Data-driven End-to-End Equivalence Checking of HLS Synthesis (DEEQ)

Review

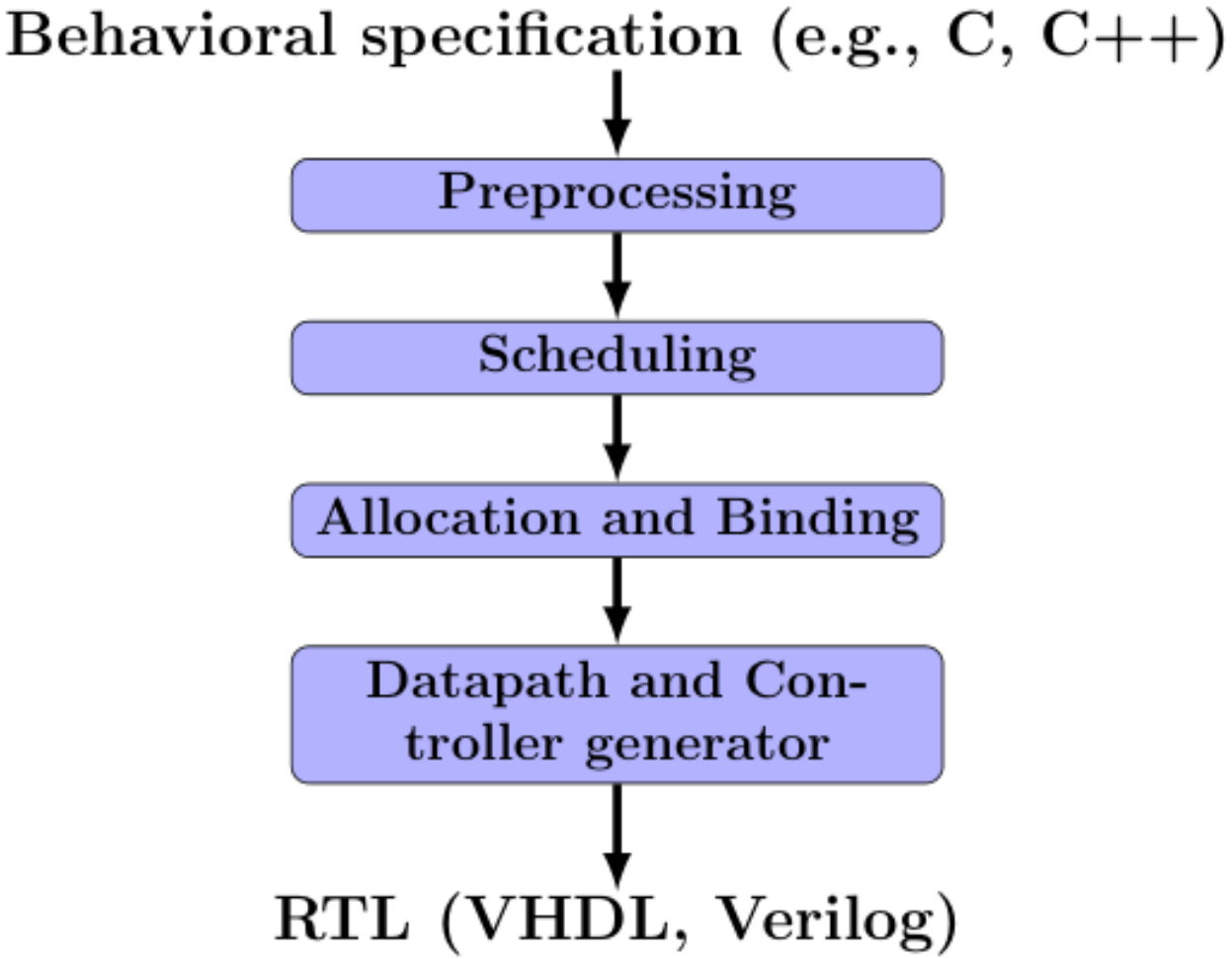
Lakshita Singhal, Prashant Pandey, Chaitanya Tejaswi,
Archisman Dey, Manchikatla Navya Sri

.

CS577: C-Based VLSI Design

Indian Institute of Technology, Guwahati
April 23, 2022

HLS


Behavioral specification (e.g., C, C++)
Preprocessing
Scheduling
Allocation and Binding
Datapath and Controller generator
RTL (VHDL, Verilog)

Is the problem of HLS Verification still relevant?

- Most works target ”Scheduling”, not ”Allocation/Binding” & ”Datapath Generation” phases.
- C/Verilog equivalence checking (using intermediate info) (6).
- v2c (Verilog to C) (7) generates incorrect C code using VivadoHLS.
- Fuzzy-Search suggested VivadoHLS generates 2.5% incorrect RTL from C (5).
- C/Verilog equivalence checking (without using intermediate info) $\Leftarrow$ *ThisWork*!

C & RTL-C Equivalence Checking Workflow (1), (2)

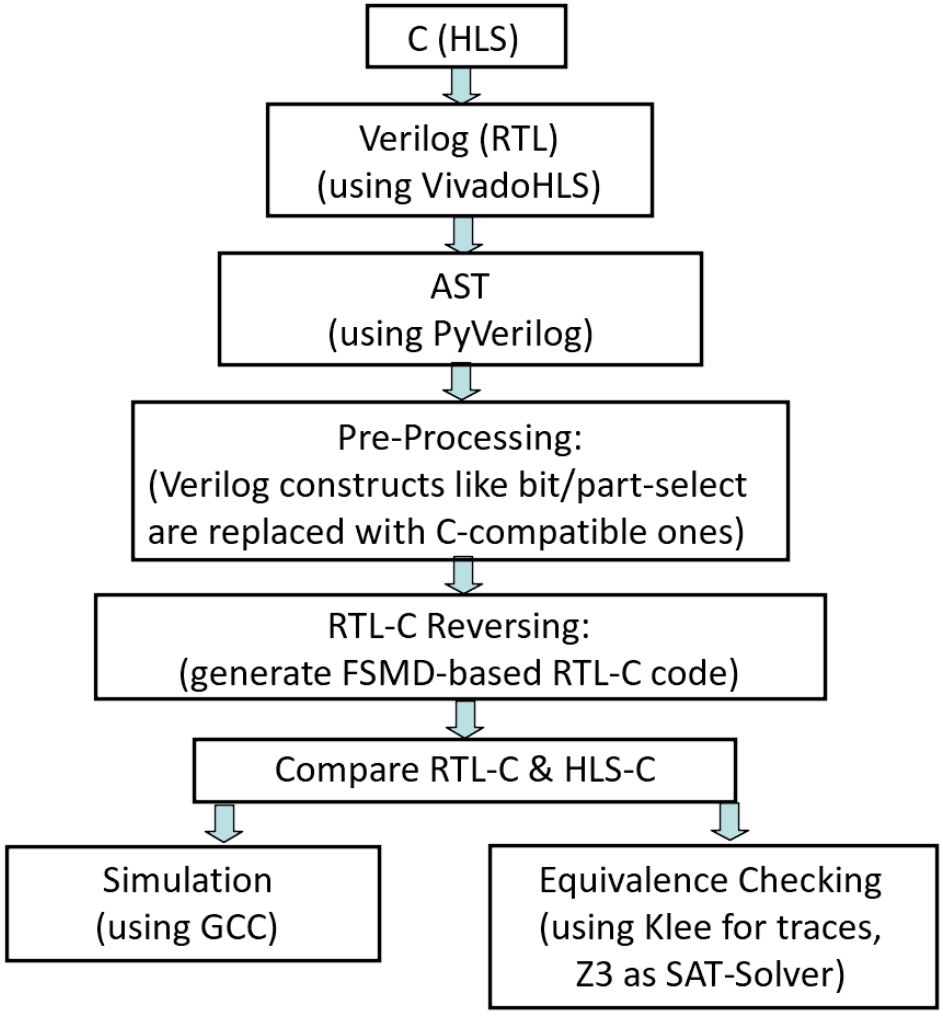


Figure: C & RTL-C Equivalence Checking Workflow (1), (2)

C & RTL-C Equivalence Checking Workflow (1)

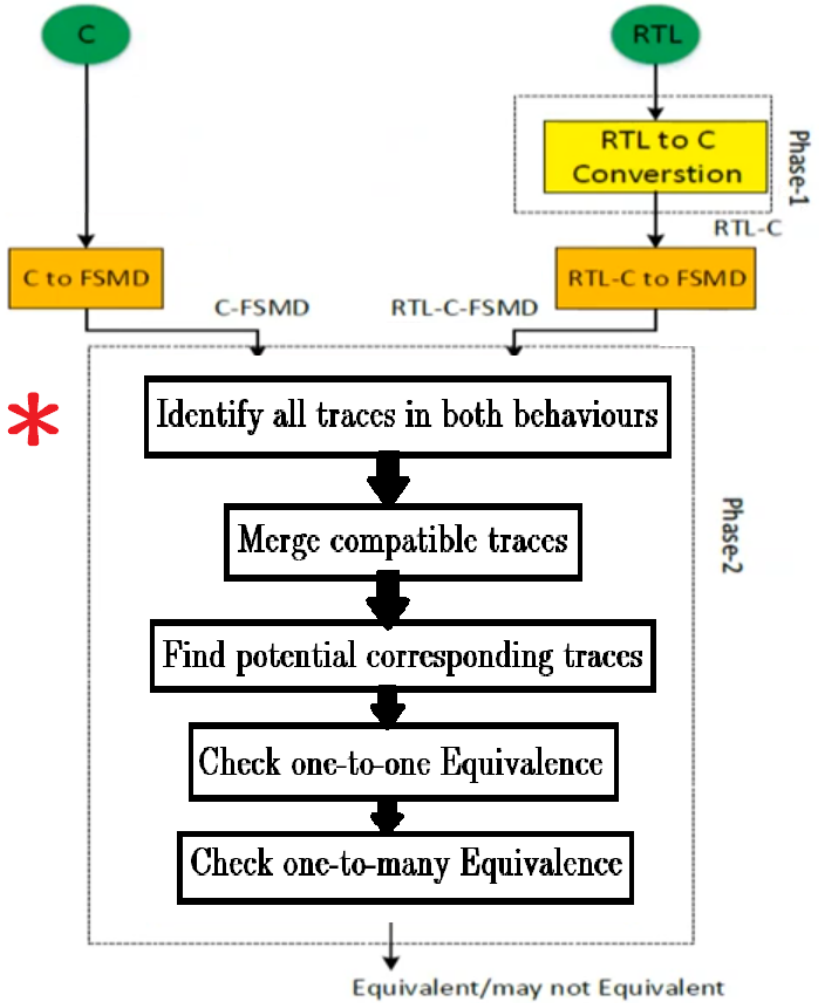


Figure: C & RTL-C Equivalence Checking Workflow (1)

C & RTL-C Equivalence Checking Workflow (1)

Algorithm : DEEQ (C, RTL-C)

Input: Input-C, RTL-C
Result: Equivalent, May not equivalent

1 T_0 = findTrace(C); T_1 = findTrace(RTL-C);
2 T_0 = mergeTrace (T_0); T_1 = mergeTrace (T_1);
3 copyT$_0$= T$_0$; copyT$_1$= T$_1$;
4 **while** $T_0 \neq \phi$ **do**
5 τ_0 = select a trace from T_0;
6 TC = getTestcase(τ_0);
7 τ_1 = getCorrespondingTrace(T_1, TC);
8 **if** $((c_{\tau_0} \equiv c_{\tau_1}) \wedge (s_{\tau_0} \equiv s_{\tau_1}))$ **then**
9 //τ_0 and τ_1 are equivalent
10 removeTrace (τ_0, copyT$_0$);
11 removeTrace (τ_1, copyT$_1$);
12 **else if** $((c_{\tau_0} \equiv c_{\tau_1}) \wedge (s_{\tau_0} \neq s_{\tau_1}))$ **then**
13 Report "Not Equivalent (NEq)" and Exit;
14 **endif**
15 removeTrace (τ_0, T$_0$);
16 **endwhile**
17 **if** $(|copyT_0| \neq NULL)$ **then**
18 **foreach** $\tau_0 \in copyT_0$ **do**
19 $c_\tau = \phi$;
20 **foreach** $\tau_1 \in copyT_1$ **do**
21 **if** $(c_{\tau_0} \wedge c_{\tau_1} \neq \phi)$ **then**
22 **if** $(c_{\tau_0} \wedge c_{\tau_1} \wedge s_{\tau_0} \neq s_{\tau_1})$ **then**
23 Report "Not Equivalent" and Exit;
24 **else**
25 $c_\tau = c_\tau \vee c_{\tau_1}$;
26 **if** $(c_\tau \neq c_{\tau_0})$ **then**
27 Report "Not Equivalent" and Exit;
28 Report Equivalent (Eq);

Generate all traces in both behaviours
Merge compatible traces

Find corresponding traces in both behaviours

Check 1-to-1 Equivalence

Equivalence-Checking using Z3

Check 1-to-Many Equivalence

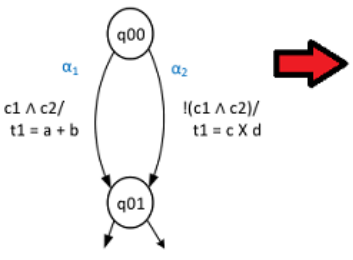
Figure: C & RTL-C Equivalence Checking Workflow (1), (2)

Equivalence Checking Example: if/else (1)

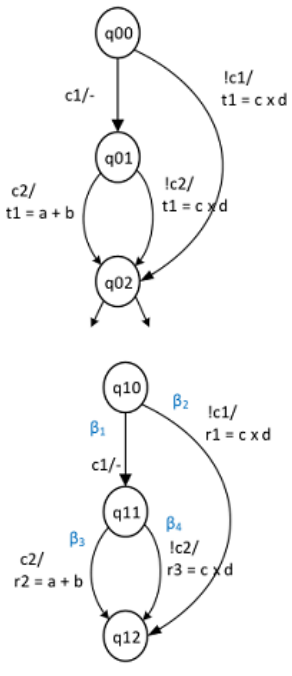
C-codeblock presented to SAT solver in CNF Form

```
if (c1 ∧ c2){
    t1 = a + b;
}
else{
   t1 = c x d;
}
```

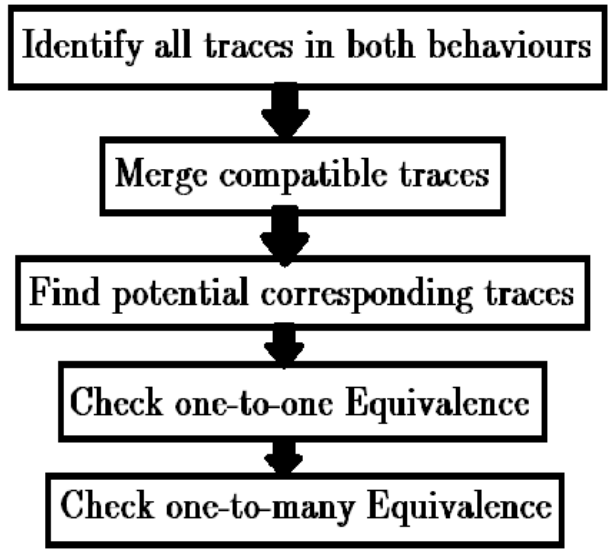
```
if(c1){
    if(c2)
        t1 = a + b;
    else
        t1 = c x d;
}
else{
    t1 = c x d;
}
```



Traces: α1, α2



Traces:
$\beta_1\beta_3, \beta_1\beta_4, \beta_2$



SAT (CNF Form)

```
(and (or x1 x2)
     (or x1 x3 x8)
     (or (not x2) (not x3) x4)
     (or (not x4) x5 x7)
     (or (not x4) x6 x8)
     (or (not x5) (not x6))
     (or x7 (not x8))
     (or x7 (not x9) x10))
```

Figure: Equivalence-Checking of FSMDs before/after pre-processing

Equivalence Checking Example: GCD (3)

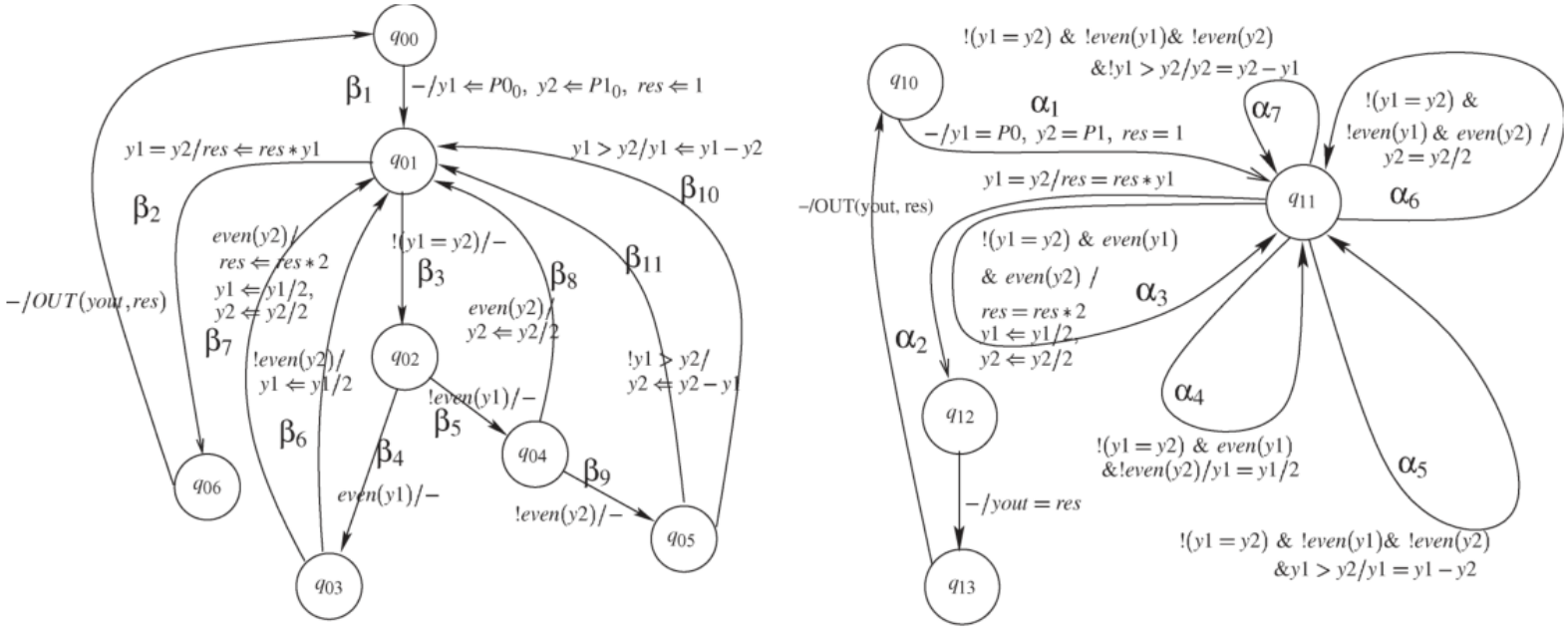


Figure: Equivalence-Checking of FSMDs of GCD before/after scheduling

Setup

1. DEEQ is written in Python
2. Tested on benchmarks - CHStone (4)
3. VivadoHLS generates RTL (verilog) from HLS-C (C code of benchmarks)
4. pyVerilog extracts AST from RTL; rewrite to form RTL-C using FastSim
5. Traces of both behaviours are obtained using Klee
6. Equivalence of two traces is verified using SMT Solver (Z3)

Results

Bench	#in	#out	C code		RTL code		RTL-C		Traces			Equivalent		Not Equivalent	
			#line	#var	#line	#regs	#line	#var	#C	#RTLC	#merged	time (s)	result	time (s)	result
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)
Waka	20	3	33	32	270	12	382	126	3	4	(3, 3)	1.709	Eq	0.669s	NEq
Arf	11	4	53	43	351	19	607	158	4	4	(3, 3)	1.890	Eq	0.949	NEq
Parker	6	1	62	14	196	10	275	100	12	23	(2, 2)	1.614	Eq	0.976	NEq
FindMin8	8	1	40	15	175	11	780	243	128	128	(8, 8)	22.246	Eq	17.141	NEq
MatrixAdd	2	1	48	7	734	44	2595	241	1	1	(1, 1)	1.684	Eq	0.749	NEq
SumArray	1	1	19	4	263	15	541	100	1	1	(1, 1)	0.754	Eq	0.706	NEq
Motion	10	3	52	43	413	29	881	235	1	1	(1, 1)	0.681	Eq	0.663	NEq
Dfadd	2	1	719	70	1975	113	9353	1041	67	68	(21,42)	1016.052	Eq	960.238	NEq

Figure: Results for CHStone Benchmarks

Usefulness

1. Benchmarks consist of complex **if-else** (find-Min8), **loops & arrays** (matrixAdd, sumArray), complex **arithmetic operations** (arf and motion), and **function calls** (dfadd).
2. dfadd has 9k lines of RTL-C code.
3. Since DEEQ merges compatible traces before checking equivalence, and uses a data-driven approach, it is expected to scale well to larger benchmarks.
4. Detects reported bug in VivadoHLS (5) - a large int is shifted repeatedly by array values - non-equivalence reported as 73741823 (HLS-C) and 6632959 (RTL-C).

References

[1] M. Abderehman, R. T. Reddy, and C. Karfa, "Deeq: Data-driven end-to-end equivalence checking of high-level synthesis," in *23rd International Symposium on Quality Electronic Design (ISQED'22)*, ACM/IEEE, 2022.

[2] M. Abderehman, J. Patidar, J. Oza, Y. Nigam, T. A. Khader, and C. Karfa, "Fastsim: A fast simulation framework for high-level synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2021.

[3] C. Karfa, D. Sarkar, C. Mandal, and P. Kumar, "An equivalence-checking method for scheduling verification in high-level synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 3, pp. 556–569, 2008.

[4] Y. Hara, H. Tomiyama, S. Honda, and H. Takada, "Proposal and quantitative analysis of the chstone benchmark program suite for practical c-based high-level synthesis," *Journal of Information Processing*, vol. 17, pp. 242–254, 2009.

[5] Y. Herklotz, Z. Du, N. Ramanathan, and J. Wickerson, "An empirical study of the reliability of high-level synthesis tools," in *2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 219–223, 2021.

[6] D. B. A. Leung and S. Lerner, "C-to-verilog translation validation," in *Proceedings of the 2015 ACM/IEEE International Conference on Formal Methods and Models for Codesign*, MEMOCODE '15, (USA), p. 42–47, IEEE Computer Society, 2015.

[7] R. Mukherjee, M. Tautschnig, and D. Kroening, "V2c — a verilog to c translator," in *Proceedings of the 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems - Volume 9636*, (Berlin, Heidelberg), p. 580–586, Springer-Verlag, 2016.

Thank You!