2021 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION CHINA
SHANGHAI | MAY 26, 2021

accellera SYSTEMS INITIATIVE

Enflame 燧原科技

Transaction Equivalence Formal Check(DPV) in Video Algorithm/FPU/AI Area

Minqi Bao,Enflame,Shanghai,China(bob.bao@enflame-tech.com)

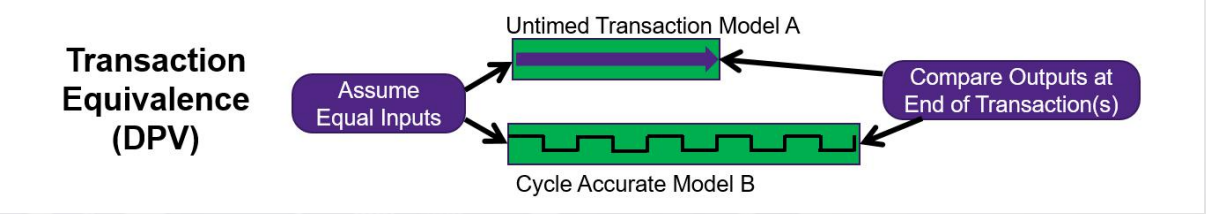
OUTLINE

- Introduction
- Methodology
  - General flow
  - Video algorithm use model and results
  - Floating point computation use model and results
  - AI computation use model and results
  - Debugging method

Introduction

All these pain points drive us to find a more efficient verification methodology. In this paper, we will use “DPV”, a transaction equivalence based formal tool, to gain more than 10X efficiency verification improvement in datapath verification. By transaction, we mean the following:

- A transaction consists of Inputs, Input State (optional), State change Outputs, Output State (optional).
- A transaction can be combinational, sequential overlapping / pipelined, or sequential non-overlapping.

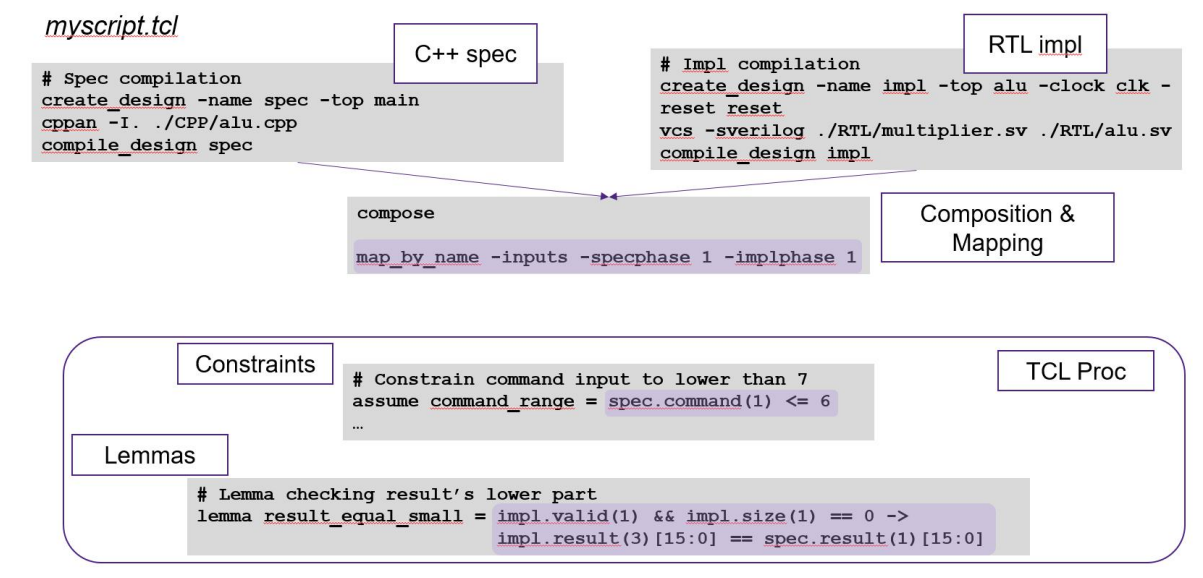


Methodology

General flow

DPV general flow is very simple and can be divided into four parts

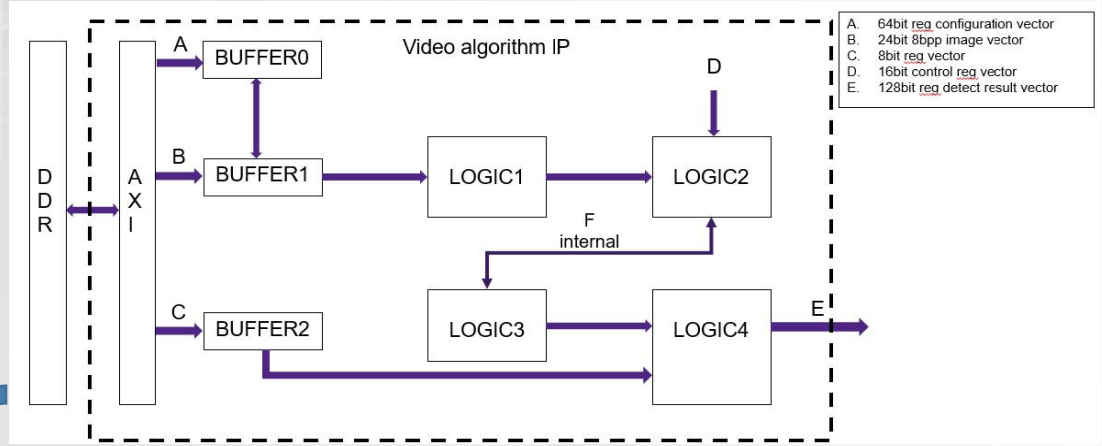
a. Compile C model – gnu compatible
b. Compile RTL – vcs compatible
c. Compose and mapping – compose automatically done by tool and mapping needs manual work usually
d. Set constrain(constraints) and checkers(lemmas) and prove them by formal engines

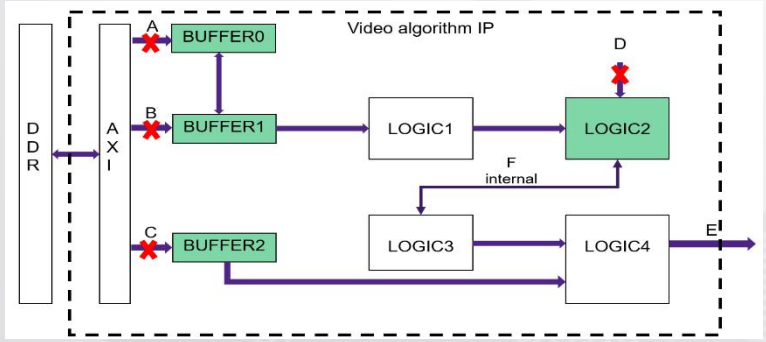


Methodology

Video algorithm use model and results

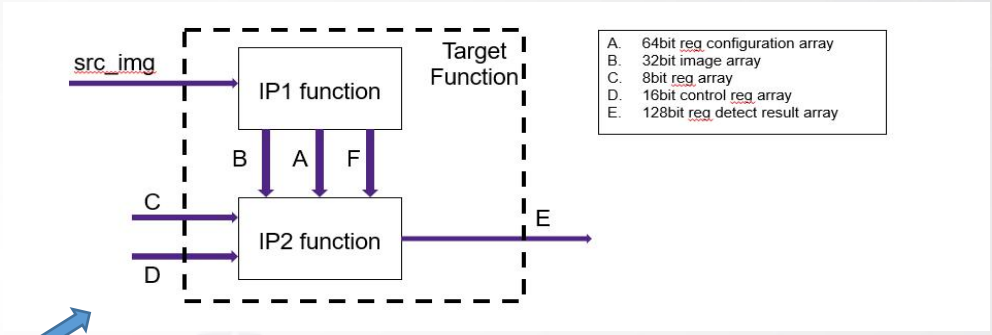
Original RTL





Optimized RTL structure for DPV

C



Corner bug found with transactional equivalence:

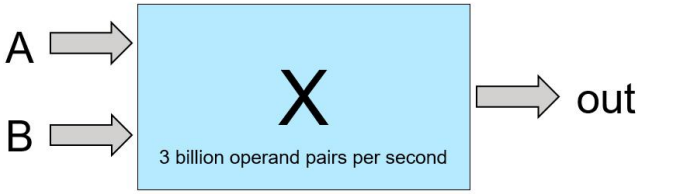
The maximum of the gradient is represented by a 11bits signal. After updated, the gradient operator is changed and the maximum of the gradient should be a 13bits signal while it keeps 11bits in design which caused the inconsistent with C++ golden

Method	Time to uncover this bug
Simulation	Days or weeks depends on specific pattern
DPV (transactional equivalence)	3 hours without any pattern effort

Methodology

Floating point computation use model and results

In FPU side, we choose a FP64 FMA with in-house radix booth encoding. It needs at least 2 weeks for all UVM testbench and coverage points setup while it just requires one-hour setup for transaction equivalence exhaustive verification.

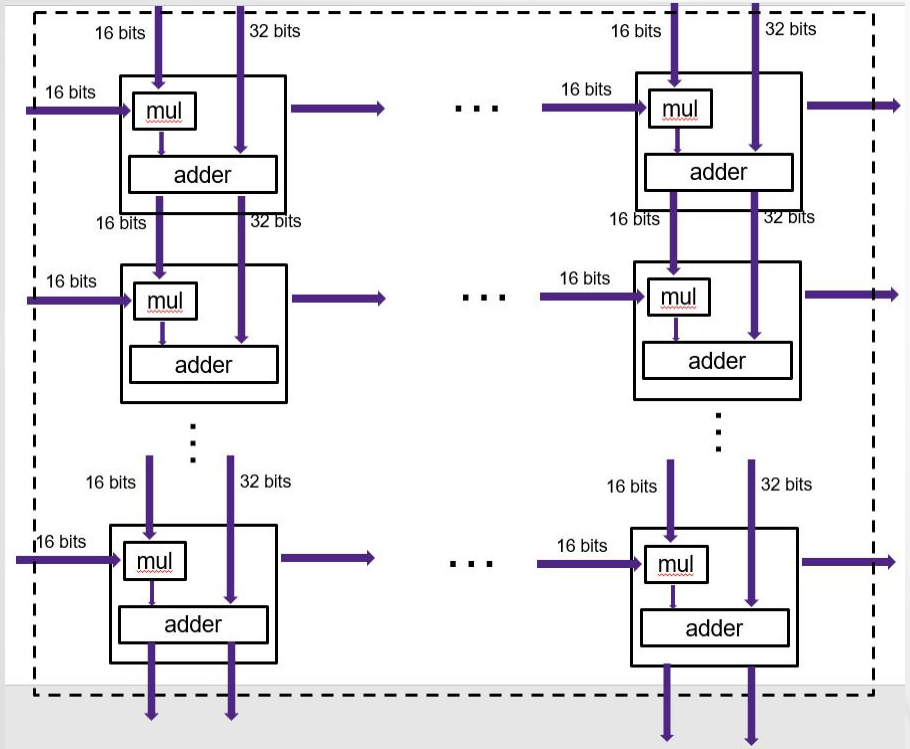


16-bit operands	32-bit operands	64-bit operands
1.5 seconds	195 years	$3.5 * 10^{21}$ years > earth age

Key Concerns	Traditional flow	Transaction Equivalence flow
Algorithm design block verification	• Develop UVM testbench • Random test to improve coverage	• Easy tcl setup and no testbench needed • Exhaustive proof
FP64 FMA	• Completed basic verification via UVM test with 14 days typically • Impossible to be exhaustive	• Setup formal check in one day • Possible to do exhaustive verification
Sign-off	• Need huge manual coverage review to meet datapath function sign off quality	• Comprehensive datapath function sign off flow

Methodology

AI computation use model and results



Scenario	Results
Matrix 8 x 8	proved in seconds
Matrix 32 x 32	proved in minutes
Matrix 64 x 64	proved in a few hours
Matrix 128 x 128	proved in 30 hours

Methodology

Debugging method