TWO SIGMA

Agile FPGA Design

July 16, 2019 - ASAP 2019

Justin Thiel
Two Sigma Securities

Legal Disclaimer

For informational purposes only.

None of the information contained in these materials constitutes a recommendation, solicitation or offer by Two Sigma and/or its affiliates (the "Two Sigma Affiliates") to buy or sell any securities, futures, options or other financial instruments or provide any investment advice or service.

Any distribution of this information without the express written consent of Two Sigma is prohibited and could be highly damaging to the business operations of the Two Sigma Affiliates.

Overview

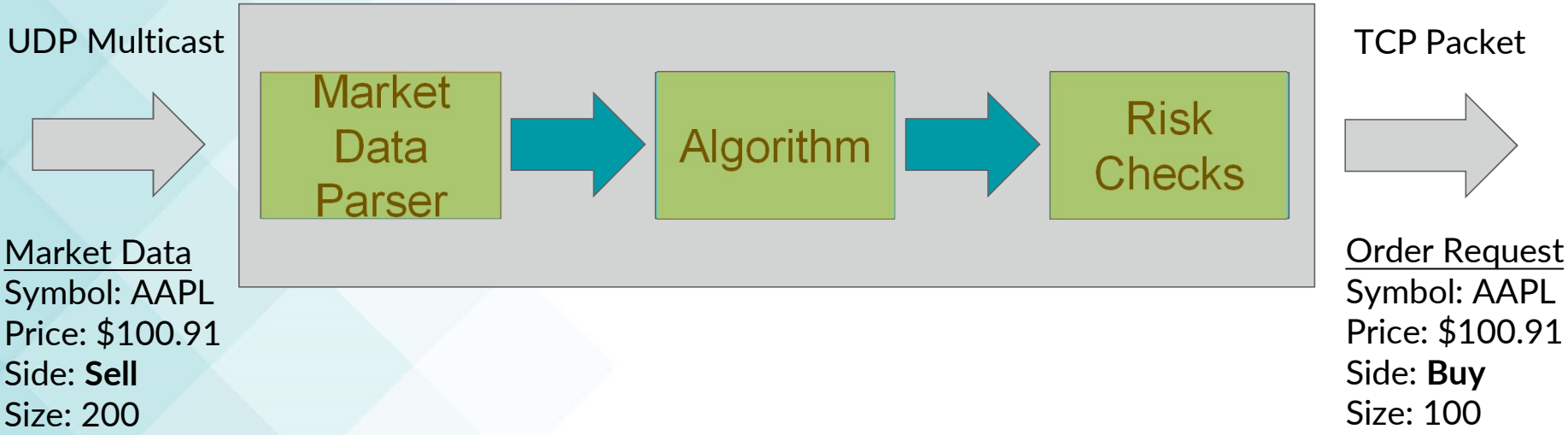
- Background
- Development Framework
- Testing Framework
- Vision

Background

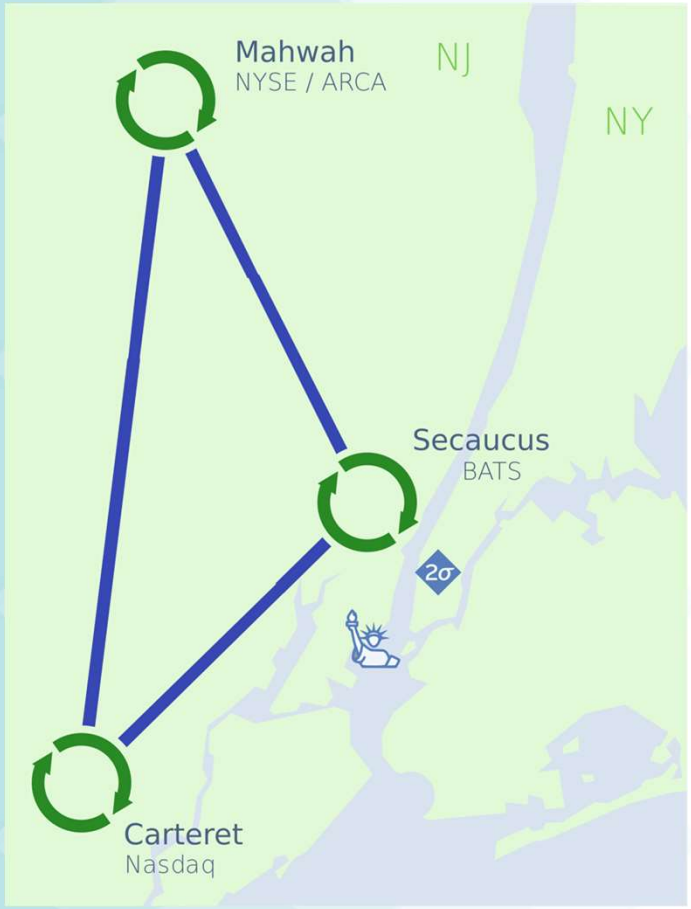
Background - Who is Two Sigma Securities?

- FINRA registered broker-dealer
- Over 100 employees
- Equities, options, and futures trading
- US and International operations

Background - Example Trading Pipeline



Background - Trading Environment



- Exchanges spread across NJ
- Exchange operators run multiple exchanges
- Systems installed at all major sites
- Need to coordinate across sites to maximize opportunities

Background - Business Challenges

Competitive Pressure to Improve Execution Speed

https://spectrum.ieee.org/tech-talk/telecom/wireless/wall-street-tries-shortwave-radio-to-make-highfrequency-trades-across-the-atlantic

Ever Increasing Market Data Rates

Feed	Peak / Minute (Events)	Average / Minute
Nasdaq Equities	1 Million	500 Thousand
Nasdaq Options	15 Million	7.5 Million

Background - Safety and Compliance

Risk Management

Knight Capital Says Trading Glitch Cost It $440 Million

BY NATHANIEL POPPER AUGUST 2, 2012 9:07 AM 356

https://dealbook.nytimes.com/2012/08/02/knight-capital-says-trading-mishap-cost-it-440-million

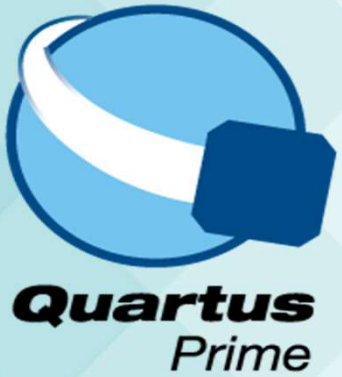
Exchange Mandated Changes

March 3, 2018: Nasdaq TotalView-ITCH Version 5.1

Released a new version of Nasdaq TotalView-ITCH documentation to add a new Operational Halt message (Section 4.2.8) to indicate the current Operational Status of a security to the trading community.

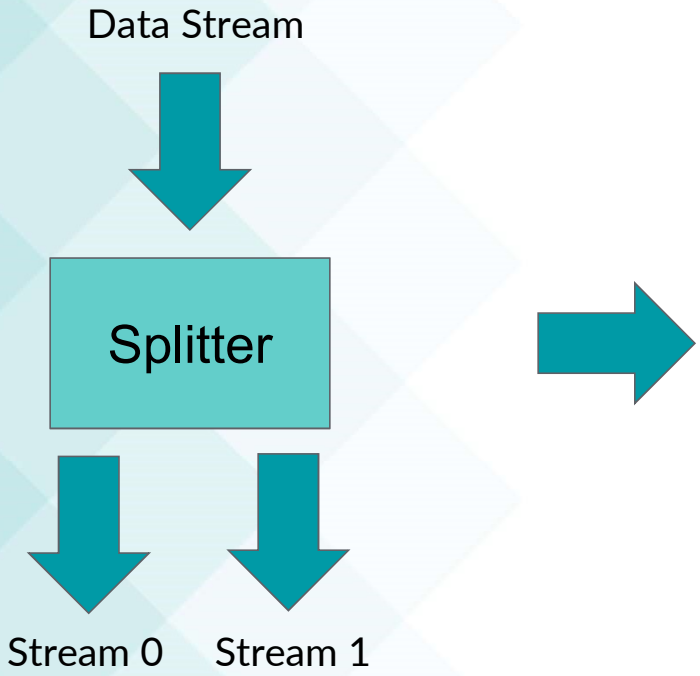
Background - Technology Challenges

Background - Technology Challenges



Development Framework

Development - Modern System Verilog




```
// Generated w/ IP Wizard in Vivado 2018.3
module axis_broadcaster_0(
  aclk, aresetn,
  s_axis_tvalid, s_axis_tready, s_axis_tdata,
  m_axis_tvalid, m_axis_tready, m_axis_tdata
);
  input aclk;
  input aresetn;

  input s_axis_tvalid;
  output s_axis_tready;
  input  [7:0] s_axis_tdata;

  output [1:0] m_axis_tvalid;
  input  [1:0] m_axis_tready;
  output [15:0] m_axis_tdata;

...
endmodule
```

Development - Modern System Verilog

```
module axis_broadcaster_0(
  aclk, aresetn,
  s_axis_tvalid, s_axis_tready, s_axis_tdata,
  m_axis_tvalid, m_axis_tready, m_axis_tdata
);
  input aclk;
  input aresetn;

  input s_axis_tvalid;
  output s_axis_tready;
  input  [7:0] s_axis_tdata;

  output [1:0] m_axis_tvalid;
  input  [1:0] m_axis_tready;
  output [15:0] m_axis_tdata;

...
endmodule
```

Old Style Portmaps

Signals for each Port

Fixed Signal Widths
Output Ports are Combined

Development - Modern System Verilog

```
interface stInterface #(
  parameter stream_cfg_t PARAMS = '0
);

logic [getStEmptyWidth(PARAMS)-1:0] empty;
logic [getStDataWidth(PARAMS)-1:0]  data;
logic val;
logic sop;
logic eop;
logic halt;

modport sink (
  input empty,
  input data,
  input val,
  input sop,
  input eop,
  output halt
);
endinterface
```

Configurable

Ports are bundled together

Fine grain control over connections

Development - Modern System Verilog

```
module axis_broadcaster_0(
  aclk, aresetn,
  s_axis_tvalid, s_axis_tready, s_axis_tdata,
  m_axis_tvalid, m_axis_tready, m_axis_tdata
);
  input aclk;
  input aresetn;

  input s_axis_tvalid;
  output s_axis_tready;
  input  [7:0]  s_axis_tdata;

  output [1:0]  m_axis_tvalid;
  input  [1:0]  m_axis_tready;
  output [15:0] m_axis_tdata;

...
endmodule
```

```
stSplit #(
  parameter stream_cfg_t PARAMS = '0
  parmeter PORTS = 2
)
(
  logic i_clk,
  logic i_rst,

  stInterface.sink i_st,
  stInterface.source o_sts[PORTS-1:0]
);

...
endmodule
```

Development - Code Generation

```
{
  “name” : “config_reg”,
  “desc” : “General Config Register”,
  “fields” : [
    {
      “name”: “exp_next_sqn”,
      “desc”: “Next expected sequence #”,
      “type” : “lu32”
    },
    {
      “name”: “exp_user_id”,
      “desc”: “Expected User ID”,
      “type” : “lu16”
    }
  ]
  “attributes” : {
    “sw_access” : “rw”,
    “hw_access” : “rw”
  }
}
```

Field Layout

Data Formats

Access Patterns

Development - Code Generation (C/C++)

```
{
  “name” : “config_reg”,
  “desc” : “General Config Register”,
  “fields” :
  [
    {
      “name”: “exp_next_sqn”,
      “desc”: “Next expected sequence #”,
      “type” : “lu32”
    },
    {
      “name”: “exp_user_id”,
      “desc”: “Expected User ID”,
      “type” : “lu16”
    }
  ]
}
```

```
#define CONFIG_REG_OFFSET 96

typedef struct config_reg_t {
  uint32_t exp_next_sqn;
  uint16_t exp_user_id;
} __PACKED config_reg_t;
```

```
// Example Use Code
config_reg_t config_reg = {
  .exp_next_sqn = 123,
  .exp_user_id  = 456
};

fpga_mm_write(
  CONFIG_REG_OFFSET,
  config_reg,
  sizeof(config_reg)
);
```

Development - Code Generation (Verilog)

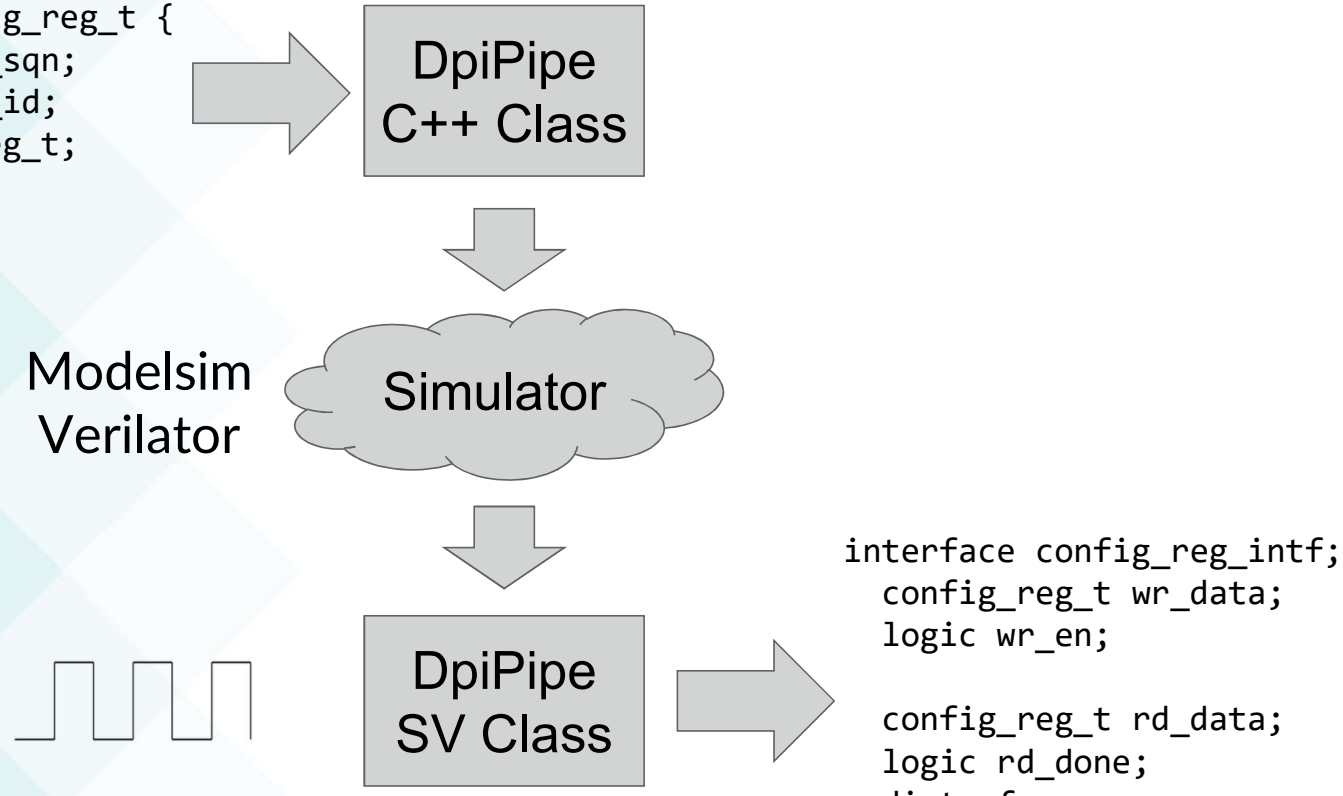
```
{
  “name” : “config_reg”,
  “desc” : “General Config Register”,
  “fields” :
  [
    {
      “name”: “exp_next_sqn”,
      “desc”: “Next expected sequence #”,
      “type” : “lu32”
    },
    {
      “name”: “exp_user_id”,
      “desc”: “Expected User ID”,
      “type” : “lu16”
    }
  ]
}
```

```
typedef struct packed {
  logic [31:0] exp_next_sqn;
  logic [15:0] exp_user_id;
} config_reg_t;

interface config_reg_intf;
  config_reg_t wr_data;
  logic wr_en;

  config_reg_t rd_data;
  logic rd_done;
endinterface

module module_regs #(
  parameter mm_cfg_t PARAMS = ‘0
)(
  mmUserInterface.sink io_mm,
  config_reg_intf      io_config_reg,
  . . .
);
```

Testing Framework

Testing - Simulation

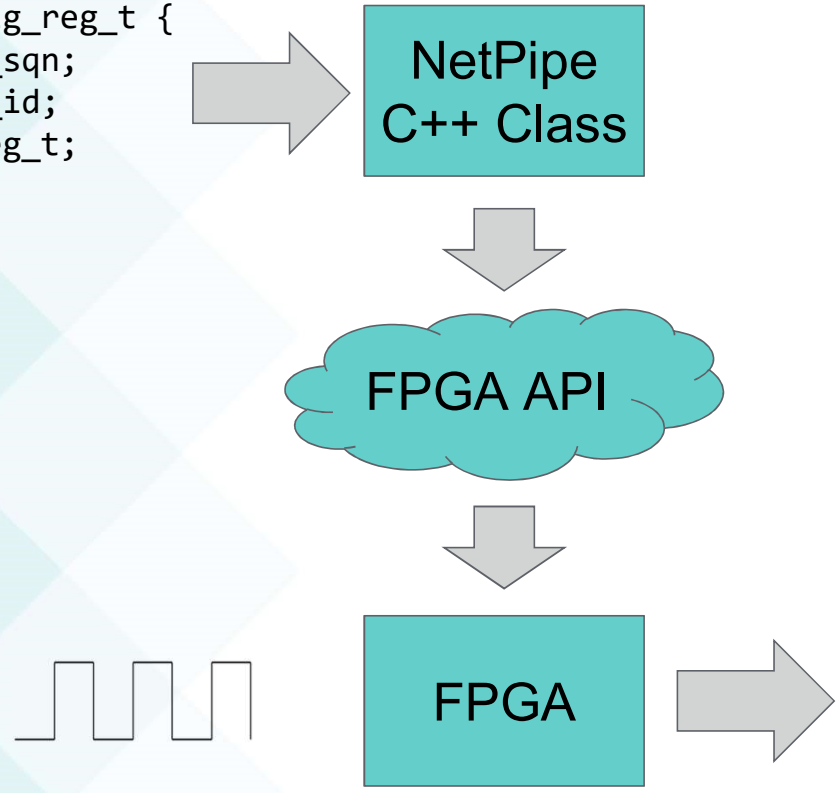
```
typedef struct config_reg_t {
  uint32_t exp_next_sqn;
  uint16_t exp_user_id;
} __PACKED config_reg_t;
```




```
interface config_reg_intf;
  config_reg_t wr_data;
  logic wr_en;

  config_reg_t rd_data;
  logic rd_done;
endinterface
```

Testing - Bare Metal

```
typedef struct config_reg_t {
  uint32_t exp_next_sqn;
  uint16_t exp_user_id;
} __PACKED config_reg_t;
```



Can reuse:

- Models
- Generators
- Checkers

```
interface config_reg_intf;
  config_reg_t wr_data;
  logic wr_en;

  config_reg_t rd_data;
  logic rd_done;
endinterface
```

Testing - Continuous Integration

Simulation Builds
Pre-push Tests
Code Review

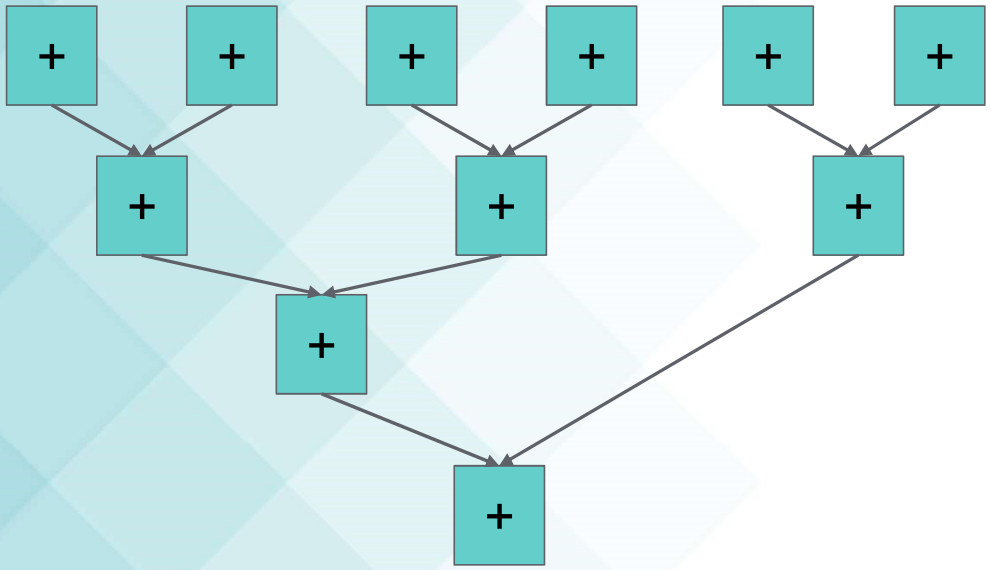
Bitstream Builds
Post-push Sims

HW Stress Tests
Integration Tests

Tech Approval
Compliance Sign Off

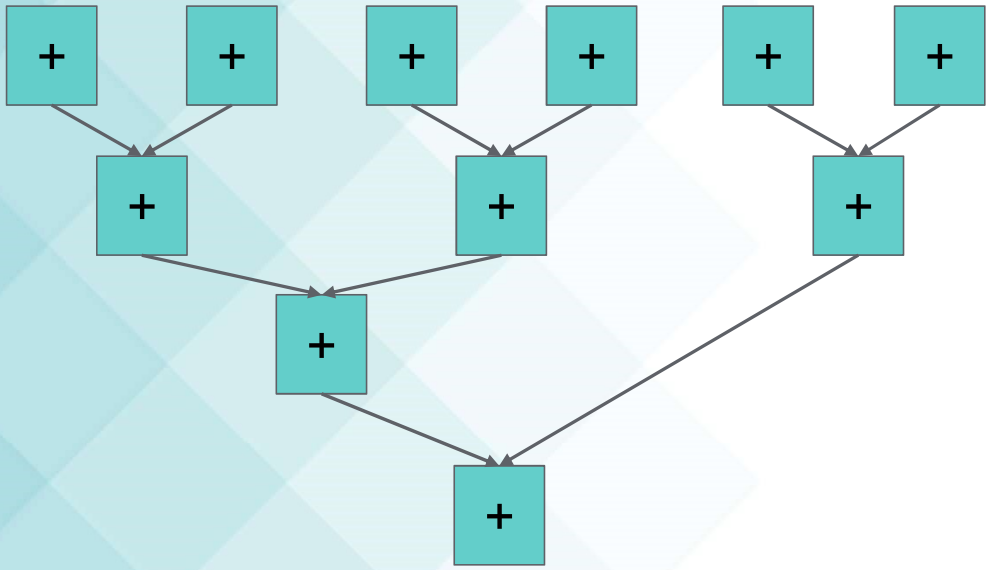
Vision

Vision - High Level Synthesis



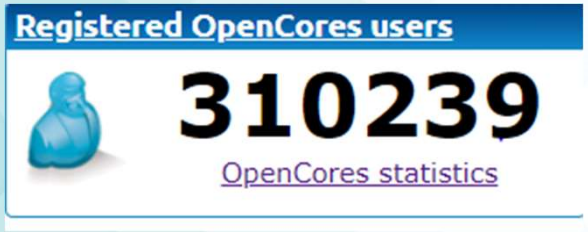

```
module adder_tree #(
  . . .
)(
  input  [WIDTH-1:0][CNT-1:0] i_datas,
  output [WIDTH-1:0] o_result,
  . . .
);

genvar s, p;

generate begin
  for (s = 0; s < STAGES; s++) begin
    . . .
    for (p = 0; p < PAIRS; i++) begin
      . . .
    end
  end
end endgenerate

assign o_result = result_q;
endmodule
```

Vision - High Level Synthesis



```
ap_uint<WIDTH+1> adder_tree (
  ap_uint<WIDTH> i_datas[CNT]
)
{
  #pragma HLS PIPELINE II=1
  ap_uint<WIDTH+1> result;

  #pragma HLS UNROLL FACTOR=2
  for (int x = 0; x < CNT; x++) {
    result += data[x];
  }

  return result;
);
```

Vision - Open Source History



- Arithmetic core 4
- Communication controller 6
- Crypto core 3
- ECC core 2
- Memory core 2
- Processor 9
- System on Chip 2
- System controller 2
- Testing / Verification 1
- Video controller 1

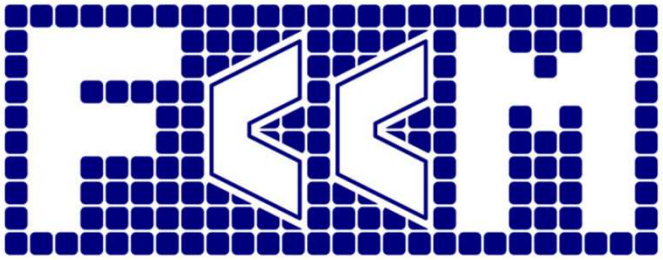
Total Certified Cores: 32

Vision - Open Source Now

IP Cores and Tutorials

Vendor Participation

CPU Research



Academic Transparency

Vision - Open Source Future

- Design tools
  - YoSys + NextPNR (github.com/YosysHQ)
- Package management
  - FuseSOC (github.com/olofk/fusesoc)
- Continuous Integration
  - LibreCores CI (librecores.org/static/librecores-ci)
- Lowering the barrier to entry
  - Education: FPGA Wars (fpgawars.github.io)
  - Access: EDA Playground (edaplayground.com)

Summary

- Changing technology and business requirements are easier to deal with if your foundation is flexible
- Vendors could do more to support modern coding constructs and provide flexible IP blocks
- Leveraging the DPI can allow for testbench reuse and simplify use of CI of FPGA designs
- HLS and the Open Source community can bring FPGAs to the masses

Questions?