# Insertion Sorting

AHA PROJECT

**N Adharsh - PES1UG21EC161**
**Akshay Anand - PES1UG21EC029**
**Nandan Vadeyar - PES2UG21EC089**
**Akash Ravi Bhat - PES1UG21EC025**
**Abhishek A Shetty – PES1UG21EC008**

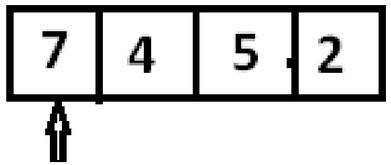
PES UNIVERSITY

# Insertion Sorting

- Insertion sort is a simple sorting algorithm that builds the final sorted array one item at a time by comparisons.
- This algorithm is efficient when sorting small data sets.
- It works in a very similar way to how we sort a hand of playing cards - each element in the list is inspected and then inserted in a new position such that all aspects to the left are smaller and all elements to the right are larger.

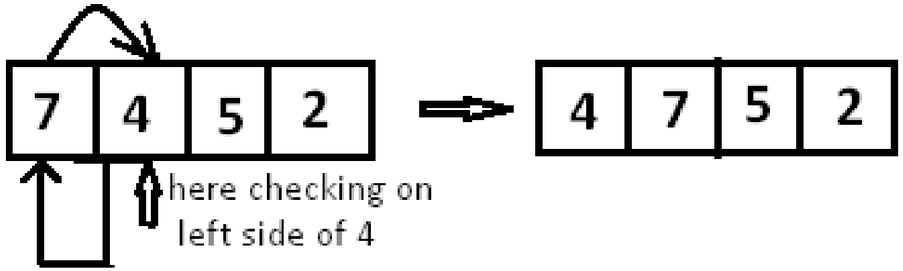
# Insertion Sorting

STEP 1.



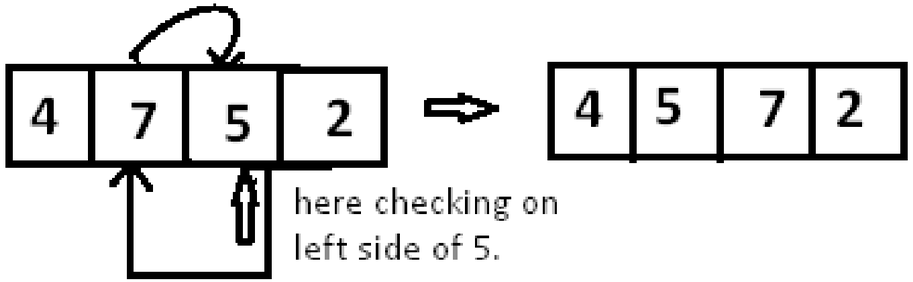

No element on left side of 7,so no change in its position .

STEP 2.



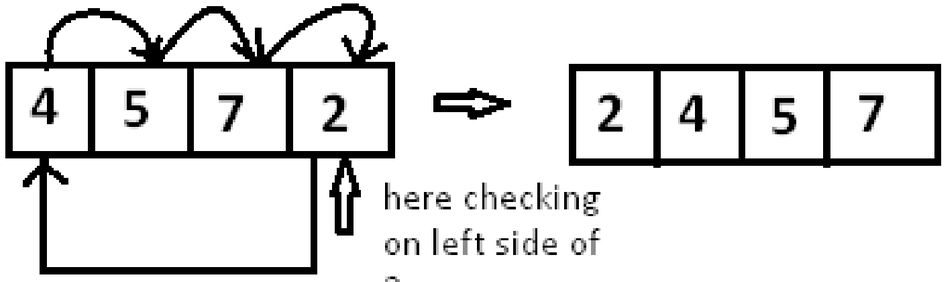

As 7>4 ,therfore 7 will be moved forward and 4 will be moved to 7's position.

STEP 3.




As 7>5,7 will be moved forward,but 4 <5,so no change in position of 4. And 5 will be moved to position of 7.

STEP 4.




As all the element on left side of 2 are greater than 2,so all the elements will be moved forward and 2 will be shifted to position of 4

# ADVANTAGES OF INSERTION SORTING

- Insertion sort operates directly on the input array, meaning it sorts the array in its original memory space **without necessitating additional memory allocation**.
- It's classified as a **stable sorting algorithm**, indicating that it maintains the relative order of equal elements in the sorted array.

PES UNIVERSITY

# HEADER FILE

```
#ifndef _SORTER_DEFS_
#define _SORTER_DEFS_

#include <cstdlib>
#include <ctime>

#include <iostream>
#include <iomanip>

using namespace std;



/*-----------------------------------------
  TYPES & CONSTANTS
-----------------------------------------*/

// this defines the total size of the external memory
const unsigned int memSize = 1024;

// this defines the type of the list elements
typedef unsigned int DATA_t;

// this defines the type of the list start address
typedef unsigned int ADDR_t;

// this defines the type of the list size
typedef unsigned int LSIZE_t;


/*-----------------------------------------
  FUNCTIONS PROTOTYPES
-----------------------------------------*/
void insertsort(ADDR_t startAddr, LSIZE_t listSize, DATA_t sortList[memSize]);

void chkResults(DATA_t testArray[memSize], ADDR_t startAddr, LSIZE_t listSize);


#endif
```

PES UNIVERSITY

# HLS CPP CODE

```
#include "sorter_defs_v1_0.h"


void insertsort(ADDR_t startAddr, LSIZE_t listSize, DATA_t sortList[memSize])
{
	/*--------------------------------------------------------
	 * Pragmas for startAddr, listSize and the control ports
	 --------------------------------------------------------*/
	// these pragmas bundle the input arguments into an AXI lite
	// interface for software control
	 #pragma HLS INTERFACE s_axilite port=startAddr   bundle=CTL
	 #pragma HLS INTERFACE s_axilite port=listSize    bundle=CTL
	 #pragma HLS INTERFACE s_axilite port=return      bundle=CTL



	/*--------------------------------------------------------
	 * Pragmas for the memory interface
	 --------------------------------------------------------*/
	// this pragma will implement the memory interface as AXI-MM
	#pragma HLS INTERFACE m_axi port=sortList offset=off

	// these pragmas together will implement the interface to the
	// memory as a single port BRAM
   //  #pragma HLS INTERFACE bram port=sortList
   //  #pragma HLS RESOURCE variable=sortList core=RAM_1P


	  for (unsigned int i = startAddr+1; i < listSize+startAddr; i++)
	  {
		int j;
		DATA_t temp;

	    temp = sortList[i];
	    j = i-1;

	    while (j >= startAddr && sortList[j] > temp)
	    {
	       sortList[j+1] = sortList[j];
		j = j-1;
	    }
	    sortList[j+1] = temp;
	  }


}
```

PES UNIVERSITY

# TESTBENCH (1/3)

```
//#define NDEBUG
#include <assert.h>


int main() {

      // test variables
    ADDR_t tb_startAddr = 13;
    LSIZE_t tb_listSize = 27;
    unsigned int tb_arraysize = 0;


      DATA_t  temp = 0;
      DATA_t tb_array[memSize];
      tb_arraysize = tb_startAddr + tb_listSize;


      // ALL SAME TEST
      cout << "---------------------------------" << endl;
      for (int i=0; i < memSize; i++) {
            tb_array[i] = 7;
      }
    insertsort(tb_startAddr, tb_listSize, tb_array);
    chkResults(tb_array, tb_startAddr, tb_listSize);
      cout << "PASSED all same case" << endl;
      cout << "---------------------------------" << endl;


    // MAX VALUE FIRST
      for (int i=0; i < memSize; i++) {
            tb_array[i] = 7;
      }
      tb_array[tb_startAddr] = 10;
      insertsort(tb_startAddr, tb_listSize, tb_array);
      chkResults(tb_array, tb_startAddr, tb_listSize);
      cout << "PASSED max value first case" << endl;
      cout << "---------------------------------" << endl;


    // MAX VALUE LAST
      for (int i=0; i < memSize; i++) {
            tb_array[i] = 7;
      }
      tb_array[tb_arraysize-1] = 10;
      insertsort(tb_startAddr, tb_listSize, tb_array);
      chkResults(tb_array, tb_startAddr, tb_listSize);
      cout << "PASSED max value last case" << endl;
      cout << "---------------------------------" << endl;


    // MIN VALUE FIRST
      for (int i=0; i < memSize; i++) {
            tb_array[i] = 12;
      }
```

PES UNIVERSITY

# TEST CASES

```
7
7
7
7
7
7
7
7
7
7
7
7
7
7
7
7
PASSED all same case
```

Vivado HLS Console

```
7
7
7
7
7
7
7
7
7
7
7
7
7
7
7
10
PASSED max value last case
```

PES UNIVERSITY

# TEST CASES

```
4
12
12
12
12
12
12
12
12
12
12
12
12
12
12
12
12
```

```
12
12
12
12
12
12
12
12
12
12
12
12
12
12
12
12
12
PASSED min value first case
```

# TESTBENCH (2/3)

```
    tb_array[tb_startAddr] = 4;
    insertsort(tb_startAddr, tb_listSize, tb_array);
    chkResults(tb_array, tb_startAddr, tb_listSize);
    cout << "PASSED min value first case" << endl;
    cout << "---------------------------------" << endl;


  // MIN VALUE LAST
    for (int i=0; i < memSize; i++) {
          tb_array[i] = 13;
    }
    tb_array[tb_arraysize-1] = 10;
    insertsort(tb_startAddr, tb_listSize, tb_array);
    chkResults(tb_array, tb_startAddr, tb_listSize);
    cout << "PASSED min value last case" << endl;
    cout << "---------------------------------" << endl;


    // INCREMENTING NUMBERS TEST
    for (int i=0; i < memSize; i++) {
          tb_array[i] = i;
    }
    insertsort(tb_startAddr, tb_listSize, tb_array);
    chkResults(tb_array, tb_startAddr, tb_listSize);
    cout << "PASSED incrementing numbers case" << endl;
    cout << "---------------------------------" << endl;


    // DECREMENTING NUMBERS TEST
    temp = memSize - 1;
    for (int i=0; i < memSize; i++) {
          tb_array[i] = temp;
          temp--;
    }
    insertsort(tb_startAddr, tb_listSize, tb_array);
    chkResults(tb_array, tb_startAddr, tb_listSize);
    cout << "PASSED decrementing numbers case" << endl;
    cout << "---------------------------------" << endl;



  // RANDOM NUMBERS TEST
    srand((unsigned)time(NULL));

    for (int i=tb_startAddr; i < memSize; i++) {
          DATA_t tb_random = rand();
          tb_array[i] = tb_random;;
    }
    insertsort(tb_startAddr, tb_listSize, tb_array);
    chkResults(tb_array, tb_startAddr, tb_listSize);
    cout << "PASSED random numbers case" << endl;
    cout << "---------------------------------" << endl;


  return 0;

}
```

PES UNIVERSITY

# TESTBENCH (3/3)

```
void chkResults(DATA_t testArray[memSize], ADDR_t startAddr, LSIZE_t listSize) {

        DATA_t  temp = 0;

        for (int i = startAddr; i < listSize+startAddr; i++) {
                assert (testArray[i] >= temp);
                temp = testArray[i];
                cout << dec << testArray[i] << endl;
        }

}
```

PES UNIVERSITY

# TEST CASES

```
13
14
15
16
17
18
19
20
21
22
23
24
25
26
```

```
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
PASSED incrementing numbers case
```

# TEST CASES

```
2419
3090
5305
5385
8749
12708
13721
14531
14570
15166
16017
17378
18447
20600
23577
24709
```

```
20600
23577
24709
25137
25570
25935
26282
27219
27744
28460
29192
30016
31278
31827
PASSED random numbers case
--------------------------------
```

PES UNIVERSITY

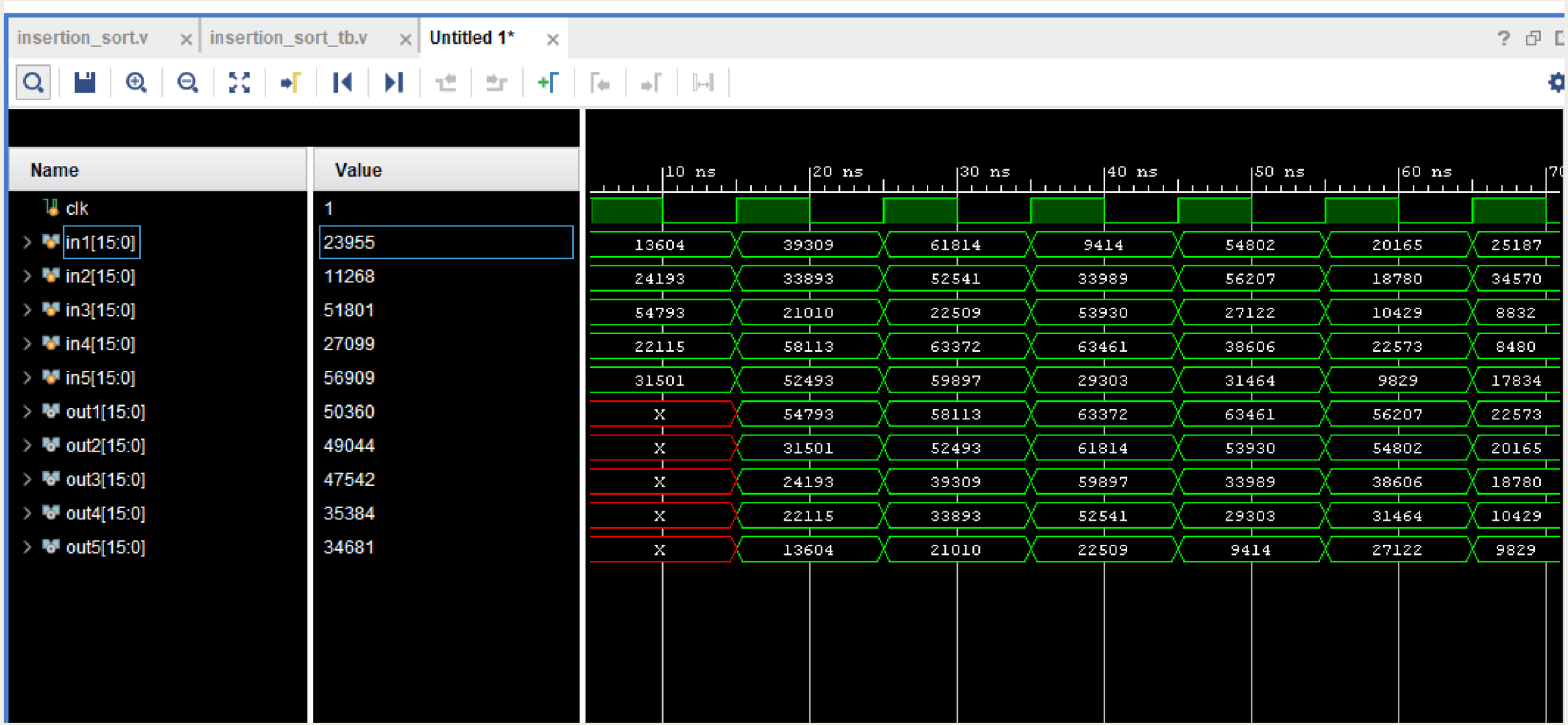
# Cosimulation Report for 'insertsort'

## Result

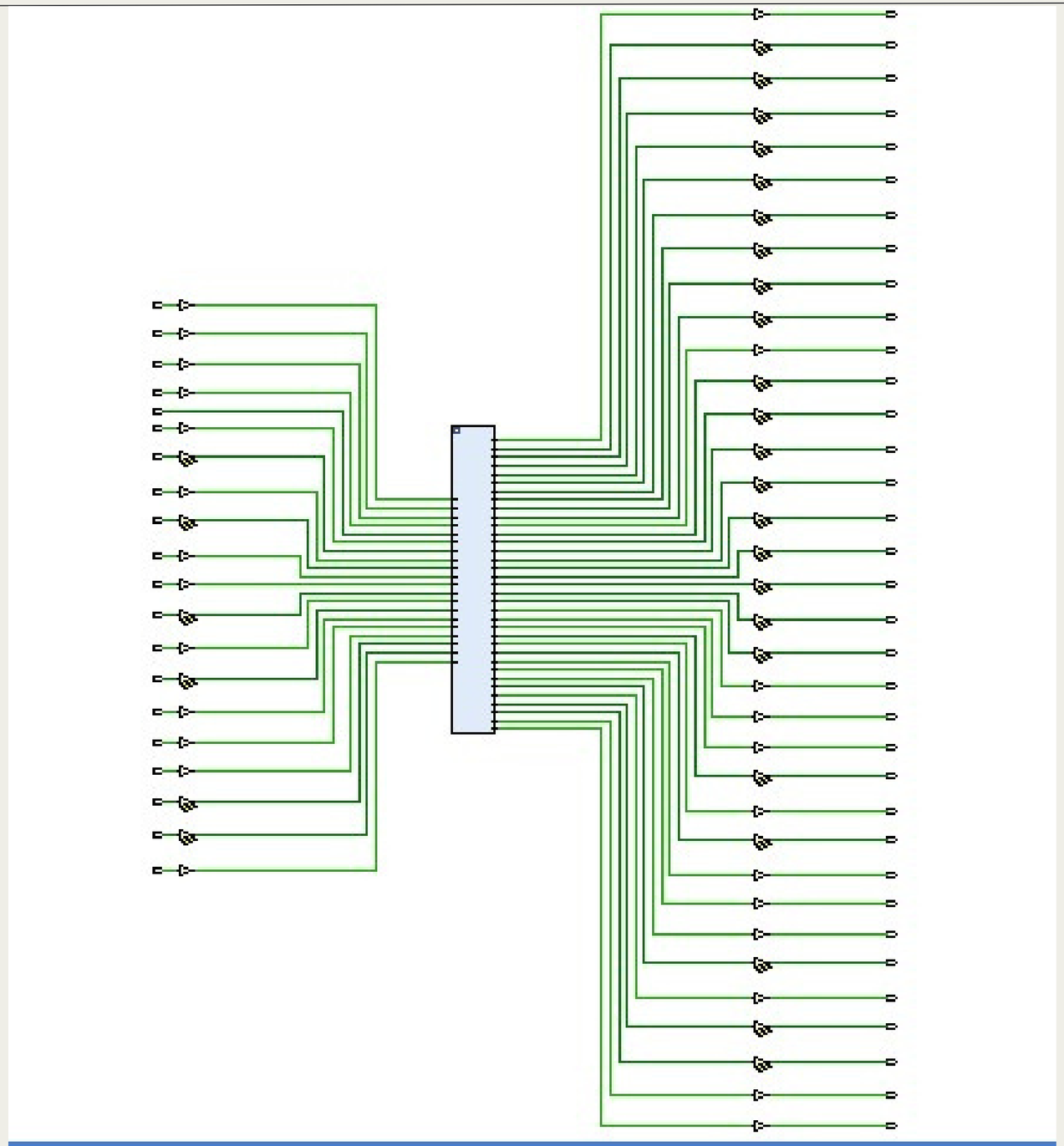
| | | Latency | | | Interval | | |
|---|---|---|---|---|---|---|---|
| RTL | Status | min | avg | max | min | avg | max |
| VHDL | NA | NA | NA | NA | NA | NA | NA |
| Verilog | Pass | 1236 | 3476 | 11783 | 1237 | 2972 | 11784 |

Export the report(.html) using the Export Wizard

# WAVEFORMS

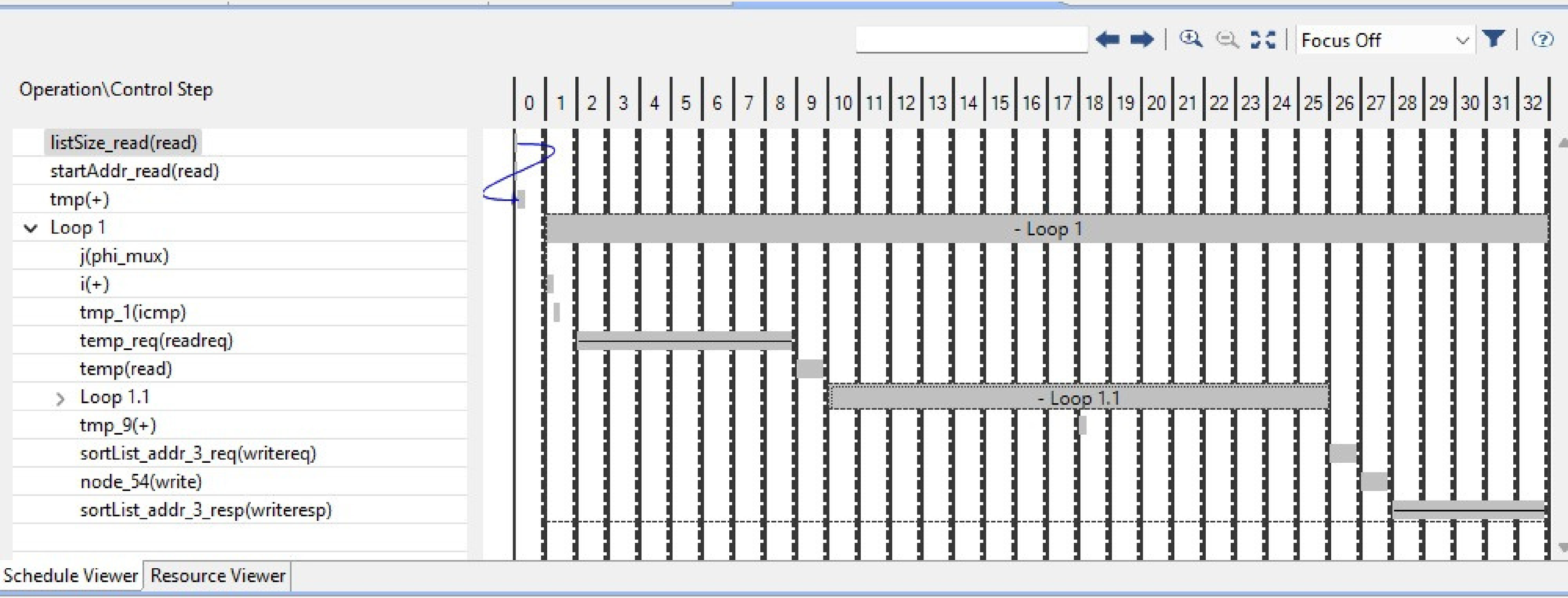

insertion_sort.v
insertion_sort_tb.v
Untitled 1*
Name
Value
clk
1
in1[15:0]
23955
in2[15:0]
11268
in3[15:0]
51801
in4[15:0]
27099
in5[15:0]
56909
out1[15:0]
50360
out2[15:0]
49044
out3[15:0]
47542
out4[15:0]
35384
out5[15:0]
34681
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
13604 39309 61814 9414 54802 20165 25187
24193 33893 52541 33989 56207 18780 34570
54793 21010 22509 53930 27122 10429 8832
22115 58113 63372 63461 38606 22573 8480
31501 52493 59897 29303 31464 9829 17834
X 54793 58113 63372 63461 56207 22573
X 31501 52493 61814 53930 54802 20165
X 24193 39309 59897 33989 38606 18780
X 22115 33893 52541 29303 31464 10429
X 13604 21010 22509 9414 27122 9829

# SYNTHESIS SCHEMATIC



PES UNIVERSITY

# SCHEDULE VIEWER




PES UNIVERSITY

# SYNTHESIS REPORT OF HLS CODE (1/7)

# Synthesis Report for 'insertsort'

## General Information

**Date:** Fri Mar 8 12:35:17 2024
**Version:** 2018.3 (Build 2405991 on Thu Dec 06 23:56:15 MST 2018)
**Project:** AHA_insertionsort
**Solution:** solution1
**Product family:** aartix7
**Target device:** xa7a12tcsg325-1q

## Performance Estimates

- **Timing (ns)**
  - **Summary**

| Clock | Target | Estimated | Uncertainty |
|---|---|---|---|
| ap_clk | 10.00 | 8.750 | 1.25 |

- **Latency (clock cycles)**
  - **Summary**

| Latency | | Interval | | Type |
|---|---|---|---|---|
| min | max | min | max | |
| ? | ? | ? | ? | none |

  - **Detail**
    - **Instance**

N/A

    - **Loop**

| Loop Name | Latency | | Iteration Latency | Initiation Interval | | Trip Count | Pipelined |
|---|---|---|---|---|---|---|---|
| | min | max | | achieved | target | | |
| - Loop 1 | ? | ? | ? | - | - | ? | no |
| + Loop 1.1 | ? | ? | 16 | - | - | ? | no |

PES UNIVERSITY

# SYNTHESIS REPORT OF HLS CODE (2/7)

## Utilization Estimates

- **Summary**

| Name | BRAM_18K | DSP48E | FF | LUT |
|---|---|---|---|---|
| DSP | - | - | - | - |





| | | | | |
|---|---|---|---|---|
| Expression | - | - | 0 | 218 |
| FIFO | - | - | - | - |
| Instance | 2 | - | 624 | 748 |
| Memory | - | - | - | - |
| Multiplexer | - | - | - | 288 |
| Register | - | - | 325 | - |
| Total | 2 | 0 | 949 | 1254 |
| Available | 40 | 40 | 16000 | 8000 |
| Utilization (%) | 5 | 0 | 5 | 15 |

- **Detail**
  - **Instance**

| Instance | Module | BRAM_18K | DSP48E | FF | LUT |
|---|---|---|---|---|---|
| insertsort_CTL_s_axi_U | insertsort_CTL_s_axi | 0 | 0 | 112 | 168 |
| insertsort_sortList_m_axi_U | insertsort_sortList_m_axi | 2 | 0 | 512 | 580 |
| Total | | 2 | 2 | 0 | 624 | 748 |

# SYNTHESIS REPORT OF HLS CODE (3/7)

- **Expression**

| Variable Name | Operation | DSP48E | FF | LUT | Bitwidth P0 | Bitwidth P1 |
|---|---|---|---|---|---|---|
| grp_fu_135_p2 | + | 0 | 0 | 39 | 32 | 1 |
| i_fu_151_p2 | + | 0 | 0 | 39 | 32 | 1 |
| j_1_fu_192_p2 | + | 0 | 0 | 39 | 32 | 2 |
| tmp_fu_145_p2 | + | 0 | 0 | 39 | 32 | 32 |
| ap_block_state11_io | and | 0 | 0 | 8 | 1 | 1 |
| tmp_1_fu_157_p2 | icmp | 0 | 0 | 18 | 32 | 32 |
| tmp_3_fu_172_p2 | icmp | 0 | 0 | 18 | 32 | 32 |
| tmp_5_fu_188_p2 | icmp | 0 | 0 | 18 | 32 | 32 |
| Total | 8 | 0 | 0 | 218 | 225 | 133 |

- **Multiplexer**

| Name | LUT | Input Size | Bits | Total Bits |
|---|---|---|---|---|
| ap_NS_fsm | 153 | 34 | 1 | 34 |

file:///C:/Users/appaa/Downloads/solution1_insertsort_csynth.html

PES UNIVERSITY

# SYNTHESIS REPORT OF HLS CODE (4/7)



Synthesis Report for 'insertsort'

| | | | | |
|---|---|---|---|---|
| ap_sig_ioackin_sortList_ARREADY | 9 | 2 | 1 | 2 |
| ap_sig_ioackin_sortList_AWREADY | 9 | 2 | 1 | 2 |
| ap_sig_ioackin_sortList_WREADY | 9 | 2 | 1 | 2 |
| j1_reg_124 | 9 | 2 | 32 | 64 |
| j_reg_114 | 9 | 2 | 32 | 64 |
| sortList_ARADDR | 15 | 3 | 32 | 96 |
| sortList_AWADDR | 15 | 3 | 32 | 96 |
| sortList_WDATA | 15 | 3 | 32 | 96 |
| sortList_blk_n_AR | 9 | 2 | 1 | 2 |
| sortList_blk_n_AW | 9 | 2 | 1 | 2 |
| sortList_blk_n_B | 9 | 2 | 1 | 2 |
| sortList_blk_n_R | 9 | 2 | 1 | 2 |
| sortList_blk_n_W | 9 | 2 | 1 | 2 |
| Total | 288 | 63 | 169 | 466 |

# SYNTHESIS REPORT OF HLS CODE (5/7)

- Register

| Name | FF | LUT | Bits | Const Bits |
|---|---|---|---|---|
| ap_CS_fsm | 33 | 0 | 33 | 0 |
| ap_reg_ioackin_sortList_ARREADY | 1 | 0 | 1 | 0 |
| ap_reg_ioackin_sortList_AWREADY | 1 | 0 | 1 | 0 |
| ap_reg_ioackin_sortList_WREADY | 1 | 0 | 1 | 0 |
| i_reg_231 | 32 | 0 | 32 | 0 |
| j1_reg_124 | 32 | 0 | 32 | 0 |
| j_1_reg_271 | 32 | 0 | 32 | 0 |
| j_reg_114 | 32 | 0 | 32 | 0 |
| reg_141 | 32 | 0 | 32 | 0 |
| sortList_addr_1_read_reg_262 | 32 | 0 | 32 | 0 |
| startAddr_read_reg_220 | 32 | 0 | 32 | 0 |
| temp_reg_246 | 32 | 0 | 32 | 0 |
| tmp_3_reg_252 | 1 | 0 | 1 | 0 |
| tmp_reg_226 | 32 | 0 | 32 | 0 |
| Total | 325 | 0 | 325 | 0 |

# SYNTHESIS REPORT OF HLS CODE (6/7)

## Interface

- Summary

| RTL Ports | Dir | Bits | Protocol | Source Object | C Type |
|---|---|---|---|---|---|
| s_axi_CTL_AWVALID | in | 1 | s_axi | CTL | scalar |
| s_axi_CTL_AWREADY | out | 1 | s_axi | CTL | scalar |
| s_axi_CTL_AWADDR | in | 5 | s_axi | CTL | scalar |
| s_axi_CTL_WVALID | in | 1 | s_axi | CTL | scalar |
| s_axi_CTL_WREADY | out | 1 | s_axi | CTL | scalar |
| s_axi_CTL_WDATA | in | 32 | s_axi | CTL | scalar |
| s_axi_CTL_WSTRB | in | 4 | s_axi | CTL | scalar |

PES UNIVERSITY

# SYNTHESIS REPORT OF HLS CODE (7/7)

| | | | | | |
|---|---|---|---|---|---|
| s_axi_CTL_ARVALID | in | 1 | s_axi | CTL | scalar |
| s_axi_CTL_ARREADY | out | 1 | s_axi | CTL | scalar |
| s_axi_CTL_ARADDR | in | 5 | s_axi | CTL | scalar |
| s_axi_CTL_RVALID | out | 1 | s_axi | CTL | scalar |
| s_axi_CTL_RREADY | in | 1 | s_axi | CTL | scalar |
| s_axi_CTL_RDATA | out | 32 | s_axi | CTL | scalar |
| s_axi_CTL_RRESP | out | 2 | s_axi | CTL | scalar |
| s_axi_CTL_BVALID | out | 1 | s_axi | CTL | scalar |
| s_axi_CTL_BREADY | in | 1 | s_axi | CTL | scalar |
| s_axi_CTL_BRESP | out | 2 | s_axi | CTL | scalar |
| ap_clk | in | 1 | ap_ctrl_hs | insertsort | return value |
| ap_rst_n | in | 1 | ap_ctrl_hs | insertsort | return value |
| interrupt | out | 1 | ap_ctrl_hs | insertsort | return value |
| m_axi_sortList_AWVALID | out | 1 | m_axi | sortList | pointer |
| m_axi_sortList_AWREADY | in | 1 | m_axi | sortList | pointer |
| m_axi_sortList_AWADDR | out | 32 | m_axi | sortList | pointer |
| m_axi_sortList_AWID | out | 1 | m_axi | sortList | pointer |
| m_axi_sortList_AWLEN | out | 8 | m_axi | sortList | pointer |
| m_axi_sortList_AWSIZE | out | 3 | m_axi | sortList | pointer |
| m_axi_sortList_AWBURST | out | 2 | m_axi | sortList | pointer |
| m_axi_sortList_AWLOCK | out | 2 | m_axi | sortList | pointer |
| m_axi_sortList_AWCACHE | out | 4 | m_axi | sortList | pointer |
| m_axi_sortList_AWPROT | out | 3 | m_axi | sortList | pointer |
| m_axi_sortList_AWQOS | out | 4 | m_axi | sortList | pointer |
| m_axi_sortList_AWREGION | out | 4 | m_axi | sortList | pointer |
| m_axi_sortList_AWUSER | out | 1 | m_axi | sortList | pointer |
| m_axi_sortList_WVALID | out | 1 | m_axi | sortList | pointer |
| m_axi_sortList_WREADY | in | 1 | m_axi | sortList | pointer |
| m_axi_sortList_WDATA | out | 32 | m_axi | sortList | pointer |
| m_axi_sortList_WSTRB | out | 4 | m_axi | sortList | pointer |
| m_axi_sortList_WLAST | out | 1 | m_axi | sortList | pointer |
| m_axi_sortList_WID | out | 1 | m_axi | sortList | pointer |
| m_axi_sortList_WUSER | out | 1 | m_axi | sortList | pointer |
| m_axi_sortList_ARVALID | out | 1 | m_axi | sortList | pointer |
| m_axi_sortList_ARREADY | in | 1 | m_axi | sortList | pointer |
| m_axi_sortList_ARADDR | out | 32 | m_axi | sortList | pointer |
| m_axi_sortList_ARID | out | 1 | m_axi | sortList | pointer |
| m_axi_sortList_ARLEN | out | 8 | m_axi | sortList | pointer |
| m_axi_sortList_ARSIZE | out | 3 | m_axi | sortList | pointer |
| m_axi_sortList_ARBURST | out | 2 | m_axi | sortList | pointer |
| m_axi_sortList_ARLOCK | out | 2 | m_axi | sortList | pointer |
| m_axi_sortList_ARCACHE | out | 4 | m_axi | sortList | pointer |
| m_axi_sortList_ARPROT | out | 3 | m_axi | sortList | pointer |
| m_axi_sortList_ARQOS | out | 4 | m_axi | sortList | pointer |

PES UNIVERSITY

# IMPLEMENTATION REPORT OF EXPORTED RTL (PLACE + ROUTE)

## Export Report for 'insertsort'

### General Information

**Report date:** Mon Mar 08 23:19:43 +0530 2021
**Project:** AHA_insertionsort
**Solution:** solution1
**Device target:** xa7a12tcsg325-1q
**Implementation tool:** Xilinx Vivado v.2018.3

### Resource Usage

| | Verilog |
|---|---|
| SLICE | 389 |
| LUT | 955 |
| FF | 1426 |
| DSP | 0 |
| BRAM | 2 |
| SRL | 40 |

### Final Timing

| | Verilog |
|---|---|
| CP required | 10.000 |
| CP achieved post-synthesis | 5.988 |
| CP achieved post-implementation | 6.353 |

Timing met

PES UNIVERSITY

# T H A N K  Y O U