TemLoPAC Plugin Source-Code

Technical Report for NSTIP Funded Project 13-INF761-10

Track - Software Engineering and Innovative Systems

Version 3.1 - September, 2016

Aamir M. Khan

PhD Embedded Systems

Consultant MODEVES Project

NSTIP, Saudi Arabia.

Acknowledgments

This project is partially funded by NSTIP (National Science Technology and Innovative Plan), Saudi Arabia under the Track "Software Engineering and Innovated Systems" bearing the project code "13-INF761-10". We would like thank the government of Saudi Arabia especially the Ministry of Science, Technology and Scientific Research for providing us an opportunity to work on the cutting-edge technologies and latest research trends in this domain.

We would like to thank Prof. Dr. Frederic Mallet from University of Nice, Sophia-Antipolis, France for his relentless support for the project. Without his ideas and suggestions, it wouldn't have been possible. We would also like to thank the reviewers of this NSTIP funding and the reviewers from various reputed conferences (like TASE, SIES) and SoSym journal for their valuable comments and suggestions. All their critical remarks helped us evolve the work to its present form.

Contents

Chapter 1

Source Code Structure

This document provides the basic structure of the TemLoPAC plugin source code. This software has two major parts:

- Transform to semantic MARTE model
- Generate CCSL/Verilog observer code

For these two distinct portions, there are two separate packages in the TemLoPAC plugin project named as *marte* and *observer* respectively, also shown in Figure 1.1 clearly.

The purpose of various Java code files in the *marte* package is explained below:

- **InteractionInterface.java**
  This class provides the functions for the front-end of Sequence Diagram to provide right-click menu option.
- **InteractionProcess.java**
  This class provides the functions to extract information from UML Sequence Diagram and then applies the MARTE stereotypes as needed. CCSL Specification is also inserted to the Model.
- **StatemachineInterface.java**
  This class provides the functions for the front-end of State machine Diagram to provide right-click menu option.

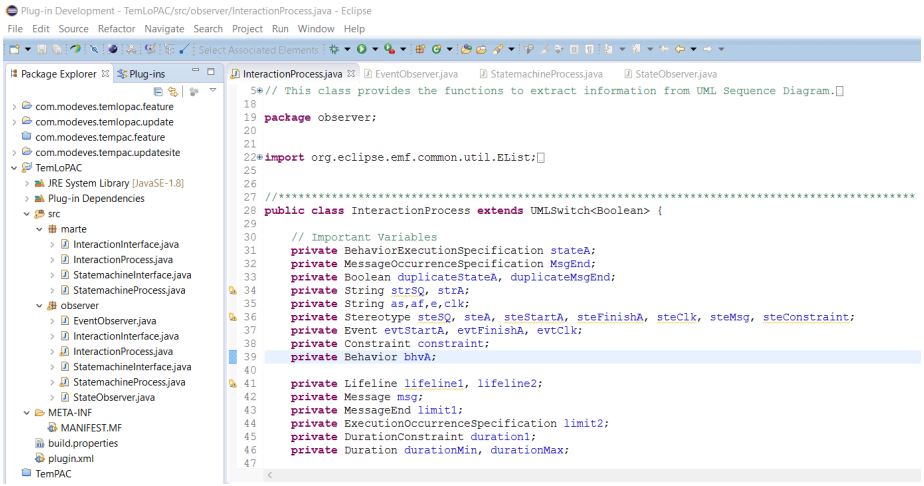

Figure 1.1: TemLoPAC Plugin Project Environment

- **StatemachineProcess.java**
  This class provides the functions to extract information from UML State machine Diagram and then applies the MARTE stereotypes as needed. CCSL Specification is also inserted to the Model.

The purpose of various Java code files in the *observer* package is explained below:

- **InteractionInterface.java**
  This class provides the functions for the front-end of Sequence Diagram to provide right-click menu option.

- **InteractionProcess.java**
  This class provides the functions to extract information from UML Sequence Diagram to generate CCSL/Verilog Observers code from MARTE Model.

- **StatemachineInterface.java**
  This class provides the functions for the front-end of State machine Diagram to provide right-click menu option.

- **StatemachineProcess.java**
  This class provides the functions to extract information from UML State machine Diagram to generate CCSL/Verilog Observers code from MARTE

Model.

- **StateObserver.java**
  This class provides the CCSL and Verilog Observers generation facility for State-State Relations
- **EventObserver.java**
  This class provides the CCSL and Verilog Observers generation facility for State-Event Relations

Chapter 2

MARTE Package

InteractionInterface.java

```
// ************************************************************************************************
// ************************************************************************************************
// ************************************************************************************************
//
// This class provides the functions for the front-end of Sequence Diagram.
// For Software to transform UML Model into MARTE Model.
//
// @package     MARTE
// @class       InteractionInterface
// @author      Aamir M. Khan
// @version     3.1
// @first       2016-02-02
// @current     2016-08-13
//
// ************************************************************************************************
// ************************************************************************************************
// ************************************************************************************************

package marte;

import org.eclipse.core.commands.AbstractHandler;
import org.eclipse.core.commands.ExecutionEvent;
import org.eclipse.core.commands.ExecutionException;
import org.eclipse.ui.IWorkbenchWindow;
import org.eclipse.ui.handlers.HandlerUtil;
import org.eclipse.jface.dialogs.MessageDialog;


import org.eclipse.core.commands.IHandler;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.jface.viewers.StructuredSelection;
import org.eclipse.ui.PlatformUI;

import org.eclipse.uml2.uml.*;


//*************************************************************************************************
public class InteractionInterface extends AbstractHandler implements IHandler {
    /**
     * The constructor.
     */
    //*********************************************************************************************
    public InteractionInterface() {
    }

    /**
     * the command has been executed, so extract extract the needed information
     * from the application context.
     */
    //*********************************************************************************************
    public Object execute(ExecutionEvent event) throws ExecutionException {

        ISelection selection = PlatformUI.
                getWorkbench().getActiveWorkbenchWindow().getActivePage().getSelection();
        if (!(selection instanceof StructuredSelection)) return null;
        Object selected = ((StructuredSelection)selection).getFirstElement();

        // The type should be guaranteed by the "isVisibleWhen"
        assert(selected instanceof Interaction);


        InteractionProcess PI = new InteractionProcess();
        PI.doSwitch((Interaction)selected);


        if(PI.msgError == null) PI.msgError = "Semantic MARTE Annotations Successfully Generated!";
        IWorkbenchWindow window = HandlerUtil.getActiveWorkbenchWindowChecked(event);
        MessageDialog.openInformation(
                window.getShell(),
                "Graphical Property Analysis",
                PI.msgError
                );

        return null;
```

```
    }
}
```

InteractionProcess.java

```
// ************************************************************************************************
// ************************************************************************************************
// ************************************************************************************************
//
// This class provides the functions to extract information from UML Sequence Diagram.
// For Software to transform UML Model into MARTE Model.
// Applies the MARTE stereotypes as needed.
// CCSL Specification is also inserted to the Model.
//
// @package     MARTE
// @class       InteractionProcess
// @author      Aamir M. Khan
// @version     3.1
// @first       2016-02-02
// @current     2016-08-13
//
// ************************************************************************************************
// ************************************************************************************************
// ************************************************************************************************

package marte;


import org.eclipse.emf.common.util.EList;
import org.eclipse.uml2.uml.*;
import org.eclipse.uml2.uml.Package;
import org.eclipse.uml2.uml.util.UMLSwitch;


//************************************************************************************************
public class InteractionProcess extends UMLSwitch<Boolean> {

    // Important Variables
    private Package pkg;

    private BehaviorExecutionSpecification stateA;
    private MessageOccurrenceSpecification MsgEnd;
    private Boolean duplicateStateA, duplicateMsgEnd;
    private String strSQ, strA;
    private Stereotype steSQ, steA, steStartA, steFinishA, steClk, steMsg, steConstraint;
    private Event evtStartA, evtFinishA, evtClk;
    private Constraint constraint;
    private Behavior bhvA;

    @SuppressWarnings("unused")
    private Lifeline lifeline1, lifeline2;
    private Message msg;
    private MessageEnd limit1;
    private ExecutionOccurrenceSpecification limit2;
    private DurationConstraint duration1;
    private Duration durationMin, durationMax;

    private enum PatternEvent {Timed,Untimed};
    PatternEvent patternType;
    private String patternKind;
    private int min,max;

    public String msgError;


    // ************************************************************************************************
    public Boolean caseInteraction(Interaction SQ) {

        System.out.println("\n-------------------------------------------------------------------------");
        System.out.println("\ncase MARTE Interaction: ");


        // ------------------------------------------------------------------------------------------------
        // Requirements: Check Elements + Stereotypes
        // GaAnalysisContext
        // ------------------------------------------------------------------------------------------------

        pkg = SQ.getPackage();
        strSQ=SQ.getName();

        if(SQ.getAppliedStereotype("MARTE::MARTE_AnalysisModel::GQAM::GaAnalysisContext")==null) {
            System.out.println("\nGaAnalysisContext Now Applied");
            steSQ = SQ.getApplicableStereotype("MARTE::MARTE_AnalysisModel::GQAM::GaAnalysisContext");
            SQ.applyStereotype(steSQ);
        } else {
            System.out.println("\nGaAnalysisContext Already Applied");
            steSQ=SQ.getAppliedStereotype("MARTE::MARTE_AnalysisModel::GQAM::GaAnalysisContext");
        }

        // Get Lifelines
        if(SQ.getLifelines().size() != 2) {
            propertyError("Error in Interaction Lifelines.");
            return false;
        }
        else {
            lifeline1 = SQ.getLifelines().get(0);
```

```
    lifeline2 = SQ.getLifelines().get(1);
}

// Message e
if(SQ.getMessages().size() != 1) {
    propertyError("Error in Interaction Message.");
    return false;
}
else {
    msg = SQ.getMessages().get(0);
    limit1 = msg.getReceiveEvent();
}


// ---------------------------------------------------------------------------------------------------
// Observation Profile
// ---------------------------------------------------------------------------------------------------

// Pattern Stereotype
if((msg.getAppliedStereotype("MODEVES::UntimedEventPattern")==null) &&
   (msg.getAppliedStereotype("MODEVES::TimedEventPattern")==null)) {
    propertyError("Error in Timed/Untimed Pattern Stereotype.");
    return false;
}
else if((msg.getAppliedStereotype("MODEVES::UntimedEventPattern")!=null) &&
        (msg.getAppliedStereotype("MODEVES::TimedEventPattern")!=null)) {
    propertyError("Error in Timed/Untimed Pattern Stereotype.");
    return false;
}

// Untimed Pattern
else if(msg.getAppliedStereotype("MODEVES::UntimedEventPattern")!=null){
    steMsg=msg.getAppliedStereotype("MODEVES::UntimedEventPattern");
    patternType=PatternEvent.Untimed;
    patternKind=(String)( ( (EnumerationLiteral)msg.getValue(steMsg, "kind") ).getName() );

    if(lifeline2.getCoveredBys().size() < 4) {
        propertyError("Error in Interaction Message or Behavior Execution Specification Structure.");
        return false;
    }
    else {
        duplicateStateA = false; duplicateMsgEnd = false;
        for(InteractionFragment os: lifeline2.getCoveredBys()) doSwitch(os);
    }

    if(MsgEnd == null || stateA == null) {
        propertyError("Error in Interaction Message or Behavior Execution Specification Structure.");
        return false;
    }
    else if((lifeline2.getCoveredBys().indexOf(MsgEnd) <= lifeline2.getCoveredBys().indexOf(stateA.getStart())) ||
            (lifeline2.getCoveredBys().indexOf(MsgEnd) >= lifeline2.getCoveredBys().indexOf(stateA.getFinish()))) {
        propertyError("Error in Interaction Message or Behavior Structure.");
        return false;
    }

    for(Constraint c: SQ.getOwnedRules())
        if(c instanceof DurationConstraint) {
            propertyError("Error regarding Duration Constraint.");
            return false;
        }

    System.out.println("\nUntimed Event Pattern Present");
    System.out.println("\nType: "+patternType);
    System.out.println("\nKind: "+patternKind);

}

// Timed Pattern
else if(msg.getAppliedStereotype("MODEVES::TimedEventPattern")!=null){
    steMsg=msg.getAppliedStereotype("MODEVES::TimedEventPattern");
    patternType=PatternEvent.Timed;
    patternKind=(String)( ( (EnumerationLiteral)msg.getValue(steMsg, "kind") ).getName() );
    evtClk=(Event)msg.getValue(steMsg, "on");

    if(evtClk.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock")==null){
        steClk = evtClk.getApplicableStereotype("MARTE::MARTE_Foundations::Time::Clock");
        evtClk.applyStereotype(steClk);
    }
    else {
        steClk = evtClk.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock");
        System.out.println("\nClock Already Present.");
    }


    System.out.println("\nTimed Event Pattern Present");
    System.out.println("\nType: "+patternType);
    System.out.println("\nKind: "+patternKind);
    System.out.println("\nOn: "+evtClk.getName());


    // Duration
    Boolean duplicate = false;
```

```
    for(Constraint c: SQ.getOwnedRules()) {
        if(c instanceof DurationConstraint) {
            if((c.getConstrainedElements().get(0)==limit1)||
               (c.getConstrainedElements().get(1)==limit1)){
                if(duplicate==true) {
                    propertyError("Error: Multiple Duration Constraints.");
                    return false;
                }
                else {
                    duplicate = true;
                    duration1 = (DurationConstraint) c;
                    durationMin = (Duration) ((DurationInterval)duration1.getSpecification()).getMin();
                    min = ((LiteralInteger) durationMin.getExpr()).getValue();
                    durationMax = (Duration) ((DurationInterval)duration1.getSpecification()).getMax();
                    max = ((LiteralInteger) durationMax.getExpr()).getValue();
                    System.out.println("\nmin: "+min);
                    System.out.println("\nmax: "+max);
                }
            }
        }
    }


    // State A
    if(duration1.getConstrainedElements().get(0)==limit1)
        limit2 = (ExecutionOccurrenceSpecification) duration1.getConstrainedElements().get(1);
    else if (duration1.getConstrainedElements().get(1)==limit1)
        limit2 = (ExecutionOccurrenceSpecification) duration1.getConstrainedElements().get(0);

    stateA = (BehaviorExecutionSpecification) limit2.getExecution();
    strA = stateA.getName();
}


// ---------------------------------------------------------------------------------------------
// MARTE Profile
// ---------------------------------------------------------------------------------------------

// Behavior_A
if(stateA.getBehavior()==null){
    System.out.println("\nBehavior Now Applied");
    bhvA = (Behavior) pkg.createPackagedElement("behavior_"+strA, UMLPackage.eINSTANCE.getOpaqueBehavior());
    stateA.setBehavior(bhvA);
}
else {
    bhvA = stateA.getBehavior();
    System.out.println("\nBehavior Already Present.");
}

// TimedProcessing
if(bhvA.getAppliedStereotype("MARTE::MARTE_Foundations::Time::TimedProcessing")==null) {
    System.out.println("\nTimedProcessing Now Applied");
    steA = bhvA.getApplicableStereotype("MARTE::MARTE_Foundations::Time::TimedProcessing");
    bhvA.applyStereotype(steA);
}
else {
    steA = bhvA.getAppliedStereotype("MARTE::MARTE_Foundations::Time::TimedProcessing");
    System.out.println("\nTimedProcessing Already Present.");
}

// Tagged Value Start A
if(bhvA.getValue(steA, "start")==null){
    System.out.println("\nClock Start A Now Applied");
    evtStartA = (Event) pkg.createPackagedElement("start_"+strA, UMLPackage.eINSTANCE.getChangeEvent());
    bhvA.setValue(steA, "start", evtStartA);

    steStartA = evtStartA.getApplicableStereotype("MARTE::MARTE_Foundations::Time::Clock");
    evtStartA.applyStereotype(steStartA);

}
else {
    evtStartA = (Event) stateA.getBehavior().getValue(steA, "start");
    if(evtStartA.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock")==null) {
        steStartA = evtStartA.getApplicableStereotype("MARTE::MARTE_Foundations::Time::Clock");
        evtStartA.applyStereotype(steStartA);
    }
    else {
        steStartA = evtStartA.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock");
        System.out.println("\nClock Start A Already Present.");
    }
}

// Tagged Value Finish A
if(bhvA.getValue(steA, "finish")==null){
    System.out.println("\nClock Finish A Now Applied");
    evtFinishA = (Event) pkg.createPackagedElement("finish_"+strA, UMLPackage.eINSTANCE.getChangeEvent());
    bhvA.setValue(steA, "finish", evtFinishA);

    steFinishA = evtFinishA.getApplicableStereotype("MARTE::MARTE_Foundations::Time::Clock");
    evtFinishA.applyStereotype(steFinishA);

}
```

```
else {
    evtFinishA = (Event) bhvA.getValue(steA, "finish");
    if(evtFinishA.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock")==null) {
        steFinishA = evtFinishA.getApplicableStereotype("MARTE::MARTE_Foundations::Time::Clock");
        evtFinishA.applyStereotype(steFinishA);
    }
    else {
        steFinishA = evtFinishA.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock");
        System.out.println("\nClock Finish A Already Present.");
    }
}


// Tagged Value On: Start A, Finish A
if(((EList<?>)bhvA.getValue(steA, "on")).size() == 0){
    System.out.println("\nClock On Now Applied");

    bhvA.setValue(steA, "on[0]", evtStartA.getStereotypeApplication(steStartA));
    bhvA.setValue(steA, "on[1]", evtFinishA.getStereotypeApplication(steFinishA));
}
else {
    System.out.println("\nClock On Already Applied");
}

// ---------------------------------------------------------------------------------------------
// Property Constraint
// ---------------------------------------------------------------------------------------------

// Constraint
if(SQ.getOwnedRules().size()==0){
    System.out.println("\nConstraint Now Created");
    constraint = SQ.createOwnedRule("constraint_"+strSQ, UMLPackage.eINSTANCE.getConstraint());
}
else if(SQ.getOwnedRules().size()==1){
    if(!(SQ.getOwnedRules().get(0) instanceof DurationConstraint)){
        System.out.println("\nConstraint Already Present");
        constraint = SQ.getOwnedRules().get(0);
    }
    else {
        System.out.println("\nConstraint Now Created");
        constraint = SQ.createOwnedRule("constraint_"+strSQ, UMLPackage.eINSTANCE.getConstraint());
    }
}
else if(SQ.getOwnedRules().size()==2){
    if(!(SQ.getOwnedRules().get(0) instanceof DurationConstraint)){
        System.out.println("\nConstraint Already Present");
        constraint = SQ.getOwnedRules().get(0);
    }
    else if(!(SQ.getOwnedRules().get(1) instanceof DurationConstraint)){
        System.out.println("\nConstraint Already Present");
        constraint = SQ.getOwnedRules().get(1);
    }
    else {
        propertyError("Error in Property Constraint.");
        return false;
    }
}
else {
    propertyError("Error in Property Constraint.");
    return false;
}


// CCSL Specification
if(constraint.getSpecification()==null){
    System.out.println("\nCCSL Specification Now Created");
    constraint.createSpecification(null,null,UMLPackage.eINSTANCE.getLiteralString());
    constraint.getSpecification().setName("CCSL");
}
else {
    System.out.println("\nCCSL Specification Already Present");
}

String CCSLSpec = getCCSLSpec();
((LiteralString)constraint.getSpecification()).setValue(CCSLSpec);


// TimedConstraint
if(constraint.getAppliedStereotype("MARTE::MARTE_Foundations::Time::TimedConstraint")==null){
    System.out.println("\nTimedConstraint Now Applied");
    steConstraint = constraint.getApplicableStereotype("MARTE::MARTE_Foundations::Time::TimedConstraint");
    constraint.applyStereotype(steConstraint);
}
else {
    steConstraint = constraint.getAppliedStereotype("MARTE::MARTE_Foundations::Time::TimedConstraint");
    System.out.println("\nTimedConstraint Already Present.");
}

constraint.setValue(steConstraint, "on[0]", evtStartA.getStereotypeApplication(steStartA));
constraint.setValue(steConstraint, "on[1]", evtFinishA.getStereotypeApplication(steFinishA));

if(patternType==PatternEvent.Timed){
```

```
            constraint.setValue(steConstraint, "on[2]", evtClk.getStereotypeApplication(steClk));
        }

        System.out.println("\n--------------------------------------------------------------------------");

        return true;
    }


    // ************************************************************************************************
    // ************************************************************************************************
    // Generate the CCSL Specification
    // ************************************************************************************************
    public String getCCSLSpec() {

        String CCSLSpec = null;

        if(patternType==PatternEvent.Timed){
            switch(patternKind){
            case "Triggers":
                CCSLSpec = "e delayedFor min on clk precedes as,\nas precedes e delayedFor max on clk,\nas alternatesWith af";
                break;
            case "Terminates":
                CCSLSpec = "e delayedFor min on clk precedes af,\naf precedes e delayedFor max on clk,\nas alternatesWith af";
                break;
            default:
                propertyError("Error: Invalid Timed Pattern");
            }
        }

        else if(patternType==PatternEvent.Untimed){
            switch(patternKind){
            case "Triggers":
                CCSLSpec = "Untimed Triggers";
                break;
            case "Terminates":
                CCSLSpec = "Untimed Terminates";
                break;
            case "Forbids":
                CCSLSpec = "e excludes as,\nas alternatesWith af";
                break;
            case "Contains":
                CCSLSpec = "Untimed Contains";
                break;
            default:
                propertyError("Error: Invalid Untimed Pattern");
            }
        }
        else propertyError("Error: Invalid Pattern Type");

        return CCSLSpec;
    }


    //************************************************************************************************
    public Boolean caseBehaviorExecutionSpecification(BehaviorExecutionSpecification bes) {
        if(duplicateStateA == true) propertyError("Error: Duplicate Behavior Execution Specification");
        else {
            duplicateStateA = true;
            stateA = bes;
            strA = bes.getName();
        }
        return true;
    }


    // ************************************************************************************************
    public Boolean caseMessageOccurrenceSpecification(MessageOccurrenceSpecification mos) {
        if(duplicateMsgEnd == true) propertyError("Error: Duplicate Message Occurrence Specification");
        else {
            duplicateMsgEnd = true;
            MsgEnd = mos;
        }
        return true;
    }


    // ************************************************************************************************
    public void propertyError(String msg){
        System.out.println("\nIncorrect Property: "+msg);
        if(msg==null) msgError = "Error Occurred!";
        else msgError = msg;
    }

}
```

StatemachineInterface.java

```
// ************************************************************************************************
// ************************************************************************************************
// ************************************************************************************************
//
// This class provides the functions for the front-end of StateMachine Diagram.
// For Software to transform UML Model into MARTE Model.
//
// @package     MARTE
// @class       StatemachineInterface
// @author      Aamir M. Khan
// @version     3.1
// @first       2016-02-02
// @current     2016-08-13
//
// ************************************************************************************************
// ************************************************************************************************
// ************************************************************************************************

package marte;

import org.eclipse.core.commands.AbstractHandler;
import org.eclipse.core.commands.ExecutionEvent;
import org.eclipse.core.commands.ExecutionException;
import org.eclipse.ui.IWorkbenchWindow;
import org.eclipse.ui.handlers.HandlerUtil;
import org.eclipse.jface.dialogs.MessageDialog;


import org.eclipse.core.commands.IHandler;
//import org.eclipse.core.commands.IHandlerListener;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.jface.viewers.StructuredSelection;
import org.eclipse.ui.PlatformUI;

import org.eclipse.uml2.uml.*;


//************************************************************************************************
public class StatemachineInterface extends AbstractHandler implements IHandler {

    /**
     * The constructor.
     */
    //************************************************************************************************
    public StatemachineInterface() {
    }

    /**
     * the command has been executed, so extract extract the needed information
     * from the application context.
     */
    //************************************************************************************************
    public Object execute(ExecutionEvent event) throws ExecutionException {

        ISelection selection = PlatformUI.
                getWorkbench().getActiveWorkbenchWindow().getActivePage().getSelection();
        if (!(selection instanceof StructuredSelection)) return null;
        Object selected = ((StructuredSelection)selection).getFirstElement();

        // The type should be guaranteed by the "isVisibleWhen"
        assert(selected instanceof StateMachine);


        StatemachineProcess SM = new StatemachineProcess();
        SM.doSwitch((StateMachine)selected);


        if(SM.msgError == null) SM.msgError = "Semantic MARTE Annotations Successfully Generated!";
        IWorkbenchWindow window = HandlerUtil.getActiveWorkbenchWindowChecked(event);
        MessageDialog.openInformation(
                window.getShell(),
                "Graphical Property Analysis",
                SM.msgError
                );
```

```
            return null;
        }
    }
```

StatemachineProcess.java

```
// ************************************************************************************************
// ************************************************************************************************
// ************************************************************************************************
//
// This class provides the functions to extract information from UML StateMachine Diagram.
// For Software to transform UML Model into MARTE Model.
// Applies the MARTE stereotypes as needed.
// CCSL Specification is also inserted to the Model.
//
// @package     MARTE
// @class       StatemachineProcess
// @author      Aamir M. Khan
// @version     3.1
// @first       2016-02-02
// @current     2016-08-13
//
// ************************************************************************************************
// ************************************************************************************************
// ************************************************************************************************

package marte;

import org.eclipse.emf.common.util.EList;
import org.eclipse.uml2.uml.*;
import org.eclipse.uml2.uml.Package;
import org.eclipse.uml2.uml.util.UMLSwitch;


public class StatemachineProcess extends UMLSwitch<Boolean> {

    // Important Variables
    private Package pkg;
    private State stateA, stateB;
    private String strSM, strA, strB;
    private Event evtStartA, evtFinishA, evtStartB, evtFinishB, evtClk;
    @SuppressWarnings("unused")
    private Stereotype steClkType, steClk, steA, steStartA, steFinishA, steB, steStartB, steFinishB;
    private Stereotype steSM, steTransition, steConstraint;
    private Transition transition;
    private Constraint constraint;

    private enum PatternState {Timed,Untimed};
    PatternState patternType;
    private String patternKind;
    private int min,max;

    public String msgError;


// ************************************************************************************************
    public Boolean caseStateMachine(StateMachine SM) {

        System.out.println("\n-------------------------------------------------------------------------------");
        System.out.println("\ncase MARTE StateMachine");


        // ---------------------------------------------------------------------------------------------------
        // Requirements: Check Elements + Stereotypes
        // GaAnalysisContext
        // ---------------------------------------------------------------------------------------------------

        if(SM.getAppliedStereotype("MARTE::MARTE_AnalysisModel::GQAM::GaAnalysisContext")==null) {
            System.out.println("\nGaAnalysisContext Now Applied");
            steSM = SM.getApplicableStereotype("MARTE::MARTE_AnalysisModel::GQAM::GaAnalysisContext");
            SM.applyStereotype(steSM);

        } else System.out.println("\nGaAnalysisContext Already Applied");

        pkg = SM.getPackage();
        strSM=SM.getName();
        steSM=SM.getAppliedStereotype("MARTE::MARTE_AnalysisModel::GQAM::GaAnalysisContext");


        // ---------------------------------------------------------------------------------------------------
        // Activation State
        // ---------------------------------------------------------------------------------------------------

        if(SM.getRegions().size() != 1) {
            propertyError("Error in State Machine Region.");
            return false;
        }
        else if(SM.getRegions().get(0).getSubvertices().size() != 2) {
            propertyError("Error in State Machine Vertices.");
            return false;
        }
        else {
            for(Vertex vx : SM.getRegions().get(0).getSubvertices()) doSwitch(vx);
        }

        // Behavior_A
        if(stateA==null) {
            propertyError("Error in Activation State.");
            return false;
        }
        else if(stateA.getDoActivity()==null){
            System.out.println("\nBehavior A Now Applied");
            stateA.createDoActivity("behavior_"+strA, UMLPackage.eINSTANCE.getOpaqueBehavior());
```

```
}
else System.out.println("\nBehavior A Already Present.");

// TimedProcessing A
if(stateA.getDoActivity().getAppliedStereotype("MARTE::MARTE_Foundations::Time::TimedProcessing")==null){
    System.out.println("\nTimedProcessing A Now Applied");
    steA = stateA.getDoActivity().getApplicableStereotype("MARTE::MARTE_Foundations::Time::TimedProcessing");
    stateA.getDoActivity().applyStereotype(steA);
}
else {
    steA = stateA.getDoActivity().getAppliedStereotype("MARTE::MARTE_Foundations::Time::TimedProcessing");
    System.out.println("\nTimedProcessing A Already Present.");
}

// Tagged Value Start A
if(stateA.getDoActivity().getValue(steA, "start")==null){
    System.out.println("\nClock A Start Now Applied");
    evtStartA = (Event) pkg.createPackagedElement("start_"+strA, UMLPackage.eINSTANCE.getChangeEvent());
    stateA.getDoActivity().setValue(steA, "start", evtStartA);

    steStartA = evtStartA.getApplicableStereotype("MARTE::MARTE_Foundations::Time::Clock");
    evtStartA.applyStereotype(steStartA);
}
else {
    evtStartA = (Event) stateA.getDoActivity().getValue(steA, "start");
    if(evtStartA.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock")==null) {
        steStartA = evtStartA.getApplicableStereotype("MARTE::MARTE_Foundations::Time::Clock");
        evtStartA.applyStereotype(steStartA);
    }
    else {
        steStartA = evtStartA.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock");
        System.out.println("\nClock A Start Already Present.");
    }
}

// Tagged Value Finish A
if(stateA.getDoActivity().getValue(steA, "finish")==null){
    System.out.println("\nClock A Finish Now Applied");
    evtFinishA = (Event) pkg.createPackagedElement("finish_"+strA, UMLPackage.eINSTANCE.getChangeEvent());
    stateA.getDoActivity().setValue(steA, "finish", evtFinishA);

    steFinishA = evtFinishA.getApplicableStereotype("MARTE::MARTE_Foundations::Time::Clock");
    evtFinishA.applyStereotype(steFinishA);
}
else {
    evtFinishA = (Event) stateA.getDoActivity().getValue(steA, "finish");
    if(evtFinishA.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock")==null) {
        steFinishA = evtFinishA.getApplicableStereotype("MARTE::MARTE_Foundations::Time::Clock");
        evtFinishA.applyStereotype(steFinishA);
    }
    else {
        steFinishA = evtFinishA.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock");
        System.out.println("\nClock A Finish Already Present.");
    }
}

// Tagged Value On: start A, finish A
if(((EList<?>)stateA.getDoActivity().getValue(steA, "on")).size() == 0){
    System.out.println("\nClock A On Now Applied");
    stateA.getDoActivity().setValue(steA, "on[0]", evtStartA.getStereotypeApplication(steStartA));
    stateA.getDoActivity().setValue(steA, "on[1]", evtFinishA.getStereotypeApplication(steFinishA));
} else {
    System.out.println("\nClock A On Already Applied");
}

// ------------------------------------------------------------------------------------------
// Transition
// ------------------------------------------------------------------------------------------

if(SM.getRegions().get(0).getTransitions().size() != 1) {
    propertyError("Error in State Transition.");
    return false;
}
else if(!SM.getRegions().get(0).getTransitions().get(0).getSource().equals(stateA)) {
    propertyError("Error in State Transition.");
    return false;
}
else transition = SM.getRegions().get(0).getTransitions().get(0);

// Pattern Stereotype
if((transition.getAppliedStereotype("MODEVES::UntimedStatePattern")==null) &&
   (transition.getAppliedStereotype("MODEVES::TimedStatePattern")==null)) {
    propertyError("Error in Timed/Untimed Pattern Stereotype.");
    return false;
}
else if((transition.getAppliedStereotype("MODEVES::UntimedStatePattern")!=null) &&
        (transition.getAppliedStereotype("MODEVES::TimedStatePattern")!=null)) {
    propertyError("Error in Timed/Untimed Pattern Stereotype.");
    return false;
}

// Untimed Pattern
else if(transition.getAppliedStereotype("MODEVES::UntimedStatePattern")!=null) {
    steTransition=transition.getAppliedStereotype("MODEVES::UntimedStatePattern");
    patternType=PatternState.Untimed;
    patternKind=(String)( ( (EnumerationLiteral)transition.getValue(steTransition, "kind") ).getName() );
    System.out.println("\nUntimed State Pattern Present");
    System.out.println("\nType: "+patternType);
```

```
    System.out.println("\nKind: "+patternKind);
}

// Timed Pattern
else if(transition.getAppliedStereotype("MODEVES::TimedStatePattern")!=null){
    steTransition=transition.getAppliedStereotype("MODEVES::TimedStatePattern");
    patternType=PatternState.Timed;
    patternKind=(String)( ( (EnumerationLiteral)transition.getValue(steTransition, "kind") ).getName() );
    min=(Integer)transition.getValue(steTransition, "min");
    max=(Integer)transition.getValue(steTransition, "max");
    evtClk=(Event)transition.getValue(steTransition, "on");

    if(evtClk.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock")==null){
        steClk = evtClk.getApplicableStereotype("MARTE::MARTE_Foundations::Time::Clock");
        evtClk.applyStereotype(steClk);
    } else {
        steClk = evtClk.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock");
        System.out.println("\nClock Already Present.");
    }

    System.out.println("\nTimed State Pattern Present");
    System.out.println("\nType: "+patternType);
    System.out.println("\nKind: "+patternKind);
    System.out.println("\nMin: "+min);
    System.out.println("\nMax: "+max);
    System.out.println("\nOn: "+evtClk.getName());
}


// ---------------------------------------------------------------------------------------------
// State B
// ---------------------------------------------------------------------------------------------

if(transition.getTarget()==null) {
    propertyError("Error in State Machine Transition or Vertices.");
    return false;
}
else {
    stateB=(State)transition.getTarget();
    strB=stateB.getName();
}

// Behavior_B
if(stateB.getDoActivity()==null){
    System.out.println("\nBehavior B Now Applied");
    stateB.createDoActivity("behavior_"+strB, UMLPackage.eINSTANCE.getOpaqueBehavior());
}
else System.out.println("\nBehavior B Already Present.");

// TimedProcessing
if(stateB.getDoActivity().getAppliedStereotype("MARTE::MARTE_Foundations::Time::TimedProcessing")==null){
    System.out.println("\nTimedProcessing B Now Applied");
    steB = stateB.getDoActivity().getApplicableStereotype("MARTE::MARTE_Foundations::Time::TimedProcessing");
    stateB.getDoActivity().applyStereotype(steB);
}
else {
    steB = stateB.getDoActivity().getAppliedStereotype("MARTE::MARTE_Foundations::Time::TimedProcessing");
    System.out.println("\nTimedProcessing B Already Present.");
}

// Event Start B
if(stateB.getDoActivity().getValue(steB, "start")==null){
    System.out.println("\nClock B Start Now Applied");
    evtStartB = (Event) pkg.createPackagedElement("start_"+strB, UMLPackage.eINSTANCE.getChangeEvent());
    stateB.getDoActivity().setValue(steB, "start", evtStartB);

    steStartB = evtStartB.getApplicableStereotype("MARTE::MARTE_Foundations::Time::Clock");
    evtStartB.applyStereotype(steStartB);
}
else {
    evtStartB = (Event) stateB.getDoActivity().getValue(steB, "start");
    if(evtStartB.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock")==null) {
        steStartB = evtStartB.getApplicableStereotype("MARTE::MARTE_Foundations::Time::Clock");
        evtStartB.applyStereotype(steStartB);
    }
    else {
        steStartB = evtStartB.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock");
        System.out.println("\nClock B Start Already Present.");
    }
}

// Event Finish B
if(stateB.getDoActivity().getValue(steB, "finish")==null){
    System.out.println("\nClock B Finish Now Applied");
    evtFinishB = (Event) pkg.createPackagedElement("finish_"+strB, UMLPackage.eINSTANCE.getChangeEvent());
    stateB.getDoActivity().setValue(steB, "finish", evtFinishB);

    steFinishB = evtFinishB.getApplicableStereotype("MARTE::MARTE_Foundations::Time::Clock");
    evtFinishB.applyStereotype(steFinishB);
}
else {
    evtFinishB = (Event) stateB.getDoActivity().getValue(steB, "finish");
    if(evtFinishB.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock")==null) {
        steFinishB = evtFinishB.getApplicableStereotype("MARTE::MARTE_Foundations::Time::Clock");
        evtFinishB.applyStereotype(steFinishB);
    }
    else {
        steFinishB = evtFinishB.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock");
```

```
            System.out.println("\nClock B Finish Already Present.");
        }
    }

    // Tagged Value On: start B, finish B
    if(((EList<?>)stateB.getDoActivity().getValue(steB, "on")).size() == 0) {
        System.out.println("\nClock B On Now Applied");
        stateB.getDoActivity().setValue(steB, "on[0]", evtStartB.getStereotypeApplication(steStartB));
        stateB.getDoActivity().setValue(steB, "on[1]", evtFinishB.getStereotypeApplication(steFinishB));
    }
    else {
        System.out.println("\nClock B On Already Applied");
    }


    // ------------------------------------------------------------------------------------------------
    // Property Constraint
    // ------------------------------------------------------------------------------------------------

    // Constraint
    if(SM.getOwnedRules().size()==0){
        System.out.println("\nConstraint Now Created");
        constraint = SM.createOwnedRule("constraint_"+strSM, UMLPackage.eINSTANCE.getConstraint());
    }
    else if(SM.getOwnedRules().size()==1){
        System.out.println("\nConstraint Already Present");
        constraint = SM.getOwnedRules().get(0);
    }
    else {
        propertyError("Error in Property Constraint.");
        return false;
    }


    // CCSL Specification
    if(constraint.getSpecification()==null){
        System.out.println("\nCCSL Specification Now Created");
        constraint.createSpecification(null,null,UMLPackage.eINSTANCE.getLiteralString());
        constraint.getSpecification().setName("CCSL");
    }
    else {
        System.out.println("\nCCSL Specification Already Present");
    }

    String CCSLSpec = getCCSLSpec();
    ((LiteralString)constraint.getSpecification()).setValue(CCSLSpec);


    // TimedConstraint
    if(constraint.getAppliedStereotype("MARTE::MARTE_Foundations::Time::TimedConstraint")==null){
        System.out.println("\nTimedConstraint Now Applied");
        steConstraint = constraint.getApplicableStereotype("MARTE::MARTE_Foundations::Time::TimedConstraint");
        constraint.applyStereotype(steConstraint);
    }
    else {
        steConstraint = constraint.getAppliedStereotype("MARTE::MARTE_Foundations::Time::TimedConstraint");
        System.out.println("\nTimedConstraint Already Present.");
    }

    constraint.setValue(steConstraint, "on[0]", evtStartA.getStereotypeApplication(steStartA));
    constraint.setValue(steConstraint, "on[1]", evtFinishA.getStereotypeApplication(steFinishA));
    constraint.setValue(steConstraint, "on[2]", evtStartB.getStereotypeApplication(steStartB));
    constraint.setValue(steConstraint, "on[3]", evtFinishB.getStereotypeApplication(steFinishB));

    if(patternType==PatternState.Timed){
        constraint.setValue(steConstraint, "on[4]", evtClk.getStereotypeApplication(steClk));
    }

    System.out.println("\n--------------------------------------------------------------------------");

    return true;
}


// ************************************************************************************************
// ************************************************************************************************
// Generate the CCSL Specification
// ************************************************************************************************
public String getCCSLSpec() {

    String CCSLSpec = null;

    if(patternType==PatternState.Timed){
        switch(patternKind){
        case "Precedes":
            CCSLSpec = "af delayedFor min on clk precedes bs,\nbs precedes af delayedFor max on clk,\nas alternatesWith af,\nbs al
            break;
        case "Starts":
            CCSLSpec = "as delayedFor min on clk precedes bs,\nbs precedes as delayedFor max on clk,\nas alternatesWith af,\nbs al
            break;
        case "Finishes":
            CCSLSpec = "af delayedFor min on clk precedes bf,\nbf precedes af delayedFor max on clk,\nas alternatesWith af,\nbs al
            break;
        default:
            propertyError("Error: Invalid Timed Pattern");
        }
    }
    else if(patternType==PatternState.Untimed){
```

```
            switch(patternKind){
            case "Precedes":
                CCSLSpec = "af precedes bs,\nas alternatesWith af,\nbs alternatesWith bf";
                break;
            case "Starts":
                CCSLSpec = "Untimed Starts";
                break;
            case "Finishes":
                CCSLSpec = "Untimed Finishes";
                break;
            case "Causes":
                CCSLSpec = "Untimed Causes";
                break;
            case "Excludes":
                CCSLSpec = "bs sampledOn as precedes af,\nas alternatesWith af";
                break;
            case "Forbids":
                CCSLSpec = "af exclusiveWith bs,\nas alternatesWith af,\nbs alternatesWith bf";
                break;
            case "Contains":
                CCSLSpec = "as precedes e_prime,\ne_prime precedes af,\ne isSubsetOf e_prime,\nas alternatesWith af";
                break;
            case "Implies":
                CCSLSpec = "Untimed Implies";
                break;
            default:
                propertyError("Error: Invalid Untimed Pattern");
            }
        }
        else propertyError("Error: Invalid Pattern Type");

        return CCSLSpec;
    }


    // ********************************************************************************************
    public Boolean caseRegion(Region rg) {
        //System.out.println("caseRegion \n");
        for(Vertex vertex : rg.getSubvertices()) doSwitch(vertex);
        for(Transition transition : rg.getTransitions()) doSwitch(transition);
        return true;
    }


    // ********************************************************************************************
    public Boolean caseState(State st) {
        //System.out.println("caseState \n");
        if(st.getAppliedStereotype("MODEVES::ActivationState") != null) {
            stateA = st;
            strA = stateA.getName();
        }
        return true;
    }


    // ********************************************************************************************
    public void propertyError(String msg){
        System.out.println("\nIncorrect Property: "+msg);
        if(msg==null) msgError = "Error Occurred!";
        else msgError = msg;
    }

}
```

Chapter 3

Observer Package

InteractionInterface.java

```
// ************************************************************************************************
// ************************************************************************************************
// ************************************************************************************************
//
// This class provides the functions for the front-end of Sequence Diagram.
// For Software to generate CCSL/Verilog Observers code from MARTE Model.
//
// @package     OBSERVER
// @class       InteractionInterface
// @author      Aamir M. Khan
// @version     3.1
// @first       2016-02-02
// @current     2016-08-13
//
// ************************************************************************************************
// ************************************************************************************************
// ************************************************************************************************

package observer;

import org.eclipse.core.commands.AbstractHandler;
import org.eclipse.core.commands.ExecutionEvent;
import org.eclipse.core.commands.ExecutionException;
import org.eclipse.ui.IWorkbenchWindow;
import org.eclipse.ui.handlers.HandlerUtil;
import org.eclipse.jface.dialogs.MessageDialog;


import org.eclipse.core.commands.IHandler;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.jface.viewers.StructuredSelection;
import org.eclipse.ui.PlatformUI;

import org.eclipse.uml2.uml.*;


//*************************************************************************************************
public class InteractionInterface extends AbstractHandler implements IHandler {

    /**
     * The constructor.
     */
    //*********************************************************************************************
    public InteractionInterface() {
    }

    /**
     * the command has been executed, so extract extract the needed information
     * from the application context.
     */
    //*********************************************************************************************
    public Object execute(ExecutionEvent event) throws ExecutionException {

        ISelection selection = PlatformUI.
                getWorkbench().getActiveWorkbenchWindow().getActivePage().getSelection();
        if (!(selection instanceof StructuredSelection)) return null;
        Object selected = ((StructuredSelection)selection).getFirstElement();

        // The type should be guaranteed by the "isVisibleWhen"
        assert(selected instanceof Interaction);


        InteractionProcess PI = new InteractionProcess();
        PI.doSwitch((Interaction)selected);


        if(PI.msgError == null) PI.msgError = "CCSL/Verilog Observer Code Successfully Generated!";
        IWorkbenchWindow window = HandlerUtil.getActiveWorkbenchWindowChecked(event);
        MessageDialog.openInformation(
                window.getShell(),
                "Graphical Property Analysis",
                PI.msgError
                );
```

```
            return null;
        }
    }
```

InteractionProcess.java

```
// ************************************************************************************************
// ************************************************************************************************
// ************************************************************************************************
//
// This class provides the functions to extract information from UML Sequence Diagram.
// For Software to generate CCSL/Verilog Observers code from MARTE Model.
//
// @package     OBSERVER
// @class       InteractionProcess
// @author      Aamir M. Khan
// @version     3.1
// @first       2016-02-02
// @current     2016-08-13
//
// ************************************************************************************************
// ************************************************************************************************
// ************************************************************************************************

package observer;


import org.eclipse.emf.common.util.EList;
import org.eclipse.uml2.uml.*;
import org.eclipse.uml2.uml.util.UMLSwitch;


//************************************************************************************************
public class InteractionProcess extends UMLSwitch<Boolean> {

    // Important Variables
    private BehaviorExecutionSpecification stateA;
    private MessageOccurrenceSpecification MsgEnd;
    private Boolean duplicateStateA, duplicateMsgEnd;
    private String strSQ, strA;
    private String as,af,e,clk;
    private Stereotype steSQ, steA, steStartA, steFinishA, steClk, steMsg, steConstraint;
    private Event evtStartA, evtFinishA, evtClk;
    private Constraint constraint;
    private Behavior bhvA;

    private Lifeline lifeline1, lifeline2;
    private Message msg;
    private MessageEnd limit1;
    private ExecutionOccurrenceSpecification limit2;
    private DurationConstraint duration1;
    private Duration durationMin, durationMax;

    private enum PatternEvent {Timed,Untimed};
    PatternEvent patternType;
    private String patternKind;
    private int min,max;

    public String msgError;


    // ************************************************************************************************
    public Boolean caseInteraction(Interaction SQ) {

        System.out.println("\n-------------------------------------------------------------------------");
        System.out.println("\ncase Observer Interaction: ");


        // ------------------------------------------------------------------------------------------------
        // Requirements: Check Elements + Stereotypes
        // GaAnalysisContext
        // ------------------------------------------------------------------------------------------------

        if(SQ.getAppliedStereotype("MARTE::MARTE_AnalysisModel::GQAM::GaAnalysisContext")==null) {
            propertyError("Error: GaAnalysisContext Stereotype Not Applied.");
            return false;
        }
        else {
            System.out.println("\nGaAnalysisContext Applied");
            strSQ=SQ.getName();
            steSQ=SQ.getAppliedStereotype("MARTE::MARTE_AnalysisModel::GQAM::GaAnalysisContext");
        }

        // Get Lifelines
        if(SQ.getLifelines().size() != 2) {
            propertyError("Error in Interaction Lifelines.");
            return false;
        }
        else {
            lifeline1 = SQ.getLifelines().get(0);
            lifeline2 = SQ.getLifelines().get(1);
        }

        // Message e
```

```
if(SQ.getMessages().size() != 1) {
    propertyError("Error in Interaction Message.");
    return false;
}
else {
    msg = SQ.getMessages().get(0);
    e = msg.getName();
    limit1 = msg.getReceiveEvent();
}


// ---------------------------------------------------------------------------------------------------
// Observation Profile
// ---------------------------------------------------------------------------------------------------

// Pattern Stereotype
if((msg.getAppliedStereotype("MODEVES::UntimedEventPattern")==null) &&
   (msg.getAppliedStereotype("MODEVES::TimedEventPattern")==null)){
    propertyError("Error in Timed/Untimed Pattern Stereotype.");
    return false;
}
else if((msg.getAppliedStereotype("MODEVES::UntimedEventPattern")!=null) &&
        (msg.getAppliedStereotype("MODEVES::TimedEventPattern")!=null)){
    propertyError("Error in Timed/Untimed Pattern Stereotype.");
    return false;
}

// Untimed Pattern
else if(msg.getAppliedStereotype("MODEVES::UntimedEventPattern")!=null){
    steMsg=msg.getAppliedStereotype("MODEVES::UntimedEventPattern");
    patternType=PatternEvent.Untimed;
    patternKind=(String)( ( (EnumerationLiteral)msg.getValue(steMsg, "kind") ).getName() );

    if(lifeline2.getCoveredBys().size() < 4) {
        propertyError("Error in Interaction Message or Behavior Execution Specification Structure.");
        return false;
    }
    else {
        duplicateStateA = false; duplicateMsgEnd = false;
        for(InteractionFragment os: lifeline2.getCoveredBys()) doSwitch(os);
    }

    if(MsgEnd == null || stateA == null) {
        propertyError("Error in Interaction Message or Behavior Execution Specification Structure.");
        return false;
    }
    else if((lifeline2.getCoveredBys().indexOf(MsgEnd) <= lifeline2.getCoveredBys().indexOf(stateA.getStart())) ||
            (lifeline2.getCoveredBys().indexOf(MsgEnd) >= lifeline2.getCoveredBys().indexOf(stateA.getFinish()))) {
        propertyError("Error in Interaction Message or Behavior Structure.");
        return false;
    }

    for(Constraint c: SQ.getOwnedRules())
        if(c instanceof DurationConstraint) {
            propertyError("Error regarding Duration Constraint.");
            return false;
        }

    System.out.println("\nUntimed Event Pattern Applied");
    System.out.println("\nType: "+patternType);
    System.out.println("\nKind: "+patternKind);

}

// Timed Pattern
else if(msg.getAppliedStereotype("MODEVES::TimedEventPattern")!=null){
    steMsg=msg.getAppliedStereotype("MODEVES::TimedEventPattern");
    patternType=PatternEvent.Timed;
    patternKind=(String)( ( (EnumerationLiteral)msg.getValue(steMsg, "kind") ).getName() );
    evtClk=(Event)msg.getValue(steMsg, "on");
    clk=evtClk.getName();

    if(evtClk.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock")==null){
        propertyError("Error in Timed Pattern Stereotype.");
        return false;
    }
    else {
        steClk = evtClk.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock");
        System.out.println("\nClock Present.");
    }

    System.out.println("\nTimed Event Pattern Applied");
    System.out.println("\nType: "+patternType);
    System.out.println("\nKind: "+patternKind);
    System.out.println("\nOn: "+clk);


    // Duration
    Boolean duplicate = false;
    for(Constraint c: SQ.getOwnedRules()) {
```

```
        if(c instanceof DurationConstraint) {
            if((c.getConstrainedElements().get(0)==limit1)||
               (c.getConstrainedElements().get(1)==limit1)){
                if(duplicate==true) {
                    propertyError("Error: Multiple Duration Constraints.");
                    return false;
                }
                else {
                    duplicate = true;
                    duration1 = (DurationConstraint) c;
                    durationMin = (Duration) ((DurationInterval)duration1.getSpecification()).getMin();
                    min = ((LiteralInteger) durationMin.getExpr()).getValue();
                    durationMax = (Duration) ((DurationInterval)duration1.getSpecification()).getMax();
                    max = ((LiteralInteger) durationMax.getExpr()).getValue();
                    System.out.println("\nmin: "+min);
                    System.out.println("\nmax: "+max);
                }
            }
        }
    }


    // State A
    if(duration1.getConstrainedElements().get(0)==limit1)
        limit2 = (ExecutionOccurrenceSpecification) duration1.getConstrainedElements().get(1);
    else if (duration1.getConstrainedElements().get(1)==limit1)
        limit2 = (ExecutionOccurrenceSpecification) duration1.getConstrainedElements().get(0);

    stateA = (BehaviorExecutionSpecification) limit2.getExecution();
    strA = stateA.getName();
}


// ---------------------------------------------------------------------------------------------------
// MARTE Profile
// ---------------------------------------------------------------------------------------------------

// Behavior A
if(stateA==null){
    propertyError("Error in Execution Specification.");
    return false;
}
else if(stateA.getBehavior()==null){
    propertyError("Error in Execution Specification Behavior.");
    return false;
}
else {
    bhvA = stateA.getBehavior();
    System.out.println("\nBehavior Present.");
}

// TimedProcessing A
if(bhvA.getAppliedStereotype("MARTE::MARTE_Foundations::Time::TimedProcessing")==null) {
    propertyError("Error: TimedProcessing Stereotype Not Applied.");
    return false;
}
else {
    steA = bhvA.getAppliedStereotype("MARTE::MARTE_Foundations::Time::TimedProcessing");
    System.out.println("\nTimedProcessing Present.");
}

// Tagged Value Start A
if(bhvA.getValue(steA, "start")==null){
    propertyError("Error in TimedProcessing Stereotype.");
    return false;
}
else {
    evtStartA = (Event) stateA.getBehavior().getValue(steA, "start");
    as = evtStartA.getName();
    if(evtStartA.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock")==null) {
        propertyError("Error in Clock Stereotype.");
        return false;
    }
    else {
        steStartA = evtStartA.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock");
        System.out.println("\nClock Start A Present.");
    }
}

// Tagged Value Finish A
if(bhvA.getValue(steA, "finish")==null){
    propertyError("Error in TimedProcessing Stereotype.");
    return false;
}
else {
    evtFinishA = (Event) bhvA.getValue(steA, "finish");
    af = evtFinishA.getName();
    if(evtFinishA.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock")==null) {
        propertyError("Error in Clock Stereotype.");
        return false;
```

```
    }
    else {
        steFinishA = evtFinishA.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock");
        System.out.println("\nClock Finish A Present.");
    }
}

// Tagged Value On: Start A, Finish A
if(((EList<?>)bhvA.getValue(steA, "on")).size() == 0){
    propertyError("Error in TimedProcessing Stereotype.");
    return false;
}


// ---------------------------------------------------------------------------------------------------
// Property Constraint
// ---------------------------------------------------------------------------------------------------

// Constraint
if(SQ.getOwnedRules().size()==1){
    if(!(SQ.getOwnedRules().get(0) instanceof DurationConstraint)){
        System.out.println("\nConstraint Present");
        constraint = SQ.getOwnedRules().get(0);
    }
    else {
        propertyError("Error in Property Constraint.");
        return false;
    }
}
else if(SQ.getOwnedRules().size()==2){
    if(!(SQ.getOwnedRules().get(0) instanceof DurationConstraint)){
        System.out.println("\nConstraint Present");
        constraint = SQ.getOwnedRules().get(0);
    }
    else if(!(SQ.getOwnedRules().get(1) instanceof DurationConstraint)){
        System.out.println("\nConstraint Present");
        constraint = SQ.getOwnedRules().get(1);
    }
    else {
        propertyError("Error in Property Constraint.");
        return false;
    }
}
else {
    propertyError("Error in Property Constraint.");
    return false;
}

// CCSL Specification
if(constraint.getSpecification()==null){
    propertyError("Error in Constraint Specification.");
    return false;
}

// TimedConstraint
if(constraint.getAppliedStereotype("MARTE::MARTE_Foundations::Time::TimedConstraint")==null){
    propertyError("Error in Timed Constraint Stereotype.");
    return false;
}
else {
    steConstraint = constraint.getAppliedStereotype("MARTE::MARTE_Foundations::Time::TimedConstraint");
    System.out.println("\nTimedConstraint Present.");
}

System.out.println("\n-------------------------------------------------------------------------");


// ---------------------------------------------------------------------------------------------------
// ---------------------------------------------------------------------------------------------------
// Generate the Observer Code
// ---------------------------------------------------------------------------------------------------

EventObserver EO = new EventObserver();

if(patternType==PatternEvent.Timed){
    switch(patternKind){
    case "Triggers":
        EO.generateTimedTriggers(strA, as, af, e, clk, min, max);
        break;
    case "Terminates":
        EO.generateTimedTerminates(strA, as, af, e, clk, min, max);
        break;
    default:
        propertyError("Error: Invalid Timed Pattern");
    }
}
else if(patternType==PatternEvent.Untimed){
    switch(patternKind){
    case "Triggers":
        EO.generateTriggers(strA, as, af, e);
```

```
                break;
            case "Terminates":
                EO.generateTerminates(strA, as, af, e);
                break;
            case "Forbids":
                EO.generateForbids(strA, as, af, e);
                break;
            case "Contains":
                EO.generateContains(strA, as, af, e);
                break;
            default:
                propertyError("Error: Invalid Untimed Pattern");
            }
        }
        else propertyError("Error: Invalid Pattern Type");


        // ---------------------------------------------------------------------------------------------

        EO.pw1.close();
        EO.pw2.close();
        EO.pw3.close();

        return true;
    }


    //*****************************************************************************************
    public Boolean caseBehaviorExecutionSpecification(BehaviorExecutionSpecification bes) {
        if(duplicateStateA == true) propertyError("Error: Duplicate Behavior Execution Specification");
        else {
            duplicateStateA = true;
            stateA = bes;
            strA = bes.getName();
        }
        return true;
    }


    // ******************************************************************************************
    public Boolean caseMessageOccurrenceSpecification(MessageOccurrenceSpecification mos) {
        if(duplicateMsgEnd == true) propertyError("Error: Duplicate Message Occurrence Specification");
        else {
            duplicateMsgEnd = true;
            MsgEnd = mos;
        }
        return true;
    }


    // ******************************************************************************************
    public void propertyError(String msg){
        System.out.println("\nIncorrect Property: "+msg);
        if(msg==null) msgError = "Error Occurred!";
        else msgError = msg;
    }

}
```

StatemachineInterface.java

```
// ************************************************************************************************
// ************************************************************************************************
// ************************************************************************************************
//
// This class provides the functions for the front-end of StateMachine Diagram.
// For Software to generate CCSL/Verilog Observers code from MARTE Model.
//
// @package     OBSERVER
// @class       StatemachineInterface
// @author      Aamir M. Khan
// @version     3.1
// @first       2016-02-02
// @current     2016-08-13
//
// ************************************************************************************************
// ************************************************************************************************
// ************************************************************************************************

package observer;

import org.eclipse.core.commands.AbstractHandler;
import org.eclipse.core.commands.ExecutionEvent;
import org.eclipse.core.commands.ExecutionException;
import org.eclipse.ui.IWorkbenchWindow;
import org.eclipse.ui.handlers.HandlerUtil;
import org.eclipse.jface.dialogs.MessageDialog;


import org.eclipse.core.commands.IHandler;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.jface.viewers.StructuredSelection;
import org.eclipse.ui.PlatformUI;

import org.eclipse.uml2.uml.*;


//*************************************************************************************************
public class StatemachineInterface extends AbstractHandler implements IHandler {
    /**
     * The constructor.
     */
    //*********************************************************************************************
    public StatemachineInterface() {
    }

    /**
     * the command has been executed, so extract extract the needed information
     * from the application context.
     */
    //*********************************************************************************************
    public Object execute(ExecutionEvent event) throws ExecutionException {

        ISelection selection = PlatformUI.
                getWorkbench().getActiveWorkbenchWindow().getActivePage().getSelection();
        if (!(selection instanceof StructuredSelection)) return null;
        Object selected = ((StructuredSelection)selection).getFirstElement();

        // The type should be guaranteed by the "isVisibleWhen"
        assert(selected instanceof StateMachine);


        StatemachineProcess SM = new StatemachineProcess();
        SM.doSwitch((StateMachine)selected);


        if(SM.msgError == null) SM.msgError = "CCSL/Verilog Observer Code Successfully Generated!";
        IWorkbenchWindow window = HandlerUtil.getActiveWorkbenchWindowChecked(event);
        MessageDialog.openInformation(
                window.getShell(),
                "Graphical Property Analysis",
                SM.msgError
                );

        return null;
```

```
    }
}
```

StatemachineProcess.java

```
// ************************************************************************************************
// ************************************************************************************************
// ************************************************************************************************
//
// This class provides the functions to extract information from UML StateMachine Diagram.
// For Software to generate CCSL/Verilog Observers code from MARTE Model.
//
// @package     OBSERVER
// @class       StatemachineProcess
// @author      Aamir M. Khan
// @version     3.1
// @first       2016-02-02
// @current     2016-08-13
//
// ************************************************************************************************
// ************************************************************************************************
// ************************************************************************************************

package observer;


import org.eclipse.emf.common.util.EList;
import org.eclipse.uml2.uml.*;
import org.eclipse.uml2.uml.util.UMLSwitch;


//************************************************************************************************
public class StatemachineProcess extends UMLSwitch<Boolean> {

    // Important Variables
    private State stateA, stateB;
    private OpaqueBehavior bhvA, bhvB;
    private String strSM, strA, strB;
    private String as,af,bs,bf,clk;
    private Stereotype steSM, steA, steStartA, steFinishA, steB, steStartB, steFinishB, steClk, steTransition, steConstraint;
    private Event evtStartA, evtFinishA, evtStartB, evtFinishB, evtClk;
    private Transition transition;
    private Constraint constraint;

    private enum PatternState {Timed,Untimed};
    PatternState patternType;
    private String patternKind;
    private int min,max;

    public String msgError;


    // ************************************************************************************************
    public Boolean caseStateMachine(StateMachine SM) {

        System.out.println("\n-------------------------------------------------------------------------");
        System.out.println("\ncaseStateMachine");

        StateObserver SO = new StateObserver();

        // ------------------------------------------------------------------------------------------------
        // Requirements: Check Elements + Stereotypes
        // GaAnalysisContext
        // ------------------------------------------------------------------------------------------------

        if(SM.getAppliedStereotype("MARTE::MARTE_AnalysisModel::GQAM::GaAnalysisContext")==null) {
            propertyError("Error: GaAnalysisContext Stereotype Not Applied.");
            return false;
        }
        else {
            System.out.println("\nGaAnalysisContext Present");
            strSM=SM.getName();
            steSM=SM.getAppliedStereotype("MARTE::MARTE_AnalysisModel::GQAM::GaAnalysisContext");
        }


        // ------------------------------------------------------------------------------------------------
        // Activation State
        // ------------------------------------------------------------------------------------------------

        if(SM.getRegions().size() != 1) {
            propertyError("Error in State Machine Region.");
            return false;
        }
        else if(SM.getRegions().get(0).getSubvertices().size() != 2) {
            propertyError("Error in State Machine Vertices.");
            return false;
        }
        else {
            for(Vertex vx : SM.getRegions().get(0).getSubvertices()) doSwitch(vx);
        }

        // Behavior_A
        if(stateA==null) {
            propertyError("Error in Activation State.");
            return false;
```

```
}
else if(stateA.getDoActivity()==null) {
    propertyError("Error in Activation State Behavior.");
    return false;
}
else {
    bhvA = (OpaqueBehavior) stateA.getDoActivity();
    System.out.println("\nBehavior A Present.");
}

// TimedProcessing A
if(bhvA.getAppliedStereotype("MARTE::MARTE_Foundations::Time::TimedProcessing")==null){
    propertyError("Error: TimedProcessing Stereotype Not Applied.");
    return false;
}
else {
    steA = bhvA.getAppliedStereotype("MARTE::MARTE_Foundations::Time::TimedProcessing");
    System.out.println("\nTimedProcessing A Present.");
}

// Tagged Value Start A
if(bhvA.getValue(steA, "start")==null){
    propertyError("Error in TimedProcessing Stereotype.");
    return false;
}
else {
    evtStartA = (Event) bhvA.getValue(steA, "start");
    as = evtStartA.getName();
    if(evtStartA.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock")==null) {
        propertyError("Error in Clock Stereotype.");
        return false;
    }
    else {
        steStartA = evtStartA.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock");
        System.out.println("\nClock Start A Present.");
    }
}

// Tagged Value Finish A
if(bhvA.getValue(steA, "finish")==null){
    propertyError("Error in TimedProcessing Stereotype.");
    return false;
}
else {
    evtFinishA = (Event) bhvA.getValue(steA, "finish");
    af = evtFinishA.getName();
    if(evtFinishA.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock")==null) {
        propertyError("Error in Clock Stereotype.");
        return false;
    }
    else {
        steFinishA = evtFinishA.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock");
        System.out.println("\nClock Finish A Present.");
    }
}

// Tagged Value On: Start A, Finish A
if(((EList<?>)bhvA.getValue(steA, "on")).size() == 0){
    propertyError("Error in TimedProcessing Stereotype.");
    return false;
}


// ------------------------------------------------------------------------------------------
// Transition
// ------------------------------------------------------------------------------------------

if(SM.getRegions().get(0).getTransitions().size() != 1) {
    propertyError("Error in State Transition.");
    return false;
}
else if(!SM.getRegions().get(0).getTransitions().get(0).getSource().equals(stateA)) {
    propertyError("Error in State Transition.");
    return false;
}
else transition = SM.getRegions().get(0).getTransitions().get(0);

// Pattern Stereotype
if((transition.getAppliedStereotype("MODEVES::UntimedStatePattern")==null) &&
   (transition.getAppliedStereotype("MODEVES::TimedStatePattern")==null)) {
    propertyError("Error in Timed/Untimed Pattern Stereotype.");
    return false;
}
else if((transition.getAppliedStereotype("MODEVES::UntimedStatePattern")!=null) &&
        (transition.getAppliedStereotype("MODEVES::TimedStatePattern")!=null)) {
    propertyError("Error in Timed/Untimed Pattern Stereotype.");
    return false;
}

// Untimed Pattern
else if(transition.getAppliedStereotype("MODEVES::UntimedStatePattern")!=null) {
```

```
    steTransition=transition.getAppliedStereotype("MODEVES::UntimedStatePattern");
    patternType=PatternState.Untimed;
    patternKind=(String)( ( (EnumerationLiteral)transition.getValue(steTransition, "kind") ).getName() );
    System.out.println("\nTransition Pattern Applied");
    System.out.println("\nType: "+patternType);
    System.out.println("\nKind: "+patternKind);
}

// Timed Pattern
else if(transition.getAppliedStereotype("MODEVES::TimedStatePattern")!=null){
    steTransition=transition.getAppliedStereotype("MODEVES::TimedStatePattern");
    patternType=PatternState.Timed;
    patternKind=(String)( ( (EnumerationLiteral)transition.getValue(steTransition, "kind") ).getName() );
    min=(Integer)transition.getValue(steTransition, "min");
    max=(Integer)transition.getValue(steTransition, "max");
    evtClk=(Event)transition.getValue(steTransition, "on");
    clk=evtClk.getName();

    if(evtClk.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock")==null){
        propertyError("Error in Timed Pattern Stereotype.");
        return false;
    } else {
        steClk = evtClk.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock");
        System.out.println("\nClock Present.");
    }

    System.out.println("\nTimed State Pattern Applied");
    System.out.println("\nType: "+patternType);
    System.out.println("\nKind: "+patternKind);
    System.out.println("\nMin: "+min);
    System.out.println("\nMax: "+max);
    System.out.println("\nOn: "+clk);
}


// ------------------------------------------------------------------------------------------------
// State B
// ------------------------------------------------------------------------------------------------

if(transition.getTarget()==null) {
    propertyError("Error in State Machine Transition or Vertices.");
    return false;
}
else {
    stateB=(State)transition.getTarget();
    strB=stateB.getName();
}

// Behavior B
if(stateB.getDoActivity()==null){
    propertyError("Error in State Behavior.");
    return false;
}
else {
    bhvB = (OpaqueBehavior) stateB.getDoActivity();
    System.out.println("\nBehavior B Present.");
}

// TimedProcessing
if(bhvB.getAppliedStereotype("MARTE::MARTE_Foundations::Time::TimedProcessing")==null){
    propertyError("Error: TimedProcessing Stereotype Not Applied.");
    return false;
}
else {
    steB = bhvB.getAppliedStereotype("MARTE::MARTE_Foundations::Time::TimedProcessing");
    System.out.println("\nTimedProcessing B Present.");
}

// Tagged Value Start B
if(bhvB.getValue(steB, "start")==null){
    propertyError("Error in TimedProcessing Stereotype.");
    return false;
}
else {
    evtStartB = (Event) bhvB.getValue(steB, "start");
    bs = evtStartB.getName();
    if(evtStartB.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock")==null) {
        propertyError("Error in Clock Stereotype.");
        return false;
    }
    else {
        steStartB = evtStartB.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock");
        System.out.println("\nClock Start B Present.");
    }
}

// Tagged Value Finish B
if(bhvB.getValue(steB, "finish")==null){
    propertyError("Error in TimedProcessing Stereotype.");
    return false;
}
```

```
else {
    evtFinishB = (Event) bhvB.getValue(steB, "finish");
    bf = evtFinishB.getName();
    if(evtFinishB.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock")==null) {
        propertyError("Error in Clock Stereotype.");
        return false;
    }
    else {
        steFinishB = evtFinishB.getAppliedStereotype("MARTE::MARTE_Foundations::Time::Clock");
        System.out.println("\nClock Finish B Present.");
    }
}

// Tagged Value On: Start B, Finish B
if(((EList<?>)bhvB.getValue(steB, "on")).size() == 0){
    propertyError("Error in TimedProcessing Stereotype.");
    return false;
}


// ---------------------------------------------------------------------------------------------------
// Property Constraint
// ---------------------------------------------------------------------------------------------------

// Constraint
if(SM.getOwnedRules().size()==1){
    System.out.println("\nConstraint Present");
    constraint = SM.getOwnedRules().get(0);
}
else {
    propertyError("Error in Property Constraint.");
    return false;
}


// CCSL Specification
if(constraint.getSpecification()==null){
    propertyError("Error in Constraint Specification.");
    return false;
}

// TimedConstraint
if(constraint.getAppliedStereotype("MARTE::MARTE_Foundations::Time::TimedConstraint")==null){
    propertyError("Error in Timed Constraint Stereotype.");
    return false;
}
else {
    steConstraint = constraint.getAppliedStereotype("MARTE::MARTE_Foundations::Time::TimedConstraint");
    System.out.println("\nTimedConstraint Present.");
}


// ---------------------------------------------------------------------------------------------------
// ---------------------------------------------------------------------------------------------------
// Generate the Observer Code
// ---------------------------------------------------------------------------------------------------


if(patternType==PatternState.Timed){
    switch(patternKind){
    case "Precedes":
        SO.generateTimedPrecedes(strA, strB, as, af, bs, bf, clk, min, max);
        break;
    case "Starts":
        SO.generateTimedStarts(strA, strB, as, af, bs, bf, clk, min, max);
        break;
    case "Finishes":
        SO.generateTimedFinishes(strA, strB, as, af, bs, bf, clk, min, max);
        break;
    default:
        propertyError("Error: Invalid Timed Pattern");
    }
}
else if(patternType==PatternState.Untimed){
    switch(patternKind){
    case "Precedes":
        SO.generatePrecedes(strA, strB, as, af, bs, bf);
        break;
    case "Starts":
        SO.generateStarts(strA, strB, as, af, bs, bf);
        break;
    case "Finishes":
        SO.generateFinishes(strA, strB, as, af, bs, bf);
        break;
    case "Causes":
        SO.generateCauses(strA, strB, as, af, bs, bf);
        break;
    case "Contains":
        SO.generateContains(strA, strB, as, af, bs, bf);
        break;
    case "Implies":
```

```
                SO.generateImplies(strA, strB, as, af, bs, bf);
                break;
            case "Forbids":
                SO.generateForbids(strA, strB, as, af, bs, bf);
                break;
            case "Excludes":
                SO.generateExcludes(strA, strB, as, af, bs, bf);
                break;
            default:
                propertyError("Error: Invalid Untimed Pattern");
            }
        }
        else propertyError("Error: Invalid Pattern Type");

        System.out.println("\n------------------------------------------------------------------------");

        SO.pw1.close();
        SO.pw2.close();
        SO.pw3.close();

        return true;
    }


    // ***********************************************************************************************
    public Boolean caseRegion(Region rg) {
        //System.out.println("caseRegion \n");
        for(Vertex vertex : rg.getSubvertices()) doSwitch(vertex);
        for(Transition transition : rg.getTransitions()) doSwitch(transition);
        return true;
    }


    // ***********************************************************************************************
    public Boolean caseState(State st) {
        //System.out.println("caseState \n");
        if(st.getAppliedStereotype("MODEVES::ActivationState") != null) {
            stateA = st;
            strA = stateA.getName();
        }
        return true;
    }


    // ***********************************************************************************************
    public void propertyError(String msg){
        System.out.println("\nIncorrect Property: "+msg);
        if(msg==null) msgError = "Error Occurred!";
        else msgError = msg;
    }

}
```

StateObserver.java

```
// ************************************************************************************************
// ************************************************************************************************
// ************************************************************************************************
//
// This class provides the CCSL and Verilog Observers generation facility for State-State Relations
//
// @package     OBSERVER
// @class       StateObserver
// @author      Aamir M. Khan
// @version     3.1
// @first       2016-02-02
// @current     2016-08-13
//
// ************************************************************************************************
// ************************************************************************************************
// ************************************************************************************************

package observer;


import java.io.FileOutputStream;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;


//************************************************************************************************
public class StateObserver {

    // Important Variables
    public PrintWriter pw1,pw2,pw3;


    // ********************************************************************************************
    // Constructor
    public StateObserver(){
        // create three files: CCSLPlus.txt, CCSL.txt, SystemVerilog.sv
        try {
            System.out.println("\nFile Output Initialization ");
            pw1 = new PrintWriter( new OutputStreamWriter(
                    new FileOutputStream("CCSLPlus.txt"),"UTF-8"));
            pw2 = new PrintWriter( new OutputStreamWriter(
                    new FileOutputStream("CCSL.txt"),"UTF-8"));
            pw3 = new PrintWriter( new OutputStreamWriter(
                    new FileOutputStream("SystemVerilog.sv"),"UTF-8"));

        } catch (Exception e) {
            pw1.print("ERROR OCCURED");
            pw2.print("ERROR OCCURED");
            pw3.print("ERROR OCCURED");
            }
    }


    // ********************************************************************************************
    public void generateTimedPrecedes(String strA,String strB,String as,String af,String bs,String bf,String clk,int min,int max) {
        System.out.println("\ngenerateTimedPrecedes");

        // CCSL+
        // A precedes B by [m,n] on clk
        pw1.print(strA);
        pw1.print(" precedes ");
        pw1.print(strB);
        pw1.print(" by [" + min + "," + max + "] on " + clk);


        // CCSL
        // af delayedFor min on clk precedes bs
        pw2.print(af + " delayedFor " + min + " on " + clk + " precedes " + bs);
        pw2.println("");

        // bs precedes af delayedFor max on clk
        pw2.print(bs + " precedes " + af + " delayedFor " + max + " on " + clk);
        pw2.println("");


        // as alternatesWith af
        pw2.print(as);
        pw2.print(" alternatesWith ");
        pw2.print(af);
        pw2.println("");

        // bs alternatesWith bf
        pw2.print(bs);
        pw2.print(" alternatesWith ");
        pw2.print(bf);


        // SystemVerilog
        // ---------------------------------------
        pw3.println("// @Generated");
        pw3.println("// " + strA + " precedes " + strB + " by [" + min + "," + max + "] on " + clk);
        pw3.println("");
        pw3.println("module Precedes (");
        pw3.println("  input " + as + ",");
        pw3.println("  input " + af + ",");
        pw3.println("  input " + bs + ",");
        pw3.println("  input " + bf + ",");
```

```
pw3.println("  input " + clk + ",");
pw3.println("  output violation");
pw3.println("  );");
pw3.println("  parameter min=" + min + ",max=" + max + ";");
pw3.println("  ");
pw3.println("  reg valid;");
pw3.println("  int unsigned d[max-1];");
pw3.println("  int unsigned ds;");
pw3.println("  int unsigned df;");
pw3.println("  int unsigned i;");
pw3.println("  ");
pw3.println("  int unsigned FSM;");
pw3.println("  reg v;");
pw3.println("  ");
pw3.println("  always @ (" + as + " or " + af + " or " + bs + " or " + bf + " or " + clk + ")");
pw3.println("  begin ");
pw3.println("    case(FSM)");
pw3.println("      // ------------------------------------------------------");
pw3.println("         0: // State A'B'");
pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1'b0)
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin ");
pw3.println("             if(i==0) begin FSM=0;valid=1'b1; end");
pw3.println("             else if(i>0 && d[ds]<max-1)");
pw3.println("             begin");
pw3.println("               FSM=0;valid=1'b1;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin");
pw3.println("             if(i==0 || d[ds]<max)");
pw3.println("             begin");
pw3.println("               valid=1'b1;FSM=2;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin");
pw3.println("             if(i>0 && d[ds]>=min && d[ds]<=max)");
pw3.println("             begin");
pw3.println("               i--; FSM=1;");
pw3.println("               ds=(ds+1) % max;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin");
pw3.println("             if(i>0 && d[ds]>=min-1 && d[ds]<max)");
pw3.println("             begin");
pw3.println("               i--;FSM=1;valid=1'b1;");
pw3.println("               ds=(ds+1) % max;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin");
pw3.println("             if(i>0 && d[ds]>=min && d[ds]<=max)");
pw3.println("             begin");
pw3.println("               i--; FSM=3;");
pw3.println("               ds=(ds+1) % max;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b1 && " + clk + "==1
pw3.println("           begin");
pw3.println("             if(i>0 && d[ds]>=min-1 && d[ds]<max)");
pw3.println("             begin");
pw3.println("               i--;FSM=3;valid=1'b1;");
pw3.println("               ds=(ds+1) % max;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end     ");
pw3.println("           else ");
pw3.println("           begin");
pw3.println("             FSM=4;v=1'b1;");
pw3.println("           end");
pw3.println("         // ------------------------------------------------------");
pw3.println("         1: // State A'B");
```

```
pw3.println("          if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1'b0)
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          begin ");
pw3.println("            if(i==0) begin FSM=1;valid=1'b1; end");
pw3.println("            else if(i>0 && d[ds]<max-1)");
pw3.println("            begin");
pw3.println("              FSM=1;valid=1'b1;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=4;v=1'b1;");
pw3.println("            end");
pw3.println("          end");
pw3.println("          else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          begin");
pw3.println("            if(i==0 || d[ds]<max)");
pw3.println("            begin");
pw3.println("              valid=1'b1;FSM=3;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("          end");
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1 && " + clk + "==1
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1 && " + clk + "==1
pw3.println("          begin");
pw3.println("            if(i==0 || d[ds]<max)");
pw3.println("            begin");
pw3.println("              FSM=0;valid=1'b1;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=4;v=1'b1;");
pw3.println("            end");
pw3.println("          end ");
pw3.println("          else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1 && " + clk + "==1
pw3.println("          else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1 && " + clk + "==1
pw3.println("          begin");
pw3.println("            if(i==0 || d[ds]<max)");
pw3.println("            begin");
pw3.println("              FSM=2;valid=1'b1;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=4;v=1'b1;");
pw3.println("            end");
pw3.println("          end   ");
pw3.println("          else ");
pw3.println("          begin");
pw3.println("            FSM=4;v=1'b1;");
pw3.println("          end");
pw3.println("        // -------------------------------------------------------");
pw3.println("        2: // State AB'");
pw3.println("          if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1'b0)
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          begin ");
pw3.println("            if(i==0) begin FSM=2;valid=1'b1; end");
pw3.println("            else if(i>0 && d[ds]<max-1)");
pw3.println("            begin");
pw3.println("              FSM=2;valid=1'b1;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=4;v=1'b1;");
pw3.println("            end");
pw3.println("          end");
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          begin");
pw3.println("            FSM=0;");
pw3.println("            if(valid==1'b1)");
pw3.println("            begin");
pw3.println("              i++;valid=1'b0;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("          end ");
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          begin");
pw3.println("            if(valid==1'b0 && i==0 || d[ds]<max)");
pw3.println("            begin");
pw3.println("              valid=1'b1;FSM=0;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else if(valid==1'b1 && i==0 || d[ds]<max)");
pw3.println("            begin");
pw3.println("              i++;valid=1'b0;FSM=0;");
pw3.println("              d[(df+1) % max]=0;");
pw3.println("              df=(df+1) % max;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=4;v=1'b1;");
pw3.println("            end");
pw3.println("          end");
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0 && " + clk + "==1
```

```
pw3.println("           begin");
pw3.println("             if(i>0 && d[ds]>=min && d[ds]<=max)");
pw3.println("             begin");
pw3.println("               i--; FSM=3;");
pw3.println("               ds=(ds+1) % max;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin");
pw3.println("             if(i>0 && d[ds]>=min-1 && d[ds]<max)");
pw3.println("             begin");
pw3.println("               i--;valid=1'b1;FSM=3;");
pw3.println("               ds=(ds+1) % max;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end ");
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b1 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b1 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin");
pw3.println("             if(i==0 || d[ds]<max)");
pw3.println("             begin");
pw3.println("               valid=1'b1;FSM=1;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end   ");
pw3.println("           else ");
pw3.println("           begin");
pw3.println("             FSM=4;v=1'b1;");
pw3.println("           end");
pw3.println("         // -------------------------------------------------------");
pw3.println("         3: // State AB");
pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1'b0)
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin ");
pw3.println("             if(i==0) begin FSM=3;valid=1'b1; end");
pw3.println("             else if(i>0 && d[ds]<max-1)");
pw3.println("             begin");
pw3.println("               FSM=3;valid=1'b1;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin");
pw3.println("             FSM=1;");
pw3.println("             if(valid==1'b1)");
pw3.println("             begin");
pw3.println("               i++;valid=1'b0;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("           end ");
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin");
pw3.println("             if(valid==1'b0 && i==0 || d[ds]<max)");
pw3.println("             begin");
pw3.println("               valid=1'b1;FSM=1;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else if(valid==1'b1 && i==0 || d[ds]<max)");
pw3.println("             begin");
pw3.println("               i++;valid=1'b0;FSM=1;");
pw3.println("               d[(df+1) % max]=0;");
pw3.println("               df=(df+1) % max;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1 && " + clk + "==1
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1 && " + clk + "==1
pw3.println("           begin");
pw3.println("             if(i==0 || d[ds]<max)");
pw3.println("             begin");
pw3.println("               FSM=2;valid=1'b1;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end ");
```

```
        pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b1 && " + clk + "==1
        pw3.println("           begin");
        pw3.println("             FSM=0;");
        pw3.println("             if(valid==1'b1)");
        pw3.println("             begin");
        pw3.println("               i++;valid=1'b0;");
        pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
        pw3.println("             end");
        pw3.println("           end");
        pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b1 && " + clk + "==1
        pw3.println("        begin");
        pw3.println("          if(valid==1'b0 && i==0 || d[ds]<max)");
        pw3.println("             begin");
        pw3.println("               valid=1'b1;FSM=0;");
        pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
        pw3.println("             end");
        pw3.println("             else if(valid==1'b1 && i==0 || d[ds]<max)");
        pw3.println("             begin");
        pw3.println("               i++;valid=1'b0;FSM=0;");
        pw3.println("               d[(df+1) % max]=0;");
        pw3.println("               df=(df+1) % max;");
        pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
        pw3.println("             end");
        pw3.println("             else");
        pw3.println("             begin");
        pw3.println("               FSM=4;v=1'b1;");
        pw3.println("             end");
        pw3.println("        end");
        pw3.println("           else ");
        pw3.println("           begin");
        pw3.println("             FSM=4;v=1'b1;");
        pw3.println("           end");
        pw3.println("         // -------------------------------------------------------");
        pw3.println("         default:  // Violation ");
        pw3.println("         begin");
        pw3.println("           FSM=4;");
        pw3.println("           v=1'b1;");
        pw3.println("         end     ");
        pw3.println("      endcase");
        pw3.println("   end");
        pw3.println("   ");
        pw3.println("  assign violation = v;");
        pw3.println("           ");
        pw3.println("  initial ");
        pw3.println("  begin");
        pw3.println("    valid=1'b1;");
        pw3.println("    ds=0;");
        pw3.println("    df=max-1;");
        pw3.println("    d[max-1]=0;");
        pw3.println("    i=0;");
        pw3.println("    FSM = 0;");
        pw3.println("    v = 0;");
        pw3.println("  end");
        pw3.println("endmodule");

    }


    // ********************************************************************************************
    public void generateTimedStarts(String strA,String strB,String as,String af,String bs,String bf,String clk,int min,int max) {
        System.out.println("\ngenerateTimedStarts");

        // CCSL+
        // A starts B after [m,n] on clk
        pw1.print(strA);
        pw1.print(" starts ");
        pw1.print(strB);
        pw1.print(" after [" + min + "," + max + "] on " + clk);


        // CCSL
        // as delayedFor min on clk precedes bs
        pw2.print(as + " delayedFor " + min + " on " + clk + " precedes " + bs);
        pw2.println("");

        // bs precedes as delayedFor max on clk
        pw2.print(bs + " precedes " + as + " delayedFor " + max + " on " + clk);
        pw2.println("");

        // as alternatesWith af
        pw2.print(as);
        pw2.print(" alternatesWith ");
        pw2.print(af);
        pw2.println("");

        // bs alternatesWith bf
        pw2.print(bs);
        pw2.print(" alternatesWith ");
        pw2.print(bf);


        // SystemVerilog
        pw3.println("// @Generated");
        pw3.println("// " + strA + " starts " + strB + " after [" + min + "," + max + "] on " + clk);
        pw3.println("");
        pw3.println("module Starts (");
        pw3.println("  input " + as + ",");
        pw3.println("  input " + af + ",");
```

```
pw3.println("  input " + bs + ",");
pw3.println("  input " + bf + ",");
pw3.println("  input " + clk + ",");
pw3.println("  output violation");
pw3.println("  );");
pw3.println("  parameter min=" + min + ",max=" + max + ";");
pw3.println("  ");
pw3.println("  reg valid;");
pw3.println("  int unsigned d[max-1];");
pw3.println("  int unsigned ds;");
pw3.println("  int unsigned df;");
pw3.println("  int unsigned i;");
pw3.println("  ");
pw3.println("  int unsigned FSM;");
pw3.println("  reg v;");
pw3.println("  ");
pw3.println("  always @ (" + as + " or " + af + " or " + bs + " or " + bf + " or " + clk + ")");
pw3.println("  begin ");
pw3.println("    case(FSM)");
pw3.println("      // ------------------------------------------------------");
pw3.println("         0: // State A'B'");
pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1'b0)
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin ");
pw3.println("             if(i==0) begin FSM=0;valid=1'b1; end");
pw3.println("             else if(i>0 && d[ds]<max-1)");
pw3.println("             begin");
pw3.println("               FSM=0;valid=1'b1;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin");
pw3.println("             if(i==0 || d[ds]<max)");
pw3.println("             begin");
pw3.println("               valid=1'b1;FSM=2;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin");
pw3.println("             if(i>0 && d[ds]>=min && d[ds]<=max)");
pw3.println("             begin");
pw3.println("               i--; FSM=1;");
pw3.println("               ds=(ds+1) % max;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin");
pw3.println("             if(i>0 && d[ds]>=min-1 && d[ds]<max)");
pw3.println("             begin");
pw3.println("               i--;FSM=1;valid=1'b1;");
pw3.println("               ds=(ds+1) % max;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin");
pw3.println("             if(i>0 && d[ds]>=min && d[ds]<=max)");
pw3.println("             begin");
pw3.println("               i--; FSM=3;");
pw3.println("               ds=(ds+1) % max;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin");
pw3.println("             if(i>0 && d[ds]>=min-1 && d[ds]<max)");
pw3.println("             begin");
pw3.println("               i--;FSM=3;valid=1'b1;");
pw3.println("               ds=(ds+1) % max;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end     ");
pw3.println("           else ");
pw3.println("           begin");
pw3.println("             FSM=4;v=1'b1;");
pw3.println("           end");
```

```
pw3.println("         // ------------------------------------------------------");
pw3.println("         1: // State A'B");
pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1'b0)
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin ");
pw3.println("             if(i==0) begin FSM=1;valid=1'b1; end");
pw3.println("             else if(i>0 && d[ds]<max-1)");
pw3.println("             begin");
pw3.println("               FSM=1;valid=1'b1;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin");
pw3.println("             if(i==0 || d[ds]<max)");
pw3.println("             begin");
pw3.println("               valid=1'b1;FSM=3;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1 && " + clk + "==1
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1 && " + clk + "==1
pw3.println("           begin");
pw3.println("             if(i==0 || d[ds]<max)");
pw3.println("             begin");
pw3.println("               FSM=0;valid=1'b1;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end ");
pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1 && " + clk + "==1
pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1 && " + clk + "==1
pw3.println("           begin");
pw3.println("             if(i==0 || d[ds]<max)");
pw3.println("             begin");
pw3.println("               FSM=2;valid=1'b1;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end   ");
pw3.println("           else ");
pw3.println("           begin");
pw3.println("             FSM=4;v=1'b1;");
pw3.println("           end");
pw3.println("         // ------------------------------------------------------");
pw3.println("         2: // State AB'");
pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1'b0)
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin ");
pw3.println("             if(i==0) begin FSM=2;valid=1'b1; end");
pw3.println("             else if(i>0 && d[ds]<max-1)");
pw3.println("             begin");
pw3.println("               FSM=2;valid=1'b1;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin");
pw3.println("             FSM=0;");
pw3.println("             if(valid==1'b1)");
pw3.println("             begin");
pw3.println("               i++;valid=1'b0;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("           end ");
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin");
pw3.println("             if(valid==1'b0 && i==0 || d[ds]<max)");
pw3.println("             begin");
pw3.println("               valid=1'b1;FSM=0;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else if(valid==1'b1 && i==0 || d[ds]<max)");
pw3.println("             begin");
pw3.println("               i++;valid=1'b0;FSM=0;");
pw3.println("               d[(df+1) % max]=0;");
pw3.println("               df=(df+1) % max;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
```

```
pw3.println("          end");
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          begin");
pw3.println("            if(i>0 && d[ds]>=min && d[ds]<=max)");
pw3.println("            begin");
pw3.println("              i--; FSM=3;");
pw3.println("              ds=(ds+1) % max;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=4;v=1'b1;");
pw3.println("            end");
pw3.println("          end");
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          begin");
pw3.println("            if(i>0 && d[ds]>=min-1 && d[ds]<max)");
pw3.println("            begin");
pw3.println("              i--;valid=1'b1;FSM=3;");
pw3.println("              ds=(ds+1) % max;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=4;v=1'b1;");
pw3.println("            end");
pw3.println("          end ");
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b1 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b1 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          begin");
pw3.println("            if(i==0 || d[ds]<max)");
pw3.println("            begin");
pw3.println("              valid=1'b1;FSM=1;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=4;v=1'b1;");
pw3.println("            end");
pw3.println("          end   ");
pw3.println("          else ");
pw3.println("          begin");
pw3.println("            FSM=4;v=1'b1;");
pw3.println("          end");
pw3.println("        // ----------------------------------------------------------");
pw3.println("        3: // State AB");
pw3.println("          if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1'b0)
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          begin ");
pw3.println("            if(i==0) begin FSM=3;valid=1'b1; end");
pw3.println("            else if(i>0 && d[ds]<max-1)");
pw3.println("            begin");
pw3.println("              FSM=3;valid=1'b1;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=4;v=1'b1;");
pw3.println("            end");
pw3.println("          end");
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          begin");
pw3.println("            FSM=1;");
pw3.println("            if(valid==1'b1)");
pw3.println("            begin");
pw3.println("              i++;valid=1'b0;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("          end ");
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          begin");
pw3.println("            if(valid==1'b0 && i==0 || d[ds]<max)");
pw3.println("            begin");
pw3.println("              valid=1'b1;FSM=1;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else if(valid==1'b1 && i==0 || d[ds]<max)");
pw3.println("            begin");
pw3.println("              i++;valid=1'b0;FSM=1;");
pw3.println("              d[(df+1) % max]=0;");
pw3.println("              df=(df+1) % max;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=4;v=1'b1;");
pw3.println("            end");
pw3.println("          end");
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1 && " + clk + "==1
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1 && " + clk + "==1
pw3.println("          begin");
pw3.println("            if(i==0 || d[ds]<max)");
pw3.println("            begin");
pw3.println("              FSM=2;valid=1'b1;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=4;v=1'b1;");
```

```
    pw3.println("             end");
    pw3.println("           end ");
    pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b1 && " + clk + "==1
    pw3.println("           begin");
    pw3.println("             FSM=0;");
    pw3.println("             if(valid==1'b1)");
    pw3.println("             begin");
    pw3.println("               i++;valid=1'b0;");
    pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
    pw3.println("             end");
    pw3.println("           end");
    pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b1 && " + clk + "==1
    pw3.println("         begin");
    pw3.println("           if(valid==1'b0 && i==0 || d[ds]<max)");
    pw3.println("             begin");
    pw3.println("               valid=1'b1;FSM=0;");
    pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
    pw3.println("             end");
    pw3.println("             else if(valid==1'b1 && i==0 || d[ds]<max)");
    pw3.println("             begin");
    pw3.println("               i++;valid=1'b0;FSM=0;");
    pw3.println("               d[(df+1) % max]=0;");
    pw3.println("               df=(df+1) % max;");
    pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
    pw3.println("             end");
    pw3.println("             else");
    pw3.println("             begin");
    pw3.println("               FSM=4;v=1'b1;");
    pw3.println("             end");
    pw3.println("         end");
    pw3.println("           else ");
    pw3.println("           begin");
    pw3.println("             FSM=4;v=1'b1;");
    pw3.println("           end");
    pw3.println("          // -------------------------------------------------------");
    pw3.println("          default:  // Violation ");
    pw3.println("          begin");
    pw3.println("            FSM=4;");
    pw3.println("            v=1'b1;");
    pw3.println("          end     ");
    pw3.println("       endcase");
    pw3.println("    end");
    pw3.println("    ");
    pw3.println("  assign violation = v;");
    pw3.println("              ");
    pw3.println("  initial ");
    pw3.println("  begin");
    pw3.println("    valid=1'b1;");
    pw3.println("    ds=0;");
    pw3.println("    df=max-1;");
    pw3.println("    d[max-1]=0;");
    pw3.println("    i=0;");
    pw3.println("    FSM = 0;");
    pw3.println("    v = 0;");
    pw3.println("  end");
    pw3.println("endmodule");

}

// ************************************************************************************************
public void generateTimedFinishes(String strA,String strB,String as,String af,String bs,String bf,String clk,int min,int max) {
    System.out.println("\ngenerateTimedFinishes");

    // CCSL+
    // A finishes B after [m,n] on clk
    pw1.print(strA);
    pw1.print(" finishes ");
    pw1.print(strB);
    pw1.print(" after [" + min + "," + max + "] on " + clk);


    // CCSL
    // af delayedFor min on clk precedes bf
    pw2.print(af + " delayedFor " + min + " on " + clk + " precedes " + bf);
    pw2.println("");

    // bf precedes af delayedFor max on clk
    pw2.print(bf + " precedes " + af + " delayedFor " + max + " on " + clk);
    pw2.println("");

    // as alternatesWith af
    pw2.print(as);
    pw2.print(" alternatesWith ");
    pw2.print(af);
    pw2.println("");

    // bs alternatesWith bf
    pw2.print(bs);
    pw2.print(" alternatesWith ");
    pw2.print(bf);


    // SystemVerilog
    pw3.println("// @Generated");
    pw3.println("// " + strA + " finishes " + strB + " after [" + min + "," + max + "] on " + clk);
    pw3.println("");
    pw3.println("module Finishes (");
    pw3.println("  input " + as + ",");
```

```
pw3.println("  input " + af + ",");
pw3.println("  input " + bs + ",");
pw3.println("  input " + bf + ",");
pw3.println("  input " + clk + ",");
pw3.println("  output violation");
pw3.println("  );");
pw3.println("  parameter min=" + min + ",max=" + max + ";");
pw3.println("  ");
pw3.println("  reg valid;");
pw3.println("  int unsigned d[max-1];");
pw3.println("  int unsigned ds;");
pw3.println("  int unsigned df;");
pw3.println("  int unsigned i;");
pw3.println("  ");
pw3.println("  int unsigned FSM;");
pw3.println("  reg v;");
pw3.println("  ");
pw3.println("  always @ (" + as + " or " + af + " or " + bs + " or " + bf + " or " + clk + ")");
pw3.println("  begin ");
pw3.println("    case(FSM)");
pw3.println("      // ------------------------------------------------------");
pw3.println("         0: // State A'B'");
pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1'b0)
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin ");
pw3.println("             if(i==0) begin FSM=0;valid=1'b1; end");
pw3.println("             else if(i>0 && d[ds]<max-1)");
pw3.println("             begin");
pw3.println("               FSM=0;valid=1'b1;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin");
pw3.println("             if(i==0 || d[ds]<max)");
pw3.println("             begin");
pw3.println("               valid=1'b1;FSM=2;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin");
pw3.println("             if(i>0 && d[ds]>=min && d[ds]<=max)");
pw3.println("             begin");
pw3.println("               i--; FSM=1;");
pw3.println("               ds=(ds+1) % max;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin");
pw3.println("             if(i>0 && d[ds]>=min-1 && d[ds]<max)");
pw3.println("             begin");
pw3.println("               i--;FSM=1;valid=1'b1;");
pw3.println("               ds=(ds+1) % max;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin");
pw3.println("             if(i>0 && d[ds]>=min && d[ds]<=max)");
pw3.println("             begin");
pw3.println("               i--; FSM=3;");
pw3.println("               ds=(ds+1) % max;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("           begin");
pw3.println("             if(i>0 && d[ds]>=min-1 && d[ds]<max)");
pw3.println("             begin");
pw3.println("               i--;FSM=3;valid=1'b1;");
pw3.println("               ds=(ds+1) % max;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=4;v=1'b1;");
pw3.println("             end");
pw3.println("           end     ");
pw3.println("           else ");
pw3.println("           begin");
pw3.println("             FSM=4;v=1'b1;");
```

```
pw3.println("          end");
pw3.println("        // ------------------------------------------------------");
pw3.println("        1: // State A'B");
pw3.println("          if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1'b0)
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          begin ");
pw3.println("            if(i==0) begin FSM=1;valid=1'b1; end");
pw3.println("            else if(i>0 && d[ds]<max-1)");
pw3.println("            begin");
pw3.println("              FSM=1;valid=1'b1;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=4;v=1'b1;");
pw3.println("            end");
pw3.println("          end");
pw3.println("          else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          begin");
pw3.println("            if(i==0 || d[ds]<max)");
pw3.println("            begin");
pw3.println("              valid=1'b1;FSM=3;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("          end");
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1 && " + clk + "==1
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1 && " + clk + "==1
pw3.println("          begin");
pw3.println("            if(i==0 || d[ds]<max)");
pw3.println("            begin");
pw3.println("              FSM=0;valid=1'b1;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=4;v=1'b1;");
pw3.println("            end");
pw3.println("          end ");
pw3.println("          else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1 && " + clk + "==1
pw3.println("          else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1 && " + clk + "==1
pw3.println("          begin");
pw3.println("            if(i==0 || d[ds]<max)");
pw3.println("            begin");
pw3.println("              FSM=2;valid=1'b1;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=4;v=1'b1;");
pw3.println("            end");
pw3.println("          end   ");
pw3.println("          else ");
pw3.println("          begin");
pw3.println("            FSM=4;v=1'b1;");
pw3.println("          end");
pw3.println("        // ------------------------------------------------------");
pw3.println("        2: // State AB'");
pw3.println("          if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1'b0)
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          begin ");
pw3.println("            if(i==0) begin FSM=2;valid=1'b1; end");
pw3.println("            else if(i>0 && d[ds]<max-1)");
pw3.println("            begin");
pw3.println("              FSM=2;valid=1'b1;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=4;v=1'b1;");
pw3.println("            end");
pw3.println("          end");
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          begin");
pw3.println("            FSM=0;");
pw3.println("            if(valid==1'b1)");
pw3.println("            begin");
pw3.println("              i++;valid=1'b0;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("          end ");
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          begin");
pw3.println("            if(valid==1'b0 && i==0 || d[ds]<max)");
pw3.println("            begin");
pw3.println("              valid=1'b1;FSM=0;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else if(valid==1'b1 && i==0 || d[ds]<max)");
pw3.println("            begin");
pw3.println("              i++;valid=1'b0;FSM=0;");
pw3.println("              d[(df+1) % max]=0;");
pw3.println("              df=(df+1) % max;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=4;v=1'b1;");
```

```
pw3.println("            end");
pw3.println("          end");
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          begin");
pw3.println("            if(i>0 && d[ds]>=min && d[ds]<=max)");
pw3.println("            begin");
pw3.println("              i--; FSM=3;");
pw3.println("              ds=(ds+1) % max;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=4;v=1'b1;");
pw3.println("            end");
pw3.println("          end");
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          begin");
pw3.println("            if(i>0 && d[ds]>=min-1 && d[ds]<max)");
pw3.println("            begin");
pw3.println("              i--;valid=1'b1;FSM=3;");
pw3.println("              ds=(ds+1) % max;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=4;v=1'b1;");
pw3.println("            end");
pw3.println("          end ");
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b1 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b1 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          begin");
pw3.println("            if(i==0 || d[ds]<max)");
pw3.println("            begin");
pw3.println("              valid=1'b1;FSM=1;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=4;v=1'b1;");
pw3.println("            end");
pw3.println("          end   ");
pw3.println("          else ");
pw3.println("          begin");
pw3.println("            FSM=4;v=1'b1;");
pw3.println("          end");
pw3.println("        // ----------------------------------------------------");
pw3.println("        3: // State AB");
pw3.println("          if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1'b0)
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          begin ");
pw3.println("            if(i==0) begin FSM=3;valid=1'b1; end");
pw3.println("            else if(i>0 && d[ds]<max-1)");
pw3.println("            begin");
pw3.println("              FSM=3;valid=1'b1;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=4;v=1'b1;");
pw3.println("            end");
pw3.println("          end");
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          begin");
pw3.println("            FSM=1;");
pw3.println("            if(valid==1'b1)");
pw3.println("            begin");
pw3.println("              i++;valid=1'b0;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("          end ");
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b0 && " + clk + "==1
pw3.println("          begin");
pw3.println("            if(valid==1'b0 && i==0 || d[ds]<max)");
pw3.println("            begin");
pw3.println("              valid=1'b1;FSM=1;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else if(valid==1'b1 && i==0 || d[ds]<max)");
pw3.println("            begin");
pw3.println("              i++;valid=1'b0;FSM=1;");
pw3.println("              d[(df+1) % max]=0;");
pw3.println("              df=(df+1) % max;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=4;v=1'b1;");
pw3.println("            end");
pw3.println("          end");
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1 && " + clk + "==1
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1 && " + clk + "==1
pw3.println("          begin");
pw3.println("            if(i==0 || d[ds]<max)");
pw3.println("            begin");
pw3.println("              FSM=2;valid=1'b1;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
```

```
    pw3.println("              FSM=4;v=1'b1;");
    pw3.println("            end");
    pw3.println("          end ");
    pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b1 && " + clk + "==1
    pw3.println("          begin");
    pw3.println("            FSM=0;");
    pw3.println("            if(valid==1'b1)");
    pw3.println("            begin");
    pw3.println("              i++;valid=1'b0;");
    pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
    pw3.println("            end");
    pw3.println("          end");
    pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b1 && " + clk + "==1
    pw3.println("        begin");
    pw3.println("          if(valid==1'b0 && i==0 || d[ds]<max)");
    pw3.println("            begin");
    pw3.println("              valid=1'b1;FSM=0;");
    pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
    pw3.println("            end");
    pw3.println("            else if(valid==1'b1 && i==0 || d[ds]<max)");
    pw3.println("            begin");
    pw3.println("              i++;valid=1'b0;FSM=0;");
    pw3.println("              d[(df+1) % max]=0;");
    pw3.println("              df=(df+1) % max;");
    pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
    pw3.println("            end");
    pw3.println("            else");
    pw3.println("            begin");
    pw3.println("              FSM=4;v=1'b1;");
    pw3.println("            end");
    pw3.println("        end");
    pw3.println("          else ");
    pw3.println("          begin");
    pw3.println("            FSM=4;v=1'b1;");
    pw3.println("          end");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         default:  // Violation ");
    pw3.println("         begin");
    pw3.println("           FSM=4;");
    pw3.println("           v=1'b1;");
    pw3.println("         end     ");
    pw3.println("      endcase");
    pw3.println("   end");
    pw3.println("   ");
    pw3.println("  assign violation = v;");
    pw3.println("           ");
    pw3.println("  initial ");
    pw3.println("  begin");
    pw3.println("    valid=1'b1;");
    pw3.println("    ds=0;");
    pw3.println("    df=max-1;");
    pw3.println("    d[max-1]=0;");
    pw3.println("    i=0;");
    pw3.println("    FSM = 0;");
    pw3.println("    v = 0;");
    pw3.println("  end");
    pw3.println("endmodule");

}


// ********************************************************************************************
// ********************************************************************************************
// UNTIMED PATTERNS
// ********************************************************************************************
// ********************************************************************************************


// ********************************************************************************************
public void generatePrecedes(String strA,String strB,String as,String af,String bs,String bf) {
    System.out.println("\ngeneratePrecedes");

    // CCSL+
    // A precedes B
    pw1.print(strA);
    pw1.print(" precedes ");
    pw1.print(strB);


    // CCSL
    // af precedes bs
    pw2.print(af);
    pw2.print(" precedes ");
    pw2.print(bs);
    pw2.println("");

    // as alternatesWith af
    pw2.print(as);
    pw2.print(" alternatesWith ");
    pw2.print(af);
    pw2.println("");

    // bs alternatesWith bf
    pw2.print(bs);
    pw2.print(" alternatesWith ");
    pw2.print(bf);
```

```
// SystemVerilog
pw3.println("// @Generated");
pw3.println("// " + strA + " precedes " + strB);
pw3.println("");
pw3.println("module Precedes (");
pw3.println("  input " + as + ",");
pw3.println("  input " + af + ",");
pw3.println("  input " + bs + ",");
pw3.println("  input " + bf + ",");
pw3.println("  output violation");
pw3.println("  );");
pw3.println("  ");
pw3.println("  int unsigned delta;");
pw3.println("  int unsigned FSM;");
pw3.println("  reg v;");
pw3.println("  ");
pw3.println("  always @ (" + as + " or " + af + " or " + bs + " or " + bf + ")");
pw3.println("  begin ");
pw3.println("    case(FSM)");
pw3.println("         // ------------------------------------------------------");
pw3.println("         0: // State " + strA + "'" + strB + "'");
pw3.println("         begin");
pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=0;      // empty"
pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=2; // " + as
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0)        // " + bs
pw3.println("           begin");
pw3.println("               if (delta>0) ");
pw3.println("               begin");
pw3.println("                   delta--;");
pw3.println("                   FSM=1;  ");
pw3.println("               end");
pw3.println("               else");
pw3.println("               begin");
pw3.println("                   FSM=4;");
pw3.println("                   v=1'b1;");
pw3.println("               end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0)       // " + as
pw3.println("           begin");
pw3.println("               if (delta>0) ");
pw3.println("               begin");
pw3.println("                   delta--;");
pw3.println("                   FSM=3;  ");
pw3.println("               end");
pw3.println("               else");
pw3.println("               begin");
pw3.println("                   FSM=4;");
pw3.println("                   v=1'b1;");
pw3.println("               end");
pw3.println("           end");
pw3.println("           else // Violation");
pw3.println("           begin");
pw3.println("             FSM=4;v=1'b1;");
pw3.println("           end");
pw3.println("         end");
pw3.println("         // ------------------------------------------------------");
pw3.println("         1: // State " + strA + "'" + strB);
pw3.println("         begin");
pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=1;      // empty"
pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=3; // " + as
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1) FSM=0; // " + bf
pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1) FSM=2; // " + as
pw3.println("           else // Violation");
pw3.println("           begin");
pw3.println("             FSM=4;v=1'b1;");
pw3.println("           end");
pw3.println("         end");
pw3.println("         // ------------------------------------------------------");
pw3.println("         2: // State " + strA + strB + "'");
pw3.println("         begin");
pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=2;     // empty")
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b0)       // " + af)
pw3.println("           begin");
pw3.println("               delta++;");
pw3.println("               FSM=0;  ");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0)       // " + bs)
pw3.println("           begin");
pw3.println("               if (delta>0) ");
pw3.println("               begin");
pw3.println("                   delta--;");
pw3.println("                   FSM=3;  ");
pw3.println("               end");
pw3.println("               else");
pw3.println("               begin");
pw3.println("                   FSM=4;");
pw3.println("                   v=1'b1;");
pw3.println("               end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b1 && " + bf + "==1'b0) FSM=1; // " + af
pw3.println("           else // Violation");
pw3.println("           begin");
pw3.println("             FSM=4;v=1'b1;");
pw3.println("           end");
pw3.println("         end");
pw3.println("         // ------------------------------------------------------");
pw3.println("         3: // State " + strA + strB);
pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=3;     // empty")
```

```
        pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b0)       // " + af)
        pw3.println("           begin");
        pw3.println("               delta++;");
        pw3.println("               FSM=1;  ");
        pw3.println("           end");
        pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1) FSM=2;// " + bf)
        pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b1)       // " + af
        pw3.println("           begin");
        pw3.println("               delta++;");
        pw3.println("               FSM=0;  ");
        pw3.println("           end");
        pw3.println("           else // Violation");
        pw3.println("           begin");
        pw3.println("             FSM=4;v=1'b1;");
        pw3.println("           end");
        pw3.println("         // -------------------------------------------------------");
        pw3.println("         default:  // State 4: Violation ");
        pw3.println("         begin");
        pw3.println("           FSM=4;");
        pw3.println("           v=1'b1;");
        pw3.println("         end     ");
        pw3.println("      endcase");
        pw3.println("   end");
        pw3.println("   ");
        pw3.println("  assign violation = v;");
        pw3.println("           ");
        pw3.println("endmodule");
    }


// ************************************************************************************************
    public void generateStarts(String strA,String strB,String as,String af,String bs,String bf) {
        System.out.println("\ngenerateStarts");

        // CCSL+
        // A starts B
        pw1.print(strA);
        pw1.print(" starts ");
        pw1.print(strB);


        // CCSL
        // bs isSubclockOf as
        pw2.print(bs);
        pw2.print(" isSubclockOf ");
        pw2.print(as);
        pw2.println("");

        // as alternatesWith af
        pw2.print(as);
        pw2.print(" alternatesWith ");
        pw2.print(af);
        pw2.println("");

        // bs alternatesWith bf
        pw2.print(bs);
        pw2.print(" alternatesWith ");
        pw2.print(bf);


        // SystemVerilog
        pw3.println("// @Generated");
        pw3.println("// " + strA + " starts " + strB);
        pw3.println("");
        pw3.println("module Starts (");
        pw3.println("  input " + as + ",");
        pw3.println("  input " + af + ",");
        pw3.println("  input " + bs + ",");
        pw3.println("  input " + bf + ",");
        pw3.println("  output violation");
        pw3.println("  );");
        pw3.println("  ");
        pw3.println("  int unsigned FSM;");
        pw3.println("  reg v;");
        pw3.println("  ");
        pw3.println("  always @ (" + as + " or " + af + " or " + bs + " or " + bf + ")");
        pw3.println("  begin ");
        pw3.println("    case(FSM)");
        pw3.println("      // -------------------------------------------------------");
        pw3.println("        0: // State " + strA + "'" + strB + "'");
        pw3.println("          if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=0;      // empty"
        pw3.println("          else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0) FSM=3; // " + as
        pw3.println("          else // Violation");
        pw3.println("          begin");
        pw3.println("            FSM=4;v=1'b1;");
        pw3.println("          end");
        pw3.println("        // -------------------------------------------------------");
        pw3.println("        1: // State " + strA + "'" + strB);
        pw3.println("          if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=1;      // empty"
        pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1) FSM=0; // " + bf
        pw3.println("          else // Violation");
        pw3.println("          begin");
        pw3.println("            FSM=4;v=1'b1;");
        pw3.println("          end");
        pw3.println("        // -------------------------------------------------------");
        pw3.println("        2: // State " + strA + strB + "'");
        pw3.println("          if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=2;      // empty"
        pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=0; // " + af
```

```
    pw3.println("           else // Violation");
    pw3.println("           begin");
    pw3.println("             FSM=4;v=1'b1;");
    pw3.println("           end");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         3: // State " + strA + strB);
    pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=3;      // empty"
    pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=1; // " + af
    pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1) FSM=2; // " + bf
    pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b1) FSM=0; // " + af
    pw3.println("           else // Violation");
    pw3.println("           begin");
    pw3.println("             FSM=4;v=1'b1;");
    pw3.println("           end");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         default:  // Violation ");
    pw3.println("         begin");
    pw3.println("           FSM=4;");
    pw3.println("           v=1'b1;");
    pw3.println("         end     ");
    pw3.println("      endcase");
    pw3.println("   end");
    pw3.println("   ");
    pw3.println("  assign violation = v;");
    pw3.println("           ");
    pw3.println("endmodule");

}


// ************************************************************************************************
public void generateFinishes(String strA,String strB,String as,String af,String bs,String bf) {
    System.out.println("\ngenerateFinishes");

    // CCSL+
    // A finishes B
    pw1.print(strA);
    pw1.print(" finishes ");
    pw1.print(strB);


    // CCSL
    // bf isSubclockOf af
    pw2.print(bf);
    pw2.print(" isSubclockOf ");
    pw2.print(af);
    pw2.println("");

    // as alternatesWith af
    pw2.print(as);
    pw2.print(" alternatesWith ");
    pw2.print(af);
    pw2.println("");

    // bs alternatesWith bf
    pw2.print(bs);
    pw2.print(" alternatesWith ");
    pw2.print(bf);


    // SystemVerilog
    pw3.println("// @Generated");
    pw3.println("// " + strA + " finishes " + strB);
    pw3.println("");
    pw3.println("module Finishes (");
    pw3.println("  input " + as + ",");
    pw3.println("  input " + af + ",");
    pw3.println("  input " + bs + ",");
    pw3.println("  input " + bf + ",");
    pw3.println("  output violation");
    pw3.println("  );");
    pw3.println("  ");
    pw3.println("  int unsigned FSM;");
    pw3.println("  reg v;");
    pw3.println("  ");
    pw3.println("  always @ (" + as + " or " + af + " or " + bs + " or " + bf + ")");
    pw3.println("  begin ");
    pw3.println("    case(FSM)");
    pw3.println("      // -------------------------------------------------------");
    pw3.println("         0: // State " + strA + "'" + strB + "'");
    pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=0;      // empty"
    pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=2; // " + as
    pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0) FSM=1; // " + bs
    pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0) FSM=3; // " + as
    pw3.println("           else // Violation");
    pw3.println("           begin");
    pw3.println("             FSM=4;v=1'b1;");
    pw3.println("           end");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         1: // State " + strA + "'" + strB);
    pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=1;      // empty"
    pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=3; // " + as
    pw3.println("           else // Violation");
    pw3.println("           begin");
    pw3.println("             FSM=4;v=1'b1;");
    pw3.println("           end");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         2: // State " + strA + strB + "'");
```

```
    pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=2;      // empty"
    pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0) FSM=3; // " + bs
    pw3.println("           else // Violation");
    pw3.println("           begin");
    pw3.println("             FSM=4;v=1'b1;");
    pw3.println("           end");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         3: // State " + strA + strB);
    pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=3;      // empty"
    pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b1) FSM=0; // " + af
    pw3.println("           else // Violation");
    pw3.println("           begin");
    pw3.println("             FSM=4;v=1'b1;");
    pw3.println("           end");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         default:  // Violation ");
    pw3.println("         begin");
    pw3.println("           FSM=4;");
    pw3.println("           v=1'b1;");
    pw3.println("         end     ");
    pw3.println("      endcase");
    pw3.println("   end");
    pw3.println("   ");
    pw3.println("  assign violation = v;");
    pw3.println("           ");
    pw3.println("endmodule");
}


// ************************************************************************************************
public void generateCauses(String strA,String strB,String as,String af,String bs,String bf) {
    System.out.println("\ngenerateCauses");

    // CCSL+
    // A causes B
    pw1.print(strA);
    pw1.print(" causes ");
    pw1.print(strB);


    // CCSL
    // bs isSubclockOf af
    pw2.print(bs);
    pw2.print(" isSubclockOf ");
    pw2.print(af);
    pw2.println("");

    // as alternatesWith af
    pw2.print(as);
    pw2.print(" alternatesWith ");
    pw2.print(af);
    pw2.println("");

    // bs alternatesWith bf
    pw2.print(bs);
    pw2.print(" alternatesWith ");
    pw2.print(bf);


    // SystemVerilog
    pw3.println("// @Generated");
    pw3.println("// " + strA + " causes " + strB);
    pw3.println("");
    pw3.println("module Causes (");
    pw3.println("  input " + as + ",");
    pw3.println("  input " + af + ",");
    pw3.println("  input " + bs + ",");
    pw3.println("  input " + bf + ",");
    pw3.println("  output violation");
    pw3.println("  );");
    pw3.println("  ");
    pw3.println("  int unsigned FSM;");
    pw3.println("  reg v;");
    pw3.println("  ");
    pw3.println("  always @ (" + as + " or " + af + " or " + bs + " or " + bf + ")");
    pw3.println("  begin ");
    pw3.println("    case(FSM)");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         0: // State " + strA + "'" + strB + "'");
    pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=0;      // empty"
    pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=2; // " + as
    pw3.println("           else // Violation");
    pw3.println("           begin");
    pw3.println("             FSM=4;v=1'b1;");
    pw3.println("           end");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         1: // State " + strA + "'" + strB);
    pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=1;      // empty"
    pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=3; // " + as
    pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1) FSM=0; // " + bf
    pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1) FSM=2; // " + as
    pw3.println("           else // Violation");
    pw3.println("           begin");
    pw3.println("             FSM=4;v=1'b1;");
    pw3.println("           end");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         2: // State " + strA + strB + "'");
    pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=2;      // empty"
```

```
    pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b1 && " + bf + "==1'b0) FSM=1; // " + af
    pw3.println("           else // Violation");
    pw3.println("           begin");
    pw3.println("             FSM=4;v=1'b1;");
    pw3.println("           end");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         3: // State " + strA + strB);
    pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=3;       // empty"
    pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1) FSM=2; // " + bf
    pw3.println("           else // Violation");
    pw3.println("           begin");
    pw3.println("             FSM=4;v=1'b1;");
    pw3.println("           end");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         default:  // Violation ");
    pw3.println("         begin");
    pw3.println("           FSM=4;");
    pw3.println("           v=1'b1;");
    pw3.println("         end     ");
    pw3.println("      endcase");
    pw3.println("   end");
    pw3.println("   ");
    pw3.println("  assign violation = v;");
    pw3.println("          ");
    pw3.println("endmodule");
}


// ********************************************************************************************
public void generateContains(String strA,String strB,String as,String af,String bs,String bf) {
    System.out.println("\ngenerateContains");

    // CCSL+
    // A contains B
    pw1.print(strA);
    pw1.print(" contains ");
    pw1.print(strB);


    // CCSL
    // as precedes bsTemp
    pw2.print(as);
    pw2.print(" precedes ");
    pw2.print("bsTemp");
    pw2.println("");

    // bfTemp precedes af
    pw2.print("bfTemp");
    pw2.print(" precedes ");
    pw2.print(af);
    pw2.println("");

    // bs isSubclockOf bsTemp
    pw2.print(bs);
    pw2.print(" isSubclockOf ");
    pw2.print("bsTemp");
    pw2.println("");

    // bf isSubclockOf bfTemp
    pw2.print(bf);
    pw2.print(" isSubclockOf ");
    pw2.print("bfTemp");
    pw2.println("");

    // as alternatesWith af
    pw2.print(as);
    pw2.print(" alternatesWith ");
    pw2.print(af);
    pw2.println("");

    // bs alternatesWith bf
    pw2.print(bs);
    pw2.print(" alternatesWith ");
    pw2.print(bf);

    // bsTemp alternatesWith bfTemp
    pw2.print("bsTemp alternatesWith bfTemp");



    // SystemVerilog
    pw3.println("// @Generated");
    pw3.println("// " + strA + " contains " + strB);
    pw3.println("");
    pw3.println("module Contains (");
    pw3.println("  input " + as + ",");
    pw3.println("  input " + af + ",");
    pw3.println("  input " + bs + ",");
    pw3.println("  input " + bf + ",");
    pw3.println("  output violation");
    pw3.println("  );");
    pw3.println("  ");
    pw3.println("  int unsigned FSM;");
    pw3.println("  reg v;");
    pw3.println("  ");
    pw3.println("  always @ (" + as + " or " + af + " or " + bs + " or " + bf + ")");
    pw3.println("  begin ");
    pw3.println("    case(FSM)");
```

```
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         0: // State " + strA + "'" + strB + "'");
    pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=0;      // empty"
    pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=2; // " + as
    pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0) FSM=3; // " + as
    pw3.println("           else // Violation");
    pw3.println("           begin");
    pw3.println("             FSM=4;v=1'b1;");
    pw3.println("           end");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         2: // State " + strA + strB + "'");
    pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=2;      // empty"
    pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=0; // " + af
    pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0) FSM=3; // " + bs
    pw3.println("           else // Violation");
    pw3.println("           begin");
    pw3.println("             FSM=4;v=1'b1;");
    pw3.println("           end");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         3: // State " + strA + strB);
    pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=3;      // empty"
    pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1) FSM=2; // " + bf
    pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b1) FSM=0; // " + af
    pw3.println("           else // Violation");
    pw3.println("           begin");
    pw3.println("             FSM=4;v=1'b1;");
    pw3.println("           end");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         default:  // Violation ");
    pw3.println("         begin");
    pw3.println("           FSM=4;");
    pw3.println("           v=1'b1;");
    pw3.println("         end     ");
    pw3.println("       endcase");
    pw3.println("    end");
    pw3.println("    ");
    pw3.println("  assign violation = v;");
    pw3.println("           ");
    pw3.println("endmodule");
}


// ********************************************************************************************
public void generateImplies(String strA,String strB,String as,String af,String bs,String bf) {
    System.out.println("\ngenerateImplies");

    // CCSL+
    // A implies B
    pw1.print(strA);
    pw1.print(" implies ");
    pw1.print(strB);


    // CCSL
    // bs isSubclockOf as
    pw2.print(bs);
    pw2.print(" isSubclockOf ");
    pw2.print("as");
    pw2.println("");

    // bf isSubclockOf af
    pw2.print(bf);
    pw2.print(" isSubclockOf ");
    pw2.print("af");
    pw2.println("");

    // as alternatesWith af
    pw2.print(as);
    pw2.print(" alternatesWith ");
    pw2.print(af);
    pw2.println("");

    // bs alternatesWith bf
    pw2.print(bs);
    pw2.print(" alternatesWith ");
    pw2.print(bf);



    // SystemVerilog
    pw3.println("// @Generated");
    pw3.println("// " + strA + " implies " + strB);
    pw3.println("");
    pw3.println("module Implies (");
    pw3.println("  input " + as + ",");
    pw3.println("  input " + af + ",");
    pw3.println("  input " + bs + ",");
    pw3.println("  input " + bf + ",");
    pw3.println("  output violation");
    pw3.println("  );");
    pw3.println("  ");
    pw3.println("  int unsigned FSM;");
    pw3.println("  reg v;");
    pw3.println("  ");
    pw3.println("  always @ (" + as + " or " + af + " or " + bs + " or " + bf + ")");
    pw3.println("  begin ");
    pw3.println("    case(FSM)");
    pw3.println("         // -------------------------------------------------------");
```

```
    pw3.println("         0: // State " + strA + "'" + strB + "'");
    pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=0;      // empty"
    pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0) FSM=3; // " + as
    pw3.println("           else // Violation");
    pw3.println("           begin");
    pw3.println("             FSM=4;v=1'b1;");
    pw3.println("           end");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         3: // State " + strA + strB);
    pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=3;      // empty"
    pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b1) FSM=0; // " + af
    pw3.println("           else // Violation");
    pw3.println("           begin");
    pw3.println("             FSM=4;v=1'b1;");
    pw3.println("           end");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         default:  // Violation ");
    pw3.println("         begin");
    pw3.println("           FSM=4;");
    pw3.println("           v=1'b1;");
    pw3.println("         end     ");
    pw3.println("       endcase");
    pw3.println("    end");
    pw3.println("    ");
    pw3.println("  assign violation = v;");
    pw3.println("           ");
    pw3.println("endmodule");
}


// ********************************************************************************************
public void generateForbids(String strA,String strB,String as,String af,String bs,String bf) {
    System.out.println("\ngenerateForbids");

    // CCSL+
    // A forbids B
    pw1.print(strA);
    pw1.print(" forbids ");
    pw1.print(strB);


    // CCSL
    // af sampledOn bs precedes bf
    pw2.print(af);
    pw2.print(" sampledOn ");
    pw2.print(bs);
    pw2.print(" precedes ");
    pw2.print(bf);
    pw2.println("");

    // as alternatesWith af
    pw2.print(as);
    pw2.print(" alternatesWith ");
    pw2.print(af);
    pw2.println("");

    // bs alternatesWith bf
    pw2.print(bs);
    pw2.print(" alternatesWith ");
    pw2.print(bf);


    // SystemVerilog
    pw3.println("// @Generated");
    pw3.println("// " + strA + " forbids " + strB);
    pw3.println("");
    pw3.println("module Forbids (");
    pw3.println("  input " + as + ",");
    pw3.println("  input " + af + ",");
    pw3.println("  input " + bs + ",");
    pw3.println("  input " + bf + ",");
    pw3.println("  output violation");
    pw3.println("  );");
    pw3.println("  ");
    pw3.println("  int unsigned FSM;");
    pw3.println("  reg v;");
    pw3.println("  ");
    pw3.println("  always @ (" + as + " or " + af + " or " + bs + " or " + bf + ")");
    pw3.println("  begin ");
    pw3.println("    case(FSM)");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         0: // State " + strA + "'" + strB + "'");
    pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=0;      // empty"
    pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=2; // " + as
    pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0) FSM=1; // " + bs
    pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0) FSM=3; // " + as
    pw3.println("           else // Violation");
    pw3.println("           begin");
    pw3.println("             FSM=4;v=1'b1;");
    pw3.println("           end");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         1: // State " + strA + "'" + strB);
    pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=1;      // empty"
    pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=3; // " + as
    pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1) FSM=0; // " + bf
    pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1) FSM=2; // " + as
    pw3.println("           else // Violation");
    pw3.println("           begin");
```

```
        pw3.println("             FSM=4;v=1'b1;");
        pw3.println("           end");
        pw3.println("         // -------------------------------------------------------");
        pw3.println("         2: // State " + strA + strB + "'");
        pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=2;      // empty"
        pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=0; // " + af
        pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0) FSM=3; // " + bs
        pw3.println("           else // Violation");
        pw3.println("           begin");
        pw3.println("             FSM=4;v=1'b1;");
        pw3.println("           end");
        pw3.println("         // -------------------------------------------------------");
        pw3.println("         3: // State " + strA + strB);
        pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=3;      // empty"
        pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1) FSM=2; // " + bf
        pw3.println("           else // Violation");
        pw3.println("           begin");
        pw3.println("             FSM=4;v=1'b1;");
        pw3.println("           end");
        pw3.println("         // -------------------------------------------------------");
        pw3.println("         default:  // Violation ");
        pw3.println("         begin");
        pw3.println("           FSM=4;");
        pw3.println("           v=1'b1;");
        pw3.println("         end     ");
        pw3.println("      endcase");
        pw3.println("   end");
        pw3.println("   ");
        pw3.println("  assign violation = v;");
        pw3.println("           ");
        pw3.println("endmodule");

}


// ********************************************************************************************
public void generateExcludes(String strA,String strB,String as,String af,String bs,String bf) {
    System.out.println("\ngenerateExcludes");

    // CCSL+
    // A excludes B
    pw1.print(strA);
    pw1.print(" excludes ");
    pw1.print(strB);


    // CCSL
    // bs sampledOn as precedes af
    pw2.print(bs);
    pw2.print(" sampledOn ");
    pw2.print(as);
    pw2.print(" precedes ");
    pw2.print(af);
    pw2.println("");

    // as sampledOn bs precedes bf
    pw2.print(as);
    pw2.print(" sampledOn ");
    pw2.print(bs);
    pw2.print(" precedes ");
    pw2.print(bf);
    pw2.println("");

    // as alternatesWith af
    pw2.print(as);
    pw2.print(" alternatesWith ");
    pw2.print(af);

    // bs alternatesWith bf
    pw2.print(bs);
    pw2.print(" alternatesWith ");
    pw2.print(bf);


    // SystemVerilog
    pw3.println("// @Generated");
    pw3.println("// " + strA + " excludes " + strB);
    pw3.println("");
    pw3.println("module Excludes (");
    pw3.println("  input " + as + ",");
    pw3.println("  input " + af + ",");
    pw3.println("  input " + bs + ",");
    pw3.println("  input " + bf + ",");
    pw3.println("  output violation");
    pw3.println("  );");
    pw3.println("  ");
    pw3.println("  int unsigned FSM;");
    pw3.println("  reg v;");
    pw3.println("  ");
    pw3.println("  always @ (" + as + " or " + af + " or " + bs + " or " + bf + ")");
    pw3.println("  begin ");
    pw3.println("    case(FSM)");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         0: // State " + strA + "'" + strB + "'");
    pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=0;      // empty"
    pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=1; // " + as
    pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b1 && " + bf + "==1'b0) FSM=2; // " + bs
    pw3.println("           else // Violation");
```

```
        pw3.println("           begin");
        pw3.println("             FSM=4;v=1'b1;");
        pw3.println("           end");
        pw3.println("         // ------------------------------------------------------");
        pw3.println("         1: // State " + strA + "'" + strB);
        pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=1;      // empty"
        pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b1 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=0; // " + af
        pw3.println("           else // Violation");
        pw3.println("           begin");
        pw3.println("             FSM=4;v=1'b1;");
        pw3.println("           end");
        pw3.println("         // ------------------------------------------------------");
        pw3.println("         2: // State " + strA + strB + "'");
        pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b0) FSM=2;      // empty"
        pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + bs + "==1'b0 && " + bf + "==1'b1) FSM=0; // " + bf
        pw3.println("           else // Violation");
        pw3.println("           begin");
        pw3.println("             FSM=4;v=1'b1;");
        pw3.println("           end");
        pw3.println("         // ------------------------------------------------------");
        pw3.println("         default:  // Violation ");
        pw3.println("         begin");
        pw3.println("           FSM=4;");
        pw3.println("           v=1'b1;");
        pw3.println("         end     ");
        pw3.println("       endcase");
        pw3.println("    end");
        pw3.println("    ");
        pw3.println("   assign violation = v;");
        pw3.println("           ");
        pw3.println("endmodule");

    }

}
```

EventObserver.java

```
// ********************************************************************************************
// ********************************************************************************************
// ********************************************************************************************
//
// This class provides the CCSL and Verilog Observers generation facility for State-Event Relations
//
// @package     OBSERVER
// @class       EventObserver
// @author      Aamir M. Khan
// @version     3.1
// @first       2016-02-02
// @current     2016-08-13
//
// ********************************************************************************************
// ********************************************************************************************
// ********************************************************************************************

package observer;


import java.io.FileOutputStream;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;

//*********************************************************************************************
public class EventObserver {

    // Important Variables
    public PrintWriter pw1,pw2,pw3;


    // *********************************************************************************************
    // Constructor
    public EventObserver(){
        // create three files: CCSLPlus.txt, CCSL.txt, SystemVerilog.sv
        try {
            System.out.println("\nFile Output Initialization ");
            pw1 = new PrintWriter( new OutputStreamWriter(
                    new FileOutputStream("CCSLPlus.txt"),"UTF-8"));
            pw2 = new PrintWriter( new OutputStreamWriter(
                    new FileOutputStream("CCSL.txt"),"UTF-8"));
            pw3 = new PrintWriter( new OutputStreamWriter(
                    new FileOutputStream("SystemVerilog.sv"),"UTF-8"));

        } catch (Exception e) {
            pw1.print("ERROR OCCURED");
            pw2.print("ERROR OCCURED");
            pw3.print("ERROR OCCURED");
            }
    }


    // *********************************************************************************************
    public void generateTimedTriggers(String strA,String as,String af,String e,String clk,int min,int max) {
        System.out.println("\ngenerateTimedTriggers");

        // CCSL+
        // e triggers A after [m,n] on clk
        pw1.print(e);
        pw1.print(" triggers ");
        pw1.print(strA);
        pw1.print(" after [" + min + "," + max + "] on " + clk);


        // CCSL
        // e delayedFor min on clk precedes as
        pw2.print(e + " delayedFor " + min + " on " + clk + " precedes " + as);
        pw2.println("");

        // as precedes e delayedFor max on clk
        pw2.print(as + " precedes " + e + " delayedFor " + max + " on " + clk);
        pw2.println("");

        // as alternatesWith af
        pw2.print(as);
        pw2.print(" alternatesWith ");
        pw2.print(af);
        pw2.println("");


        // SystemVerilog
        // ---------------------------------------
        pw3.println("// @Generated");
        pw3.println("// " + e + " triggers " + strA + " after [" + min + "," + max + "] on " + clk);
        pw3.println("");
        pw3.println("module Triggers (");
        pw3.println("  input " + as + ",");
        pw3.println("  input " + af + ",");
        pw3.println("  input " + e + ",");
        pw3.println("  input " + clk + ",");
        pw3.println("  output violation");
        pw3.println("  );");
        pw3.println("  parameter min=" + min + ",max=" + max + ";");
        pw3.println("  ");
        pw3.println("  reg valid;");
        pw3.println("  int unsigned d[max-1];");
        pw3.println("  int unsigned ds;");
```

```
pw3.println("  int unsigned df;");
pw3.println("  int unsigned i;");
pw3.println("  ");
pw3.println("  int unsigned FSM;");
pw3.println("  reg v;");
pw3.println("  ");
pw3.println("  always @ (" + as + " or " + af + " or " + e + " or " + clk + ")");
pw3.println("   begin ");
pw3.println("    case(FSM)");
pw3.println("      // ------------------------------------------------------");
pw3.println("         0: // State A'");
pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b0 && " + clk + "==1'b0) FSM=0; // empty");
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b0 && " + clk + "==1'b1)   // " + clk);
pw3.println("           begin ");
pw3.println("             if(i==0) begin FSM=0;valid=1'b1; end");
pw3.println("             else if(i>0 && d[ds]<(max-1))");
pw3.println("             begin");
pw3.println("               FSM=0;valid=1'b1;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=2;v=1'b1;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b1 && " + clk + "==1'b0)   // " + e);
pw3.println("           begin");
pw3.println("             FSM=0;");
pw3.println("             if(valid==1'b1)");
pw3.println("             begin");
pw3.println("               valid=1'b0; i++;");
pw3.println("               d[(df+1) % max]=0;");
pw3.println("               df=(df+1) % max;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b1 && " + clk + "==1'b1)   // " + e + ","
pw3.println("           begin");
pw3.println("             FSM=0;");
pw3.println("             if(valid==1'b0 && i==0 || d[ds]<max)");
pw3.println("             begin");
pw3.println("               valid=1'b1;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else if(valid==1'b1 && i==0 || d[ds]<max)");
pw3.println("             begin");
pw3.println("               valid=1'b0; i++;");
pw3.println("               d[(df+1) % max]=0;");
pw3.println("               df=(df+1) % max;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=2;v=1'b1;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + clk + "==1'b0)              // " + as);
pw3.println("           begin");
pw3.println("             if(i>0 && d[ds]>=min && d[ds]<=max)");
pw3.println("             begin");
pw3.println("               i--;FSM=1;");
pw3.println("               ds=(ds+1) % max;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=2;v=1'b1;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + clk + "==1'b1)              // " + as + "," + clk)
pw3.println("           begin");
pw3.println("             if(i>0 && d[ds]>=min-1 && d[ds]<max)");
pw3.println("             begin");
pw3.println("               i--;valid=1'b1;FSM=1;");
pw3.println("               ds=(ds+1) % max;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=2;v=1'b1;");
pw3.println("             end");
pw3.println("           end     ");
pw3.println("           else ");
pw3.println("           begin");
pw3.println("             FSM=2;v=1'b1;");
pw3.println("           end");
pw3.println("         // ------------------------------------------------------");
pw3.println("         1: // State A");
pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b0 && " + clk + "==1'b0) FSM=1;  // empty");
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b0 && " + clk + "==1'b1)    // " + clk);
pw3.println("           begin ");
pw3.println("             if(i==0) begin FSM=1;valid=1'b1; end");
pw3.println("             else if(i>0 && d[ds]<(max-1))");
pw3.println("             begin");
pw3.println("               FSM=1;valid=1'b1;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=2;v=1'b1;");
```

```
    pw3.println("            end");
    pw3.println("          end");
    pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b1 && " + clk + "==1'b0)    // " + e);
    pw3.println("          begin");
    pw3.println("            FSM=1;");
    pw3.println("            if(valid==1'b1)");
    pw3.println("            begin");
    pw3.println("              valid=1'b0; i++;");
    pw3.println("              d[(df+1) % max]=0;");
    pw3.println("              df=(df+1) % max;");
    pw3.println("            end");
    pw3.println("          end");
    pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b1 && " + clk + "==1'b1)    // " + e + ",
    pw3.println("          begin");
    pw3.println("            FSM=1;");
    pw3.println("            if(valid==1'b0 && i==0 || d[ds]<max)");
    pw3.println("            begin");
    pw3.println("              valid=1'b1;");
    pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
    pw3.println("            end");
    pw3.println("            else if(valid==1'b1 && i==0 || d[ds]<max)");
    pw3.println("            begin");
    pw3.println("              valid=1'b0; i++;");
    pw3.println("              d[(df+1) % max]=0;");
    pw3.println("              df=(df+1) % max;");
    pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
    pw3.println("            end");
    pw3.println("            else");
    pw3.println("            begin");
    pw3.println("              FSM=2;v=1'b1;");
    pw3.println("            end");
    pw3.println("          end");
    pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b1 && " + clk + "==1'b0)               // " + af);
    pw3.println("          begin");
    pw3.println("            FSM=0;");
    pw3.println("            if(valid==1'b1)");
    pw3.println("            begin");
    pw3.println("              valid=1'b0; i++;");
    pw3.println("              d[(df+1) % max]=0;");
    pw3.println("              df=(df+1) % max;");
    pw3.println("            end");
    pw3.println("          end");
    pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b1 && " + clk + "==1'b1)               // " + af + "," + clk
    pw3.println("          begin");
    pw3.println("            FSM=0;");
    pw3.println("            if(valid==1'b0 && i==0 || d[ds]<max)");
    pw3.println("            begin");
    pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
    pw3.println("            end");
    pw3.println("            else if(valid==1'b1 && i==0 || d[ds]<max)");
    pw3.println("            begin");
    pw3.println("              valid=1'b0; i++;");
    pw3.println("              d[(df+1) % max]=0;");
    pw3.println("              df=(df+1) % max;");
    pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
    pw3.println("            end");
    pw3.println("            else");
    pw3.println("            begin");
    pw3.println("              FSM=2;v=1'b1;");
    pw3.println("            end");
    pw3.println("          end     ");
    pw3.println("          else ");
    pw3.println("          begin");
    pw3.println("            FSM=2;v=1'b1;");
    pw3.println("          end");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         default: // Violation");
    pw3.println("      begin");
    pw3.println("        FSM=2;v=1'b1;");
    pw3.println("      end");
    pw3.println("    endcase");
    pw3.println("   end");
    pw3.println("   ");
    pw3.println("  assign violation = v;");
    pw3.println("        ");
    pw3.println("  initial ");
    pw3.println("  begin");
    pw3.println("    valid=1'b1;");
    pw3.println("    ds=0;");
    pw3.println("    df=max-1;");
    pw3.println("    d[max-1]=0;");
    pw3.println("    i=0;");
    pw3.println("    FSM = 0;");
    pw3.println("    v = 0;");
    pw3.println("  end");
    pw3.println("endmodule");
}

// ********************************************************************************************
public void generateTimedTerminates(String strA,String as,String af,String e,String clk,int min,int max) {
    System.out.println("\ngenerateTimedTerminates");

    // CCSL+
    // e terminates A after [m,n] on clk
    pw1.print(e);
    pw1.print(" terminates ");
    pw1.print(strA);
    pw1.print(" after [" + min + "," + max + "] on " + clk);
```

```
// CCSL
// e delayedFor min on clk precedes af
pw2.print(e + " delayedFor " + min + " on " + clk + " precedes " + af);
pw2.println("");

// af precedes e delayedFor max on clk
pw2.print(af + " precedes " + e + " delayedFor " + max + " on " + clk);
pw2.println("");

// as alternatesWith af
pw2.print(as);
pw2.print(" alternatesWith ");
pw2.print(af);
pw2.println("");


// SystemVerilog
// ---------------------------------------
pw3.println("// @Generated");
pw3.println("// " + e + " terminates " + strA + " after [" + min + "," + max + "] on " + clk);
pw3.println("");
pw3.println("module Terminates (");
pw3.println("  input " + as + ",");
pw3.println("  input " + af + ",");
pw3.println("  input " + e + ",");
pw3.println("  input " + clk + ",");
pw3.println("  output violation");
pw3.println("  );");
pw3.println("  parameter min=" + min + ",max=" + max + ";");
pw3.println("  ");
pw3.println("  reg valid;");
pw3.println("  int unsigned d[max-1];");
pw3.println("  int unsigned ds;");
pw3.println("  int unsigned df;");
pw3.println("  int unsigned i;");
pw3.println("  ");
pw3.println("  int unsigned FSM;");
pw3.println("  reg v;");
pw3.println("  ");
pw3.println("  always @ (" + as + " or " + af + " or " + e + " or " + clk + ")");
pw3.println("   begin ");
pw3.println("    case(FSM)");
pw3.println("      // -------------------------------------------------------");
pw3.println("         0: // State A'");
pw3.println("           if(" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b0 && " + clk + "==1'b0) FSM=0; // empty");
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b0 && " + clk + "==1'b1)   // " + clk);
pw3.println("           begin ");
pw3.println("             if(i==0) begin FSM=0;valid=1'b1; end");
pw3.println("             else if(i>0 && d[ds]<(max-1))");
pw3.println("             begin");
pw3.println("               FSM=0;valid=1'b1;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=2;v=1'b1;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b1 && " + clk + "==1'b0)   // " + e);
pw3.println("           begin");
pw3.println("             FSM=0;");
pw3.println("             if(valid==1'b1)");
pw3.println("             begin");
pw3.println("               valid=1'b0; i++;");
pw3.println("               d[(df+1) % max]=0;");
pw3.println("               df=(df+1) % max;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b1 && " + clk + "==1'b1)   // " + e + ","
pw3.println("           begin");
pw3.println("             FSM=0;");
pw3.println("             if(valid==1'b0 && i==0 || d[ds]<max)");
pw3.println("             begin");
pw3.println("               valid=1'b1;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else if(valid==1'b1 && i==0 || d[ds]<max)");
pw3.println("             begin");
pw3.println("               valid=1'b0; i++;");
pw3.println("               d[(df+1) % max]=0;");
pw3.println("               df=(df+1) % max;");
pw3.println("               for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("             end");
pw3.println("             else");
pw3.println("             begin");
pw3.println("               FSM=2;v=1'b1;");
pw3.println("             end");
pw3.println("           end");
pw3.println("           else if(" + as + "==1'b1 && " + af + "==1'b0 && " + clk + "==1'b0)              // " + as);
pw3.println("           begin");
pw3.println("             if(i>0 && d[ds]>=min && d[ds]<=max)");
pw3.println("             begin");
pw3.println("               i--;FSM=1;");
pw3.println("               ds=(ds+1) % max;");
pw3.println("             end");
pw3.println("             else");
```

```
pw3.println("            begin");
pw3.println("              FSM=2;v=1'b1;");
pw3.println("            end");
pw3.println("          end");
pw3.println("          else if(" + as + "==1'b1 && " + af + "==1'b0 && " + clk + "==1'b1)              // " + as + "," + clk)
pw3.println("          begin");
pw3.println("            if(i>0 && d[ds]>=min-1 && d[ds]<max)");
pw3.println("            begin");
pw3.println("              i--;valid=1'b1;FSM=1;");
pw3.println("              ds=(ds+1) % max;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=2;v=1'b1;");
pw3.println("            end");
pw3.println("          end     ");
pw3.println("          else ");
pw3.println("          begin");
pw3.println("            FSM=2;v=1'b1;");
pw3.println("          end");
pw3.println("        // -------------------------------------------------------");
pw3.println("        1: // State A");
pw3.println("          if(" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b0 && " + clk + "==1'b0) FSM=1;  // empty");
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b0 && " + clk + "==1'b1)    // " + clk);
pw3.println("          begin ");
pw3.println("            if(i==0) begin FSM=1;valid=1'b1; end");
pw3.println("            else if(i>0 && d[ds]<(max-1))");
pw3.println("            begin");
pw3.println("              FSM=1;valid=1'b1;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=2;v=1'b1;");
pw3.println("            end");
pw3.println("          end");
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b1 && " + clk + "==1'b0)    // " + e);
pw3.println("          begin");
pw3.println("            FSM=1;");
pw3.println("            if(valid==1'b1)");
pw3.println("            begin");
pw3.println("              valid=1'b0; i++;");
pw3.println("              d[(df+1) % max]=0;");
pw3.println("              df=(df+1) % max;");
pw3.println("            end");
pw3.println("          end");
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b1 && " + clk + "==1'b1)    // " + e + ",
pw3.println("          begin");
pw3.println("            FSM=1;");
pw3.println("            if(valid==1'b0 && i==0 || d[ds]<max)");
pw3.println("            begin");
pw3.println("              valid=1'b1;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else if(valid==1'b1 && i==0 || d[ds]<max)");
pw3.println("            begin");
pw3.println("              valid=1'b0; i++;");
pw3.println("              d[(df+1) % max]=0;");
pw3.println("              df=(df+1) % max;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=2;v=1'b1;");
pw3.println("            end");
pw3.println("          end");
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b1 && " + clk + "==1'b0)              // " + af);
pw3.println("          begin");
pw3.println("            FSM=0;");
pw3.println("            if(valid==1'b1)");
pw3.println("            begin");
pw3.println("              valid=1'b0; i++;");
pw3.println("              d[(df+1) % max]=0;");
pw3.println("              df=(df+1) % max;");
pw3.println("            end");
pw3.println("          end");
pw3.println("          else if(" + as + "==1'b0 && " + af + "==1'b1 && " + clk + "==1'b1)              // " + af + "," + clk
pw3.println("          begin");
pw3.println("            FSM=0;");
pw3.println("            if(valid==1'b0 && i==0 || d[ds]<max)");
pw3.println("            begin");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else if(valid==1'b1 && i==0 || d[ds]<max)");
pw3.println("            begin");
pw3.println("              valid=1'b0; i++;");
pw3.println("              d[(df+1) % max]=0;");
pw3.println("              df=(df+1) % max;");
pw3.println("              for(int unsigned x=ds;x<=ds+i-1;x++) d[x % max]++;");
pw3.println("            end");
pw3.println("            else");
pw3.println("            begin");
pw3.println("              FSM=2;v=1'b1;");
pw3.println("            end");
pw3.println("          end     ");
pw3.println("          else ");
pw3.println("          begin");
```

```
    pw3.println("             FSM=2;v=1'b1;");
    pw3.println("           end");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         default: // Violation");
    pw3.println("       begin");
    pw3.println("         FSM=2;v=1'b1;");
    pw3.println("       end");
    pw3.println("     endcase");
    pw3.println("    end");
    pw3.println("    ");
    pw3.println("  assign violation = v;");
    pw3.println("        ");
    pw3.println("  initial ");
    pw3.println("  begin");
    pw3.println("    valid=1'b1;");
    pw3.println("    ds=0;");
    pw3.println("    df=max-1;");
    pw3.println("    d[max-1]=0;");
    pw3.println("    i=0;");
    pw3.println("    FSM = 0;");
    pw3.println("    v = 0;");
    pw3.println("  end");
    pw3.println("endmodule");
}


// ********************************************************************************************
// ********************************************************************************************
// ********************************************************************************************
// ********************************************************************************************


// ********************************************************************************************
public void generateTriggers(String strA,String as,String af,String e) {
    System.out.println("\ngenerateTriggers");

    // CCSL+
    // e triggers A
    pw1.print(e);
    pw1.print(" triggers ");
    pw1.print(strA);


    // CCSL
    // e isSubclockOf as
    pw2.print(e + " isSubclockOf " + as);
    pw2.println("");

    // as alternatesWith af
    pw2.print(as);
    pw2.print(" alternatesWith ");
    pw2.print(af);
    pw2.println("");


    // SystemVerilog
    // ----------------------------------------
    pw3.println("// @Generated");
    pw3.println("// " + e + " triggers " + strA);
    pw3.println("");
    pw3.println("module Triggers (");
    pw3.println("  input " + as + ",");
    pw3.println("  input " + af + ",");
    pw3.println("  input " + e + ",");
    pw3.println("  output violation");
    pw3.println("  );");
    pw3.println("  ");
    pw3.println("  int unsigned FSM;");
    pw3.println("  reg v;");
    pw3.println("  ");
    pw3.println("  always @ (" + as + " or " + af + " or " + e + ")");
    pw3.println("   begin ");
    pw3.println("    case (FSM)");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         0 : // State " + strA + "'");
    pw3.println("         begin");
    pw3.println("           if (" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b0) FSM=0;       // empty");
    pw3.println("           else if (" + as + "==1'b1 && " + af + "==1'b0 && " + e + "==1'b0) FSM=1;  // " + as);
    pw3.println("           else if (" + as + "==1'b1 && " + af + "==1'b0 && " + e + "==1'b1) FSM=1;  // " + as + "," + e);
    pw3.println("           else ");
    pw3.println("           begin");
    pw3.println("             FSM=2; ");
    pw3.println("             v=1'b1;");
    pw3.println("           end");
    pw3.println("         end");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         1 : // State " + strA);
    pw3.println("         begin");
    pw3.println("           if (" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b0) FSM=1;       // empty");
    pw3.println("           else if (" + as + "==1'b0 && " + af + "==1'b1 && " + e + "==1'b0) FSM=0;  // " + af);
    pw3.println("           else ");
    pw3.println("           begin");
    pw3.println("             FSM=2; ");
    pw3.println("             v=1'b1;");
    pw3.println("           end");
    pw3.println("         end");
    pw3.println("         // -------------------------------------------------------");
    pw3.println("         default : // Violation");
```

```
        pw3.println("         begin");
        pw3.println("           FSM=2;");
        pw3.println("           v=1'b1;");
        pw3.println("         end");
        pw3.println("    endcase");
        pw3.println("   end");
        pw3.println("   ");
        pw3.println("  assign violation = v;");
        pw3.println("           ");
        pw3.println("  initial ");
        pw3.println("  begin");
        pw3.println("    FSM = 0;");
        pw3.println("    v = 0;");
        pw3.println("  end");
        pw3.println("endmodule");
    }

    // ************************************************************************************************
    public void generateTerminates(String strA,String as,String af,String e) {
        System.out.println("\ngenerateTerminates");

        // CCSL+
        // e terminates A
        pw1.print(e);
        pw1.print(" terminates ");
        pw1.print(strA);


        // CCSL
        // e isSubclockOf af
        pw2.print(e + " isSubclockOf " + af);
        pw2.println("");

        // as alternatesWith af
        pw2.print(as);
        pw2.print(" alternatesWith ");
        pw2.print(af);
        pw2.println("");


        // SystemVerilog
        // --------------------------------------

        pw3.println("// @Generated");
        pw3.println("// " + e + " terminates " + strA);
        pw3.println("");
        pw3.println("module Terminates (");
        pw3.println("  input " + as + ",");
        pw3.println("  input " + af + ",");
        pw3.println("  input " + e + ",");
        pw3.println("  output violation");
        pw3.println("  );");
        pw3.println("");
        pw3.println("  int unsigned FSM;");
        pw3.println("  reg v;");
        pw3.println("");
        pw3.println("  always @ (" + as + " or " + af + " or " + e + ")");
        pw3.println("   begin");
        pw3.println("    case (FSM)");
        pw3.println("         // ------------------------------------------------------");
        pw3.println("         0 : // State A'");
        pw3.println("         begin");
        pw3.println("           if (" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b0) FSM=0;       // empty");
        pw3.println("           else if (" + as + "==1'b1 && " + af + "==1'b0 && " + e + "==1'b0) FSM=1;  // " + as);
        pw3.println("           else ");
        pw3.println("           begin");
        pw3.println("             FSM=2; ");
        pw3.println("             v=1'b1;");
        pw3.println("           end");
        pw3.println("         end");
        pw3.println("         // ------------------------------------------------------");
        pw3.println("         1 : // State A");
        pw3.println("         begin");
        pw3.println("           if (" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b0) FSM=1;       // empty");
        pw3.println("           else if (" + as + "==1'b0 && " + af + "==1'b1 && " + e + "==1'b0) FSM=0;  // " + af);
        pw3.println("           else if (" + as + "==1'b0 && " + af + "==1'b1 && " + e + "==1'b1) FSM=0;  // " + af + "," + e);
        pw3.println("           else ");
        pw3.println("           begin");
        pw3.println("             FSM=2; ");
        pw3.println("             v=1'b1;");
        pw3.println("           end");
        pw3.println("         end");
        pw3.println("         // ------------------------------------------------------");
        pw3.println("         default : // Violation");
        pw3.println("         begin");
        pw3.println("           FSM=2;");
        pw3.println("           v=1'b1;");
        pw3.println("         end");
        pw3.println("    endcase");
        pw3.println("   end");
        pw3.println("   ");
        pw3.println("  assign violation = v;");
        pw3.println("           ");
        pw3.println("  initial ");
        pw3.println("  begin");
        pw3.println("    FSM = 0;");
        pw3.println("    v = 0;");
        pw3.println("  end");
```

```
    pw3.println("endmodule");

}


// ********************************************************************************************
public void generateForbids(String strA,String as,String af,String e) {
    System.out.println("\ngenerateForbids");

    // CCSL+
    // e forbids A
    pw1.print(e);
    pw1.print(" forbids ");
    pw1.print(strA);


    // CCSL
    // e sampledOn as precedes af
    pw2.print(e + " sampledOn " + as + " precedes " + af);
    pw2.println("");

    // as alternatesWith af
    pw2.print(as);
    pw2.print(" alternatesWith ");
    pw2.print(af);
    pw2.println("");


    // SystemVerilog
    // ---------------------------------------

    pw3.println("// @Generated");
    pw3.println("// " + e + " forbids " + strA);
    pw3.println("");
    pw3.println("module Forbids (");
    pw3.println("  input " + as + ",");
    pw3.println("  input " + af + ",");
    pw3.println("  input " + e + ",");
    pw3.println("  output violation");
    pw3.println("  );");
    pw3.println("");
    pw3.println("  int unsigned FSM;");
    pw3.println("  reg v;");
    pw3.println("");
    pw3.println("  always @ (" + as + " or " + af + " or " + e + ")");
    pw3.println("   begin");
    pw3.println("    case (FSM)");
    pw3.println("         // ------------------------------------------------------");
    pw3.println("         0 : // State A'");
    pw3.println("         begin");
    pw3.println("           if (" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b0) FSM=0;       // empty");
    pw3.println("           else if (" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b1) FSM=0;  // e");
    pw3.println("           else if (" + as + "==1'b1 && " + af + "==1'b0 && " + e + "==1'b0) FSM=1;  // as");
    pw3.println("           else ");
    pw3.println("           begin");
    pw3.println("             FSM=2; ");
    pw3.println("             v=1'b1;");
    pw3.println("           end");
    pw3.println("         end");
    pw3.println("         // ------------------------------------------------------");
    pw3.println("         1 : // State A");
    pw3.println("         begin");
    pw3.println("           if (" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b0) FSM=1;       // empty");
    pw3.println("           else if (" + as + "==1'b0 && " + af + "==1'b1 && " + e + "==1'b0) FSM=0;  // af");
    pw3.println("           else ");
    pw3.println("           begin");
    pw3.println("             FSM=2; ");
    pw3.println("             v=1'b1;");
    pw3.println("           end");
    pw3.println("         end");
    pw3.println("         // ------------------------------------------------------");
    pw3.println("         default : // Violation");
    pw3.println("         begin");
    pw3.println("           FSM=2;");
    pw3.println("           v=1'b1;");
    pw3.println("         end");
    pw3.println("    endcase");
    pw3.println("   end");
    pw3.println("   ");
    pw3.println("  assign violation = v;");
    pw3.println("           ");
    pw3.println("  initial ");
    pw3.println("  begin");
    pw3.println("    FSM = 0;");
    pw3.println("    v = 0;");
    pw3.println("  end");
    pw3.println("endmodule");

}

// ********************************************************************************************
public void generateContains(String strA,String as,String af,String e) {
    System.out.println("\ngenerateContains");

    // CCSL+
    // A contains e
    pw1.print(strA);
    pw1.print(" contains ");
```

```
        pw1.print(e);


        // CCSL
        // as sampledOn e precedes af
        pw2.print(as + " sampledOn " + e + " precedes " + af);
        pw2.println("");

        // as alternatesWith af
        pw2.print(as);
        pw2.print(" alternatesWith ");
        pw2.print(af);
        pw2.println("");


        // SystemVerilog
        // --------------------------------------

        pw3.println("// @Generated");
        pw3.println("// " + strA + " contains " + e);
        pw3.println("");
        pw3.println("module Contains (");
        pw3.println("  input " + as + ",");
        pw3.println("  input " + af + ",");
        pw3.println("  input " + e + ",");
        pw3.println("  output violation");
        pw3.println("  );");
        pw3.println("");
        pw3.println("  int unsigned FSM;");
        pw3.println("  reg v;");
        pw3.println("");
        pw3.println("  always @ (" + as + " or " + af + " or " + e + ")");
        pw3.println("   begin");
        pw3.println("    case (FSM)");
        pw3.println("         // ------------------------------------------------------------");
        pw3.println("         0 : // State " + strA + "'");
        pw3.println("         begin");
        pw3.println("           if (" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b0) FSM=0;       // empty");
        pw3.println("           else if (" + as + "==1'b1 && " + af + "==1'b0 && " + e + "==1'b0) FSM=1;  // " + as);
        pw3.println("           else if (" + as + "==1'b1 && " + af + "==1'b0 && " + e + "==1'b1) FSM=1;  // " + as + "," + e);
        pw3.println("           else ");
        pw3.println("           begin");
        pw3.println("             FSM=2; ");
        pw3.println("             v=1'b1;");
        pw3.println("           end");
        pw3.println("         end");
        pw3.println("         // ------------------------------------------------------------");
        pw3.println("         1 : // State " + strA);
        pw3.println("         begin");
        pw3.println("           if (" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b0) FSM=1;       // empty");
        pw3.println("           else if (" + as + "==1'b0 && " + af + "==1'b0 && " + e + "==1'b1) FSM=1;  // " + e);
        pw3.println("           else if (" + as + "==1'b0 && " + af + "==1'b1 && " + e + "==1'b0) FSM=0;  // " + af);
        pw3.println("           else if (" + as + "==1'b0 && " + af + "==1'b1 && " + e + "==1'b1) FSM=0;  // " + af + "," + e);
        pw3.println("           else ");
        pw3.println("           begin");
        pw3.println("             FSM=2; ");
        pw3.println("             v=1'b1;");
        pw3.println("           end");
        pw3.println("         end");
        pw3.println("         // ------------------------------------------------------------");
        pw3.println("         default : // Violation");
        pw3.println("         begin");
        pw3.println("           FSM=2;");
        pw3.println("           v=1'b1;");
        pw3.println("         end");
        pw3.println("    endcase");
        pw3.println("   end");
        pw3.println("   ");
        pw3.println("  assign violation = v;");
        pw3.println("           ");
        pw3.println("  initial ");
        pw3.println("  begin");
        pw3.println("    FSM = 0;");
        pw3.println("    v = 0;");
        pw3.println("  end");
        pw3.println("endmodule");

    }
}
```