

CatapultC Synthesis

Dhruv Jain

Brief Explanation

The screenshot displays the Xilinx ISE software interface for a project named 'sort.v4'. The interface is divided into several panes:

- Task Bar:** Contains a list of synthesis tasks: Add Input Files, Setup Design, Architecture Constr..., Resource Constraints, Schedule, and Generate RTL.
- Project Files:** Shows the project structure for 'sort.v4 (Passed Extract)', including Input Files (main.cpp), Output Files, and Synthesis.
- Design: sort:** Displays a block diagram of the 'sort' core. The diagram shows a hierarchy: 'sort' contains 'Interface Control' and 'Interface'. 'Interface' contains 'core', which contains 'main', which contains 'for', which contains 'for:for'.
- C-CORE Settings:** A panel on the right with settings for the C-CORE. It includes a checkbox for 'Create Combinational Block', 'Effort Level' set to 'medium', 'Input Delay', 'Output Delay', and a checkbox for 'Safe FSM'. Below these are 'Low-Power Settings' with checkboxes for 'Enable Clock Gating' and 'Idle Signal:'. At the bottom are 'Settings' and 'Advanced' tabs.
- Transcript:** A panel at the bottom showing the compilation process. It includes a status bar with '0 Errors', '0 Warnings', '0 Infos', '# 0 Comments', and '0 Commands'. The transcript text shows the following messages:

```
# Message
# Reading component library '$MGC_HOME\pkgs\siflibs\psr2009a_up2\mgc_Xilinx-VIRTEX-II-4s1_beh_psr.lib' [mgc_Xilinx-VI
# Reading component library '$MGC_HOME\pkgs\siflibs\mgc_busdefs.lib' [mgc_busdefs]...
# Reading component library '$MGC_HOME\pkgs\siflibs\stdops.lib' [STDOPS]...
# Reading component library '$MGC_HOME\pkgs\siflibs\mgc_ionport.lib' [mgc_ionport]...
```

Selection Sort

- ▶ Selection Sort code for 8 numbers was written in CatapultC.
 - ▶ Output simulated with Xilinx 12.1 ise.
 - ▶ All other parameters were kept at their default values.

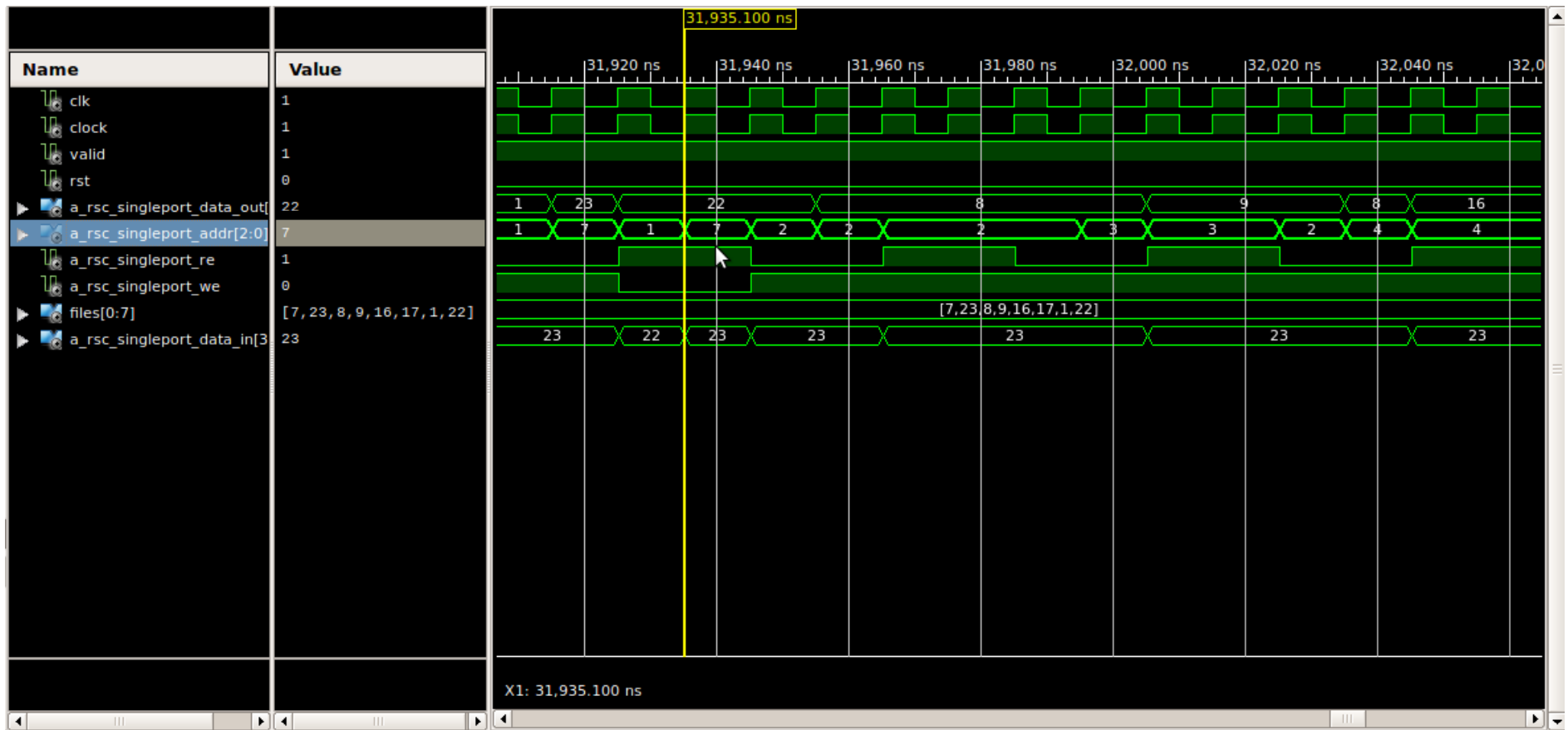
Parameters	Value
Board	Virtex 4
RAM	Single port Dist. RAM
Frequency	100Mhz
Minimum Input Arrival Time	7.007ns*
Maximum output time after time	10.999ns*
Primary Goal	Latency

*these parameters not taken care of when compiling with CatapultC so the design had to be modified again.
The results remain nearly the same.

** further optimized



Simulation with ISE 12.1



Results for Selection Sort

Parameter	CatapultC	Xilinx Ise 12.1
Maximum Frequency	100Mhz (~)	149.5390Mhz**
Critical Path Delay	10.882ns	6.678ns
Total Cycles needed for appearance of output	138	138
Slices/Area	Post DP and FSM: 7% Post Assignment: 13%	7%



Results for Selection Sort (CatapultC)

- ▶ Comparison of results with loop kept rolled (default) and full loop unrolling for 8 numbers

Parameter	Without unroll	With unroll
Total time	430ns	1.42us
Cycles	43	138
Area Score	297.3 (59%, 30%, 11%)*	2356.0 (70%, 18%, 12%)*

- ▶ Results are not according to expectations
 - ▶ Possible reason: for small datapath, major time consumed by FSM

*Percentage used by (MUX, Func, Logic)



Results for Selection Sort (CatapultC)

- ▶ Comparison of results with loop kept rolled (default) and full loop unrolling for 99 numbers

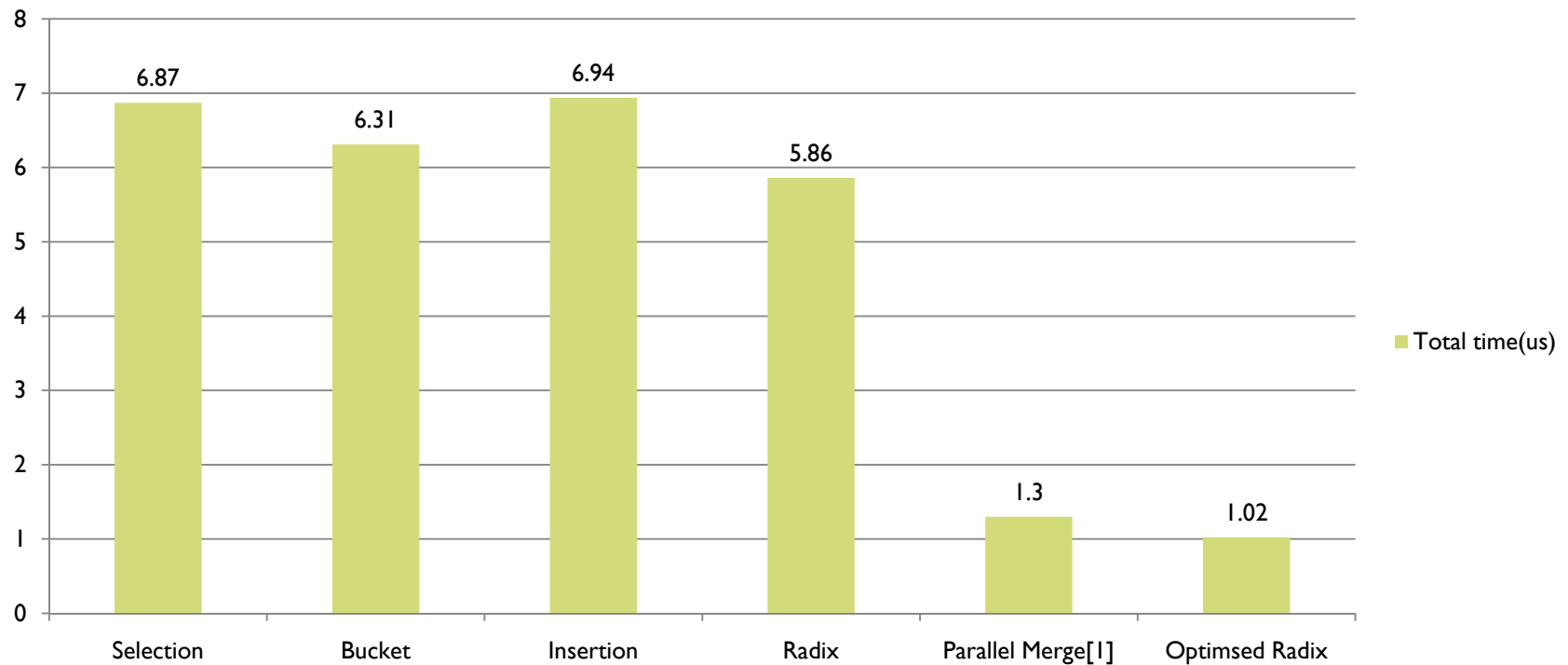
Parameter	Without unroll	With unroll
Total time	6.87us	5.42us
Area Score	365.0 (59%, 30%, 11%)	27228.7 (73%, 9%, 11%)

- ▶ These results show that our explanation was indeed correct.
 - ▶ Datapath has increased -> Less contribution from FSM



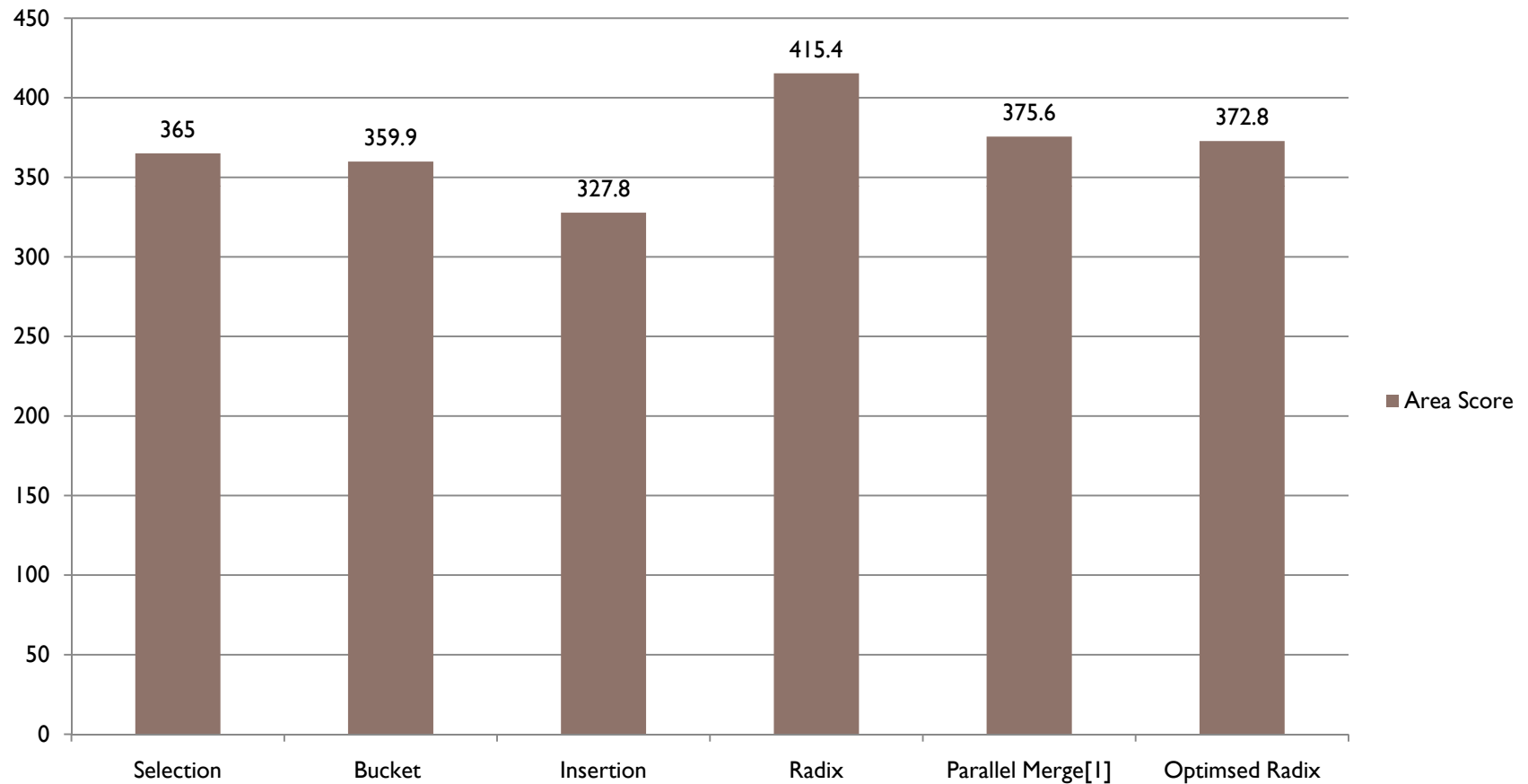
Sorting Results for different sort

- ▶ All sorting algorithms manually coded
 - ▶ Possible lack of optimization
 - ▶ All are without unroll and primary goal was latency. Numbers = 99



Sorting Results for different sorts

- ▶ Memory results were as per the expectations



Radix Sort

```
#pragma hls_design top

#define MAX 99

void radixsort(int a[MAX])
{
    int i,b[MAX],m=0,exp=1;

    for(i=0;i<MAX;i++)
    {
        if(a[i]>m)
            m=a[i];
    }

    while(m/exp>0)
    {
        int bucket[10]={0};
        for(i=0;i<MAX;i++)
            bucket[a[i]/exp%10]++;
        for(i=1;i<10;i++)
            bucket[i]+=bucket[i-1];
        for(i=MAX-1;i>=0;i--)
            b[--bucket[a[i]/exp%10]]=a[i];
        for(i=0;i<MAX;i++)
            a[i]=b[i];
        exp*=10;
    }
}
```

Optimized Radix Sort

```
void radixsort(ac_int<8, false> a[MAX])
{
    ac_int<8, false> i;
    ac_int<8, false> m = 99;
    ac_int<9, false> exp = 1;
    ac_int<1, false> flag = 0;

    /* m can be kept as fixed */

    while(m/exp>0)
    {
        int bucket[2]={0};                //Only two buckets for each bit (2 base)
                                           //Bucket is used as an index so its type is kept as int to prevent type casting again

        for(i=0;i<MAX;i++)
            bucket[a[i] || exp]++;        //Division replaced by bitwise OR (for saving type conversions)

        //for(i=1;i<10;i++)                //No need for this loop

        bucket[1]+=bucket[0];

        for(i=MAX-1;i>=0;i--)
            b[--bucket[a[i] || exp]]=a[i];

        //for(i=0;i<MAX;i++)                //No need for assignment. Just reuse by copy/pasting the code
        //    a[i]=b[i];

        exp = (ac_int<9, false>) exp << 1;    //Multiplication replaced by left shift
    }
}
```



Optimized Radix Sort Contd..

```
if(m/exp > 0)                                //If MAX is under even multiple of 2, this can be removed too!
{
    bucket[1]=bucket[0]= {0};
    for(i=0;i<MAX;i++)
        bucket[b[i] || exp]++;

    //for(i=1;i<10;i++)

    bucket[1]+=bucket[0];

    for(i=MAX-1;i>=0;i--)
        a[--bucket[b[i] || exp]]=b[i];

    //for(i=0;i<MAX;i++)
    //  a[i]=b[i];
    exp = (ac_int<9, false>) exp << 1;
}
else    flag = 1;
}

if(flag == 1)                                // Copy at the end if required. Woudn't make much difference in time
    for(i=0;i<MAX;i++)
        a[i]=b[i];
}
```



Comparison with VHDL implementation of systolic sort @DHD Project

▶ Xilinx Ise 9.1 results

Vertex II, Single Port RAM

Clock frequency: 162.824Mhz

Critical Path delay: 6.142ns

Minimum input arrival time before clock: 1.802ns

Maximum output required time after clock: 3.473ns

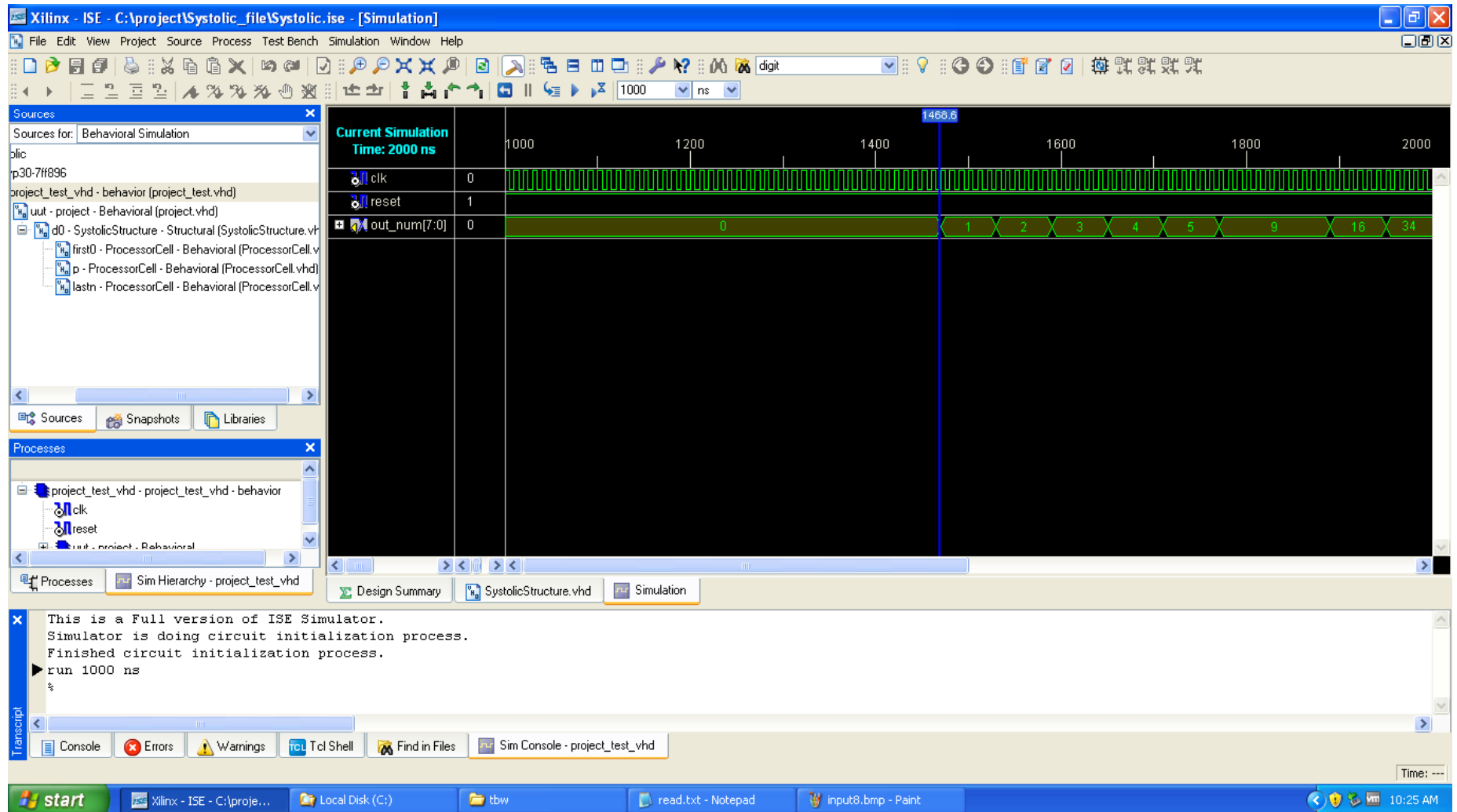
▶ These parameters were filled for CatapultC optimized radix sort

▶ Results

Parameter	Xiilnx 9.1 Ise	CatapultC
Total time for appearance of the entire output	6.14us	9.21us



Simulation for Systolic Sort ISE 9.1



Conclusions

- ▶ CatapultC is a great tool for HLS since it supports variable bit length.
- ▶ Various possible configuration options for optimisation
 - ▶ Optimization Parameter (Area/Delay)
 - ▶ Interface (RAM/Registers)
 - ▶ Effort Level
 - ▶ Loop unrolling
 - ▶ Clock manipulation
 - ▶ Power Efficient Design (Clock Gating)
 - ▶ Input/Output Delays
 - ▶ Handshaking signals
- ▶ “Scheduling” window for verification
- ▶ But, still manual coding with HDL is most efficient.



References

[1] Szell, Feher. “ Parallel sorting algorithms in FPGA”. Online. [Available]

http://home.mit.bme.hu/~szell/szell_parallel_sorting_2006.pdf

- ▶ Data clustering for bioinformatics using parallel merge sort

[2] Catapult C manual reference.

- ▶ Available inside CatapultC Tool's help section

Special thanks to Anand Isaac

