Universitat Oberta de Catalunya

Treball Final Màster de Recerca

VerilCIRC Metrics

– Metric-aware verification of digital circuits –

Estudiant:	Enric Rubio Manrique
Consultor:	David Bañeres Besora
Estudis:	Màster Universitari en Programari Lliure
Semestre:	2016-17 / 1
Data:	10 de gener de 2017

Índex de continguts

Resum

El projecte VerilCIRC Metrics té l'objectiu de crear una eina útil per a l'ensenyament a distància del disseny de circuits digitals. S'integra en un projecte més ampli anomenat VerilCIRC, que actualment ja proporciona als alumnes les eines necessàries per resoldre els exercicis que se'ls assignen.

VerilCIRC ja disposa d'un sistema per avaluar les solucions proposades pels alumnes en termes de funcionalitat. Amb el projecte VerilCIRC Metrics es vol afegir la capacitat d'avaluar les solucions en funció de la seva qualitat, en termes d'optimització de recursos i estalvi d'àrea i retard dels circuits.

Des del començament el projecte pretén treure profit de les eines existents de programari lliure per a la síntesi de circuits digitals. Algunes d'elles permeten consultar mètriques de complexitat dels dissenys que manipulen.

El resultat és una aplicació que, a partir dels dissenys dels alumnes, realitza les adaptacions necessàries perquè les eines lliures de síntesi puguin llegir-los, fer-ne la valoració de la qualitat del disseny i, finalment, emmagatzemar els resultats un fitxer de sortida de format estàndard.

En el seu estat actual, el projecte ja permet de contrastar la qualitat de diverses solucions per a un mateix exercici i trobar-ne les millors. També ha servit per conèixer el programa Yosys i les possibilitats que ofereix per a futurs projectes.

Capítol 1. Introducció

1.1. Justificació de l'interès de la proposta per a la societat

En les assignatures bàsiques de l'àrea d'Arquitectura de Computadors, com per exemple Fonaments de Computadors, els alumnes aprenen conceptes bàsics dels sistemes digitals.

Una de les competències que s'ha d'adquirir en aquestes assignatures bàsiques es l'anàlisi i síntesi de circuits combinacionals i seqüencials: l'alumne ha de ser capaç de dissenyar un circuit a partir d'una especificació.

El problema que apareix en aquest tipus d'exercicis és que l'alumne no sap si la seva solució és correcta, atès que no pot tenir una resposta personalitzada per a cada exercici.

Actualment en l'assignatura de Fonaments de Computadors, els alumnes utilitzen una eina de programari anomenada VerilCIRC (http://wyoming.uoc.es/pubver/) que els permet de verificar de forma automàtica si un circuit digital que ells han sintetitzat és funcionalment equivalent a la solució proposada per l'equip docent. En cas d'error se li ofereix un retorn personalitzat indicant-li un contraexemple on el seu circuit és erroni.

Una mancança de l'eina actual és que comprova si els circuits són funcionalment equivalents però no té en compte la qualitat de la solució, és a dir, si s'ha utilitzat un nombre excessiu de components o bé el temps de càlcul del circuit (retard) no acompleix els requisits de l'exercici.

1.2. Objectius de la recerca

L'objectiu inicial d'aquest TFM és l'estudi d'eines (lliures o comercials) de verificació automàtica de circuits que permetin tenir en compte mètriques de càlcul d'àrea o de retard del circuit.

El segon objectiu serà la utilització o implementació d'una eina lliure que permeti verificar circuits dissenyats amb el programa VerilUOC, de forma automàtica i tenint en compte aquestes mètriques per tal d'indicar si el circuit acompleix uns criteris de qualitat.

Els resultats d'aquest treball seran utilitzats en l'assignatura de Fonaments de Computadors del Grau d'enginyeria Informàtica de la UOC i seran supervisats per part dels seus membres.

1.3. Enfocament i mètode seguit

Per a la implementació del nou programari s'ha aplicat la següent metodologia:

- S'ha analitzat el conjunt d'eines lliures disponibles de simulació i síntesi digital, amb la finalitat d'explorar la seva utilitat en el càlcul de mètriques de complexitat de circuits digitals.
- Entre totes les eines lliures analitzades, s'ha considerat que la millor opció era utilitzar el programa de síntesi Yosys, que treballa en estreta connexió amb Berkeley-ABC, una altra eina lliure molt potent.
- L'ús de Yosys requereix treballar amb fitxers en format Verilog, mentre que els exercicis a analitzar estan en el format CIRC del programa VerilUOC. Per aquest motiu s'ha realitzat un seguit d'adaptacions sobre una eina ja existent en el projecte VerilCIRC, el *parser* de fitxers CIRC, que permeten d'utilitzar-lo com a conversor de fitxers CIRC al format Verilog.
- S'ha desenvolupat l'aplicació principal del projecte, que treu profit del conversor a Verilog i del programa Yosys per analitzar les mètriques de complexitat dels dissenys en format CIRC. Els resultats es lliuren en un arxiu en un format estàndard que pugui llegir-se amb facilitat des d'altres programes, i que també permeti comparar les diferents solucions per a un mateix circuit.
- Finalment es realitzen totes les proves necessàries per assegurar que el funcionament de les aplicacions és correcte.
- Tots els programes s'han desenvolupat en llenguatge JAVA.

1.4. Productes obtinguts

Els resultats d'aquest projecte es manifesten en les següents aplicacions:

- L'aplicació CircParser, que és una adaptació d'una eina ja existent en el projecte VerilCIRC, i que permet de convertir a format Verilog els dissenys del programa VerilUOC desats en format CIRC.
- L'aplicació ParsedVerilogProcessor, que pren com a entrada els fitxers Verilog generats pel programa CircParser, i els adequa per fer-los compatibles amb les eines lliures (Yosys i altres) que treballen amb fitxers Verilog.
- L'aplicació VerilCircMetrics, que treu profit de les dues anteriors i del programa de síntesi Yosys per analitzar els exercicis dels alumnes i generar taules amb els resultats de les seves mètriques de complexitat, indicadors de la qualitat de les solucions. El programa pot ser utilitzat de forma autònoma, o bé cridat des d'una altra aplicació o integrat dins de qualsevol altra aplicació JAVA.

1.5. Estructura del document - descripció dels altres capítols

- El capítol 2, «Plantejament del projecte» fa una relació dels materials de partida per iniciar el projecte, descriu les decisions en les seves primeres fases, i també fa una relació de les eines de programari lliure que s'han estudiat i que poden ser útils per dur-lo a terme.
- El capítol 3, «Eines de programari creades» descriu dos programes desenvolupats en el decurs del projecte, per tal de facilitar les tasques i les proves malgrat que no són pròpiament el producte final. Aquests programes són CircParser i ParsedVerilogProcessor.
- El capítol 4, «El programa VerilCircMetrics» descriu en profunditat el producte principal d'aquest projecte, que és el programa encarregat de fer els càlculs de mètriques de complexitat dels dissenys en format CIRC. Es detallen els requisits per a la seva posada en marxa, unes instruccions d'ús del programa, una extensa descripció del seu funcionament intern i, per últim, un exemple de realització manual de les tasques que el programa efectua automàticament.
- Per últim, el capítol 5, «Conclusions i obtenció de resultats» exposa una valoració dels resultats obtinguts, resumeix les incidències que s'han presentat durant les proves i fa una proposta de treballs futurs basats en aquest projecte.

Capítol 2. Plantejament del projecte

2.1. Eines de partida.

2.1.1. Fitxers de dades.

El director del projecte ha proporcionat un conjunt d'arxius que serviran de punt de partida del projecte i proporcionen material per a les proves. Concretament són aquests:

- Un document anomenat `format_circ.doc` amb una breu explicació de l'estructura del format CIRC.
- Un fitxer anomenat `correct_circ_students.tgz`, que inclou 7395 circuits correctes realitzats per alumnes de l'assignatura de disseny de circuits digitals. Tots ells estan en els formats CIRC i Verilog i el nom de l'arxiu segueix el format `<username>_<semester>_<num_exercise>_<num_try>_<date_upload>`, on s'especifiquen, per aquest ordre: el nom de l'alumne, el semestre, el nombre de l'exercici, el nombre de l'intent i la data d'entrega.
- Un fitxer anomenat `correct_circ.zip` que inclou 141 solucions als mateixos exercicis, realitzades pels professors de l'assignatura, també en els formats CIRC i Verilog.
- Malauradament els exercicis dels alumnes estan identificats per un nombre d'exercici, mentre que els exercicis dels professors estan identificats pel seu nom. Per aquest motiu se subministra un fitxer anomenat `exercise_veriluoc.sql`, importat d'una base de dades MySQL, i que inclou la correspondència entre els nombres dels exercicis i els seus noms.

2.1.2. Parser d'arxius CIRC.

El director del projecte també ha proporcionat el codi font d'un *parser* de fitxers CIRC, que facilita la tasca de llegir el seu contingut però que també permet de convertir-los a codi Verilog.

Aquest conversor ha estat de gran utilitat, doncs la conversió a Verilog és el primer pas per poder enviar una descripció dels circuits dissenyats amb VerilUOC a les eines lliures més conegudes de simulació i síntesi.

El *parser* de fitxers CIRC genera un subconjunt de Verilog funcional i molt simple, únicament amb declaracions «`input`», «`output`», «`reg`» i «`assign`», i blocs «`always`», sense subcircuits. Aquest resultat és perfectament utilitzable per les eines de simulació i síntesi, malgrat que algunes particularitats de la sintaxi generada obliguen a fer-hi algunes modificacions, que seran detallades més endavant.

2.2. Plantejament inicial.

Per al càlcul de les mètriques de complexitat es va partir de dos plantejaments d'inici: d'una banda, la interpretació directa del contingut del fitxer CIRC; de l'altra, la conversió del fitxer CIRC a format Verilog i posterior interpretació per part d'una eina de programari lliure.

2.2.1. Opció 1: interpretació directa del fitxer CIRC

Amb l'ajuda del *parser* de fitxers CIRC, subministrat pel director del projecte, existeix la possibilitat d'analitzar l'estructura del propi circuit, comptabilitzant nombres de nodes, de portes i de registres. Per als càlculs s'atorgaria a cadascun dels components de llibreria un pes arbitrari, tant en retard com en superfície. Amb totes aquestes dades es podria calcular les dependències entre nodes i el retard de la cadena més desfavorable.

Avantatges d'aquest plantejament:

- Permet treballar directament amb els components de VerilUOC (que és allò que l'alumne ha introduït i vist en el seu monitor) en lloc de realitzar els càlculs a partir d'una reducció a expressions matemàtiques inherent a tota conversió a Verilog.
- Permet decidir exactament quins paràmetres es volen utilitzar per valorar la complexitat del circuit. L'ús de programes externs no dóna tanta llibertat, ja que només es disposa de les mètriques que aquest programa proporcioni.

Inconvenients:

- En el cas que es vulgui «reciclar» el codi font d'alguna aplicació lliure per fer els càlculs directament sobre la representació d'un fitxer CIRC, amb tota seguretat caldrà fer conversions que podrien ser tan o més complexes que una conversió a Verilog. Si pel contrari es vol partir de zero, aquest plantejament obliga a desenvolupar tota una metodologia de càlcul de complexitat, en certa manera «reinventar la roda» atès que hi ha programari lliure que ja executa aquestes funcions, molt avançat i que s'ha desenvolupat considerant les necessitats dels equips d'investigació i les característiques dels dispositius de lògica programable reals.
- Aquest camí no ofereix cap sortida visible a l'hora de realitzar el càlcul de la «millor alternativa possible», en el sentit que si es vol optimitzar un circuit que inclou components de llibreria com ara multiplexors, descodificadors i components aritmètics, serà molt més complicat que fer-ho amb un circuit format per portes lògiques. Només es podria aconseguir amb tècniques de *technology mapping*, el desenvolupament de les quals va molt més enllà dels terminis d'aquest projecte.

2.2.2. Opció 2: Conversió del fitxer CIRC a format Verilog

Es tracta d'adaptar el propi *parser* de fitxers CIRC (que de fet és un conversor de CIRC a Verilog que forma part del projecte VerilCIRC) per tal que pugui treballar de forma autònoma i generar arxius Verilog que puguin ser llegits per eines lliures d'anàlisi,

simulació i síntesi. Aquestes eines serien les encarregades de realitzar els càlculs requerits en aquest projecte. Els resultats d'aquests càlculs es recopilen en un arxiu de format estàndard per al seu procés posterior.

Avantatges d'aquest plantejament:

- La major part de la feina de convertir els fitxers CIRC a Verilog ja està feta: la fa el *parser* del projecte VerilCIRC.
- Treballar amb el format Verilog obre la porta a utilitzar eines de síntesi molt potents, i per tant accedir a opcions de simplificació, optimització i *technology mapping*. Això últim és imprescindible si es vol obtenir una «solució òptima» per als exercicis.
- Les mètriques de complexitat (càlculs d'àrea i retard) estaran basades en la representació real del circuit en portes simples o en qualsevol tecnologia de dispositius programables que pugui gestionar-se des del programari de síntesi.

Inconvenients:

- Utilitzar aquestes eines de síntesi per a l'objectiu d'aquest projecte requereix l'aprenentatge del seu ús en una certa profunditat.
- Cal fer moltes proves per assegurar que la conversió de fitxers CIRC a Verilog és totalment compatible amb les eines que es voldran utilitzar.
- Segurament la informació estadística que proporcioni el programa de síntesi no estarà expressada explícitament en els termes «àrea total del circuit» o bé «retard màxim del circuit». Per tenir una estimació d'aquests valors, caldrà fer una interpretació del conjunt de dades que les eines proporcionaran. Tampoc aquesta informació es podrà obtenir en un format estàndard, sinó que s'haurà d'obtenir analitzant i filtrant la sortida del programa cap a la consola.

2.2.3. Conclusió

Després de valorar els avantatges i inconvenients dels dos plantejaments, s'ha optat per la segona opció. Els problemes que planteja es consideren assumibles i a canvi ofereix moltes més possibilitats sense exigir forçosament més treball.

2.3. Selecció de les eines lliures a utilitzar

2.3.1. Icarus Verilog

També conegut com iVerilog [ICV001], el seu desenvolupament es va iniciar el 1998. Suporta les versions de l'estàndard de 1995, 2001 i 2005 i una part de SystemVerilog. Pot funcionar en plataformes Linux, FreeBSD, OpenSolaris, AIX, Windows i Mac OS X. També existeix una versió en línia amb interfície web. Està disponible en els repositoris oficials d'algunes distribucions de Linux, com per exemple Ubuntu i totes les seves derivades.

Amb fama de ser el millor simulador Verilog lliure, Icarus Verilog millora contínuament i s'ha utilitzat per al treball real de disseny en entorns empresarials, com a simulador i com a sintetitzador. Pot utilitzar-se associat al conjunt d'eines lliures gEDA per al disseny de maquinari.

En versions anteriors Icarus Verilog havia estat una eina de simulació i síntesi, per aquest motiu la intenció inicial en el projecte VerilCIRC Metrics era aprofitar la seva capacitat de síntesi. Malauradament les darreres versions han perdut el suport d'aquesta funcionalitat i el programa dóna errors quan s'intenta. De tota manera és una eina útil per comprovar si són correctes els fitxers Verilog generats per l'eina de conversió del projecte.

Adreça web: http://iverilog.icarus.com/

2.3.2. Verilator

És un compilador que a partir de la definició d'un circuit en codi Verilog genera codi equivalent en C / C++, adequat per a la simulació a gran velocitat.

En el context d'aquest projecte, Verilator és una eina útil per comprovar si la sintaxi generada pel conversor de CIRC a Verilog és correcta. En aquest sentit, Verilator és més estricte que Icarus Verilog,

És un simulador compilat de codi obert, de molt alta velocitat, que a partir del codi Verilog genera codi en C++ o SystemC. Ha estat adoptat principalment en el món acadèmic [VRL001] i del programari lliure, però empreses com NXP, ARM o Infineon l'han utilitzat i/o han contribuït al seu desenvolupament. Està disponible en els repositoris oficials d'algunes distribucions de Linux, com per exemple Ubuntu i totes les seves derivades.

Adreça web: http://www.veripool.org/wiki/verilator

2.3.3. Berkeley-ABC

És un creixent programari per a la síntesi i verificació de circuits de lògica seqüencial [AMI001]. Combina l'optimització lògica escalable, i un *technology mapping* orientat a minimitzar els retards dels circuits.

ABC proporciona una implementació experimental d'algoritmes innovadors per a la síntesi i verificació seqüencial, i també un entorn de programació per elaborar aplicacions similars.

ABC té una gran potència però malauradament no té un suport complet per a la lectura de fitxers Verilog. Malgrat això, gràcies a la combinació amb un altre programa lliure (Yosys) se n'ha pogut treure profit per a aquest projecte.

Està disponible ens repositoris oficials de la distribució Ubuntu i derivats.

Pàgina web: https://people.eecs.berkeley.edu/~alanmi/abc/abc.htm

2.3.4. Yosys

Yosys és un programa que es va crear per cobrir l'absència d'eines lliures de síntesi extensibles i adaptables a qualsevol arquitectura [CLW002]. Treu profit del programa preexistent ABC, amb el qual intercanvia informació en el format BLIF, per realitzar optimitzacions avançades a nivell de portes.

Permet sintetitzar gran part de l'estàndard Verilog 2005 i s'ha provat amb un ampli ventall de dispositius reals, incloent les CPU OpenRISC 1200, OpenMSP430 i altres.

El que fa que Yosys sigui especialment útil per a aquest projecte, és que pot llegir arxius Verilog i realitzar-hi operacions de *technology mapping*, a més de realitzar les optimitzacions i transformacions necessàries en el disseny per tal que pugui ser enviat al programa ABC i aprofitar així totes les opcions que aquest ofereix.

El programa s'utilitza des de la línia de comandes, però permet de visualitzar diagrames gràficament utilitzant les eines externes «`xdot`» i «`graphviz`».

Yosys és un programa extraordinari que permet l'anàlisi, l'optimització i la síntesi de dissenys, dirigides al seu desplegament en dispositius reals de lògica programable. El programa presenta infinitat de possibilitats, el domini de les quals seria un projecte complet per ell mateix.

Està disponible ens repositoris oficials de la distribució Ubuntu i derivats.

Pàgina web: http://www.clifford.at/yosys/

Capítol 3. Eines de programari creades

Per assolir els objectius del projecte s'han desenvolupat tres aplicacions en llenguatge Java. Totes elles poden utilitzar-se des de la línia de comandes o mitjançant una interfície gràfica.

- CircParser – un convertidor de fitxers CIRC a format Verilog basat en el *parser* del projecte VerilCIRC subministrat pel director del projecte.
- ParsedVerilogProcessor – un postprocessador que modifica els fitxers Verilog generats pel *parser*, amb l'objectiu d'adequar-los als requeriments d'entrada de l'aplicació Yosys utilitzada per al càlcul de mètriques de complexitat.
- VerilCircMetrics – el programa que realitza el càlcul de les mètriques de complexitat, tant de fitxers individuals com de directoris sencers. Aquest programa és el producte principal del projecte i té el seu propi capítol 4, «El programa VerilCircMetrics» a la pàgina 27.

3.1. Conversió de CIRC a Verilog (CircParser)

CircParser és un programa en Java basat en el *parser* de fitxers CIRC proporcionat pel director del projecte. Aquest programa pren com a entrada un fitxer de VerilUOC amb l'extensió CIRC i genera un fitxer Verilog de comportament equivalent.

Els fitxers resultants són lleugerament diferents dels Verilog proporcionats com a exemples per al càlcul del projecte. Això pot deure's a què hagin estat generats amb una versió del *parser* diferent del codi font proporcionat pel director del projecte.

En el desenvolupament d'aquesta eina s'ha procurat de mantenir inalterat el paquet «`com.dbaneres.file.veriluoc`», de manera que només actualitzant aquest paquet a les versions més actuals, automàticament quedi actualitzada l'eina sense haver de fer cap altra modificació.

3.1.1. Ús en mode línia de comandes

Utilització de l'eina:

```
java -jar CircParser.jar [-o <outputfile>] [-l] <inputfiles...>
```

Significat de les opcions:

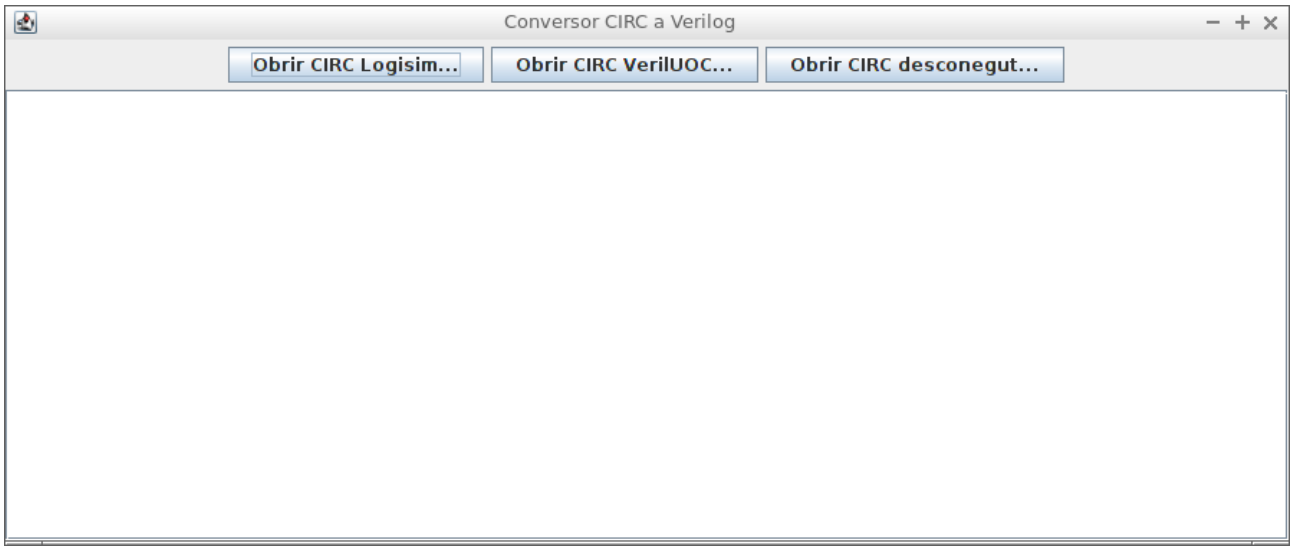
- `-o <outputfile>`: on `<outputfile>` és el nom del fitxer Verilog resultant de la conversió. En el cas de no especificar aquest paràmetre, el fitxer de sortida tindrà exactament el mateix nom que el CIRC original, amb el sufix «`.v`» afegit a continuació.
- `-l`: significa que el fitxer CIRC està desat en el format «antic» generat pel programa Logisim (darrera versió, 2.7.1) o bé per les primeres versions de VerilUOC que eren compatibles amb aquest format. En cas de no indicar-se aquest paràmetre, s'assumeix que el fitxer CIRC ha estat desat en el format «nou»,

generat per les darreres versions de VerilUOC. Per a més informació sobre les diferències entre els dos formats, consultar l'apartat 3.1.3, «Diferències entre versions dels fitxers CIRC» a la pàgina 16.

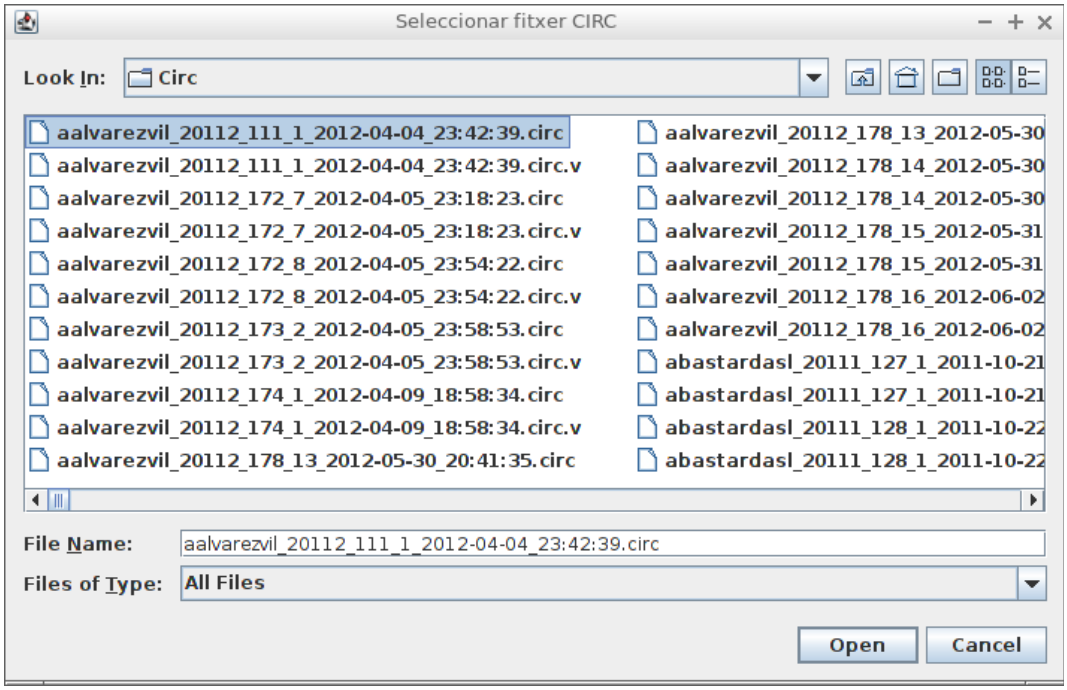
- `-g`: intenta endevinar la versió amb què ha estat generat el fitxer CIRC. Per aconseguir-ho, primer intenta fer la conversió amb el «format nou» corresponent a les versions més actuals de VerilUOC. Després comprova que en fitxer resultant no aparegui el text «`error_3`». Si apareix, significa que el *parser* ha trobat elements «`wire`» mal connectats, probablement a causa de la mida incorrecta dels components. Aleshores el programa intentarà realitzar una nova conversió amb el «format antic» corresponent a les primeres versions de VerilUOC o fitxers generats per Logisim. En cas que persisteixi el text «`error_3`», es considerarà que la conversió no és possible i s'enviarà una notificació a la consola. Per a més informació sobre les diferències entre els dos tipus de fitxers, consultar l'apartat 3.1.3, «Diferències entre versions dels fitxers CIRC» a la pàgina 16.
- `<inputfiles...>` és una llista de fitxers CIRC a convertir, separats per espais. Si s'ha indicat l'opció «`-g`» els fitxers poden ser una barreja de diferents versions i el programa tractarà de detectar-les individualment.

3.1.2. Ús en mode interactiu

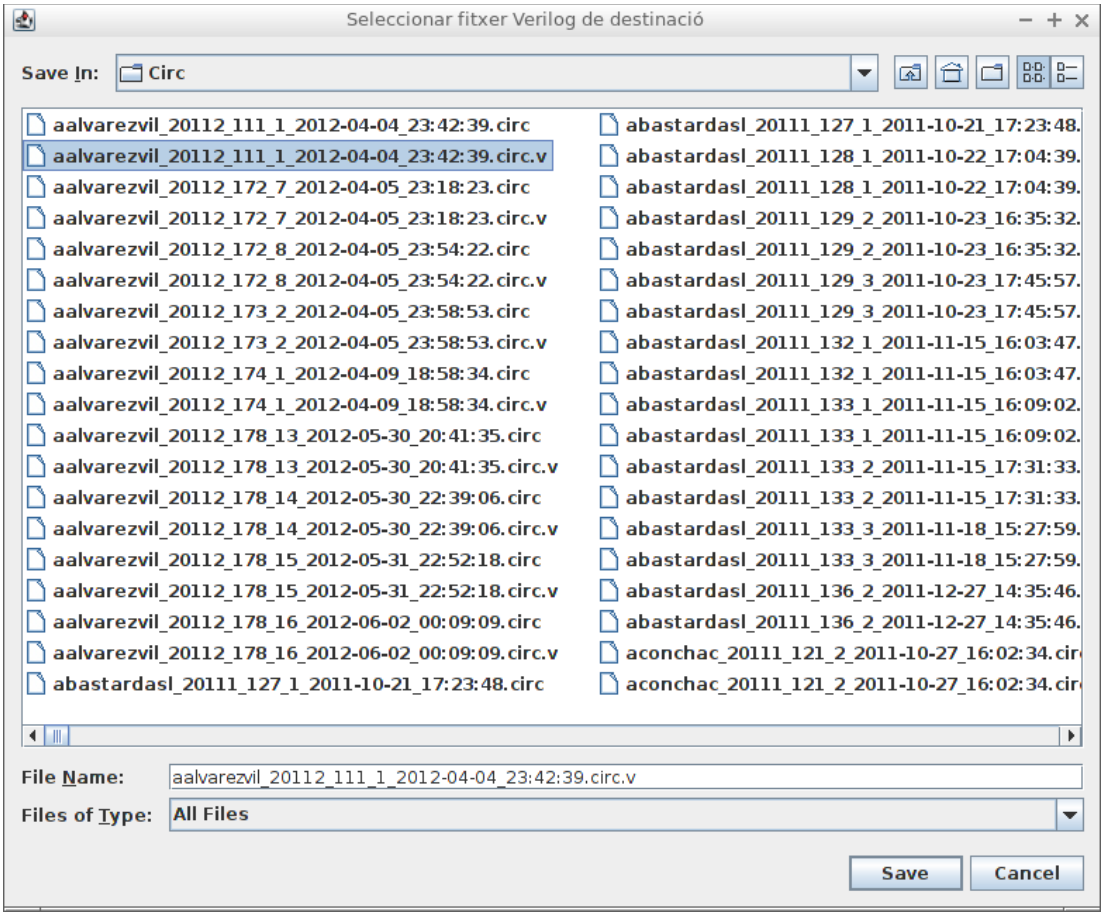
En cas que no s'hagi indicat cap fitxer d'entrada, el programa mostrarà una senzilla interfície d'usuari que permet la selecció dels fitxers de forma interactiva.



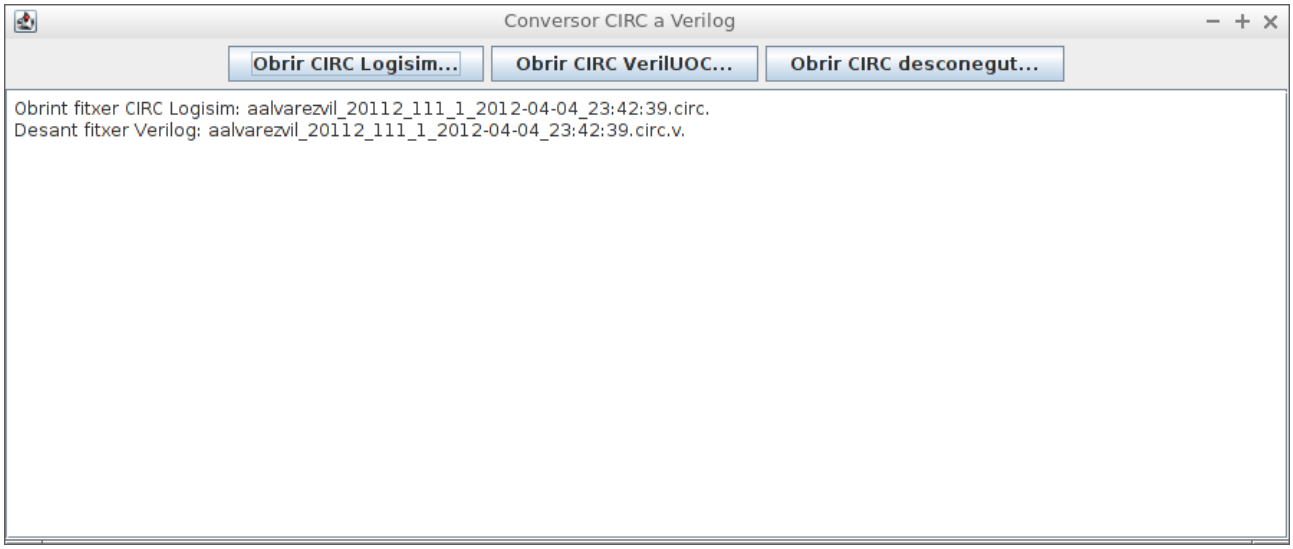
Les tres opcions permeten obrir arxius de tipus 0, 1 o desconegut, respectivament. En prémer qualsevol d'elles apareix un diàleg de selecció de fitxer.



El següent diàleg sol·licita un nom per al fitxer Verilog de destinació:



Un cop completada l'operació, la pròpia finestra de l'aplicació actua com a visor del registre de missatges:



Es poden realitzar tantes conversions com es desitgi. El programa finalitza quan es tanca la finestra principal.

3.1.3. Diferències entre versions dels fitxers CIRC

El programa VerilUOC està basat en Logisim però inclou alguns canvis per adaptar-lo a la funció concreta a què està destinat. Entre aquests canvis hi ha modificacions a la seva llibreria de components, que afecten les dimensions dels mateixos.

Aquests canvis, però, no afecten per igual tots els fitxers CIRC subministrats com a exemples. Aquells fitxers que van ser generats amb les versions més antigues del programa VerilUOC conserven el format original i la compatibilitat amb el programa Logisim. Pel contrari, els fitxers CIRC generats amb les darreres versions de VerilUOC utilitzen el nou format.

Quan s'intenta llegir un fitxer CIRC amb el format incorrecte, el *parser* inclou en la seva sortida uns missatges amb el text «`error_3`» indicant que el circuit conté cables que no estan connectats enlloc. Aquest error no hauria de produir-se en circuits que ja s'ha comprovat que són correctes, per tant se suposa que la causa de l'error és el canvi en la mida d'algun component del circuit.

Els components afectats pel canvi de versió són els següents:

- `comp_Adder` – Afecta les dimensions del component i obliga orientació «sud» fixa, a diferència del format original en què l'orientació és «est» fixa.
- `comp_Comparator` – Afecta les dimensions del component i obliga orientació «sud» fixa, a diferència del format original en què l'orientació és «est» fixa. El mode de comparació és obligatòriament sense signe, havent-se eliminat de la interfície d'usuari del programa l'opció de comparació amb signe.
- `comp_Counter` – Afecta les dimensions del component i obliga orientació «sud» fixa, a diferència del format original en què l'orientació és «est» fixa.
- `comp_DflipFlop` – Afecta només les dimensions del component.

- `comp_Negator` – Afecta les dimensions del component i obliga orientació «sud» fixa, a diferència del format original en què l'orientació és «est» fixa. D'altra banda, aquest component (negador aritmètic) no apareix a la interfície d'usuari de VerilUOC, per tant en les darreres versions ja ni tan sols pot incloure's en els dissenys.
- `comp_RAM` – Afecta només la posició d'una etiqueta de text.
- `comp_Register` – Afecta les dimensions del component i obliga orientació «sud» fixa, a diferència del format original en què l'orientació és «est» fixa.
- `comp_Shifter` – Afecta només les dimensions del component. D'altra banda, aquest component (desplaçament de *bits* aritmètic) no apareix a la interfície d'usuari de VerilUOC, per tant en les darreres versions ja ni tan sols pot incloure's en els dissenys.

El *parser* original del projecte VerilUOC permet d'interpretar els dos tipus d'arxiu, però necessita que se li indiqui el tipus mitjançant un paràmetre numèric, corresponent el «tipus 0» a la versió antiga i el «tipus 1» a la versió nova.

Malauradament no existeix cap indicador a l'interior dels fitxers CIRC que permeti identificar el tipus d'arxiu a partir del seu contingut. En el programa CircParser s'ha inclòs una opció que intenta deduir el tipus d'arxiu (veure descripció de l'opció «`-g`» a la pàgina 14 (apartat 3.1.1).

3.1.4. Modificacions realitzades sobre el parser de VerilCIRC

Durant les proves realitzades s'han trobat problemes per interpretar alguns dels fitxers CIRC subministrats com a exemple, que s'han pogut resoldre mitjançant petites modificacions en el codi del *parser* original del projecte VerilCIRC. Els següents apartats expliquen aquestes modificacions.

a) Component comparador en mode «complement a 2»

Aplicant el programa CircParser als fitxers CIRC d'exemple, molts d'ells produeixen l'error «`error_5_mode_comparator`». Estudiant el codi del *parser* s'ha comprovat que aquest error es produeix quan l'esquema inclou un comparador aritmètic en mode «complement a 2». Es dóna la circumstància que cap de les versions de VerilUOC analitzades (2.2.2, 2.3.1 i 3.0.0) inclouen aquesta opció en el seu entorn de treball (només permeten comparació sense signe). Tenint això en compte, s'ha fet una modificació en el codi del *parser* per tal que internament, durant la conversió, es consideri sempre el comparador sense signe, independentment del mode que indiqui el fitxer CIRC.

La modificació s'ha efectuat sobre el mètode `comp_Comparator` de la classe `VerilogXMLWriter.WriteContext` dins del paquet `com.dbaneres.file.veriluoc`. El codi original és:

```
...
        posless = posequal.movepoint(compattr.facing,  0, 20);
        posgreater = posequal.movepoint(compattr.facing, 0, -20);
    }

    if(compattr.is2complement == true)
        xurro += "error_5_mode_comparator";
    else
        xurro += VerilogComponents.comp_COMPARATOR(...
...
```

El codi modificat queda de la següent manera:

```
...
        posless = posequal.movepoint(compattr.facing,  0, 20);
        posgreater = posequal.movepoint(compattr.facing, 0, -20);
    }

    compattr.is2complement = false; /* Added by: erubioman */
    if(compattr.is2complement == true)
        xurro += "error_5_mode_comparator";
    else
        xurro += VerilogComponents.comp_COMPARATOR(...
...
```

S'ha comprovat que els fitxers Verilog generats pel CircParser després d'aquest canvi són equivalents als que es va subministrar com exemple per als mateixos exercicis.

b) Error 3 per errors de connexions

Alguns circuits no poden ser processats amb cap de les dues versions del *parser*. Per exemple, tots aquests exercicis resolts pel professor:

```
cmp2.circ
codif.circ
cont1a.circ
cram2.circ
demux.circ
desp2l.circ
desp2ra.circ
desp2r.circ
litr.circ
mult4.circ
mux.circ
muxval.circ
sm4.circ
sr4.circ
ual2b.circ
```

La relació completa es troba a l'apartat 5.2.1, «Fitxers que no s'han pogut convertir de format CIRC a Verilog» a la pàgina 56. Tots ells tenen en comú el fet de començar amb la línia

```
<project version="1.0" source="2.3.2">
```

a diferència de la majoria de circuits que comencen amb la línia

```
<project source="2.7.1" version="1.0">
```

Tanmateix, no tots els esquemes emmagatzemats amb la versió 2.3.2 donen errors.

c) Excepció per error en camp «max» del component «Counter»

Una altra modificació que s'ha efectuat sobre el *parser* es deu a una excepció que es produeix durant la seva execució sobre alguns fitxers CIRC. Són els següents:

```
dbaneres_20111_135_2_2011-11-09_15:13:28.circ
dbaneres_20111_135_5_2011-11-11_14:51:42.circ
dbaneres_20111_135_7_2011-11-11_14:52:11.circ
```

En tots ells s'ha observat que contenen fragments similars a:

```
    <comp lib="4" name="Counter" loc="(430,300)">
      <a name="width" val="6" />
      <a name="max" val="4" />
      <a name="trigger" val="rising" />
      <a name="label" val="" />
      <a name="labelfont" val="SansSerif plain 12" />
    </comp>
```

Mitjançant depuració del *parser*, s'ha comprovat que l'error es produeix a causa de l'atribut «`max`» amb valor 4. El procediment encarregat de processar aquesta línia aparentment espera un valor hexadecimal i en descarta els dos primers caràcters, provocant una excepció quan la cadena que representa el valor té una llargada menor que 2. L'error està en la següent funció:

```
void com.dbaneres.file.veriluoc.XMLCompReader.ReadAttributes(Element comp)
...
    else if (name.equals("max")) {
        maxcount = Integer.parseInt(childNode.getAttribute("val").substring(2),16);
    }
    else if (name.equals("value")) { //constant
        maxcount = Integer.parseInt(childNode.getAttribute("val").substring(2),16);
    }
...
```

I ha estat substituït per:

```
void com.dbaneres.file.veriluoc.XMLCompReader.ReadAttributes(Element comp)
...
    else if (name.equals("max")) {
        String val = childNode.getAttribute("val");
        maxcount = val.length() > 2 ? Integer.parseInt(val.substring(2), 16) : 0;
    }
    else if (name.equals("value")) { //constant
        String val = childNode.getAttribute("val");
        maxcount = val.length() > 2 ? Integer.parseInt(val.substring(2), 16) : 0;
    }
...
```

d) Fitxers CIRC amb coordenades negatives en elements «wire».

Alguns exercicis, un cop processats pel *parser* generen codi Verilog incorrecte. El motiu és que el fitxer CIRC conté cables amb coordenades negatives en algun dels seus extrems. Quan es dóna aquesta circumstància, en el fitxer Verilog resultant es troben identificadors d'elements «`wire`» amb el caràcter guió, que no hi està permès.

Els fitxers CIRC afectats són els següents:

```
dgalindoj_20121_210_26_2012-12-11_21:35:34.circ
dgalindoj_20121_210_27_2012-12-11_21:36:11.circ
dgalindoj_20121_210_28_2012-12-11_21:36:29.circ
dgalindoj_20121_210_29_2012-12-13_18:27:35.circ
dgalindoj_20121_210_30_2012-12-13_18:29:21.circ
dgalindoj_20121_210_31_2012-12-28_19:08:00.circ
```

Per resoldre aquest problema es proposa una petita modificació en el *parser*, als mètodes `toverilog()` de la classe `com.dbaneres.file.veriluoc.Location`, en vistes a evitar la presència de guions en els identificadors dels elements «`wire`», quan alguna de les coordenades dels seus extrems és negativa.

Estat original del programa:

```
        public String toverilog() {
                return  "v_"+ x/10 + "_" + y/10 + "_";
        }

        public String toverilog(int bit) {
                return  "v_"+ x/10 + "_" + y/10 + "_" + bit;
        }

        public String toverilog(int bit, int negatein) {
            String neg = "";
            if(negatein == 1) neg = "~";
                return  neg + "v_"+ x/10 + "_" + y/10 + "_" + bit;
        }

        public String toverilog(int bit, int negatein, int invert) {
            String neg = "";
            if((negatein == 1 && invert == 0) || (negatein == 0 && invert == 1)) neg = "~";
                return  neg + "v_"+ x/10 + "_" + y/10 + "_" + bit;
        }
```

Modificació proposada:

```
    private String _toverilog() {
        return "v_"+ x/10 + "_" + y/10 + "_";
    }

        public String toverilog() {
            return _toverilog().replace('-',  '_');
        }

        public String toverilog(int bit) {
                return toverilog() + bit;
        }

        public String toverilog(int bit, int negatein) {
            String neg = "";
            if(negatein == 1) neg = "~";
                return neg + toverilog() + bit;
        }

        public String toverilog(int bit, int negatein, int invert) {
            String neg = "";
            if((negatein == 1 && invert == 0) || (negatein == 0 && invert == 1)) neg = "~";
                return neg + toverilog() + bit;
        }
```

S'ha comprovat que aquesta modificació, a part de solucionar els casos concrets on es presentava l'error, no té cap efecte en la resta de fitxers: ni en el codi Verilog generat ni en les estadístiques lliurades per VerilCircMetrics.

3.2. Tractament de circuits Verilog (ParsedVerilogProcessor)

Els fitxers Verilog proporcionats per a les proves, així com els generats pel CircParser, tenen unes particularitats en la seva sintaxi que els fan incompatibles amb les eines lliures comprovades (Icarus, Verilator, Berkeley-ABC, Yosys).

Aquestes particularitats trobades són:

- Presència del caràcter punt («.») en el nom del mòdul. No acceptat per Icarus, Verilator, Berkeley-ABC ni Yosys.
- Utilització d'elements «`wire`» abans de la seva definició. Provoca errors en Berkeley-ABC i alertes per definició implícita a Yosys.
- Utilització de constants no definides (`TRUE`, `FALSE`). Provoca errors a Berkeley-ABC i a Verilator, així com alertes per definició implícita a Yosys.

Amb la finalitat de resoldre aquest problema s'ha creat l'eina ParsedVerilogProcessor.

3.2.1. Descripció del programa

ParsedVerilogProcessor és un programa en Java que llegeix els fitxers Verilog generats pel *parser* original de VerilCIRC i els fa les adaptacions necessàries per tal que puguin ser llegits per les eines lliures Icarus, Verilator, Berkeley-ABC i Yosys.

S'ha decidit fer aquestes adaptacions en una eina separada de postprocés, per evitar fer canvis en el *parser* del projecte VerilCIRC que és el nucli del programa CircParser.

Les modificacions que es realitzen sobre el fitxer Verilog original són les següents:

- Canvi del nom del mòdul principal del circuit. Aquest nom conté originalment un caràcter punt («.») que no és acceptat per les eines de compilació, simulació i síntesi. La correcció aplicada consisteix en canviar el nom per un altre que només contingui caràcters alfanumèrics i guions baixos.
- Definició dels elements «`wire`» amb nom `TRUE` i `FALSE`, assignats a valors constants 1 i 0, respectivament, sempre que no estiguin ja definits prèviament.
- Reordenació de les definicions del cablejat («`wire`») atenent a les dependències entre ells, de manera que cap cable sigui utilitzat en clàusules «`assign`» o blocs «`always`» sense haver estat definit prèviament en una declaració «`wire`».
- Eliminació de cables redundants. Per a cada cable dibuixat en l'esquema de VerilUOC, el *parser* original genera una munió d'elements «`wire`» (un per cada segment recte) i després els assigna tots entre ells. Això provoca un nombre molt alt d'elements «`wire`» que no es correspon amb el nombre real de connexions del circuit, i que més tard, segons quin sigui el procediment d'anàlisi del circuit, pot produir un resultat equívoc. El programa ParsedVerilogProcessor elimina tots els elements «`wire`» que són còpies d'un altre, i els substitueix per l'original.
- Eliminació de nodes no referenciats. Si s'indica l'opció corresponent, el programa ParsedVerilogProcessor detecta i elimina tots aquells nodes que no tenen cap efecte a les sortides del circuit ni als elements de memòria.

3.2.2. Ús en mode línia de comandes

Utilització de l'eina `ParsedVerilogProcessor.jar`:

```
java -jar ParsedVerilogProcessor.jar [-o <outputfile>] [-s] [-w] [-r] [-a]
```

Significat de les opcions:

`-o <outputfile>`: on `<outputfile>` és el nom del fitxer Verilog resultant de la conversió. En el cas de no especificar aquest paràmetre, el fitxer de sortida tindrà exactament el mateix nom que el Verilog original, amb el sufix «`.v`» afegit a continuació.

`-s`: reordena els elements «`wire`» en funció de les dependències existents entre ells, de manera que tots ells siguin declarats abans de ser utilitzats en sentències «assign» o en blocs «`always`».

`-w`: simplifica els elements «`wire`» redundants, és a dir, agrupa en un únic «`wire`» tots els segments d'una mateixa connexió.

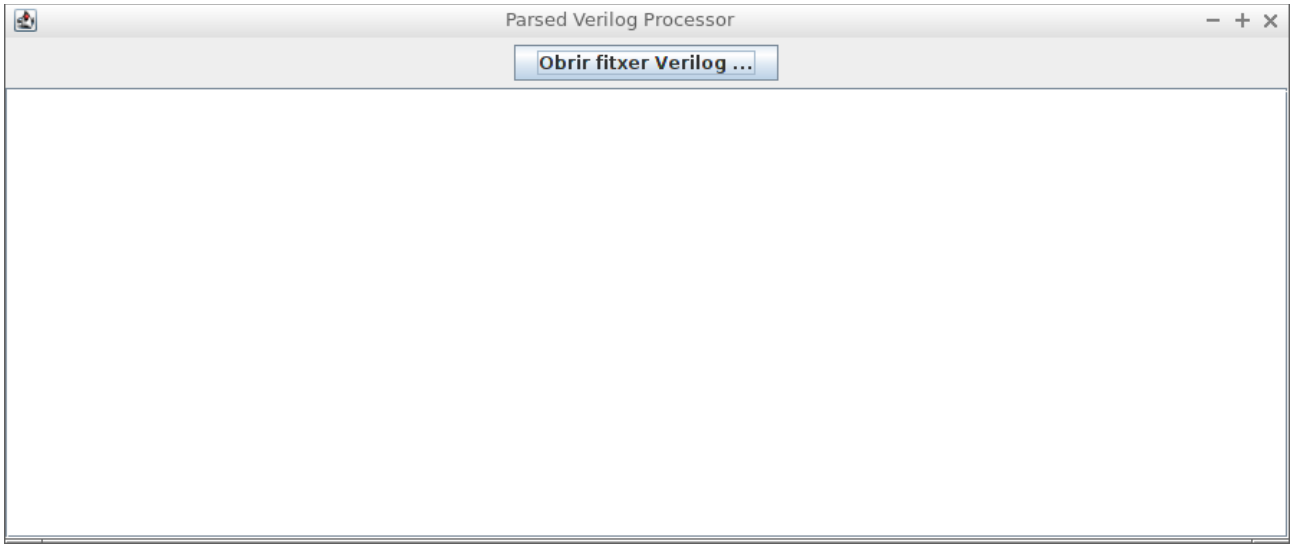
`-r`: elimina de forma recursiva les sentències «`assign`», el resultat de les quals no tingui reflex a cap sortida o a cap element «`reg`». També elimina tots els elements «`wire`» que siguin sobrants pel mateix motiu.

`-a`: aplica totes les transformacions associades a les opcions «`-s`», «`-w`» i «`-r`».

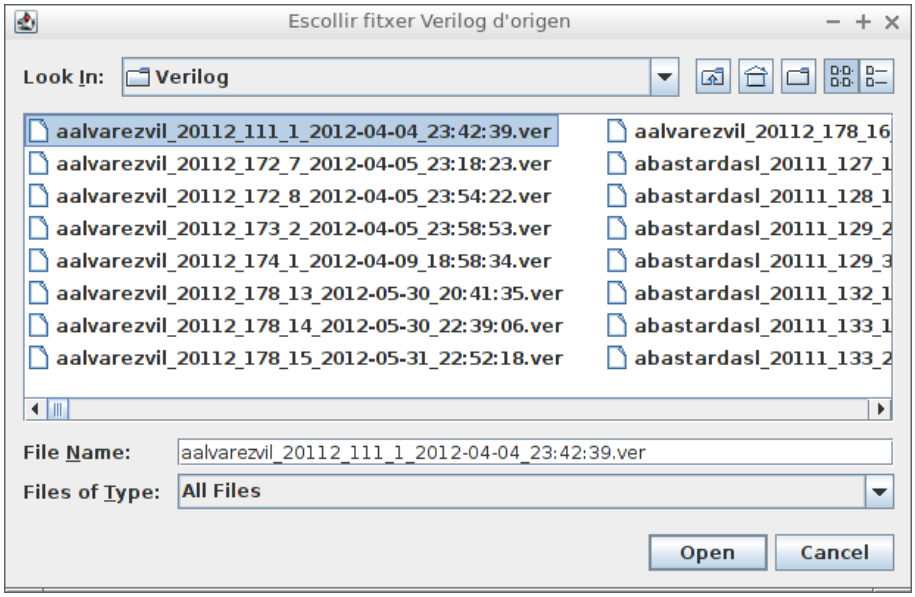
Independentment de les opcions que s'hagi indicat, el fitxer Verilog resultant sempre tindrà el mòdul principal reanomenat, de manera que no inclogui en el seu nom cap caràcter considerat il·legal per les eines Icarus, Verilator, Berkeley-ABC o Yosys.

3.2.3. Ús en mode interactiu

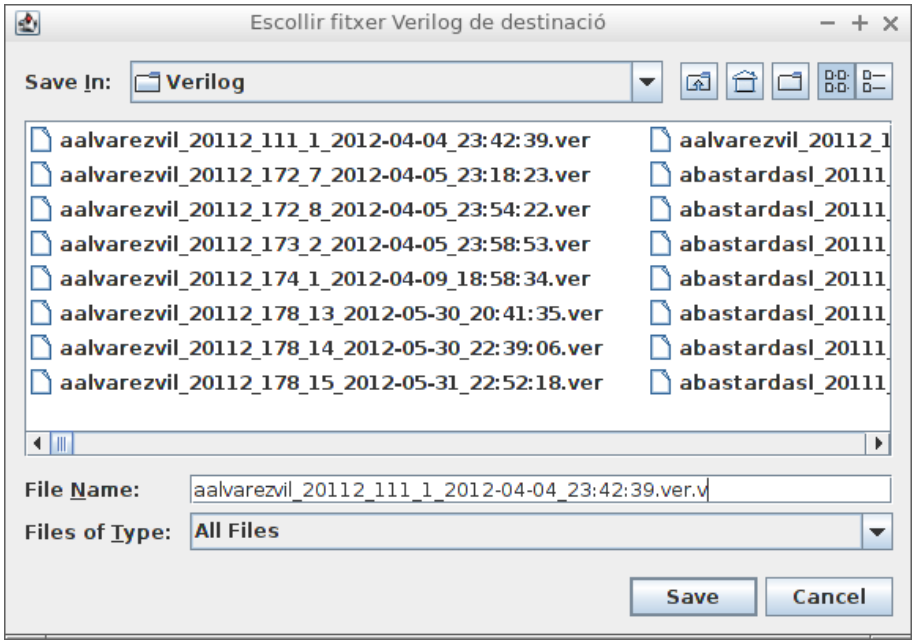
En cas que no s'hagi indicat cap fitxer d'entrada, el programa mostrarà una senzilla interfície d'usuari que permet la selecció dels fitxers de forma interactiva.



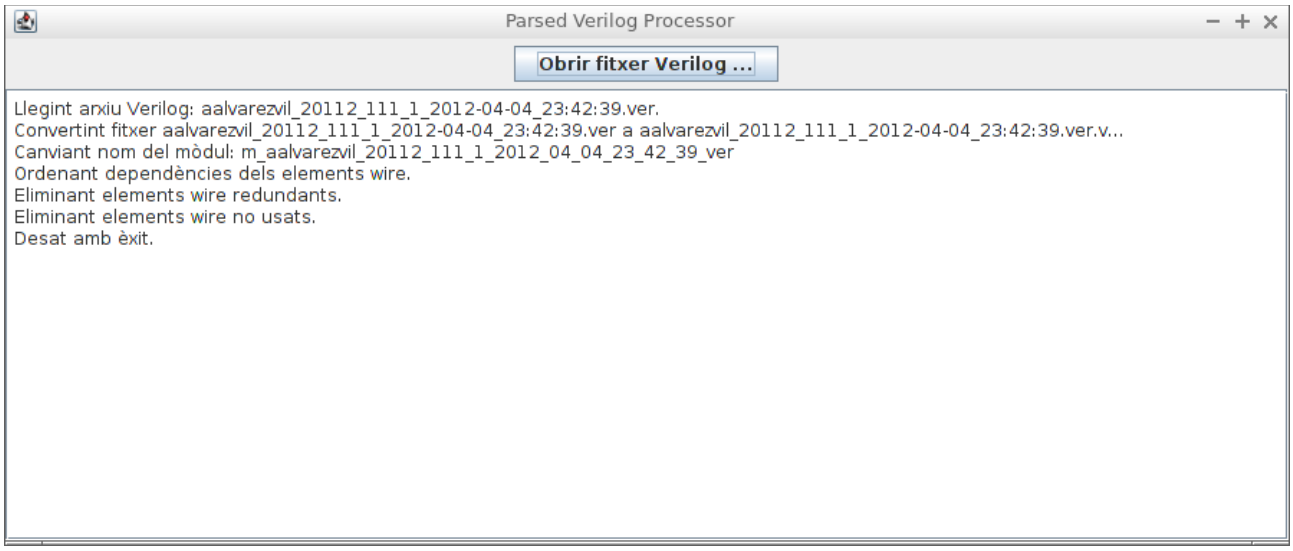
La finestra conté un espai de text que servirà com a registre de missatges, i un únic botó que permet obrir el diàleg que inicia el procés de conversió.



El següent diàleg sol·licita un nom per al fitxer Verilog de destinació:



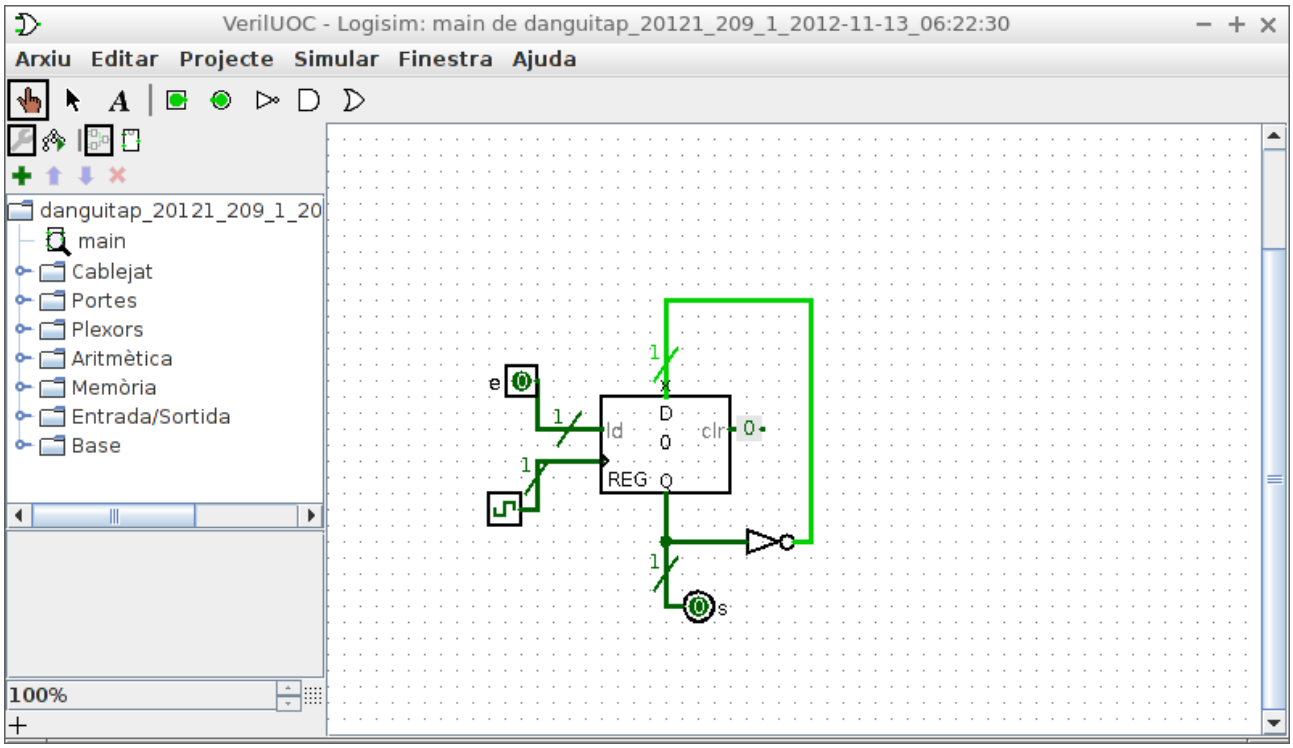
Un cop completada l'operació, la finestra de l'aplicació mostra els missatges que s'han anat generant al llarg de la conversió. Són els mateixos missatges que s'haurien produït en modalitat línia de comandes.



Es poden realitzar tantes conversions com es desitgi. El programa finalitza quan es tanca la finestra principal.

3.2.4. Exemple d'ús

S'ha agafat com a exemple un dels esquemes realitzats pels alumnes. S'ha triat l'esquema «`danguitap_20121_209_1_2012-11-13_06:22:30.circ`» per ser senzill i incloure almenys un element de memòria:



Convertint aquest esquema a Verilog utilitzant CircParser s'obté el següent resultat:

```
$ java -jar ~/bin/CircParser.jar danguitap_20121_209_1_2012-11-13_06\:22\:30.circ -o danguita1.v
Obrint fitxer CIRC VerilUOC: danguitap_20121_209_1_2012-11-13_06:22:30.circ.
Desant fitxer Verilog: danguita1.v.
$ cat danguita1.v
module danguitap_20121_209_1_2012-11-13_06:22:30.circ(clk_0, e_0,s_0);
input clk_0;
input e_0;
reg regxxx0_0;
output s_0;
assign v_12_32_0 = clk_0;
wire v_12_32_0;
assign v_29_34_0 = ~v_26_34_0;
wire v_29_34_0;
assign v_27_27_0 = 0;
wire v_27_27_0;
assign v_13_24_0 = e_0;
wire v_13_24_0;
always @(posedge v_17_29_0 or posedge v_25_27_0)
//defverif @(posedge v_17_29_0 or posedge v_25_27_0 or posedge v_17_27_0)
if(v_25_27_0)
  begin
   regxxx0_0 <= 0;
  end
else
  begin
        if(v_17_27_0)
          begin
            if(v_17_29_0)
              begin
                regxxx0_0 <= v_21_25_0;
              end
          end
   end
assign v_21_31_0= regxxx0_0;
wire v_21_31_0;
assign s_0 = v_22_38_0;
assign v_21_34_0 = v_21_31_0;
wire v_21_34_0;
assign v_30_34_0 = v_29_34_0;
wire v_30_34_0;
assign v_25_27_0 = v_27_27_0;
wire v_25_27_0;
assign v_30_19_0 = v_30_34_0;
wire v_30_19_0;
assign v_13_27_0 = v_13_24_0;
wire v_13_27_0;
assign v_17_27_0 = v_13_27_0;
wire v_17_27_0;
assign v_26_34_0 = v_21_34_0;
wire v_26_34_0;
assign v_21_38_0 = v_21_34_0;
wire v_21_38_0;
assign v_22_38_0 = v_21_38_0;
wire v_22_38_0;
assign v_21_19_0 = v_30_19_0;
wire v_21_19_0;
assign v_21_25_0 = v_21_19_0;
wire v_21_25_0;
assign v_13_32_0 = v_12_32_0;
wire v_13_32_0;
assign v_13_29_0 = v_13_32_0;
wire v_13_29_0;
assign v_17_29_0 = v_13_29_0;
wire v_17_29_0;
$
```

Executant ParsedVerilogProcessor sobre aquest fitxer Verilog, l'arxiu resultant queda tal com es mostra a continuació:

```
$ java -jar ~/bin/ParsedVerilogProcessor.jar -a danguita1.v
Convertint fitxer danguita1.v a danguita1.v.v...
Canviant nom del mòdul: m_danguita1_v
Ordenant dependències dels elements wire.
Eliminant elements wire redundants.
Eliminant elements wire no usats.
Desat amb èxit.
$ cat danguita1.v.v
module m_danguita1_v(clk_0, e_0,s_0);
input clk_0;
input e_0;
reg regxxx0_0;
output s_0;
assign s_0 = regxxx0_0;
wire v_29_34_0;
assign v_29_34_0 = ~regxxx0_0;
wire v_27_27_0;
assign v_27_27_0 = 0;
always @(posedge clk_0 or posedge v_27_27_0)
//defverif @(posedge clk_0 or posedge v_27_27_0 or posedge e_0)
if(v_27_27_0)
  begin
   regxxx0_0 <= 0;
  end
else
  begin
        if(e_0)
          begin
            if(clk_0)
              begin
                regxxx0_0 <= v_29_34_0;
              end
          end
   end
endmodule
$
```

Pot observar-se el canvi del nom del mòdul, també la reducció del nombre de cables i la reorganització de les declaracions per evitar referències a elements abans de la seva declaració. Malgrat els canvis, el fitxer Verilog resultant és funcionalment equivalent a l'original.

Capítol 4. El programa VerilCircMetrics

Aquest és el programa que sintetitza tot el treball realitzat en aquest projecte. A partir dels fitxers CIRC produïts pels alumnes i el professor, genera una taula en format CSV amb tota la informació de mètriques que pugui obtenir-se de la combinació dels programes Yosys i Berkeley-ABC.

En resum, les funcions que aquest programa duu a terme són les següents:

- Associació dels exercicis proposats pel professor amb les solucions creades pels alumnes, a partir d'una taula de noms d'exercicis en format CSV (veure l'apartat 4.1.4, «Format del fitxer CSV de nombres i noms d'exercicis», a la pàgina 28).
- Anàlisi, mitjançant el programa Yosys, dels exercicis del professor i dels alumnes. Cadascun dels exercicis està representat per un fitxer CIRC o bé pel seu equivalent en format Verilog, o bé per una combinació de tots dos.
- En cas que l'exercici estigui representat en format CIRC, fa la conversió a format Verilog mitjançant l'eina CircParser (apartat 3.1 a la pàgina 13), com a pas introductori per al càlcul de mètriques de complexitat.
- Adaptació dels fitxers Verilog (tant si han estat generats pel CircParser com si s'han subministrat directament) mitjançant l'eina ParsedVerilogProcessor (apartat 3.2 a la pàgina 21), que hi efectua les modificacions necessàries per tal que el programa Yosys els pugui llegir.
- Procés dins del programa Yosys, a partir d'un script que proporciona un resultat indicatiu de la complexitat del circuit, amb l'ajuda del programa Berkeley-ABC.
- Creació d'una taula en format CSV, que associa els noms i nombres dels exercicis, amb nom de l'alumne o professor, dates, hores, nombre d'intents i resultats de les mètriques de complexitat. S'ha escollit aquest format perquè permet la seva ràpida conversió a qualsevol altre i també la importació des de bases de dades, així com la seva manipulació directa des de qualsevol programari de fulls de càlcul.

4.1. Requisits per al funcionament

4.1.1. Programari instal·lat

- Es requereix tenir instal·lat l'entorn d'execució JAVA. El programa s'ha desenvolupat amb OpenJDK versió 8 però s'ha procurat mantenir compatibilitat a partir de la versió 6.
- Es requereix tenir instal·lat el programa Yosys. Algunes distribucions de Linux conegudes ja l'inclouen en els seus repositoris, com per exemple Ubuntu 16.04 LTS i totes les seves derivades. Cal tenir en compte que Yosys té les seves pròpies dependències. Si s'instal·la des dels repositoris oficials d'alguna distribució no hi hauria d'haver problemes.

4.1.2. Rutes i scripts

És necessari que en el directori actiu en el moment d'invocar l'aplicació, o bé en algun directori inclòs en el PATH del sistema, existeixin amb permisos d'execució els següents *scripts*:

```
veriluoc_yosys_script.sh
veriluoc_yosys_opt_script.sh
```

També es requereix que en el directori actiu en el moment d'invocar l'aplicació, o bé en el directori especificat per la variable d'entorn `VERILCIRCMETRICS_PATH`, estigui present el següent *script*:

```
verilcirc_metrics.abc
```

4.1.3. Directori de treball

El directori de treball ha de tenir permisos d'escriptura, ja que el programa necessita crear uns arxius temporals que seran utilitzats per a l'intercanvi d'informació entre els programes VerilCircMetrics i Yosys. Aquests arxius són destruïts automàticament després d'haver-los utilitzat. Els seus noms són els següents:

```
    temp_file.v
    temp_file.v.stats
```

4.1.4. Format del fitxer CSV de nombres i noms d'exercicis

En diferents apartats d'aquest document es fa referència a una taula en format CSV que el programa VerilCircMetrics espera rebre com a paràmetre per poder associar les solucions dels alumnes i del professor a un mateix exercici. Aquesta taula ha de tenir les següents característiques:

- És un fitxer de text on cada línia representa una filera de la taula.
- Dins de cada filera, les diferents dades estan separades pel símbol punt i coma «;».
- Les dades no tenen cometes que les envoltin, l'únic delimitador és el símbol «;».

- La primera dada de cada filera és numèrica i representa el nombre d'exercici.
- La segona dada és un text igual al nom de l'exercici (que equival al nom del fitxer de la solució del professor, sense l'extensió).
- Es permeten més dades (per exemple, comentaris) però seran ignorades pel programa VerilCircMetrics.
- El fitxer no conté una primera fila de capçaleres. Només conté dades.

Com a exemple, es mostra l'extracte del fitxer CSV utilitzat en aquest projecte, obtingut d'una exportació de la base de dades MySQL subministrada pel director:

```
$ cat exercise_veriluoc.csv
111;and2;Test exercise referenced in the VerilUOC wiki
112;funcio1;
113;funcio2;
114;mux;
115;muxval;
116;demux;
117;codif;
118;desp2l;
119;desp2r;
120;desp2ra;
...

...
344;vbiestable;flip-flop
345;vreg8b;8-bit register
346;vcoun8b;8-bit counter
347;20132_PR_1C;Exercise Final Project 1C Spring semester 20132
348;20132_PR_1D;Exercise Final Project 1D Spring semester 20132
349;20132_PR_1B;Exercise Final Project 1B Spring semester 20132
$
```

4.2. Ús del programa

El programa pot executar-se directament en línia de comandes o bé de forma interactiva.

4.2.1. Ús en mode línia de comandes: anàlisi massiva d'exercicis

En aquest mode el programa analitzarà mètriques de tots els exercicis que estiguin presents en un directori o grup de directoris. Els resultats s'emmagatzemaran en un fitxer en format CSV. Per obtenir una descripció detallada del significat de cadascun dels camps d'aquesta taula, es pot consultar l'apartat 5.1, «Resultats proporcionats pel programa» a la pàgina 54.

Per treballar en mode massiu cal invocar el programa `VerilCircMetrics.jar` amb el següent format:

```
java -jar VerilCircMetrics.jar
        [-c <inCsvNames>]
        [-o <outCsvTable]
        [-n <exNumber>]
        exercisesDirectory [...exercisesDirectory]
```

Significat de les opcions:

`-c <inCsvNames>` – és el nom del fitxer CSV que conté les associacions dels nombres i els noms dels exercicis. Aquest fitxer permet d'associar les solucions proposades pel professor amb les proposades pels alumnes (veure l'apartat 4.1.4, «Format del fitxer CSV de nombres i noms d'exercicis», a la pàgina 28). En cas de no indicar aquest paràmetre, en el fitxer de resultats la columna «`ExerciseNumber`» apareixerà buida en els exercicis del professor, i també la columna «`ExerciseName`» apareixerà buida en els exercicis dels alumnes.

`-o <outCsvTable>` – és el nom que tindrà el fitxer CSV de sortida, que inclou la taula amb els resultats. Si no s'indica cap nom, el fitxer de sortida tindrà com a nom per defecte «`VerilCircMetrics.csv`».

`-n <exNumber>` – permet d'analitzar només els exercicis amb un nombre d'exercici determinat, descartant-ne la resta. En realitat el paràmetre `<exNumber>` és una expressió regular de Java, de manera que amb caràcters comodí es poden filtrar conjunts d'exercicis.

`exercisesDirectory [...exercicesDirectory]` – és una llista de directoris on es cercaran (de forma recursiva) els fitxers CIRC i Verilog corresponents a les solucions als exercicis, tant del professor com dels alumnes. No hi ha cap requisit respecte del repartiment dels fitxers: poden estar els CIRC en un directori i els Verilog en un altre, o els exercicis dels alumnes i professors junts o separats. Tampoc cal que tots els exercicis tinguin associat un fitxer CIRC i també un fitxer Verilog: el programa efectuarà els càlculs amb la informació que tingui disponible. Es considera que un fitxer CIRC i un fitxer Verilog pertanyen a una mateixa solució quan ambdós tenen el mateix nom d'arxiu, un d'ells amb l'extensió «`.circ`» i l'altre amb una extensió que sigui una part de la cadena «`.verilog`» (habitualment «`.v`», «`.ver`» o bé «`.verilog`»).

4.2.2. Ús en mode línia de comandes: anàlisi d'exercicis individuals

VerilCircMetrics també permet l'anàlisi de fitxers individuals en format CIRC o Verilog. En aquest cas els resultats no s'emmagatzemen en una taula CSV sinó que es mostren directament a la consola. Per obtenir una descripció detallada del significat de cadascun dels camps que es mostraran, consultar l'apartat 5.1, «Resultats proporcionats pel programa» a la pàgina 54.

La forma d'invocar el programa és la següent:

```
java -jar VerilCircMetrics.jar fitxer [...fitxer]
```

És a dir, només cal executar el programa indicant a continuació el nom del fitxer, o una llista de noms separats per espais. No s'accepten caràcters comodí en els noms de fitxer. La sortida mostrada pel programa en aquesta situació serà similar al següent exemple. En el primer cas s'analitza un fitxer del tipus CIRC.

```
$ java -jar ~/bin/VerilCircMetrics.jar StudentsCirc/acladellas_20131_137_1_2013-10-20_10:20:55.circ
Intentant processar fitxer CIRC.
Creant Verilog a partir de fitxer CIRC: StudentsCirc/acladellas_20131_137_1_2013-10-20_10:20:55.circ
Renaming module to m_acladellas_20131_137_1_2013_10_20_10_20_55
Sorting wire dependencies.
Removing redundant wires.
20 mètriques trobades.

----- Resultats:
YosysWires = 24
YosysProcesses = 0
YosysCells = 11
D_FlipFlops = 0
MuxGates = 0
InvGates = 4
AndGates = 5
OrGates = 2
AbcNodes = 11
AbcLevels = 6
OptYosysWires = 15
OptYosysProcesses = 0
OptYosysCells = 10
OptD_FlipFlops = 0
OptMuxGates = 0
OptInvGates = 3
OptAndGates = 5
OptOrGates = 2
OptAbcNodes = 10
OptAbcLevels = 5
----- Fi dels resultats.
```

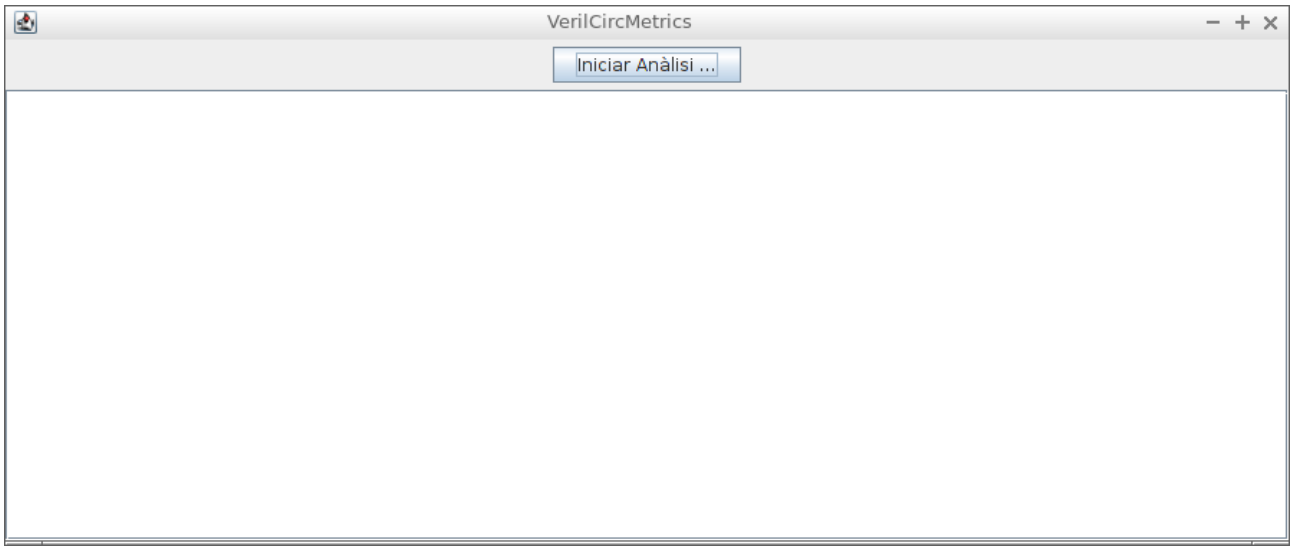
En el següent exemple s'executa la mateixa comanda, però en aquest cas es fa sobre la versió Verilog del mateix exercici de l'exemple anterior. Observi's que els resultats són idèntics.

```
$ java -jar ~/bin/VerilCircMetrics.jar StudentsVerilog/acladellas_20131_137_1_2013-10-20_10:20:55.ver
Processant fitxer StudentsVerilog/acladellas_20131_137_1_2013-10-20_10:20:55.ver
Translating file acladellas_20131_137_1_2013-10-20_10:20:55.ver into temp_file.v...
Renaming module to m_acladellas_20131_137_1_2013_10_20_10_20_55_ver
Sorting wire dependencies.
Removing redundant wires.
Successfully saved.
20 mètriques trobades.

----- Resultats:
YosysWires = 24
YosysProcesses = 0
YosysCells = 11
D_FlipFlops = 0
MuxGates = 0
InvGates = 4
AndGates = 5
OrGates = 2
AbcNodes = 11
AbcLevels = 6
OptYosysWires = 15
OptYosysProcesses = 0
OptYosysCells = 10
OptD_FlipFlops = 0
OptMuxGates = 0
OptInvGates = 3
OptAndGates = 5
OptOrGates = 2
OptAbcNodes = 10
OptAbcLevels = 5
----- Fi dels resultats.

$
```

L'opció d'anàlisi de fitxers individuals permet d'integrar VerilCircMetrics en altres eines que en podran gestionar els resultats. Per exemple, podria ser invocat des del propi programa VerilUOC per obtenir una anàlisi del circuit al mateix temps que s'està dissenyant.

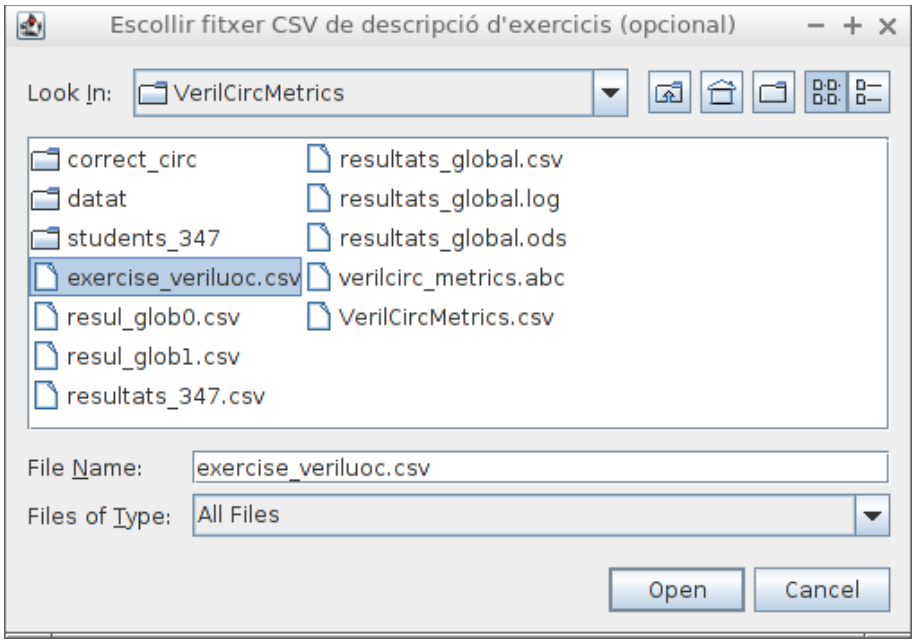
4.2.3. Ús en mode interactiu

En cas que el programa VerilCircMetrics s'executi sense indicar cap paràmetre ni cap directori per explorar, el programa mostrarà una interfície senzilla que permet la selecció interactiva de fitxers i directoris. No permet, però, el filtratge de fitxers segons el seu nombre d'exercici, ni tampoc l'anàlisi de fitxers individuals.

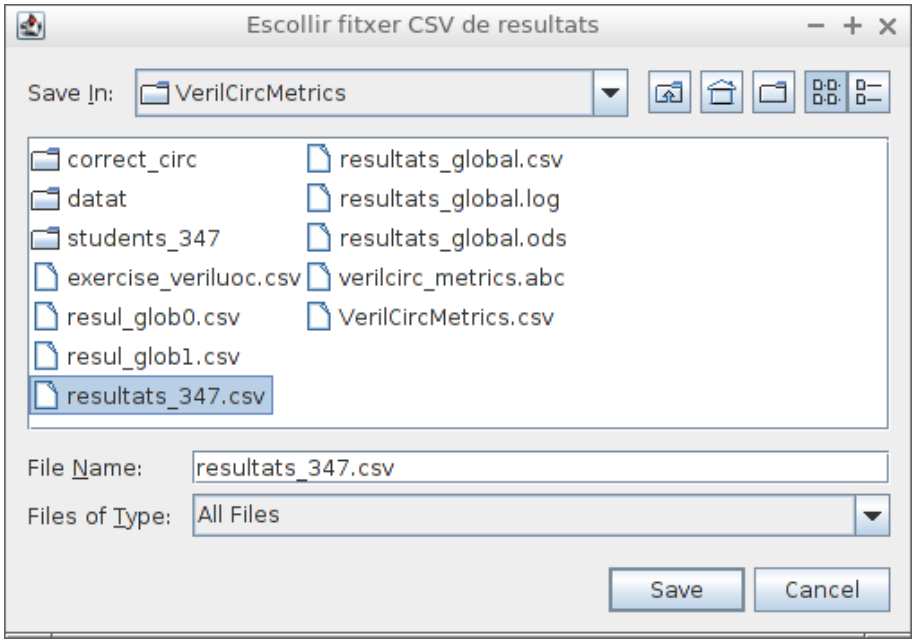


La primera pantalla no ofereix més opció que iniciar la anàlisi, o bé tancar la finestra i així sortir del programa. La part en blanc és una finestra de text on anirà apareixent un registre de cada operació que es realitza. És la mateixa informació que apareixeria per la consola si s'hagués invocat el programa des de la línia de comandes.

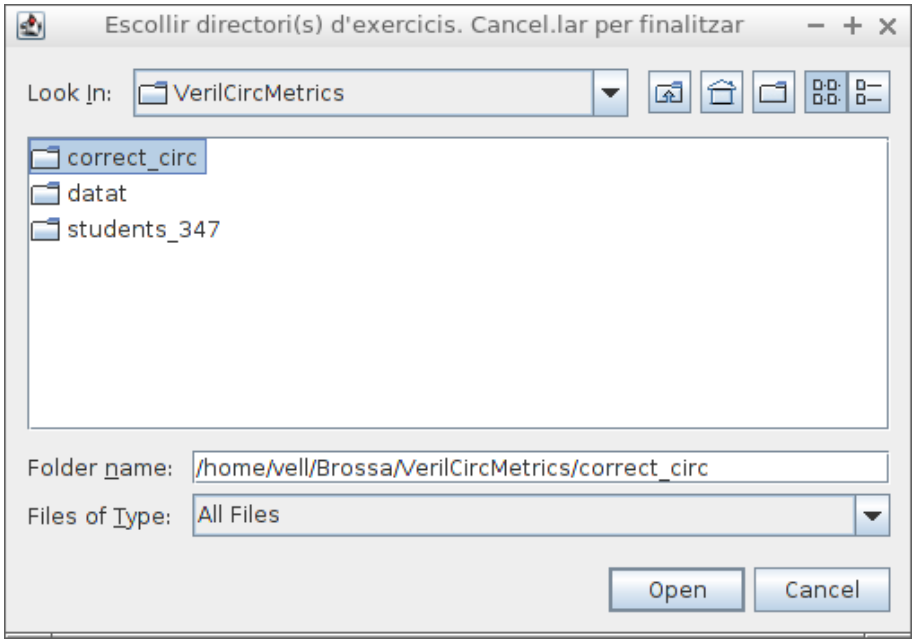
En cas de prémer el botó, un primer diàleg permet d'obrir l'arxiu CSV d'equivalències entre noms i nombres d'exercicis (veure l'apartat 4.1.4, «Format del fitxer CSV de nombres i noms d'exercicis», a la pàgina 28):



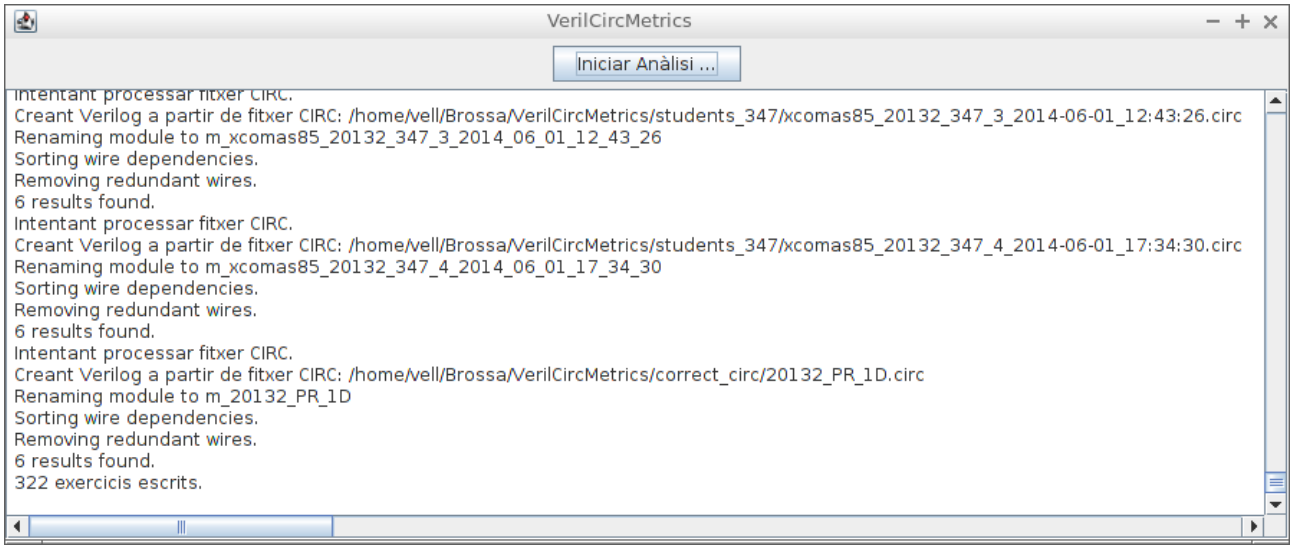
Es pot seleccionar un fitxer, o bé prémer l'opció de cancel·lar i no utilitzar el fitxer CSV. A continuació un segon diàleg demana el nom del fitxer CSV de resultats. En cas de no seleccionar-ne cap, es cancel·la l'operació i es torna a la pantalla inicial.



Si es tria un nom, es passa al següent diàleg que permet d'escollir un directori o estan emmagatzemats els fitxers CIRC i/o Verilog corresponents a les solucions dels exercicis:



Després d'escollir el directori, aquest apareixerà a la finestra de missatges i tornarà a aparèixer el mateix diàleg, oferint l'oportunitat de triar més directoris. Això és útil si les solucions del professor i les dels alumnes són en rutes diferents. Quan ja s'han triat tots els directoris desitjats, es prem el botó de cancel·lació per passar a la següent fase que és la anàlisi dels fitxers. A la finestra de missatges anirà apareixent un registre de totes les operacions realitzades.



4.3. Descripció del funcionament del programa

4.3.1. Dependències d'altres mòduls i llibreries

El programa utilitza els mòduls IO (*input / output*) i CLI (*command line interface*) de les llibreries *Apache Commons*. Són sota llicència *Apache* (lliure) i descarregables a les següents adreces web:

- https://commons.apache.org/proper/commons-cli/download_cli.cgi
- https://commons.apache.org/proper/commons-io/download_io.cgi

Addicionalment, el programa inclou alguns mòduls importats de les eines CircParser i ParsedVerilogProcessor. La inclusió d'aquests mòduls evita haver d'utilitzar ambdues eines com a programes externs.

Mòduls importats de CircParser:

- `com.dbaneres.file.veriluoc.Cache.java`
- `com.dbaneres.file.veriluoc.Location.java`
- `com.dbaneres.file.veriluoc.VerilogComponents.java`
- `com.dbaneres.file.veriluoc.VerilogXMLWriter.java`
- `com.dbaneres.file.veriluoc.XMLCompReader.java`

Mòduls importats de ParsedVerilogProcessor:

- `com.erubioman.verilog.processor.ParsedVerilogSorter.java`

4.3.2. Funcionament com a eina d'anàlisi massiva

En els següents punts es detallen les operacions que realitza el programa VerilCircMetrics quan és invocat per a efectuar una anàlisi massiva de tots els exercicis presents en un directori o conjunt de directoris. Tots els mètodes esmentats formen part de la classe `VerilCircMetrics`.

a) Lectura de la taula CSV d'associació d'exercicis

En cas que s'hagi indicat un fitxer CSV amb les equivalències dels nombres dels exercicis i els noms de les solucions del professor, s'executarà el mètode `readCsvTable`.

Aquest mètode llegeix el fitxer CSV (veure l'apartat 4.1.4, «Format del fitxer CSV de nombres i noms d'exercicis», a la pàgina 28) i crea dos estructures de tipus «`Map`», encarregades de fer la correcta associació entre la solució del professor i les solucions dels alumnes:

- `mapNamesNumbers` – té com a clau el nom de l'exercici, i com a valor el nombre associat. Aquest mapa s'utilitza per deduir el nombre d'exercici quan es troba el fitxer amb la solució del professor.
- `MapNumbersNames` – té com a clau el nombre de l'exercici, i com a valor un grup de dades que engloba: primer el nom de l'exercici; després un comptador indicant el nombre de solucions del professor trobades (només pot ser 0 o 1); i finalment un

comptador indicant el nombre de solucions d'alumnes trobades per a aquest mateix exercici. Tant el nom de l'exercici com els comptadors de solucions sortiran reflectits a la taula de resultats.

b) Lectura dels directoris

S'executa el mètode `readExercisesInDirectory`, per a cadascun dels directoris indicats com a paràmetre. Els directoris s'exploren de forma recursiva per cercar fitxers amb l'extensió «`.circ`» o bé una extensió que sigui una part de la cadena «`.verilog`».

Mentre es van trobant els fitxers, es van afegint en una estructura de tipus «`Map`» anomenada `filesMap`, on la clau és el nom del fitxer (sense extensió), i el valor associat és una parella de cadenes que indiquen respectivament les rutes completes dels fitxers CIRC i Verilog que comparteixen aquest nom de fitxer. En aquesta fase el programa només acumula la llista dels fitxers, i encara no sap si es tracta de fitxers corresponents a la solució del professor o a les solucions dels alumnes.

c) Associació dels fitxers dels exercicis

El mètode `readExercisesInFilesMap` recorre tot el mapa de fitxers `filesMap` creat en l'etapa anterior, i juntament amb els altres dos mapes `mapNumbersNames` i `mapNamesNumbers`, ja disposa de tota la informació necessària per crear una taula d'exercicis completa i llesta per analitzar.

Dels propis noms dels fitxers es dedueix si és un exercici d'un alumne o bé és la solució proposada pel professor. Consultar en l'apartat 2.1.1, «Fitxers de dades.» (pàgina 8) les condicions de la nomenclatura dels fitxers dels exercicis dels alumnes.

El resultat d'aquesta etapa serà una estructura de tipus «`Set`» anomenada `exercisesSet`, i formada per elements «`Exercise`». Cadascun d'aquests elements conté els següents camps:

- `fullName` – és el nom de l'arxiu sense extensió
- `userName` – és el nom de l'autor de l'exercici, deduït a partir del nom del fitxer si és d'un alumne, o bé «`0_Teacher`» si l'autor és el professor.
- `semester` – és el semestre en què s'ha realitzat l'exercici, deduït a partir del nom del fitxer si és d'un alumne, o bé en blanc si l'autor és el professor.
- `exerciseNum` – és el nombre de l'exercici, deduït a partir del nom del fitxer si és d'un alumne.
- `numTry` – és el nombre de l'intent, deduït a partir del nom del fitxer si és d'un alumne, o bé en blanc si l'autor és el professor.
- `date` – és la data de l'exercici, deduïda a partir del nom del fitxer si és d'un alumne, o bé en blanc si l'autor és el professor.
- `time` – és l'hora de l'exercici, deduïda a partir del nom del fitxer si és d'un alumne, o bé en blanc si l'autor és el professor.

En aquesta etapa també es descarten els exercicis repetits, i es comptabilitza el nombre total de solucions per a cada exercici, informació que s'inclourà després a la taula de resultats.

d) Anàlisi dels fitxers

La darrera fase és l'execució del mètode `outputTable`, que amb la informació de les taules generades en les etapes anteriors, va analitzant els fitxers un per un, i alhora va enviant les dades generades cap a la taula de resultats.

Per a cada exercici trobat a `exercisesSet` es fabrica una línia de dades en format CSV, separades pel caràcter «;», que es desarà al fitxer de resultats (detalls a l'apartat 5.1.1, «Format del fitxer CSV resultant», a la pàgina 54). Per fer-ho s'efectuen les següents operacions:

- Es recopilen les dades que no requereixen cap càlcul i que estan distribuïdes a les diferents taules generades en les passes anteriors.
- Si per a l'exercici actual existeix un fitxer CIRC associat, crida el mètode `getResultsFromCircFile`, explicat en el punt «e», «Obtenció del resultat a partir del fitxer CIRC», i se n'afegeixen els resultats a la línia en construcció.
- Si la passa anterior ha fallat, o no hi havia un fitxer CIRC associat a l'exercici, i pel contrari sí que hi ha un fitxer Verilog associat, aleshores es crida el mètode `getResultsFromVerilogFile`, explicat en el punt «f», «Obtenció del resultat a partir del fitxer Verilog», i se n'afegeixen els resultats a la línia en construcció.
- Per acabar, s'afegeix el text «`Circ`», «`Verilog`» o «`None`» per indicar si les dades s'han obtingut a partir del fitxer CIRC, del fitxer Verilog o de cap dels dos (anàlisi fallida).

Un cop completada l'escriptura de totes les línies de la taula de resultats, el programa finalitza. Els punts següents detallen el funcionament dels diferents mètodes referenciats per `outputTable`.

e) Obtenció del resultat a partir del fitxer CIRC

Si es disposa d'un fitxer CIRC per analitzar, el mètode `getResultsFromCircFile` intentarà fer-ne la conversió a Verilog i després analitzar-lo. Les passes per aconseguir-ho són les següents:

- Es crida el *parser* de fitxers CIRC, importat del programa CircParser, i representat per la classe `VerilogXMLWriter`. Seguint el mateix procediment del programa CircParser, per fer la conversió de CIRC a Verilog s'intentarà deduir si el fitxer CIRC és d'una versió antiga o nova (veure descripció de l'opció «`-g`» de CircParser a l'apartat 3.1.1, «Ús en mode línia de comandes», a la pàgina 13).
- En cas que la passa anterior hagi tingut èxit, se n'envia el resultat (un «`String`» que conté un programa Verilog sencer) al mètode `Translate` de la classe `ParsedVerilogSorter` (importada del programa ParsedVerilogProcessor), que s'encarrega d'adaptar-ne la sintaxi als requeriments del programa Yosys, i a continuació s'escriu el resultat en un fitxer temporal anomenat «`temp_file.v`».
- Es crida el mètode `getResultsFromTempFile` (veure punt «g» d'aquest mateix apartat) i es retorna el seu resultat.

f) Obtenció del resultat a partir del fitxer Verilog

Si es disposa d'un fitxer Verilog per analitzar, el mètode `getResultsFromVerilogFile` intentarà adaptar-ne la sintaxi als requeriments del programa Yosys per després analitzar-lo. Les passes per aconseguir-ho són les següents:

- S'envia el fitxer al mètode `Translate` de la classe `ParsedVerilogSorter` (importada del programa ParsedVerilogProcessor) i s'escriu el resultat en un fitxer temporal anomenat «`temp_file.v`».
- Es crida el mètode `getResultsFromTempFile` (veure punt «g» d'aquest mateix apartat) i es retorna el seu resultat.

g) Obtenció del resultat a partir del fitxer temporal

Aquesta passa executa dos programes externs anomenats `veriluoc_yosys_script.sh` i `veriluoc_yosys_opt_script.sh` . Tots dos fan la mateixa funció, que és enviar el fitxer temporal «`temp_file.v`» al programa Yosys, però ho fan amb paràmetres diferents: en el primer cas es fa només l'anàlisi del circuit, i en el segon cas se'n fa, a més, l'optimització per calcular el «millor resultat possible» de l'exercici.

Les crides es fan mitjançant el mètode `executeYosysScript`, que invoca el *script* i després en recull els resultats, que hauran quedat emmagatzemats en un fitxer temporal anomenat «`temp_file.v.stats`» . Com que el contingut d'aquest fitxer és una redirecció de la sortida de consola del programa Yosys, cal una seqüència d'una certa complexitat per tal d'extreure'n la informació que després s'afegirà a la taula de resultats.

El contingut dels *scripts* és el següent:

```
~/bin$ cat veriluoc_yosys_script.sh
#!/bin/bash

Nomfitxer="$1"
if [ "" = "$Nomfitxer" ]; then
    Nomfitxer="temp_file.v"
fi

NomScriptAbc="verilcirc_metrics.abc"
if [ "" != "$VERILCIRCMETRICS_PATH" ]; then
    NomScriptAbc=$VERILCIRCMETRICS_PATH+"/"+$NomScriptAbc
fi

yosys -p "read_verilog $Nomfitxer;
stat;
stat;
proc;
techmap;
stat;
stat;
abc -script $NomScriptAbc;
" > $Nomfitxer.stats
```

```
~/bin$ cat veriluoc_yosys_opt_script.sh
#!/bin/bash

Nomfitxer="$1"

Nomfitxer="$1"
if [ "" = "$Nomfitxer" ]; then
    Nomfitxer="temp_file.v"
fi

NomScriptAbc="verilcirc_metrics.abc"
if [ "" != "$VERILCIRCMETRICS_PATH" ]; then
    NomScriptAbc=$VERILCIRCMETRICS_PATH+"/"+$NomScriptAbc
fi

yosys -p "read_verilog $Nomfitxer;
stat;
stat;
proc;
techmap;
opt;
techmap;
stat;
stat;
abc -script $NomScriptAbc;
" > $Nomfitxer.stats
~/bin$
```

4.3.3. Funcionament com a eina d'anàlisi individual

Quan el programa VerilCircMetrics és invocat per efectuar una anàlisi de fitxers individuals, el funcionament és més senzill. Igual que en l'apartat anterior, tots els mètodes esmentats formen part de la classe `VerilCircMetrics`.

Els resultats no es desen a cap taula en format CSV. En el seu lloc són enviats a la consola amb un format fàcil de descompondre si alguna altra aplicació els hagués d'interpretar.

El procediment és el següent: per a cada nom d'arxiu rebut com a paràmetre, de la seva extensió es dedueix si és un fitxer CIRC o Verilog, i aleshores s'invoquen els mètodes `getResultsFromCircFile` o `getResultsFromVerilogFile` respectivament (veure el punt «e», «Obtenció del resultat a partir del fitxer CIRC», i el punt «f», «Obtenció del resultat a partir del fitxer Verilog», de l'apartat 4.3.2, «Funcionament com a eina d'anàlisi massiva» a partir de la pàgina 37.

4.3.4. Integració en altres aplicacions

La classe `VerilCircMetrics`, definida en el paquet `com.erubioman.verilog.metrics`, pot ser inclosa en qualsevol altra aplicació. Els únics elements públics que conté són el constructor i el mètode `analyzeFiles`.

a) Constructor

El constructor rep com a paràmetre un objecte que acompleixi la interfície `Loggable`. Això significa que aquest objecte haurà de disposar d'un mètode `addLog(String)` al qual es puguin enviar, en forma de `String`, tots els missatges que genera la classe `VerilCircMetrics` durant la seva execució.

Definició de la interfície `Loggable` al paquet `com.erubioman`:

```
package com.erubioman;

public interface Loggable {

    void addLog(String text);

}
...
```

Definició del constructor de la classe `VerilCircMetrics`:

```

VerilCircMetrics(Loggable _parent)
...
```

b) Mètode «analyzeFiles»

Aquest mètode engloba tota la funcionalitat del programa VerilCircMetrics. Per dur-la a terme, el mètode rep 4 paràmetres:

```
public int analyzeFiles(File csvTable, String[] exercisesDirectories, File outFile, String namesRegEx)
```

La descripció dels paràmetres és la següent:

- `csvTable` – objecte del tipus `File` representant l'arxiu on es cercarà la taula d'equivalències entre els nombres dels exercicis i els noms (veure l'apartat 4.1.4, «Format del fitxer CSV de nombres i noms d'exercicis», a la pàgina 28). Aquest paràmetre pot ser `null`, i en aquest cas no s'utilitzarà cap taula.
- `exercisesDirectories` – objecte del tipus *array* de `String` que inclou la llista dels noms de directoris a analitzar (en cas de voler executar una anàlisi massiva), o bé la llista de fitxers per fer-ne anàlisis individuals.
- `outFile` – objecte del tipus `File` representant l'arxiu on es desarà la taula de resultats en format CSV. Només s'utilitza en el cas d'anàlisis massives.
- `namesRegEx` – només utilitzat en el cas d'anàlisis massives, és una expressió regular contra la qual es compararan els nombres dels exercicis (no els noms), per analitzar únicament els fitxers que segueixin el patró. Pot ser `null` i aleshores no es fa cap filtratge d'exercicis.

El retorn de la funció és un nombre sencer indicant el nombre d'exercicis analitzats.

c) Exemple d'execució

Com a exemple, aquesta és la invocació a la classe `VerilCircMetrics` des de la classe `ConsoleLaunch` (l'encarregada d'executar el programa des de la línia de comandes). La construcció i execució de la classe `VerilCircMetrics` s'efectua en la mateixa línia:

```
public class ConsoleLaunch implements Loggable {

    public void addLog(String text) {
        System.out.print(text);
    }

    public boolean execute(String[] args) {
        ...
        new VerilCircMetrics(this).analyzeFiles(csvFile, exercisesArray, outputFile, filterString);
        ...
    }
```

Executant la crida d'aquesta manera, tota la sortida de consola generada durant l'execució del mètode `analyzeFiles` (incloent la informació de les mètriques de complexitat) es rebrà en forma de `String` mitjançant «*callbacks*» al mètode `addLog` del mateix objecte que ha realitzat la crida. Això és especialment útil si es crida el mètode `analyzeFiles` per analitzar fitxers individuals. Per obtenir una descripció detallada del significat de cadascun dels camps que es rebran, consultar l'apartat 5.1, «Resultats proporcionats pel programa» a la pàgina 54.

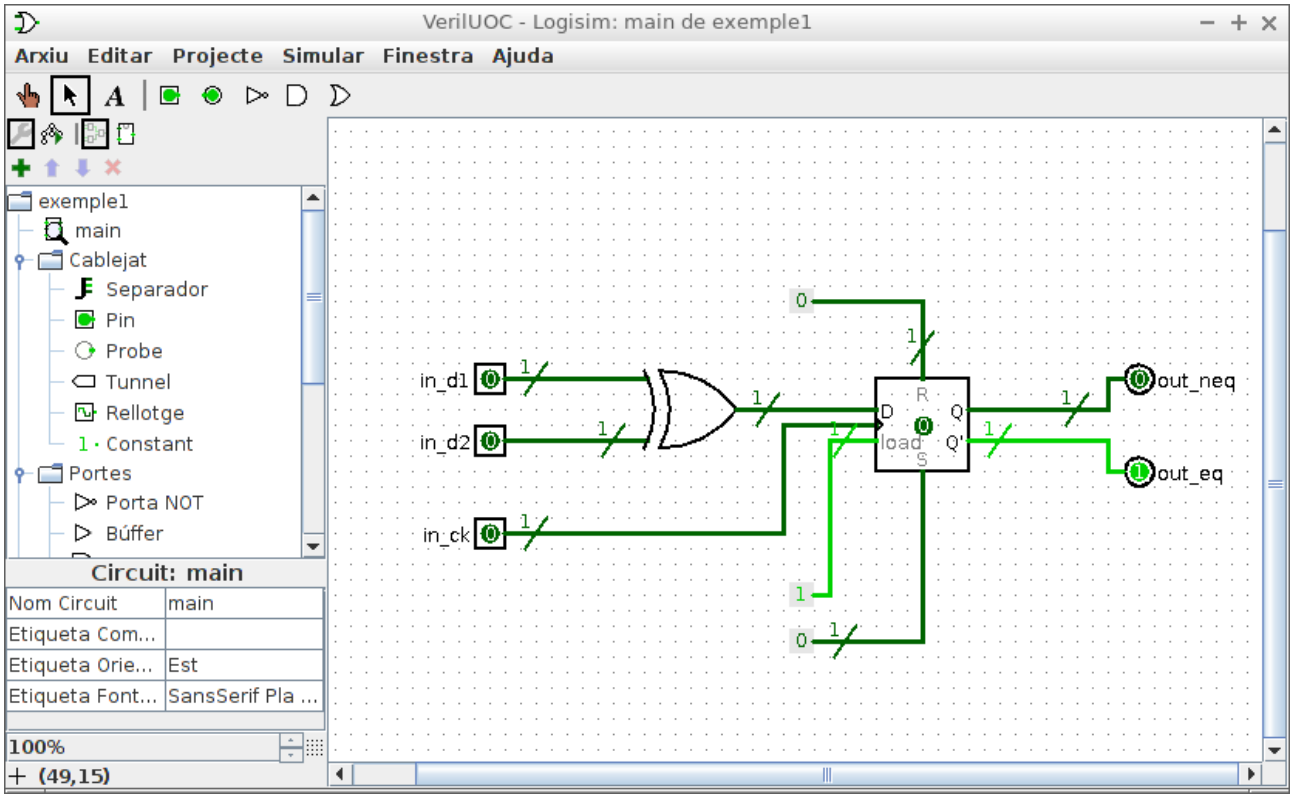
4.4. Exemple del funcionament

S'utilitzarà un disseny d'exemple del programa VerilUOC per representar els diferents processos que efectua el programa VerilCircMetrics fins arribar al resultat final de les mètriques de complexitat.

Totes les operacions que en aquest exemple es realitzen de forma manual, són les mateixes que el programa VerilCircMetrics realitza automàticament i en el mateix ordre.

4.4.1. Esquema d'exemple

S'ha creat un petit esquema molt senzill que utilitza una porta XOR i un biestable per comparar dos senyals d'entrada de forma sincronitzada amb un senyal de rellotge. Se li ha donat el nom «`exemple1.circ`»:



4.4.2. Conversió a Verilog

Per poder calcular les mètriques de complexitat del circuit, el primer pas és convertir l'esquema al format Verilog mitjançant el programa CircParser:

```
$ java -jar ~/bin/CircParser.jar exemple1.circ
Obrint fitxer CIRC VerilUOC: exemple1.circ.
Desant fitxer Verilog: exemple1.circ.v.
$
```

Així es crea el fitxer «`exemple1.circ.v`», que és en format Verilog però amb les particularitats pròpies del programa CircParser, que fan que encara no sigui apte per ser llegit pel programa Yosys. De fet, si s'intenta el resultat és aquest:

```
$ yosys
...
yosys> read_verilog exemple1.circ.v
1. Executing Verilog-2005 frontend.
Parsing Verilog input from `exemple1.circ.v' to AST representation.
ERROR: Parser error in line exemple1.circ.v:1: syntax error
$
```

Caldrà una passa prèvia que es realitza mitjançant el programa ParsedVerilogProcessor. Els paràmetres enviats al programa són els mateixos que utilitza internament el programa VerilCircMetrics:

```
$ java -jar ~/bin/ParsedVerilogProcessor.jar -s -w exemple1.circ.v
Convertint fitxer exemple1.circ.v a exemple1.circ.v.v...
Canviant nom del mòdul: m_exemple1_circ_v
Ordenant dependències dels elements wire.
Eliminant elements wire redundants.
Desat amb èxit.
$
```

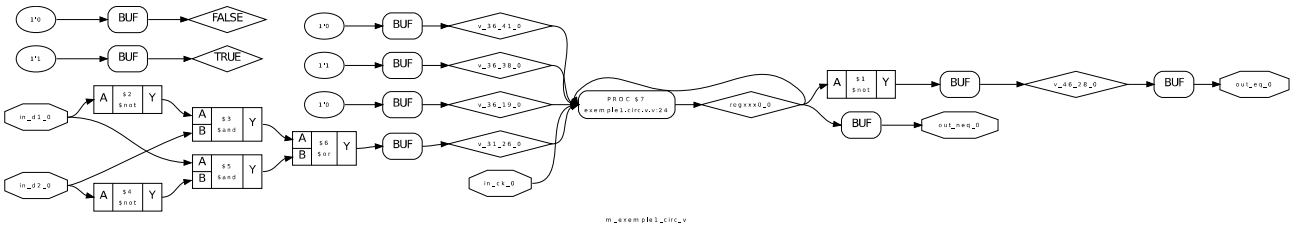
Això genera un arxiu «`exemple1.circ.v.v`» que ja pot llegir-se des del programa Yosys.

4.4.3. Primera passada: anàlisi del circuit

El programa Yosys és una eina de línia de comandes que permet executar seqüències programades d'ordres mitjançant *scripts*. Així és com l'invoca el programa VerilCircMetrics, però per aquesta prova les ordres s'introduiran manualment:

```
$ yosys
...
yosys> read_verilog exemple1.circ.v.v
1. Executing Verilog-2005 frontend.
Parsing Verilog input from `exemple1.circ.v.v' to AST representation.
Generating RTLIL representation for module `\m_exemple1_circ_v'.
Successfully finished Verilog frontend.

yosys>
```

Si en aquest moment s'introdueix la comanda «`show`» s'obté un diagrama de la representació interna de l'esquema en el programa Yosys:



En el centre de l'esquema es pot veure un element «`process`», que correspon a un bloc «`always`» del fitxer Verilog. Aquest element fa que, de moment, aquest esquema no sigui apropiat per enviar al programa Berkeley-ABC per a la seva anàlisi. Per adequar-lo, caldrà executar a continuació la comanda «`proc`» del programa Yosys, que s'encarrega de convertir els processos en combinacions d'elements biestables i multiplexors de dues entrades.

Malauradament, algunes formes de connectar els components d'un esquema (per exemple, unir les entrades «`clock`» i «`load`» d'un biestable) poden provocar que la comanda «`proc`» de Yosys falli i el programa avorti precipitadament. Els dissenys en què això passi no podran processar-se amb ABC, i per tant no se'n podran obtenir les mètriques que aquest programa proporciona. De fet, ni tan sols podran obtenir-se les mètriques que proporciona Yosys després de convertir els processos a biestables, ja que és precisament aquesta conversió la que falla.

Per aquest motiu, en aquest punt el programa VerilCircMetrics efectua una primera crida a la comanda «`stat`» per obtenir les estadístiques del circuit abans de fer-li cap modificació. En cas que més endavant la comanda «`proc`» fallés, aquestes estadístiques serien les úniques disponibles. És per aquest motiu que entre tots els circuits subministrats com a exemple, n'hi ha alguns (afortunadament una minoria) que presenten un nombre de processos diferent de zero a la taula CSV resultant.

```
yosys> stat

3. Printing statistics.

=== m_exemple1_circ_v ===

   Number of wires:                 20
   Number of wire bits:             20
   Number of public wires:          13
   Number of public wire bits:      13
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              1
   Number of cells:                  6
     $and                            2
     $not                            3
     $or                             1



yosys> stat

4. Printing statistics.
...
```

El programa VerilCircMetrics sempre executa les ordres «`stat`» dos cops seguits. Això té l'única finalitat de facilitar la lectura de la informació utilitzant el text «`Printing statistics`» com a delimitador inicial i final.

Pot veure's que Yosys ja ha fet una comptabilització del nombre de connexions i també ha fet una reducció a portes simples de tota la lògica combinacional que queda fora dels processos.

Les dades obtingudes ja són memoritzades per VerilCircMetrics però seran sobreescrites més tard si funciona la comanda «`proc`», que s'efectua a continuació:

```
yosys> proc
5. Executing PROC pass (convert processes to netlists).

5.1. Executing PROC_CLEAN pass (remove empty switches from decision trees).
Cleaned up 0 empty switches.

5.2. Executing PROC_RMDEAD pass (remove dead branches from decision trees).
Removed a total of 0 dead cases.

5.3. Executing PROC_INIT pass (extract init attributes).

5.4. Executing PROC_ARST pass (detect async resets in processes).
Found async reset \v_36_19_0 in `\m_exemple1_circ_v.$proc$exemple1.circ.v.v:24$7'.
Found async reset \v_36_41_0 in `\m_exemple1_circ_v.$proc$exemple1.circ.v.v:24$7'.

5.5. Executing PROC_MUX pass (convert decision trees to multiplexers).
Creating decoders for process `\m_exemple1_circ_v.$proc$exemple1.circ.v.v:24$7'.
  creating decoder for signal `$0\regxxx0_0[0:0]'.

5.6. Executing PROC_DLATCH pass (convert process syncs to latches).

5.7. Executing PROC_DFF pass (convert process syncs to FFs).
Creating register for signal `\m_exemple1_circ_v.\regxxx0_0' using process `\m_exemple1_circ_v.
$proc$exemple1.circ.v.v:24$7'.
Warning: Complex async reset for dff `\regxxx0_0'.
  created $dffsr cell `$procdff$26' with positive edge clock and multiple level-sensitive resets.
```

```
5.8. Executing PROC_CLEAN pass (remove empty switches from decision trees).
Found and cleaned up 2 empty switches in `\m_exemple1_circ_v.$proc$exemple1.circ.v.v:24$7'.
Removing empty process `m_exemple1_circ_v.$proc$exemple1.circ.v.v:24$7'.
Cleaned up 2 empty switches.

yosys>
```

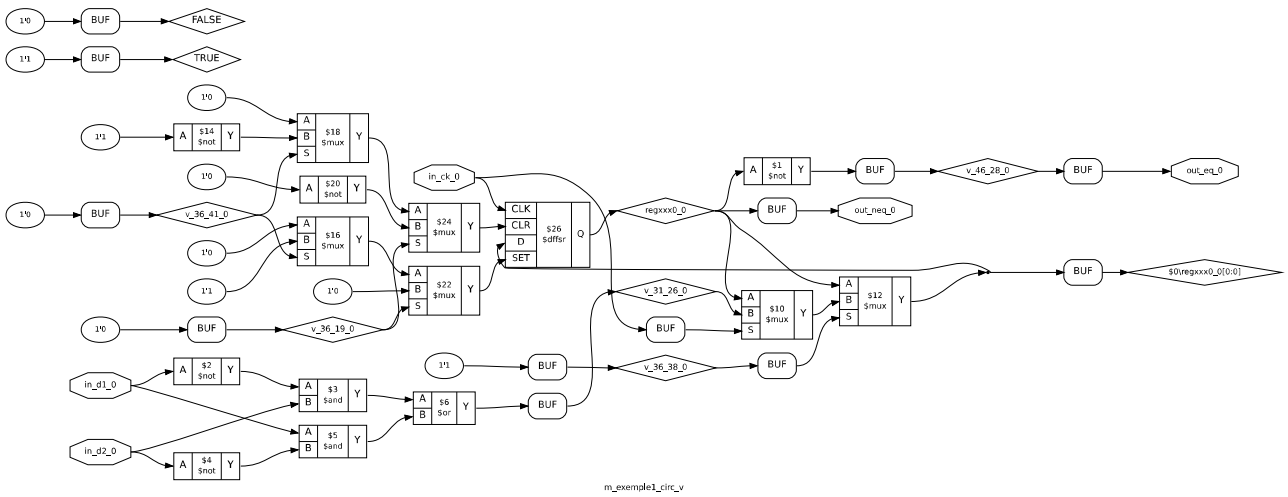
La comanda «`proc`» de Yosys agrupa de forma seqüencial una sèrie de passes destinades a transformar tots els processos del disseny (blocs «`always`»), en multiplexors i biestables del tipus *flip-flop* i *latch*. Cadascuna de les passes es correspon a una comanda més simple que també podria executar-se per separat si fos necessari. Són les següents:

- `proc_clean` – Elimina les parts buides dels processos i en darrer terme elimina el procés sencer si aquest només conté estructures buides.
- `proc_rmdead` – Identifica les branques de codi que mai arriben a executar-se, i les elimina.
- `proc_init` – Extreu les accions inicials dels processos (generades pels blocs «`initial`» del codi Verilog) i les trasllada als atributs «`init`» dels elements «`wire`» respectius.
- `proc_arst` – Identifica els *reset* asíncrons en els processos i els converteix en una representació interna diferent, que és adequada per generar cel·les del tipus *flip-flop* amb *reset* asíncron.
- `proc_mux` – Identifica els arbres de decisió dels processos (originats per les estructures `if-else` i `case`) i els converteix en arbres de cel·les del tipus multiplexor.
- `proc_dlatch` – Identifica elements *latch* en els processos i els converteix en cel·les del tipus *D-latch*.
- `proc_dff` – Identifica elements *flip-flop* en els processos i els converteix en cel·les del tipus *flip-flop*.
- `proc_clean` – Executa per segona vegada la mateixa passa amb què s'havia iniciat la seqüencia.

Després d'executar la comanda «proc», en el disseny actual ja no existeixen processos i han estat substituïts per biestables i multiplexors. Això és pot veure mitjançant l'ordre «stat» (tanmateix, el programa VerilCircMetrics no llegeix estadístiques en aquest punt):

```
yosys> stat

5. Printing statistics.

=== m_exemple1_circ_v ===

   Number of wires:                 30
   Number of wire bits:             30
   Number of public wires:          13
   Number of public wire bits:      13
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 15
     $and                            2
     $dffsr                          1
     $mux                            6
     $not                            5
     $or                             1


yosys>
```

La representació interna de l'esquema té aquest aspecte:



En aquest punt l'esquema encara no està preparat per ser enviat a ABC. Encara és necessari executar la comanda «techmap»:

```
yosys> techmap

7. Executing TECHMAP pass (map to technology primitives).

7.1. Executing Verilog-2005 frontend.
Parsing Verilog input from `<techmap.v>' to AST representation.
Generating RTLIL representation for module `\_90_simplemap_bool_ops'.
Generating RTLIL representation for module `\_90_simplemap_reduce_ops'.
Generating RTLIL representation for module `\_90_simplemap_logic_ops'.
Generating RTLIL representation for module `\_90_simplemap_compare_ops'.
Generating RTLIL representation for module `\_90_simplemap_various'.
Generating RTLIL representation for module `\_90_simplemap_registers'.
Generating RTLIL representation for module `\_90_shift_ops_shr_shl_sshl_sshr'.
Generating RTLIL representation for module `\_90_shift_shiftx'.
Generating RTLIL representation for module `\_90_fa'.
Generating RTLIL representation for module `\_90_lcu'.
Generating RTLIL representation for module `\_90_alu'.
Generating RTLIL representation for module `\_90_macc'.
Generating RTLIL representation for module `\_90_alumacc'.
```

```
Generating RTLIL representation for module `\$__div_mod_u'.
Generating RTLIL representation for module `\$__div_mod'.
Generating RTLIL representation for module `\_90_div'.
Generating RTLIL representation for module `\_90_mod'.
Generating RTLIL representation for module `\_90_pow'.
Generating RTLIL representation for module `\_90_pmux'.
Generating RTLIL representation for module `\_90_lut'.
Successfully finished Verilog frontend.
Mapping m_exemple1_circ_v.$not$exemple1.circ.v.v:11$1 ($not) with simplemap.
Mapping m_exemple1_circ_v.$not$exemple1.circ.v.v:17$2 ($not) with simplemap.
Mapping m_exemple1_circ_v.$and$exemple1.circ.v.v:17$3 ($and) with simplemap.
Mapping m_exemple1_circ_v.$not$exemple1.circ.v.v:17$4 ($not) with simplemap.
Mapping m_exemple1_circ_v.$and$exemple1.circ.v.v:17$5 ($and) with simplemap.
Mapping m_exemple1_circ_v.$or$exemple1.circ.v.v:17$6 ($or) with simplemap.
Mapping m_exemple1_circ_v.$procmux$10 ($mux) with simplemap.
Mapping m_exemple1_circ_v.$procmux$12 ($mux) with simplemap.
Mapping m_exemple1_circ_v.$auto$proc_dff.cc:105:gen_dffsr_complex$14 ($not) with simplemap.
Mapping m_exemple1_circ_v.$auto$proc_dff.cc:112:gen_dffsr_complex$16 ($mux) with simplemap.
Mapping m_exemple1_circ_v.$auto$proc_dff.cc:119:gen_dffsr_complex$18 ($mux) with simplemap.
Mapping m_exemple1_circ_v.$auto$proc_dff.cc:105:gen_dffsr_complex$20 ($not) with simplemap.
Mapping m_exemple1_circ_v.$auto$proc_dff.cc:112:gen_dffsr_complex$22 ($mux) with simplemap.
Mapping m_exemple1_circ_v.$auto$proc_dff.cc:119:gen_dffsr_complex$24 ($mux) with simplemap.
Mapping m_exemple1_circ_v.$procdff$26 ($dffsr) with simplemap.
No more expansions possible.

yosys>
```

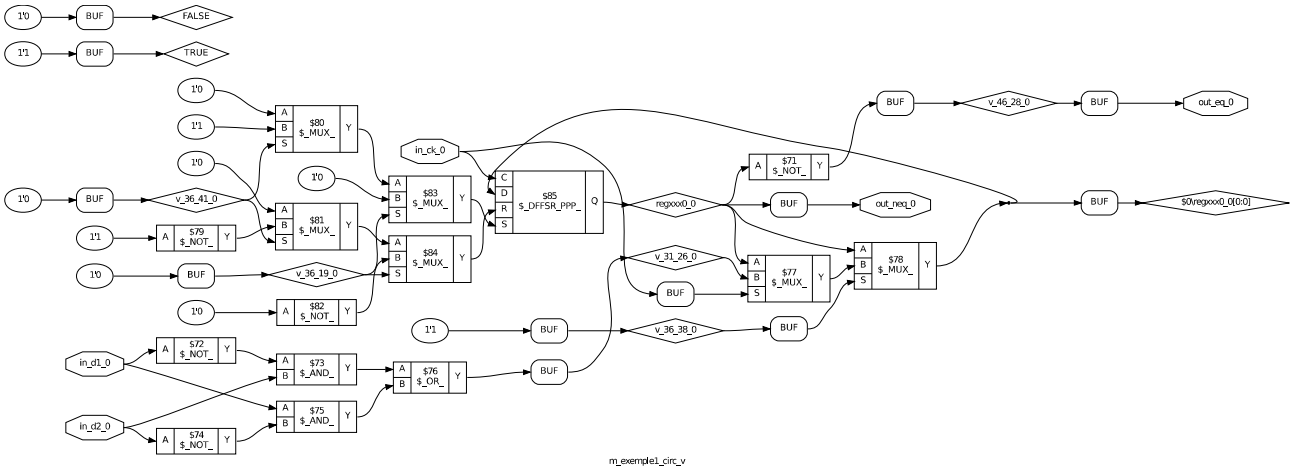
La comanda «techmap» executa un *technology mapper* molt simple [CLW137] que substitueix les cel·les del disseny amb implementacions que se li hagin donat en forma de codi font en format Verilog o *ilang* (per obtenir més detalls sobre el procés de *technology mapping*, pot consultar-se l'apartat 4.4.5, «Cerca de la millor solució amb technology mapping», a la pàgina 52).

En aquest punt l'esquema ja està preparat per enviar al programa ABC. Abans de fer-ho, el programa VerilCircMetrics realitza una lectura d'estadístiques. Els valors que s'obtenen sobreescriuen la primera lectura que s'havia realitzat.

```
yosys> stat

7. Printing statistics.

=== m_exemple1_circ_v ===

   Number of wires:                 30
   Number of wire bits:             30
   Number of public wires:          13
   Number of public wire bits:      13
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 15
     $_AND_                          2
     $_DFFSR_PPP_                    1
     $_MUX_                          6
     $_NOT_                          5
     $_OR_                           1


yosys> stat
8. Printing statistics.
...
```

Pot observar-se que en l'apartat «cells» no apareix el mateix tipus de símbols que en la primera visualització. Ara no es tracta de primitives teòriques sinó de components reals de la llibreria de Yosys per defecte, sobre els quals s'ha aplicat un *technology mapping*. El programa Yosys permet de redefinir aquesta llibreria per adaptar-la a tecnologies concretes o, fins i tot, als mateixos components de VerilUOC tal com s'explica a l'apartat 4.4.5, «Cerca de la millor solució amb technology mapping» a la pàgina 52.

Executant una nova comanda «show» es pot veure l'aspecte actual del circuit en la representació interna de Yosys:

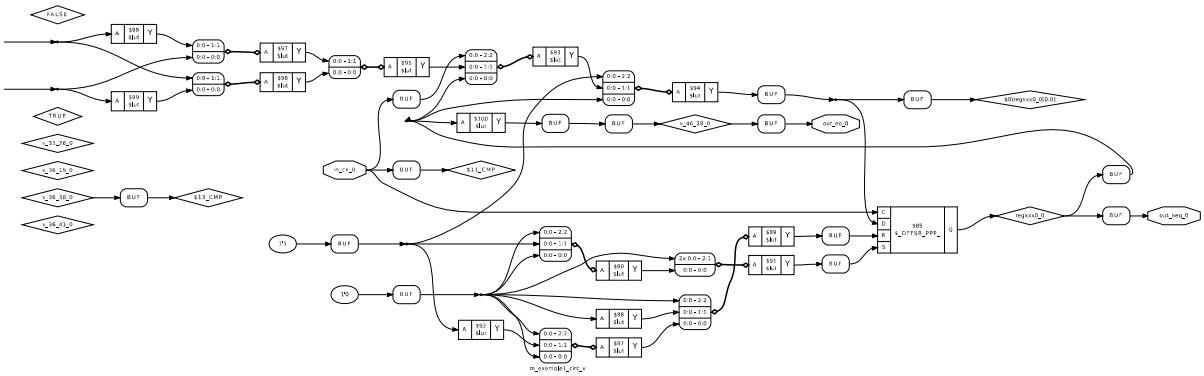


L'última passa és executar la comanda «abc», tot passant-li un paràmetre per tal que s'executi un *script* anomenat «verilcirc_metrics.abc»:

```
yosys> abc -script verilcirc_metrics.abc

10. Executing ABC pass (technology mapping using ABC).

10.1. Extracting gate netlist of module `\m_exemple1_circ_v' to `<abc-temp-dir>/input.blif'..
Extracted 14 gates and 20 wires to a netlist network with 4 inputs and 4 outputs.

10.1.1. Executing ABC.
Running ABC command: <yosys-exe-dir>/berkeley-abc -s -f <abc-temp-dir>/abc.script 2>&1
ABC: ABC command line: "source <abc-temp-dir>/abc.script".
ABC:
ABC: + read_blif <abc-temp-dir>/input.blif
ABC: + read_library <abc-temp-dir>/stdcells.genlib
ABC: Entered genlib library with 15 gates from file "<abc-temp-dir>/stdcells.genlib".
ABC: + source ~/Brossa/Analisi/verilcirc_metrics.abc
ABC: netlist       : i/o =    4/    4  lat =    0  nd =    16  edge =     29  cube =    21  lev = 5
ABC: + write_blif <abc-temp-dir>/output.blif

10.1.2. Re-integrating ABC results.
ABC RESULTS:              $lut cells:       14
ABC RESULTS:        internal signals:       12
ABC RESULTS:           input signals:        4
ABC RESULTS:          output signals:        4
Removing temp directory.

yosys>
```

Malgrat que la finalitat de la comanda «abc» de Yosys és efectuar (mitjançant el programa extern Berkeley-ABC) un *technology mapping* de la llibreria de portes interna de Yosys cap a una arquitectura de destí, en aquest punt l'únic que es vol obtenir són les mètriques del circuit que ABC pot proporcionar. Per aquest motiu el *script* executat inclou la comanda «print_stats» que mostra la informació que es pot observar en la consola. La línia clau d'aquesta informació és la que comença amb el text «ABC: netlist», i que inclou el nombre de nodes (el camp «nd») i el nombre de nivells (camp «lev»). Aquests dos camps fan referència a l'estructura de nodes AIG (*and-inverter gates* o portes AND inversores) amb què treballa internament el programa ABC, i tenen correlació amb l'àrea i el retard màxim del circuit, respectivament, per tant s'afegeixen al fitxer de sortida en format CSV.

Una crida a la comanda «show» permet de veure l'efecte que ha tingut sobre el circuit l'execució de la comanda «abc»: tota la lògica combinacional ha quedat reduïda a *look up tables* (LUT):



Això ja no és rellevant per al programa VerilCircMetrics perquè tota la informació necessària ja s'ha obtingut amb les comandes anteriors.

4.4.4. Segona passada: optimització

Per calcular la «millor solució possible» a l'exercici proposat, el programa VerilCircMetrics s'aprofita de les funcions d'optimització que proporciona el programa Yosys. Aquestes funcions eliminen tota la lògica sobrant o redundant.

Per tornar a repetir les passes des del començament, es pot buidar la memòria del programa Yosys mitjançant l'ordre «delete», o bé es pot sortir mitjançant l'ordre «exit» i tornar a invocar-lo des de la línia de comandes. Aquesta segona invocació és l'opció triada pel programa VerilCircMetrics. D'aquesta manera s'assegura que la segona passada s'efectua també en aquelles ocasions en què el programa Yosys hagi avortat per culpa d'un error durant la comanda «proc».

Les primers ordres són idèntiques a les de la primera passada, amb la diferència que després de l'ordre «techmap» s'executa una ordre «opt», i després un altre cop l'ordre «techmap».

```
$ yosys
...

yosys> read_verilog exemple1.circ.v.v
...

yosys> stat
...

yosys> stat
...

yosys> proc
...

yosys> techmap
...

yosys> opt

6. Executing OPT pass (performing simple optimizations).

6.1. Executing OPT_CONST pass (perform const folding).
```

```
Replacing $_MUX_ cell `$auto$simplemap.cc:277:simplemap_mux$78' (??1) in module `\m_exemple1_circ_v'
with constant driver `$procmux$12_Y = $procmux$10_Y'.
Replacing $_NOT_ cell `$auto$simplemap.cc:37:simplemap_not$79' (1) in module `\m_exemple1_circ_v' with
constant driver `$auto$proc_dff.cc:110:gen_dffsr_complex$15 = 1'0'.
Replacing $_MUX_ cell `$auto$simplemap.cc:277:simplemap_mux$80' (010) in module `\m_exemple1_circ_v'
with constant driver `$auto$proc_dff.cc:117:gen_dffsr_complex$17 = 1'0'.
Replacing $_MUX_ cell `$auto$simplemap.cc:277:simplemap_mux$81' (000) in module `\m_exemple1_circ_v'
with constant driver `$auto$proc_dff.cc:124:gen_dffsr_complex$19 = 1'0'.
Replacing $_NOT_ cell `$auto$simplemap.cc:37:simplemap_not$82' (0) in module `\m_exemple1_circ_v' with
constant driver `$auto$proc_dff.cc:110:gen_dffsr_complex$21 = 1'1'.
Replacing $_MUX_ cell `$auto$simplemap.cc:277:simplemap_mux$83' (000) in module `\m_exemple1_circ_v'
with constant driver `$auto$proc_dff.cc:117:gen_dffsr_complex$23 = 1'0'.
Replacing $_MUX_ cell `$auto$simplemap.cc:277:simplemap_mux$84' (010) in module `\m_exemple1_circ_v'
with constant driver `$auto$proc_dff.cc:124:gen_dffsr_complex$25 = 1'0'.

6.2. Executing OPT_SHARE pass (detect identical cells).
Finding identical cells in module `\m_exemple1_circ_v'.
Removed a total of 0 cells.

6.3. Executing OPT_MUXTREE pass (detect dead branches in mux trees).
Running muxtree optimizier on module \m_exemple1_circ_v..
  Creating internal representation of mux trees.
  No muxes found in this module.
Removed 0 multiplexer ports.

6.4. Executing OPT_REDUCE pass (consolidate $*mux and $reduce_* inputs).
  Optimizing cells in module \m_exemple1_circ_v.
Performed a total of 0 changes.

6.5. Executing OPT_SHARE pass (detect identical cells).
Finding identical cells in module `\m_exemple1_circ_v'.
Removed a total of 0 cells.

6.6. Executing OPT_RMDFF pass (remove dff with constant values).
Replaced 0 DFF cells.

6.7. Executing OPT_CLEAN pass (remove unused cells and wires).
Finding unused cells or wires in module \m_exemple1_circ_v..
  removing unused non-port wire \v_36_41_0.
  removing unused non-port wire \v_36_38_0.
  removing unused non-port wire \v_36_19_0.
  removing unused non-port wire \TRUE.
  removing unused non-port wire \FALSE.
  removed 5 unused temporary wires.

6.8. Executing OPT_CONST pass (perform const folding).

6.9. Finished OPT passes. (There is nothing left to do.)

yosys> techmap

7. Executing TECHMAP pass (map to technology primitives).
...

yosys>
```

La comanda «opt» efectua optimitzacions i neteges simples mitjançant l'execució en seqüència d'una sèrie de comandes que també podrien executar-se per separat:

- `opt_expr` – Reescriptura simple d'expressions. Aquesta passada efectua propagació de valors constants en les cel·les internes que tenen entrades constants.
- `opt_merge` – Identifica cel·les d'idèntic tipus que tenen senyals d'entrada idèntics. Aleshores les agrupa en una única cel·la.
- `opt_muxtree` – Analitza els senyals de control dels arbres de multiplexors del disseny, i després identifica les entrades que mai poden arribar a activar-se. Aleshores elimina aquestes branques mortes dels arbres de multiplexors.

- `opt_reduce` – Simplificació de components grans del tipus AND, OR i multiplexors. Aquesta passada efectua dues optimitzacions entrellaçades: d'una banda, consolida els arbres de portes AND i OR grans eliminant-ne les entrades duplicades; de l'altra, identifica les entrades duplicades dels multiplexors i les substitueix per una única entrada amb els senyals de control originals units per una operació OR.
- `opt_merge` – Repetició de la segona passa, ja comentada.
- `opt_rmdff` – Identifica els elements *flip-flop* amb entrades constants i les substitueix per un element *driver* constant.
- `opt_clean` – Identifica les connexions i cel·les que no s'utilitzen i les elimina. Altres passes sovint eliminen cel·les però conserven els cables, o bé reconnecten els cables però conserven les antigues cel·les en el disseny. Aquesta passa s'utilitza normalment per fer neteja després que altres passes hagin fet el veritable treball.
- `opt_expr` – Repetició de la primera passa, ja comentada.

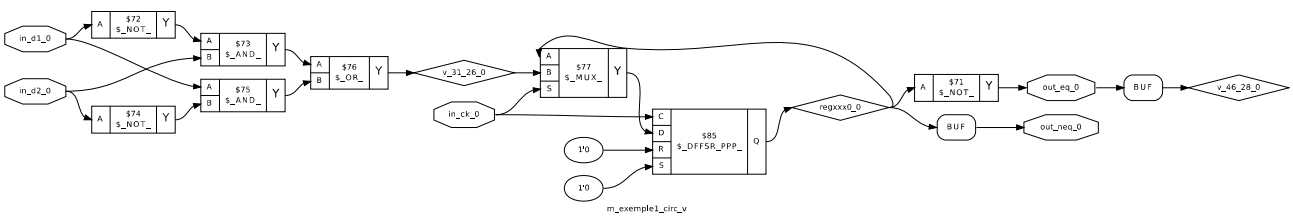
El bloc comprès des de la tercera passa («`opt_muxtree`») fins al final («`opt_expr`») es va repetint fins que Yosys detecta que ja no s'estan produint més canvis en el disseny.

Després d'haver executat «`opt`» i el segon «`techmap`», una crida a «`stat`» mostra la diferència entre aquesta passada i l'anterior:

```
yosys> stat

8. Printing statistics.

=== m_exemple1_circ_v ===

   Number of wires:                 13
   Number of wire bits:             13
   Number of public wires:           8
   Number of public wire bits:       8
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                  8
     $_AND_                          2
     $_DFFSR_PPP_                    1
     $_MUX_                          1
     $_NOT_                          3
     $_OR_                           1


yosys> stat
9. Printing statistics.
...
```

S'observa, en comparació amb la primera passada, que el nombre de cel·les s'ha reduït de 15 a 8 (és a dir, poc més de la meitat) i que en concret el nombre de cel·les MUX (les més complexes a part del biestable) s'ha reduït de 6 a 1. Són dades que s'incorporaran al fitxer de sortida en CSV.

Una nova execució de «`show`» mostra la representació actual del circuit:



Finalment, l'execució de la comanda «abc» també mostra una reducció de les mètriques. El nombre de nodes s'ha reduït de 16 a 7, i el nombre de nivells s'ha reduït de 5 a 4. Aquestes dades també aniran a parar al fitxer CSV de sortida.

```
yosys> abc -script verilcirc_metrics.abc
...
ABC: netlist       : i/o =    4/    2  lat =    0  nd =     7  edge =     12  cube =     9  lev = 4
ABC: + write_blif <abc-temp-dir>/output.blif

11.1.2. Re-integrating ABC results.
ABC RESULTS:              $lut cells:        7
ABC RESULTS:        internal signals:        5
ABC RESULTS:           input signals:        4
ABC RESULTS:          output signals:        2
Removing temp directory.

yosys>
```

Aquí acaben totes les operacions que realitza VerilCircMetrics amb l'ajuda del programa Yosys.

4.4.5. Cerca de la millor solució amb technology mapping

Technology mapping és el procés de convertir el disseny en un connexionat amb les cel·les pròpies de l'arquitectura en què es vol desplegar [CLW068].

a) Substitució de cel·les

L'ordre `techmap` de Yosys efectua una «substitució de cel·les», que és la forma més simple de *technology mapping*. Aquesta ordre pot adjuntar-se, opcionalment, amb un fitxer Verilog que inclogui definicions de cel·les simples, i aleshores el programa Yosys substitueix la seva representació interna en RTL per una de nova amb les cel·les simples del fitxer Verilog.

Si no s'adjunta cap fitxer Verilog de definicions, Yosys utilitzarà una llibreria per defecte amb definicions de portes simples, multiplexors i biestables. En gran mesura aquesta llibreria ja és adequada per representar els esquemes de VerilUOC i dóna un bon resultat.

En el context d'aquest projecte s'ha intentat de substituir la llibreria per defecte per una altra que reflecteixi més exactament els components de VerilUOC. La manca de temps no ha permès arribar al final d'aquesta via i obtenir-ne resultats. Podria ser un tema interessant per a futures investigacions.

b) Substitució de subcircuits

Per a arquitectures de destinació que posseeixen cel·les més complexes, Yosys proporciona l'ordre `extract` que pot contrastar un conjunt de mòduls contra un disseny, i identificar les parts del disseny que són idèntiques a algun dels mòduls del conjunt. Aleshores les parts identificades («subcircuits») són substituïdes per instàncies dels mòduls.

L'ordre `extract` també pot trobar variacions bàsiques dels mòduls subministrats, com per exemple entrades intercanviades en cel·les que tinguin la propietat commutativa.

c) Technology mapping a nivell de portes

En aquest nivell l'arquitectura de destí normalment es descriu mitjançant un fitxer «*Liberty*», que és un format estàndard de la indústria que s'utilitza per descriure el comportament i altres propietats de les cel·les de llibreries estàndard.

Per adaptar la lògica combinacional a l'arquitectura de destí a nivell de portes, des de Yosys es pot invocar el programa Berkeley-ABC enviant com a paràmetre una llibreria en format «*Liberty*».

El coneixement a fons de totes aquestes opcions s'escapa a la dimensió d'aquest projecte i queda per veure quin profit se'n podria obtenir de a cara millorar el càlcul de les mètriques de complexitat dels circuits.

Capítol 5. Conclusions i obtenció de resultats

5.1. Resultats proporcionats pel programa

El programa VerilCircMetrics compleix el requisit d'obtenir les mètriques de complexitat dels dissenys en format CIRC que se li subministren, i desar els resultats en un arxiu fàcil de processar. Hauria estat interessant poder aprofundir en la tècnica del *technology mapping* per poder obtenir la solució òptima basada en els propis components de VerilUOC. La potència dels programes Yosys i Berkeley-ABC suggereixen continuar investigant en aquesta direcció. Malauradament el poc temps i la complexitat del tema han obligat a triar una solució més simple i obtenir una optimització amb les portes simples de la llibreria del programa Yosys.

5.1.1. Format del fitxer CSV resultant

El fitxer CSV generat conté el resultat de l'anàlisi dels fitxers distribuït en les següents columnes:

- «`ExerciseNumber`»: és el nombre del exercici. Si es tracta de l'exercici d'un alumne, el nombre es dedueix a partir del nom del fitxer. Si es tracta de l'exercici del professor, el nombre es dedueix a partir del fitxer CSV d'equivalències de nombres i noms.
- «`UserName`» – és el nom de l'autor de l'exercici. Si es tracta de l'exercici d'un alumne, el nom es dedueix a partir del nom del fitxer. Si es tracta de l'exercici del professor, el programa genera automàticament el nom «`0_Teacher`». El fet que comenci per zero té la finalitat de situar-lo al començament quan s'ordenen les dades alfabèticament.
- «`Semester`», «`Try`», «`Date`», «`Time`» – són el semestre de l'exercici, el nombre de l'intent, la data i l'hora. Només tenen sentit per als exercicis dels alumnes. Queden en blanc si l'exercici ha estat generat pel professor.
- «`ExerciseName`» – el nom de l'exercici. En els exercicis realitzats pel professor, és igual al nom del fitxer eliminant-ne l'extensió. En els exercicis realitzats pels alumnes, el nom s'obté a partir del nombre d'exercici i del fitxer CSV d'equivalències de nombres i noms.
- «`CircPath`», «`VerilogPath`» – són les rutes completes on s'han trobat els fitxers CIRC i Verilog corresponents a aquest exercici. No és obligatori que tots dos hi siguin, però sí com a mínim un d'ells.
- «`TeacherSolutions`» – el nombre de solucions proposades pel professor per a aquest mateix nom d'exercici. Aquest camp només pot tenir els valors 0 o 1.
- «`StudentSolutions`» – el nombre de solucions proposades pels alumnes per a aquest mateix nombre d'exercici.

- «YosysWires» – mètrica de complexitat retornada pel programa Yosys representant el nombre de connexions individuals en el circuit.
- «YosysProcesses» – mètrica de complexitat retornada pel programa Yosys indicant el nombre de processos en el circuit (blocs «always»). Aquest valor hauria de ser normalment zero, atès que el procés intern del programa Yosys intenta convertir els processos en conjunts de biestables i multiplexors. En una minoria de casos això no és possible (veure explicació en l'apartat 5.2.2, «Fitxers que provoquen errors a la comanda «proc» de Yosys» a la pàgina 58).
- «YosysGates» – mètrica de complexitat retornada pel programa Yosys indicant el nombre de portes simples equivalents al circuit complet. Per defecte Yosys, en el moment de carregar el fitxer Verilog, redueix a primitives del tipus AND, OR i inversors totes les parts del disseny que estan situades fora dels processos (blocs «always»). Després de l'ordre «proc» (veure'n la descripció detallada a la pàgina 45) s'eliminen els processos i en el seu lloc s'afegeixen biestables D i multiplexors de 2 entrades. La columna «YosysGates» representa la suma dels 5 tipus de primitives, que també apareixen desglossades en les 5 columnes següents: «D_FlipFlops», «MuxGates», «InvGates», «AndGates» i «OrGates».
- «AbcNodes» – mètrica de complexitat retornada pel programa Berkeley-ABC indicant el nombre de nodes AIG (*and-inverter gates*, és a dir portes AND inversores) equivalents al circuit complet. És una mesura orientativa de l'àrea, basada en la reducció del circuit a les primitives més simples possibles.
- «AbcLevels» – mètrica de complexitat retornada pel programa Berkeley-ABC indicant la llargada de la major cadena de nodes AIG. De totes les mètriques obtingudes, aquesta és l'única que pot donar una orientació sobre el retard màxim del circuit.
- «OptYosysWires», «OptYosysProcesses», «OptYosysCells», «OptD_FlipFlops», «OptMuxGates», «OptInvGates», «OptAndGates», «OptOrGates», «OptAbcNodes» i «OptAbcLevels» – aquestes 10 columnes són una rèplica de les 10 anteriors amb el prefix «Opt», i corresponen a les mateixes mètriques però calculades després d'un procés d'optimització efectuat pel programa Yosys (la seva opció «opt», veure'n l'explicació detallada a la pàgina 50). Aquestes optimitzacions produeixen un resultat diferent per a cada solució d'un mateix exercici, de manera que seleccionant la millor de totes elles es pot aconseguir el més semblant a la «millor solució possible» que serveixi de patró de comparació per avaluar la resta.
- «ResultType» – pot tenir tres valors: «Circ» (el desitjable i més freqüent) si els resultats de l'exercici s'han pogut obtenir a partir del fitxer CIRC subministrat; «Verilog» si ha calgut utilitzar el fitxer Verilog perquè ha fallat la conversió a partir del CIRC; «None» si no s'ha pogut analitzar el circuit amb cap dels dos sistemes (per exemple, si l'arxiu CIRC no es pot convertir i a més l'arxiu Verilog conté alguna particularitat que impedeix que el programa Yosys pugui obrir-lo).

5.1.2. Temps d'execució

En un ordinador Core i5-3300 (4 nuclis a 3 GHz) s'han pogut analitzar 7542 fitxers en 46 minuts i 53 segons, la qual cosa suposa una mitjana de 0,373 segons per fitxer.

5.2. Incidències en les proves.

5.2.1. Fitxers que no s'han pogut convertir de format CIRC a Verilog

L'objectiu del programa VerilCircMetrics és poder realitzar mètriques de complexitat a partir de qualsevol esquema en format CIRC. Per poder fer-ho és necessari que pugui convertir per ell mateix tots els fitxers CIRC a format Verilog. Això s'ha pogut fer en una gran majoria dels casos, però n'hi ha una minoria amb la qual no ha estat possible per un conjunt de causes. Afortunadament es tracta de fitxers corresponents a versions antigues del programa, anteriors a l'any 2012, i no s'ha observat cap problema per gestionar fitxers més nous. Per tant, no s'espera patir aquesta mena de problemes amb fitxers CIRC generats amb versions actuals.

Dels 7536 fitxers tractats, 7433 han pogut ser analitzats directament a partir del seu original CIRC, mentre que els 109 restants han hagut de ser analitzats a partir del seu fitxer Verilog equivalent, ja que la conversió de CIRC a Verilog generava un «`error_3`» que no s'ha pogut resoldre (hi ha una explicació del que significa aquest error a l'apartat 3.1.3, «Diferències entre versions dels fitxers CIRC», a la pàgina 16). Tots els fitxers afectats són antics i tenen en comú la següent capçalera:

```
<project version="1.0" source="2.3.2">
```

Tanmateix, no tots els fitxers amb aquesta capçalera (179 en total) presenten aquest problema. Els 7357 fitxers CIRC restants, que s'han pogut convertir sense excepció, tenen aquesta altra capçalera:

```
<project source="2.7.1" version="1.0">
```

Els fitxers afectats pel problema són els següents:

```
correct_circ/cmp2.circ
correct_circ/codif.circ
correct_circ/cont1a.circ
correct_circ/cram2.circ
correct_circ/demux.circ
correct_circ/desp2l.circ
correct_circ/desp2r.circ
correct_circ/desp2ra.circ
correct_circ/litr.circ
correct_circ/mult4.circ
correct_circ/mux.circ
correct_circ/muxval.circ
correct_circ/sm4.circ
correct_circ/sr4.circ
correct_circ/ual2b.circ
datat/correct/abastardasl_20111_127_1_2011-10-21_17:23:48.circ
datat/correct/abastardasl_20111_128_1_2011-10-22_17:04:39.circ
datat/correct/abastardasl_20111_129_2_2011-10-23_16:35:32.circ
datat/correct/abastardasl_20111_129_3_2011-10-23_17:45:57.circ
datat/correct/abastardasl_20111_133_1_2011-11-15_16:09:02.circ
datat/correct/abastardasl_20111_133_2_2011-11-15_17:31:33.circ
datat/correct/abastardasl_20111_133_3_2011-11-18_15:27:59.circ
```

datat/correct/abastardasl_20111_136_2_2011-12-27_14:35:46.circ
datat/correct/cserret_20111_128_2_2011-10-29_17:23:13.circ
datat/correct/dbaneres_20111_114_3_2011-10-05_15:24:39.circ
datat/correct/dbaneres_20111_115_1_2011-10-05_15:24:08.circ
datat/correct/dbaneres_20111_116_2_2011-10-05_15:36:44.circ
datat/correct/dbaneres_20111_117_2_2011-10-05_16:26:39.circ
datat/correct/dbaneres_20111_117_3_2011-10-05_16:29:39.circ
datat/correct/dbaneres_20111_117_4_2011-10-05_16:32:31.circ
datat/correct/dbaneres_20111_117_5_2011-10-05_16:36:37.circ
datat/correct/dbaneres_20111_120_1_2011-10-05_16:47:58.circ
datat/correct/dbaneres_20111_120_3_2011-10-05_16:49:02.circ
datat/correct/dbaneres_20111_124_1_2011-10-06_14:38:45.circ
datat/correct/dbaneres_20111_124_3_2011-10-06_14:40:04.circ
datat/correct/dbaneres_20111_125_1_2011-10-06_14:42:43.circ
datat/correct/dbaneres_20111_126_10_2011-10-07_10:00:57.circ
datat/correct/dbaneres_20111_126_2_2011-10-07_09:10:59.circ
datat/correct/dbaneres_20111_126_3_2011-10-07_09:16:07.circ
datat/correct/dbaneres_20111_126_4_2011-10-07_09:16:20.circ
datat/correct/dbaneres_20111_126_5_2011-10-07_09:18:17.circ
datat/correct/dbaneres_20111_126_6_2011-10-07_09:18:25.circ
datat/correct/dbaneres_20111_127_1_2011-10-17_12:32:45.circ
datat/correct/dbaneres_20111_129_2_2011-10-17_15:43:29.circ
datat/correct/dbaneres_20111_129_7_2011-10-24_09:24:16.circ
datat/correct/dbaneres_20111_129_9_2011-10-24_09:38:52.circ
datat/correct/dbaneres_20111_130_1_2011-11-08_16:42:17.circ
datat/correct/dbaneres_20111_131_1_2011-11-08_17:32:48.circ
datat/correct/dbaneres_20111_131_18_2011-11-08_17:52:02.circ
datat/correct/dbaneres_20111_135_8_2011-11-11_14:56:38.circ
datat/correct/dbaneres_20111_136_10_2011-11-11_13:18:46.circ
datat/correct/dbaneres_20111_136_18_2011-12-01_08:56:20.circ
datat/correct/dbaneres_20111_136_5_2011-11-11_10:47:43.circ
datat/correct/dbaneres_20111_136_6_2011-11-11_10:52:48.circ
datat/correct/dbaneres_20111_136_7_2011-11-11_13:05:27.circ
datat/correct/dbaneres_20111_136_9_2011-11-11_13:13:41.circ
datat/correct/dpradosa_20111_128_2_2011-10-27_23:00:00.circ
datat/correct/dpradosa_20111_129_23_2011-11-01_12:08:31.circ
datat/correct/dpradosa_20111_129_24_2011-11-01_12:12:12.circ
datat/correct/dpradosa_20111_129_25_2011-11-01_13:12:37.circ
datat/correct/dpradosa_20111_129_26_2011-11-02_20:42:48.circ
datat/correct/dpradosa_20111_135_63_2011-11-29_10:07:30.circ
datat/correct/dpradosa_20111_135_64_2011-11-29_10:13:17.circ
datat/correct/dpradosa_20111_135_65_2011-11-29_12:11:40.circ
datat/correct/dpradosa_20111_135_66_2011-12-01_14:18:38.circ
datat/correct/dpradosa_20111_135_67_2011-12-01_14:18:47.circ
datat/correct/dpradosa_20111_135_72_2011-12-01_20:07:03.circ
datat/correct/dpradosa_20111_135_73_2011-12-01_20:07:12.circ
datat/correct/dpradosa_20111_136_164_2011-12-01_00:59:24.circ
datat/correct/dpradosa_20111_136_167_2011-12-01_15:12:39.circ
datat/correct/dpradosa_20111_136_168_2011-12-01_15:13:05.circ
datat/correct/dpradosa_20111_136_169_2011-12-01_16:17:51.circ
datat/correct/dpradosa_20111_136_170_2011-12-01_16:18:00.circ
datat/correct/dpradosa_20111_136_175_2011-12-06_12:32:59.circ
datat/correct/dpradosa_20111_136_176_2011-12-18_16:37:32.circ
datat/correct/dpradosa_20111_136_177_2011-12-18_16:51:13.circ
datat/correct/jarmenteras_20111_114_1_2011-10-31_17:48:39.circ
datat/correct/jarmenteras_20111_115_4_2011-10-31_18:26:12.circ
datat/correct/jarmenteras_20111_116_1_2011-10-31_20:16:56.circ
datat/correct/jarmenteras_20111_117_1_2011-10-31_20:39:55.circ
datat/correct/jarmenteras_20111_120_1_2011-11-09_16:09:20.circ
datat/correct/jarmenteras_20111_124_1_2011-11-01_20:24:53.circ
datat/correct/jarmenteras_20111_125_1_2011-11-01_20:36:15.circ
datat/correct/jarmenteras_20111_127_1_2011-11-02_00:08:43.circ
datat/correct/jarmenteras_20111_128_1_2011-11-02_12:05:43.circ
datat/correct/jarmenteras_20111_128_2_2011-11-02_12:06:22.circ
datat/correct/jarmenteras_20111_128_3_2011-11-02_20:52:48.circ
datat/correct/jarmenteras_20111_128_4_2011-11-02_20:54:38.circ
datat/correct/jarmenteras_20111_128_5_2011-11-02_20:56:37.circ
datat/correct/jarmenteras_20111_133_58_2011-11-22_18:50:27.circ
datat/correct/jarmenteras_20111_133_59_2011-11-22_19:10:29.circ
datat/correct/jarmenteras_20111_133_60_2011-11-22_19:24:37.circ
datat/correct/jarmenteras_20111_133_61_2011-11-22_19:27:56.circ
datat/correct/jarmenteras_20111_133_62_2011-11-22_19:29:12.circ
datat/correct/jarmenteras_20111_133_64_2011-11-22_19:31:42.circ
datat/correct/jarmenteras_20111_133_68_2011-11-22_22:44:10.circ
datat/correct/jarmenteras_20111_133_69_2011-11-22_22:48:33.circ
datat/correct/jarmenteras_20111_134_11_2011-11-22_20:10:19.circ
datat/correct/jarmenteras_20111_134_12_2011-11-22_22:53:11.circ
datat/correct/jarmenteras_20111_134_13_2011-11-22_22:56:58.circ
datat/correct/jarmenteras_20111_134_6_2011-11-22_19:14:15.circ
datat/correct/jarmenteras_20111_134_9_2011-11-22_19:56:43.circ

```
datat/correct/jlunar_20111_128_4_2011-10-30_20:10:59.circ
datat/correct/joelopezdiaz2_20111_127_1_2011-10-31_18:46:42.circ
datat/correct/joelopezdiaz2_20111_127_2_2011-11-01_12:06:39.circ
datat/correct/nsanchezrab_20111_127_2_2011-11-02_20:01:37.circ
datat/correct/nsanchezrab_20111_128_1_2011-11-01_17:45:46.circ
datat/correct/nsanchezrab_20111_128_2_2011-11-01_17:46:34.circ
datat/correct/nsanchezrab_20111_128_3_2011-11-01_22:18:14.circ
```

5.2.2. Fitxers que provoquen errors a la comanda «proc» de Yosys

Existeixen 47 fitxers CIRC dels quals no s'han pogut obtenir les mètriques calculades pel programa Berkeley-ABC. El motiu és que aquests circuits provoquen un error quan el programa Yosys executa la seva comanda «`proc`» per tal que puguin ser enviats a ABC. Són els següents:

```
correct_circ/cont1a.circ
correct_circ/cram2.circ
correct_circ/litr.circ
datat/correct/abastardasl_20111_133_1_2011-11-15_16:09:02.circ
datat/correct/abastardasl_20111_133_2_2011-11-15_17:31:33.circ
datat/correct/abastardasl_20111_133_3_2011-11-18_15:27:59.circ
datat/correct/abastardasl_20111_136_2_2011-12-27_14:35:46.circ
datat/correct/dbaneres_20111_130_1_2011-11-08_16:42:17.circ
datat/correct/dbaneres_20111_131_1_2011-11-08_17:32:48.circ
datat/correct/dbaneres_20111_131_18_2011-11-08_17:52:02.circ
datat/correct/dbaneres_20111_135_8_2011-11-11_14:56:38.circ
datat/correct/dbaneres_20111_136_10_2011-11-11_13:18:46.circ
datat/correct/dbaneres_20111_136_18_2011-12-01_08:56:20.circ
datat/correct/dbaneres_20111_136_5_2011-11-11_10:47:43.circ
datat/correct/dbaneres_20111_136_6_2011-11-11_10:52:48.circ
datat/correct/dbaneres_20111_136_7_2011-11-11_13:05:27.circ
datat/correct/dbaneres_20111_136_9_2011-11-11_13:13:41.circ
datat/correct/dpradosa_20111_135_63_2011-11-29_10:07:30.circ
datat/correct/dpradosa_20111_135_64_2011-11-29_10:13:17.circ
datat/correct/dpradosa_20111_135_65_2011-11-29_12:11:40.circ
datat/correct/dpradosa_20111_135_66_2011-12-01_14:18:38.circ
datat/correct/dpradosa_20111_135_67_2011-12-01_14:18:47.circ
datat/correct/dpradosa_20111_135_72_2011-12-01_20:07:03.circ
datat/correct/dpradosa_20111_135_73_2011-12-01_20:07:12.circ
datat/correct/dpradosa_20111_136_164_2011-12-01_00:59:24.circ
datat/correct/dpradosa_20111_136_167_2011-12-01_15:12:39.circ
datat/correct/dpradosa_20111_136_168_2011-12-01_15:13:05.circ
datat/correct/dpradosa_20111_136_169_2011-12-01_16:17:51.circ
datat/correct/dpradosa_20111_136_170_2011-12-01_16:18:00.circ
datat/correct/dpradosa_20111_136_175_2011-12-06_12:32:59.circ
datat/correct/dpradosa_20111_136_176_2011-12-18_16:37:32.circ
datat/correct/dpradosa_20111_136_177_2011-12-18_16:51:13.circ
datat/correct/jarmenteras_20111_133_58_2011-11-22_18:50:27.circ
datat/correct/jarmenteras_20111_133_59_2011-11-22_19:10:29.circ
datat/correct/jarmenteras_20111_133_60_2011-11-22_19:24:37.circ
datat/correct/jarmenteras_20111_133_61_2011-11-22_19:27:56.circ
datat/correct/jarmenteras_20111_133_62_2011-11-22_19:29:12.circ
datat/correct/jarmenteras_20111_133_64_2011-11-22_19:31:42.circ
datat/correct/jarmenteras_20111_133_68_2011-11-22_22:44:10.circ
datat/correct/jarmenteras_20111_133_69_2011-11-22_22:48:33.circ
datat/correct/jarmenteras_20111_134_11_2011-11-22_20:10:19.circ
datat/correct/jarmenteras_20111_134_12_2011-11-22_22:53:11.circ
datat/correct/jarmenteras_20111_134_13_2011-11-22_22:56:58.circ
datat/correct/jarmenteras_20111_134_6_2011-11-22_19:14:15.circ
datat/correct/jarmenteras_20111_134_9_2011-11-22_19:56:43.circ
datat/correct/mrodriguezsilva_20122_252_30_2013-06-05_00:46:25.circ
datat/correct/mrodriguezsilva_20122_252_31_2013-06-05_01:01:28.circ
```

L'error fa que els fitxers no puguin ser analitzats pel programa ABC, ja que la comanda «`proc`» de Yosys és l'encarregada d'eliminar els processos (blocs «`always`») existents en el codi Verilog, i precisament el programa ABC no pot gestionar dissenys amb processos per resoldre. En aquesta situació el programa Yosys avorta sobtadament i mostra el següent missatge:

```
yosys> read_verilog 20102_EX1_3A.verilog.v
1. Executing Verilog-2005 frontend.
Parsing Verilog input from `20102_EX1_3A.verilog.v' to AST representation.
Generating RTLIL representation for module `\m_20102_EX1_3A_verilog'.
Successfully finished Verilog frontend.

yosys> proc

2. Executing PROC pass (convert processes to netlists).

2.1. Executing PROC_CLEAN pass (remove empty switches from decision trees).
Cleaned up 0 empty switches.

2.2. Executing PROC_RMDEAD pass (remove dead branches from decision trees).
Removed a total of 0 dead cases.

2.3. Executing PROC_INIT pass (extract init attributes).

2.4. Executing PROC_ARST pass (detect async resets in processes).
Found async reset \reset_0 in `\m_20102_EX1_3A_verilog.$proc$20102_EX1_3A.verilog.v:28$145'.
Found async reset \v_22_25_0 in `\m_20102_EX1_3A_verilog.$proc$20102_EX1_3A.verilog.v:28$145'.
ERROR: Async reset \v_22_25_0 yields non-constant value 1'm for signal \regxxx0_0.
$
```

L'error sembla associat a una determinada gestió dels elements de memòria (biestables, registres i RAM) per part de la versió del *parser* utilitzada per crear els fitxers Verilog d'exemple. No afecta els fitxers Verilog generats per l'eina CircParser (apartat 3.1, pàgina 13), exceptuant-ne els 2 següents, que provoquen l'error per deficiències en el propi disseny del circuit (tenen biestables D amb els senyals CK i LD units):

```
datat/correct/mrodriguezsilva_20122_252_30_2013-06-05_00:46:25.circ
datat/correct/mrodriguezsilva_20122_252_31_2013-06-05_01:01:28.circ
```

La resta són fitxers Verilog que pertanyen a exercicis dels quals no s'ha pogut llegir el fitxer CIRC per ser d'una versió antiga (2.3.2) tal com es descriu en l'apartat anterior 5.2.1, i que per tant han hagut de ser processats a partir del fitxer Verilog d'exemple directament.

Cal esmentar que, entre els arxius Verilog d'exemple, n'hi ha 1004 que presenten aquest error (el 13 per cent del total), i que gràcies a l'ús de l'eina CircParser se n'han pogut recuperar 957, de manera que l'error persisteix només en aquells que el CircParser no és capaç de convertir.

5.2.3. Fitxers CIRC amb codificació incorrecta.

Alguns fitxers CIRC d'exemple, tal com s'han rebut, no poden ser processats pel *parser* pel fet de tenir una codificació que no es correspon amb l'estàndard UTF-8. De fet, aquests fitxers ni tan sols poden obrir-se amb el programa VerilUOC (3.0.0) ni amb el Logisim (2.7.1) ni tampoc visualitzar-los amb el navegador Firefox. Són els següents:

```
cguerrapi_20121_210_31_2012-12-27_20:32:56.circ
cguerrapi_20121_210_32_2012-12-27_22:27:08.circ
cguerrapi_20121_211_3_2012-12-23_15:45:28.circ
fegor_20121_137_6_2012-10-27_13:22:16.circ
jdomene_20131_262_4_2013-10-29_18:02:55.circ
jlledom_20121_211_1_2012-12-23_18:46:44.circ
jluciog_20121_195_1_2012-10-28_10:01:41.circ
joelopezdiaz2_20111_121_2_2011-11-01_12:00:21.circ
joelopezdiaz2_20111_121_3_2011-11-08_12:26:09.circ
joelopezdiaz2_20111_127_2_2011-11-01_12:06:39.circ
```

Tanmateix, el problema es resol obrint-los amb un editor de text i desant-los amb la codificació correcta. Es tracta d'una circumstància molt minoritària (únicament 10 fitxers CIRC sobre un total de 7536) que probablement no ha estat provocada en el moment de generar els fitxers amb VerilUOC o Logisim, sinó que podria deure's a una posterior edició dels mateixos amb un editor de text.

Quan els programes CircParser o VerilCircMetrics ensopeguen amb un problema d'aquest tipus, la conversió de CIRC a Verilog falla i s'envia una notificació a la consola de l'aplicació.

5.3. Treballs futurs

Aquest treball es podria enriquir amb algunes possibilitats de millora que han anat apareixent durant el seu desenvolupament. Per ordre de prioritat podrien ser les següents:

- Una investigació més a fons de les tècniques de *technology mapping* que es poden aplicar gràcies als programes Yosys i ABC, i que malauradament en el decurs d'aquest projecte no s'han pogut aprofitar fins al final. En teoria aquestes tècniques han de permetre arribar a la solució més òptima realitzable amb els components propis del programa VerilUOC.
- Una eina de procés automàtic de les dades generades per VerilCircMetrics. Ara mateix el programa genera unes taules amb 20 mètriques de complexitat per a cada circuit, de les quals 10 corresponen al propi circuit i les altres 10 a la versió optimitzada. Seria útil disposar d'una eina que fes una lectura ponderada de tots aquests valors i generés les avaluacions per a cada exercici. És una tasca que es pot fer fàcilment amb un full de càlcul, però seria interessant que es pogués fer automàticament.
- Unes interfícies d'usuari més elaborades per a l'execució interactiva de les aplicacions, sobretot de la principal VerilCircMetrics. Certament no era una prioritat del projecte, però és un aspecte fàcil de millorar.

5.4. Referències

- [ICV001] - C. Medrano, I. Plaza, M. Castro, F. García-Sevilla, J.D. Martínez-Calero, Josep Pou Felix, M. Corbalán – A review of electronic engineering design free software tools, 2010, IEEE EDUCON Education Engineering 2010
- [VRL001] - Vijay Nehra, Aruna Tyagi – Free Open Source Software in Electronics Engineering Education: A Survey. 2014, I.J. Modern Education and Computer Science.
- [CLW002] – Clifford Wolf – Yosys Manual, pàg.2. Design and Implementation of the Yosys Open SYnthesis Suite. 2013. – Bachelor Thesis, Vienna University of Technology. http://www.clifford.at/yosys/files/yosys_manual.pdf
- [AMI001] – Alan Mishchenko, Berkeley Verification and Synthesis Research Center – ABC: A System for Sequential Synthesis and Verification. https://people.eecs.berkeley.edu/~alanmi/abc/
- [CLW137] – Clifford Wolf – Yosys Manual, pàg.137. Design and Implementation of the Yosys Open SYnthesis Suite. 2013. – Bachelor Thesis, Vienna University of Technology. http://www.clifford.at/yosys/files/yosys_manual.pdf
- [CLW068] – Clifford Wolf – Yosys Manual, pàg.68. Design and Implementation of the Yosys Open SYnthesis Suite. 2013. – Bachelor Thesis, Vienna University of Technology. http://www.clifford.at/yosys/files/yosys_manual.pdf