RTLog Framework: Yet another open HDL and compiler, this time for Relative-Timing design

R. Simone, P. D'Angelo, I. Sztenberg, F. Badenas, F. Dominguez,
A. Ortiz, G. Makar and R. Suaya
Universidad Tecnológica Nacional - Facultad Regional Buenos Aires
Presenting author: gmakar@frba.utn.edu.ar

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Motivation

The development of a synthesis flow suitable for Relative-Timing asynchronous digital circuits design is presented. The compiler employs a four-phase handshake protocol to develop feed-forward micro pipelines. An associated HDL language, specifically devoted to describe such circuits, and a timing constraint generator are included in the present framework. Relative Timing is a mature technique for applications demanding high performance silicon with low power budget. Its design flow, on the other hand, could benefit from additional work in the automation of repetitive tasks. This work presents a user friendly framework suitable for both Relative Timing as well as synchronous designs which is compatible with industry standard Electronic Design Automation (EDA) tools.

Keywords and RTL Operations

The RTLog reserved words are:

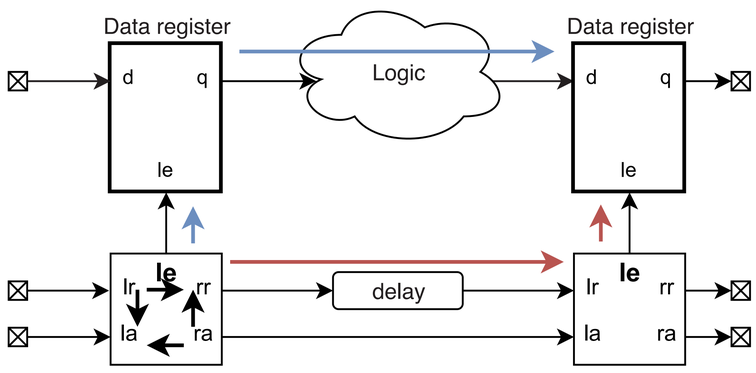
and	if	parameters
begin	in	port
block	for	reg
case	logic	rep
cat	natural	select
constant	not	when
end	or	xor
else	others	xnor

The operations between registers are:

not	Logic NOT
or	Logic OR
and	Logic AND
xor	Logic XOR
+	Addition
+x	Addition with carry
-	Subtraction
-x	Subtraction with borrow
*	Multiplication
*x	Signed Multiplication
==	Compare equal
!=	Compare different
<	Compare smaller
<=	Compare smaller or equal
>	Compare greater
>=	Compare greater or equal
>>	Right Shift
>>s	Right Shift signed
<<	Left Shift
cat	Concatenation
rep	Replication

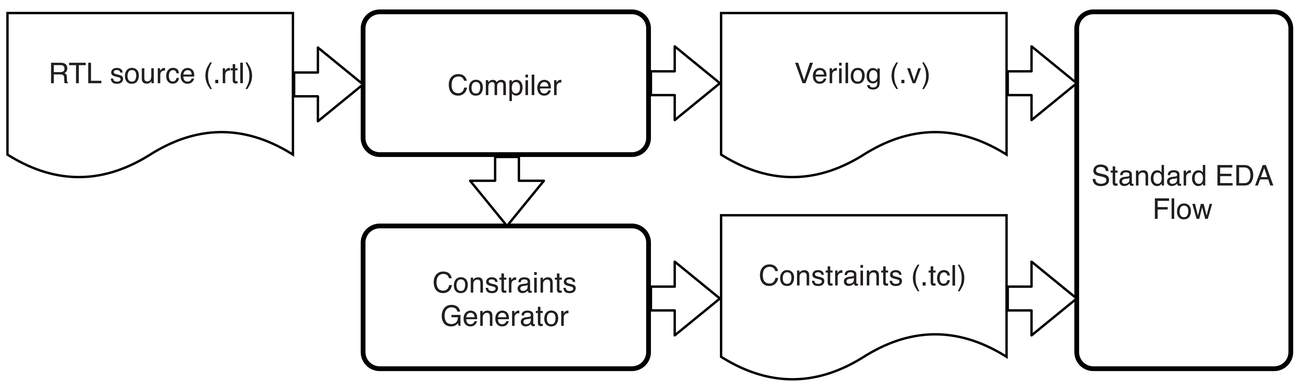
Constraints Generation

The RT methodology constrains every R2R path of the circuit to avoid timing violations and guarantee data validity. A second set of constraints prevents the synthesis tool (oriented to synchronous circuits) from removing the combinational loops in the controllers. RTLog uses feed-forward pipelines (input to output, without feedback) to build the datapaths.
Therefore the constraint generation can be automated by browsing the list of R2R, I2R and R2O paths provided by the compiler.



RTLog Flow

RTLog language is a simple HDL oriented to describe data transfers between registers. This description provides all the necessary information about the circuit structure. The compiler automates the addition of the controllers and handshake connections, which is usually tedious and error prone. The required timing constraints are also mapped into the produced design. Standard Verilog and TCL files are generated to be processed by EDA tools.



Compiler

The program flow can be conceptually divided in this sections:

- Construction of a directed graph based on the R2R paths with a synchronous circuit result
- Asynchronous blocks insertion:
  - Linear controllers for every latch (Fig 1)
  - Fork and join controller trees when necessary (Fig 2 and 3)
  - Handshake ports for external interface (Fig 4 and 5)
- Verilog source code generation for standard EDA flow

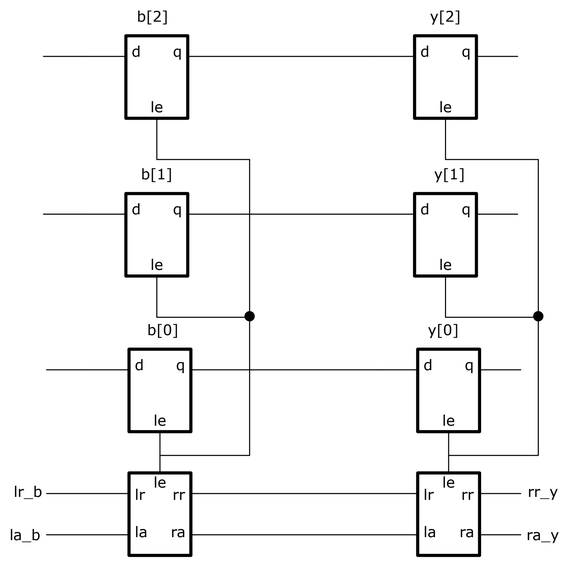


Figure 1: Addition of controllers

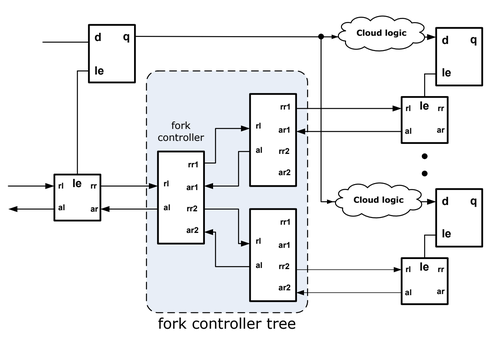


Figure 2: Fork Example

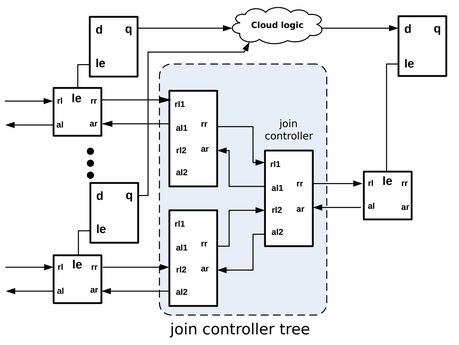


Figure 3: Join Example

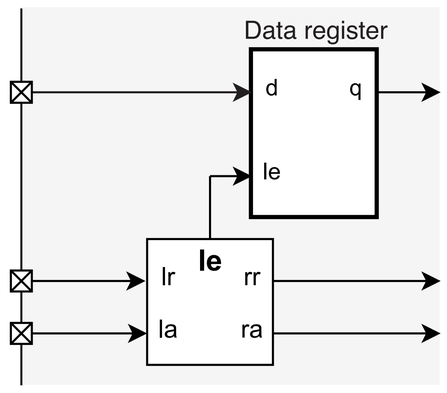


Figure 4: Input Handshake

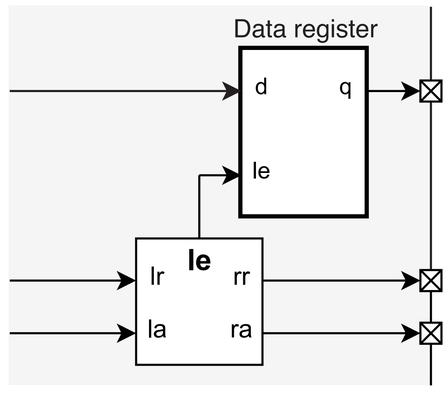


Figure 5: Output Handshake

Conclusions

This work presents a novel approach to HDL coding, RTLog, for describing asynchronous circuits that follow a four-phase handshake protocol with promising advantages over Verilog. A compiler tool to output a Verilog source code suitable for synthesis has been developed. Supporting elements were created to approximate a complete framework. The complete work is coded in C++. The current state of the project is alpha.

Main References and Code Repository

K. S. Stevens, R. Ginosar, and S. Rotem, "Relative timing [asynchronous design]" *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* , vol. 11, no. 1, pp. 129–140, Feb 2003
RTLog Repository: *https://github.com/VLSI-UTN-FRBA/RTLog*