven busses/registers.\n");
            }
            fw.append("\n");
        }
    }
```

Non-Full/Parallel Case function

```
static void CheckParallelFullCase(String [] lines, FileWriter fw)throws IOException{
    String [] words = new String[100000];
    String [] linesplit = new String[100000];
    String [] Errors = new String[100000];
    int [] caselinenum = new int[100000];
    int i=0,j=0,k=0,z=0,e=0;
    while(lines[i] != null){
        lines[i] = lines[i].trim();
        //System.out.println(lines[i]);
        linesplit = lines[i].split( regex " ");
        for(String a:linesplit){
            if(a.equals("case")){
                caselinenum[z]=i;
                z++;
            }
            words[j] = a;
            j++;
        }
        i++;
    }//split words
    int [] caseindex = new int[100000];
    i=0;
    while(words[i]!=null){
        if(words[i].equals("case")){
            //System.out.println(words[i]);

            caseindex[k]=i;
            k++;
        }
        i++;
    }//get case index
    i=0;j=0;
    while(caseindex[i]!= 0){//check full case
        if(words[caseindex[i]+2].equals("full_case") || words[caseindex[i]+3].equals("full_case")
        || words[caseindex[i]+4].equals("full_case") ||
        words[caseindex[i]+5].equals("full_case") ||
        words[caseindex[i]+6].equals("full_case")){//check for synopsis fullcase
            i++;
            continue;
```

```
68              }
69
70              j=caseindex[i];
71              while(!(words[j].equals("endcase"))){//check for default
72                  if(words[j].equals("default"))break;
73                  j++;
74              }
75              if(words[j].equals("default")){
76                  i++;
77                  continue;
78              }
79              String [] outputs = new String[100000];
80              int c=0;
81              j=caseindex[i];
82              int maxout=0;
83              while(!(words[j].equals("endcase"))){//search for output in case
84              if(words[j].equals("=") || words[j].equals("<=")){
85                  outputs[c] = words[j-1];
86                  //System.out.println(outputs[c]);
87                  c++;
88              }
89                  j++;
90                  maxout=c;
91              }
92              String [] initializations = new String[100000];
93               c=0;
94               int maxin=0;
95              j=caseindex[i];
96              while(!(words[j].equals("module"))){//search for output initialization
97              if(words[j].equals("=") || words[j].equals("<=")){
98                  initializations[c] = words[j-1];
99                  //System.out.println(initializations[c]);
100                 c++;
101             }
102                 j--;
103                 maxin=c;
104                 if(j<0)break;
105             }
106             int flag=0;
107             for(int m=0;m<maxout;m++){
108                 flag=0;
109                 for(int b=0;b<maxin;b++){
110                     if(outputs[m].equals(initializations[b])){
```

```
                    if(outputs[m].equals(initializations[b])){
                        //System.out.println("ok");
                        flag=1;
                        break;
                    }
                }
                if(flag==0){
                    break;
                }
            }
            if(flag==1){
                i++;
                continue;
            }
            //System.out.println("not Full case at line "+(caselinenum[i]+1));
            Errors[e] = "not Full case at line "+(caselinenum[i]+1);
            e++;
            i++;
        }//full case check

        i=0;j=0;
        while(caseindex[i]!= 0){
            if(words[caseindex[i]+2].equals("parallel_case") || words[caseindex[i]+3].equals("parallel_case")
            || words[caseindex[i]+4].equals("parallel_case") ||
            words[caseindex[i]+5].equals("parallel_case")||
            words[caseindex[i]+6].equals("parallel_case")){//check for synopsis fullcase
                i++;
                continue;
            }
            String [] cases = new String[100000];
            j=caseindex[i];
            int c=0;
            while(!(words[j].equals("endcase"))){//check for cases
                if(words[j].equals(":")){
                    cases[c] = words[j-1];
                    c++;
                }
                j++;
            }
            for(int o=0;o<c;o++){
                for(int l=o+1;l<c;l++){
                    if(cases[o].equals(cases[l])){
                        //System.out.println("not parallel case at line "+(caselinenum[i]+1));
```

```
153                         Errors[e] = "not parallel case at line "+(caselinenum[i]+1);
154                         e++;
155                     }
156                 }
157             }
158             i++;
159         }//parallel case check
160         //FileWriter fw = new FileWriter(path);
161         fw.append("Not Full/Parallel Cases: \n");
162         for (int s = 0 ; Errors[s]!=null ; s++){
163             fw.append("\t"+Errors[s]+" \n");
164         }
165         fw.append("\n");
166         //fw.close();
167 
168     }
```

GUI CODE (includes function that outputs in text file)

```
20 usages
public class NewJFrame1 extends javax.swing.JFrame {

    /**
     * Creates new form NewJFrame1
     */
    4 usages
    String DUTDir;
    4 usages
    String RepDir;
    2 usages
    public NewJFrame1() { initComponents(); }

    1 usage
    public void setDUTDir(File dir) { DUTDir=dir.getAbsolutePath(); }

    1 usage
    public void setRepDir(File dir) { RepDir=dir.getAbsolutePath(); }

    1 usage
    public void setTextBox1(String s) { jTextField1.setText(s); }

    1 usage
    public void setTextBox2(String s) { jTextField2.setText(s); }
    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    /unchecked/
    Generated Code

    1 usage
    private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jTextField1ActionPerformed

    }//GEN-LAST:event_jTextField1ActionPerformed

    1 usage
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton2ActionPerformed
        if(DUTDir==null)DUTDir = jTextField1.getText();
        if(RepDir==null)RepDir = jTextField2.getText();
//        jTextField1.setText(DUTDir);
//        jTextField2.setText(RepDir);
        File file = new File( DUTDir);
        String [] line = new String[100000];
```

```
202         String [] line = new String[100000];
203         int i=0;
204         Scanner scanner = null;
205         try {
206             scanner = new Scanner(file);
207         } catch (FileNotFoundException ex) {
208             Logger.getLogger(NewJFrame1.class.getName()).log(Level.SEVERE,  msg: null, ex);
209         }
210         while (scanner.hasNextLine()) {
211            line[i] = scanner.nextLine();
212            //System.out.println(line[i]);
213             i++;
214         }
215         FileWriter fw = null;
216         try {
217             fw = new FileWriter(RepDir);
218         } catch (IOException ex) {
219             Logger.getLogger(NewJFrame1.class.getName()).log(Level.SEVERE,  msg: null, ex);
220         }
221         if (jCheckBox1.isSelected())
222             try {
223                 arithmeticOperation(line, fw);
224         } catch (IOException ex) {
225             Logger.getLogger(NewJFrame1.class.getName()).log(Level.SEVERE,  msg: null, ex);
226         }
227         if (jCheckBox2.isSelected())
228             UnreachableStateChecker(line, fw);
229         if (jCheckBox3.isSelected())
230             try {
231                 unitiliazedRegs(line, fw);
232         } catch (IOException ex) {
233             Logger.getLogger(NewJFrame1.class.getName()).log(Level.SEVERE,  msg: null, ex);
234         }
235         if (jCheckBox4.isSelected()){
236             try {
237                 CheckMultipleBusses(line, fw);
238             } catch (IOException ex) {
239                 Logger.getLogger(NewJFrame1.class.getName()).log(Level.SEVERE,  msg: null, ex);
240             }
241         }
242         if (jCheckBox5.isSelected()){
243             try {
244                 CheckParallelFullCase(line, fw);
245             } catch (IOException ex) {
246                 Logger.getLogger(NewJFrame1.class.getName()).log(Level.SEVERE,  msg: null, ex);
247             }
```

```
            }
        }
        try {
            fw.close();
        } catch (IOException ex) {
            Logger.getLogger(NewJFrame1.class.getName()).log(Level.SEVERE,  msg: null, ex);
        }
        Done d = new Done( f: this);
        d.setVisible(true);
        this.setVisible(false);
    }//GEN-LAST:event_jButton2ActionPerformed

    1 usage
    private void jCheckBox1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jCheckBox1ActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_jCheckBox1ActionPerformed

    1 usage
    private void jCheckBox2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jCheckBox2ActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_jCheckBox2ActionPerformed

    1 usage
    private void jCheckBox5ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jCheckBox5ActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_jCheckBox5ActionPerformed

    1 usage
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton1ActionPerformed
        NewJFrame f1 = new NewJFrame( f1: this);
        f1.setVisible(true);
    }//GEN-LAST:event_jButton1ActionPerformed

    1 usage
    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton3ActionPerformed
        // TODO add your handling code here:
        NewJFrame11 f2 = new NewJFrame11( f1: this);
        f2.setVisible(true);
    }//GEN-LAST:event_jButton3ActionPerformed

    1 usage
    private void jCheckBox4ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jCheckBox4ActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_jCheckBox4ActionPerformed
```

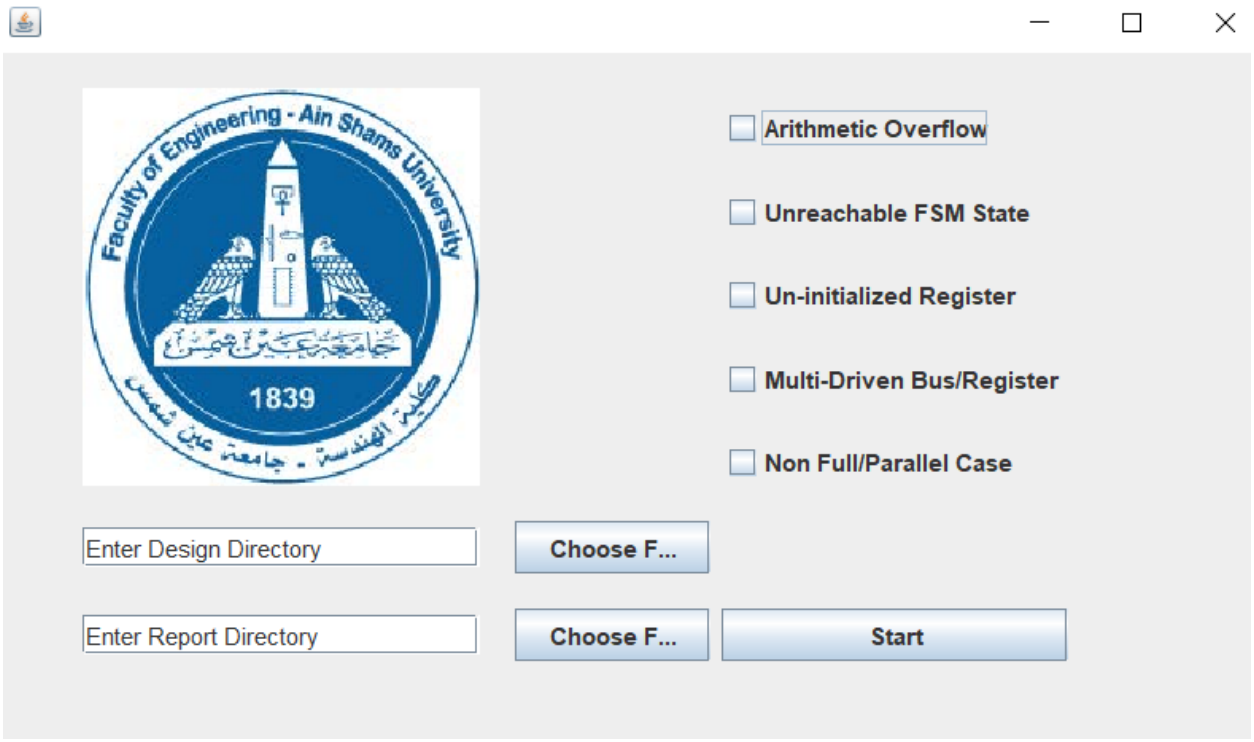
```
285
286         /**
287          * @param args the command line arguments
288          */
289         public static void main(String args[]) {
290             /* Set the Nimbus look and feel */
291             Look and feel setting code (optional)
312
313             /* Create and display the form */
314             java.awt.EventQueue.invokeLater(new Runnable() {
315                 public void run() { new NewJFrame1().setVisible(true); }
318             });
319         }
320
321         // Variables declaration - do not modify//GEN-BEGIN:variables
            5 usages
322         private javax.swing.JButton jButton1;
            5 usages
323         private javax.swing.JButton jButton2;
            5 usages
324         private javax.swing.JButton jButton3;
            6 usages
325         private javax.swing.JCheckBox jCheckBox1;
            6 usages
326         private javax.swing.JCheckBox jCheckBox2;
            5 usages
327         private javax.swing.JCheckBox jCheckBox3;
            6 usages
328         private javax.swing.JCheckBox jCheckBox4;
            6 usages
329         private javax.swing.JCheckBox jCheckBox5;
            5 usages
330         private javax.swing.JLabel jLabel1;
            7 usages
331         private javax.swing.JTextField jTextField1;
            6 usages
332         private javax.swing.JTextField jTextField2;
333         // End of variables declaration//GEN-END:variables
334     }
335
```

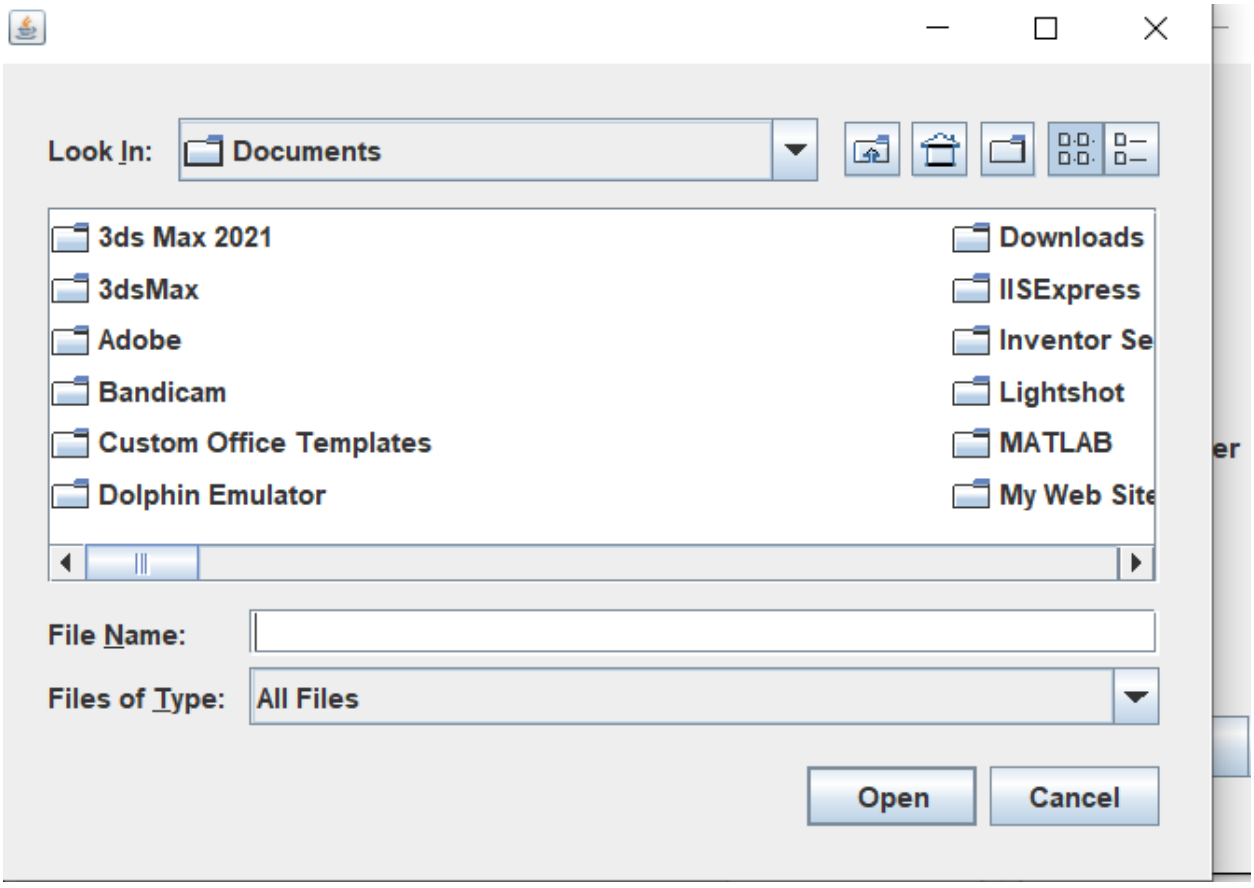
```
17      String Directory;
        3 usages
18      NewJFrame1 frame1;
        1 usage
19      public NewJFrame() { initComponents(); }
22      public NewJFrame(String dir) {
23          initComponents();
24          Directory = dir;
25      }
        1 usage
26      public NewJFrame(NewJFrame1 f1) {
27          initComponents();
28          frame1=f1;
29      }
30      /**
31       * This method is called from within the constructor to initialize the form.
32       * WARNING: Do NOT modify this code. The content of this method is always
33       * regenerated by the Form Editor.
34       */
35      /unchecked/
36      Generated Code
        1 usage
68      private void jFileChooser1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jFileChooser1ActionPerformed
69          // TODO add your handling code here:
70          File f = jFileChooser1.getSelectedFile();
71          if(f!=null){
72              frame1.setDUTDir(f);
73              frame1.setTextBox1(f.getAbsolutePath());
74          }
75          this.setVisible(false);
76      }//GEN-LAST:event_jFileChooser1ActionPerformed
77      public static void main(String args[]) {
78          /* Set the Nimbus look and feel */
79          Look and feel setting code (optional)
100         /* Create and display the form */
101         java.awt.EventQueue.invokeLater(new Runnable() {
102             public void run() {
103                 new NewJFrame().setVisible(true);
104             }
105         });
106     }
107     // Variables declaration - do not modify//GEN-BEGIN:variables
        5 usages
108     private javax.swing.JFileChooser jFileChooser1;
109     // End of variables declaration//GEN-END:variables
110 }
```

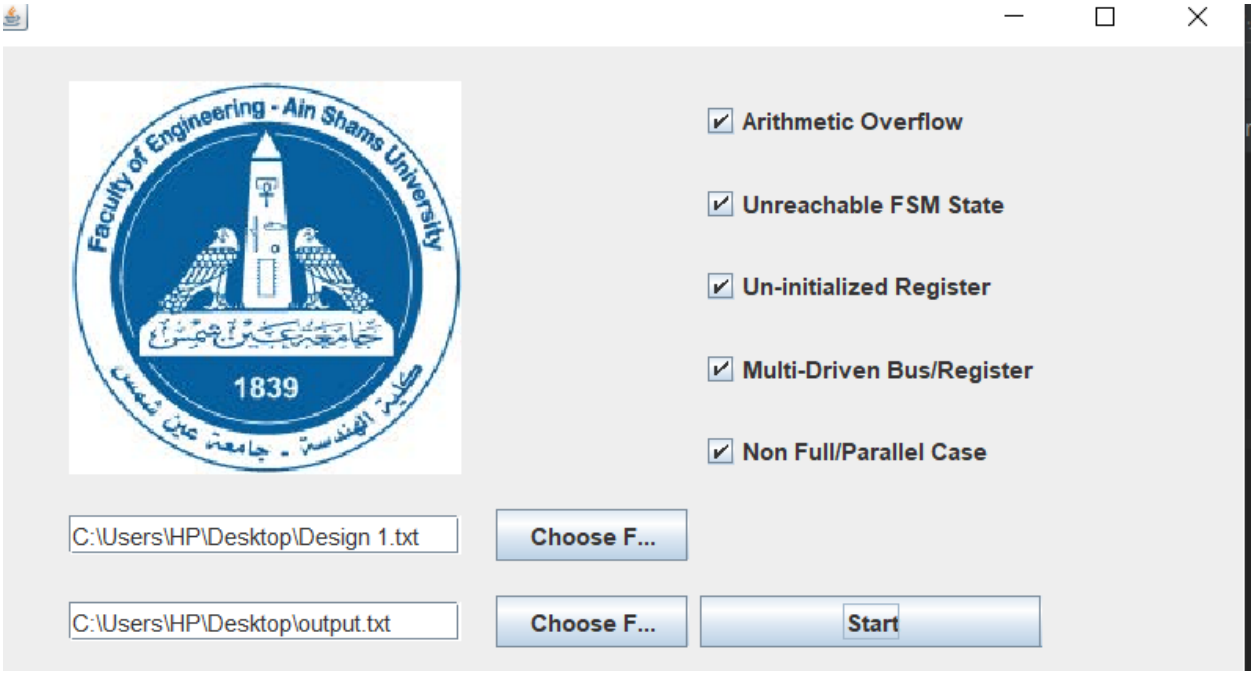
```
    public NewJFrame11() { initComponents(); }

    public NewJFrame11(String dir) {
        initComponents();
        Directory = dir;
    }
    1 usage
    public NewJFrame11(NewJFrame1 f1) {
        initComponents();
        frame1=f1;
    }
    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    /unchecked/
    Generated Code
    1 usage
    private void jFileChooser1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jFileChooser1ActionPerformed
        // TODO add your handling code here:
        File f = jFileChooser1.getSelectedFile();
        frame1.setRepDir(f);
        frame1.setTextBox2(f.getAbsolutePath());
        this.setVisible(false);
    }//GEN-LAST:event_jFileChooser1ActionPerformed

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        Look and feel setting code (optional)
        //</editor-fold>
        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() { new NewJFrame11().setVisible(true); }
        });
    }
    // Variables declaration - do not modify//GEN-BEGIN:variables
    5 usages
    private javax.swing.JFileChooser jFileChooser1;
    // End of variables declaration//GEN-END:variables
}
```

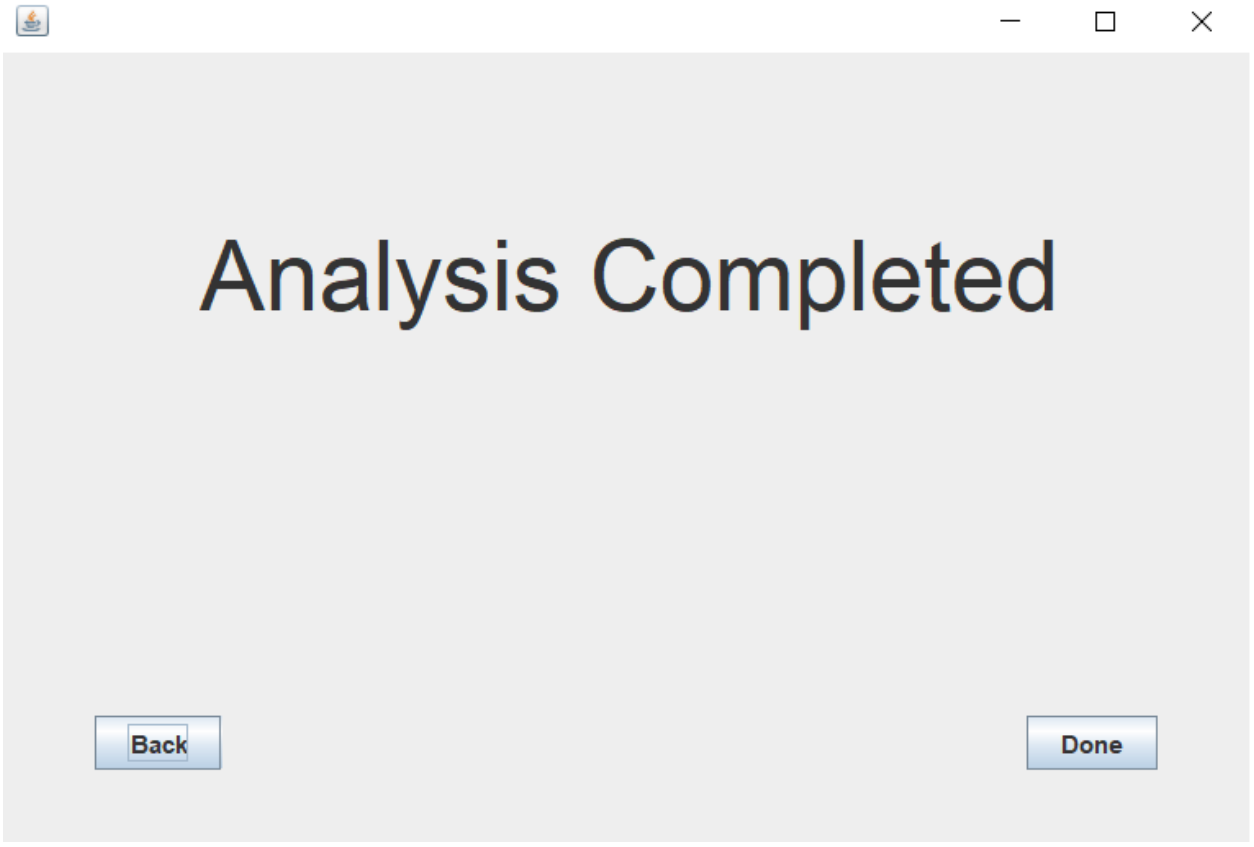
```
12  public class Done extends javax.swing.JFrame {
        3 usages
13      NewJFrame1 f;
        2 usages
14      public Done() { initComponents(); }
17
        2 usages
18      public Done(NewJFrame1 f) {
19          initComponents();
20          this.f=f;
21      }
22      /unchecked/
23      Generated Code
        1 usage
80      private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton1ActionPerformed
81          // TODO add your handling code here:
82          this.setVisible(false);
83          f.setVisible(true);
84      }//GEN-LAST:event_jButton1ActionPerformed
        1 usage
85      private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton2ActionPerformed
86          // TODO add your handling code here:
87          f.setVisible(false);
88          this.setVisible(false);
89          System.exit( status: 0);
90      }//GEN-LAST:event_jButton2ActionPerformed
91
92      /**
93       * @param args the command line arguments
94       */
95      public static void main(String args[]) {
96          /* Set the Nimbus look and feel */
97          Look and feel setting code (optional)
118
119         /* Create and display the form */
120         java.awt.EventQueue.invokeLater(new Runnable() {
121             public void run() { new Done().setVisible(true); }
124         });
125     }
        5 usages
126     private javax.swing.JButton jButton1;
        5 usages
127     private javax.swing.JButton jButton2;
        7 usages
128     private javax.swing.JLabel jLabel1;
129 }
```

GUI SCREENS


Faculty of Engineering - Ain Shams University
1839
Arithmetic Overflow
Unreachable FSM State
Un-initialized Register
Multi-Driven Bus/Register
Non Full/Parallel Case
Enter Design Directory
Choose F...
Enter Report Directory
Choose F...
Start


Look In:
Documents
3ds Max 2021
3dsMax
Adobe
Bandicam
Custom Office Templates
Dolphin Emulator
Downloads
IISExpress
Inventor Se
Lightshot
MATLAB
My Web Site
File Name:
Files of Type:
All Files
Open
Cancel


Faculty of Engineering - Ain Shams University
1839
Arithmetic Overflow
Unreachable FSM State
Un-initialized Register
Multi-Driven Bus/Register
Non Full/Parallel Case
C:\Users\HP\Desktop\Design 1.txt
Choose F...
C:\Users\HP\Desktop\output.txt
Choose F...
Start


Analysis Completed
Back
Done

Designs used

Design 1 (code and output)

```
module Up_Dn_Counter (
  input   wire  [4:0]     IN,
  input   wire            Load, Up, Down,
  input   wire            clk,
  output  reg   [4:0]     Counter,
  output  wire            High, Low
  input in1,
  input [3:0] X = 4'b1111,
  input [3:0] Y = 4'b1111,
  output [3:0] Z,
  output reg out,
  output reg out3,
  output reg out4
  );

Z = X + Y;
initial
 begin
 Load = 1'b0;
  Up = 1'b1;
 end
initial
  begin
  Load = 1'b1;
  Up = 1'b0;
end

 always @ (posedge clk)
   begin
    if (Load)
      begin
        Counter <= IN ;
      end
    else if (Down && !Low)
      begin
        Counter <= Counter - 5'b1;
      end
```

```
37          end
38        else if (Up && !High)
39          begin
40            Counter <= Counter + 5'b1;
41          end
42       case (in1)// synopsis parallel_case
43       1'b1 : out <= 1,
44              out3 <= 0,
45              out4 <= 1;
46       1'b1 : out <= 0,
47              out3 <= 1,
48              out4 <= 1;
49     endcase
50       end
51
52
53   // Down flag
54     assign Low = (Counter == 5'b0);
55
56   // Up flag
57     assign High = (Counter == 5'b11111);
58
59   endmodule
```

```
Arithmetic overflow:
        Arithmetic overflow at line 16 between X and Y
FSM States:
         No unreachable states.

Uninitialized registers:
        No uninitialized registers found.

Multi-driven buses/registers:
        Load MULTIDRIVEN in lines 19 and 24
        Up MULTIDRIVEN in lines 20 and 25

Not Full/Parallel Cases:
        not Full case at line 42
```

Design 2 (code and output)

```
1   module serializer (
2    input clk,
3    input load,
4    input inl,
5    input [7:0] in_value,
6    output reg o_bit,
7    output reg out,
8    output reg out3,
9    output reg out4,
10   output reg out5
11  );
12   reg [7:0] internal_reg ;
13   always @ (posedge clk)
14   begin
15   if (load)
16   begin
17   internal_reg <= in_value ;
18   end
19   else
20   begin
21   o_bit = internal_reg ;
22   end
23   end
24   always @ (posedge clk)
25   begin
26   load = 1'b0;
27   o_bit = 1'b1;
28   case (inl)
29      1'b1 : out <= 1,
30             out3 <= 0,
31             out4 <= 1;
32      1'b1 : out <= 0,
33             out3 <= 1,
34             out4 <= 1;
35      default : out <= 1;
36                out3 <= 1,
37                out4 <= 1;
38    endcase
39   end
40   endmodule
```

```
File  Edit  Format  View  Help
Arithmetic overflow:
        No arithmetic overflow found
FSM States:
         No unreachable states.

Uninitialized registers:
        out5 is not initialized at line 10

Multi-driven buses/registers:
        load MULTIDRIVEN in lines 15 and 26
        o_bit MULTIDRIVEN in lines 21 and 27

Not Full/Parallel Cases:
        not parallel case at line 28
```

Design 3 (code and output)

```
1  ////////////////////////////////////
2  ////////////// Moore FSM //////////////
3  ////////////////////////////////////
4
5  module LOCKER_Moore (
6  input  wire        button_0 , button_1 ,
7  input  wire        rst,
8  input  wire        clk,
9  output reg         unlock
10 );
11
12
13
14 localparam  [2:0]    IDLE = 3'b000,
15                      S1 = 3'b001,
16                      S11 = 3'b011,
17                      S011 = 3'b010,
18                      S1011 = 3'b110,
19                      UNLOCK = 3'b111 ;
20
21 reg    [2:0]         current_state,
22                      next_state ;
23
24 // state transition
25 always @(posedge clk or negedge rst)
26  begin
27   if(!rst)
28    begin
29     current_state <= IDLE ;
30    end
31   else
32    begin
33     current_state <= next_state ;
34    end
35  end
36
37 // next_state logic
38 always @(*)
39  begin
40   case (current_state)
41   IDLE : begin
42            if(button_0)
```

```
42                 if(button_0)
43                  next_state = IDLE ;
44                 else if (button_1)
45                  next_state = S1 ;
46                 else
47                  next_state = IDLE ;
48                end
49        S1 : begin
50                 if(button_0)
51                  next_state = IDLE ;
52                 else if (button_1)
53                  next_state = S11 ;
54                 else
55                  next_state = S1 ;
56               end
57        S11 : begin
58                 if(button_0)
59                  next_state = S011 ;
60                 else if (button_1)
61                  next_state = IDLE ;
62                 else
63                  next_state = S11 ;
64               end
65        S011 : begin
66                 if(button_0)
67                  next_state = IDLE ;
68                 else if (button_1)
69                  next_state = S1011 ;
70                 else
71                  next_state = S011 ;
72               end
73        S1011 : begin
74                  if(button_0)
75                    next_state = IDLE;
76                  else if (button_1)
77                    next_state = IDLE ;
78                  else
79                    next_state = S1011 ;
80                end
81        UNLOCK  : begin
```

```
81        UNLOCK  : begin
82                    next_state = IDLE ;
83                  end
84        default :   next_state = IDLE ;
85
86        endcase
87      end
88
89
90      // next_state logic
91      always @(*)
92       begin
93        case (current_state)
94        IDLE : begin
95                    unlock   =  1'b0 ;
96                  end
97        S1 : begin
98                    unlock   =  1'b0 ;
99                  end
100       S11 : begin
101                   unlock   =  1'b0 ;
102                 end
103       S011 : begin
104                   unlock   =  1'b0 ;
105                 end
106       S1011 : begin
107                   unlock   =  1'b0 ;
108                 end
109       S1011 : begin
110                   unlock   =  1'b1 ;
111                 end
112       default : begin
113                   unlock   =  1'b0 ;
114                 end
115       endcase
116      end
117
118
119     endmodule
```

```
Arithmetic overflow:
        No arithmetic overflow found
FSM States:
        UNLOCK initialized in line  is never reached.

Uninitialized registers:
        No uninitialized registers found.

Multi-driven buses/registers:
        No multi-driven busses/registers.

Not Full/Parallel Cases:
        not parallel case at line 93
```