Zynq를 활용한 SoC / FPGA 설계

HDL 및 HLS를 이용한 설계 이해 및 실습

HUINS
Human Intelligent System

CONTENTS

HUINS
Human Intelligent System

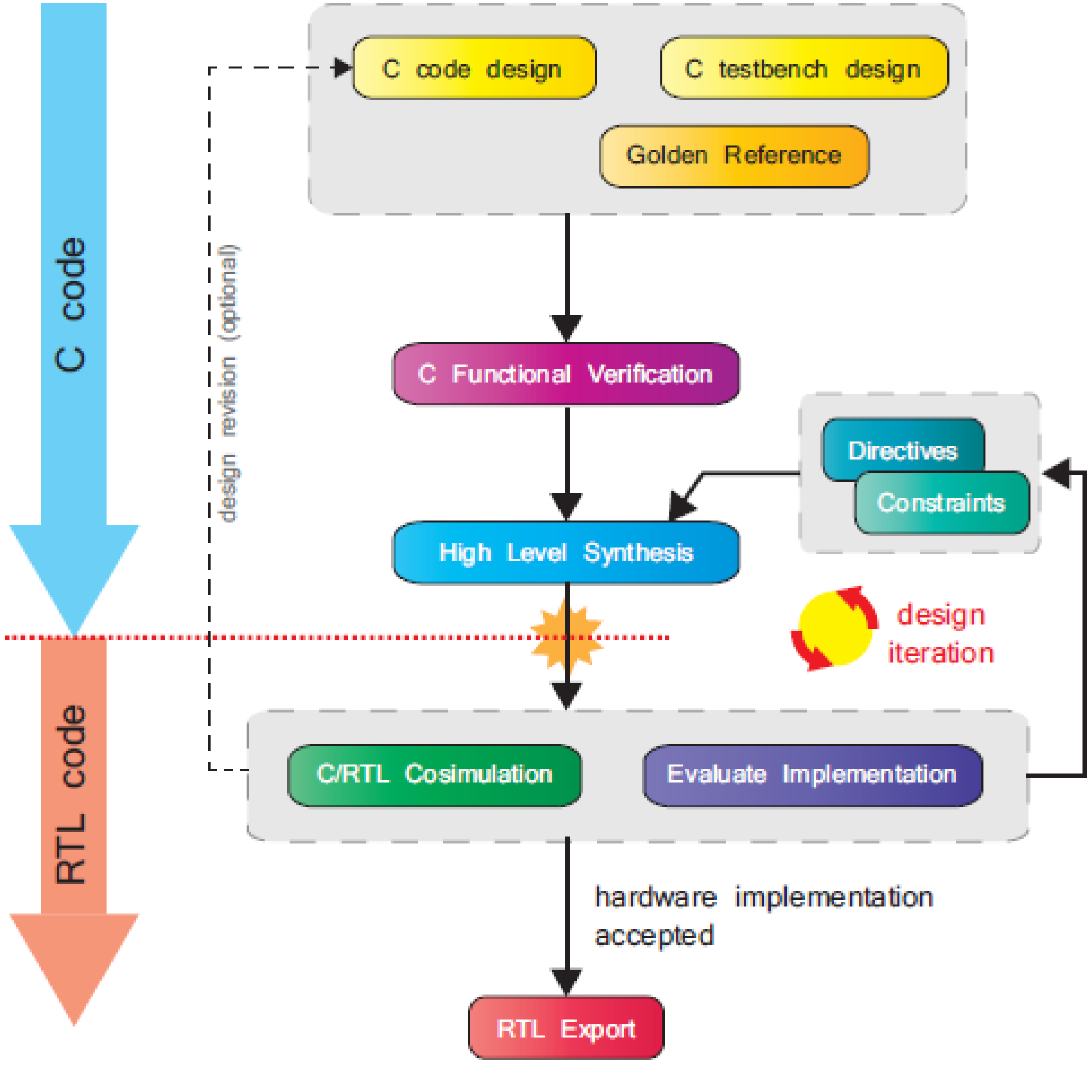
HLS를 이용한 FPGA 설계

CONTENTS

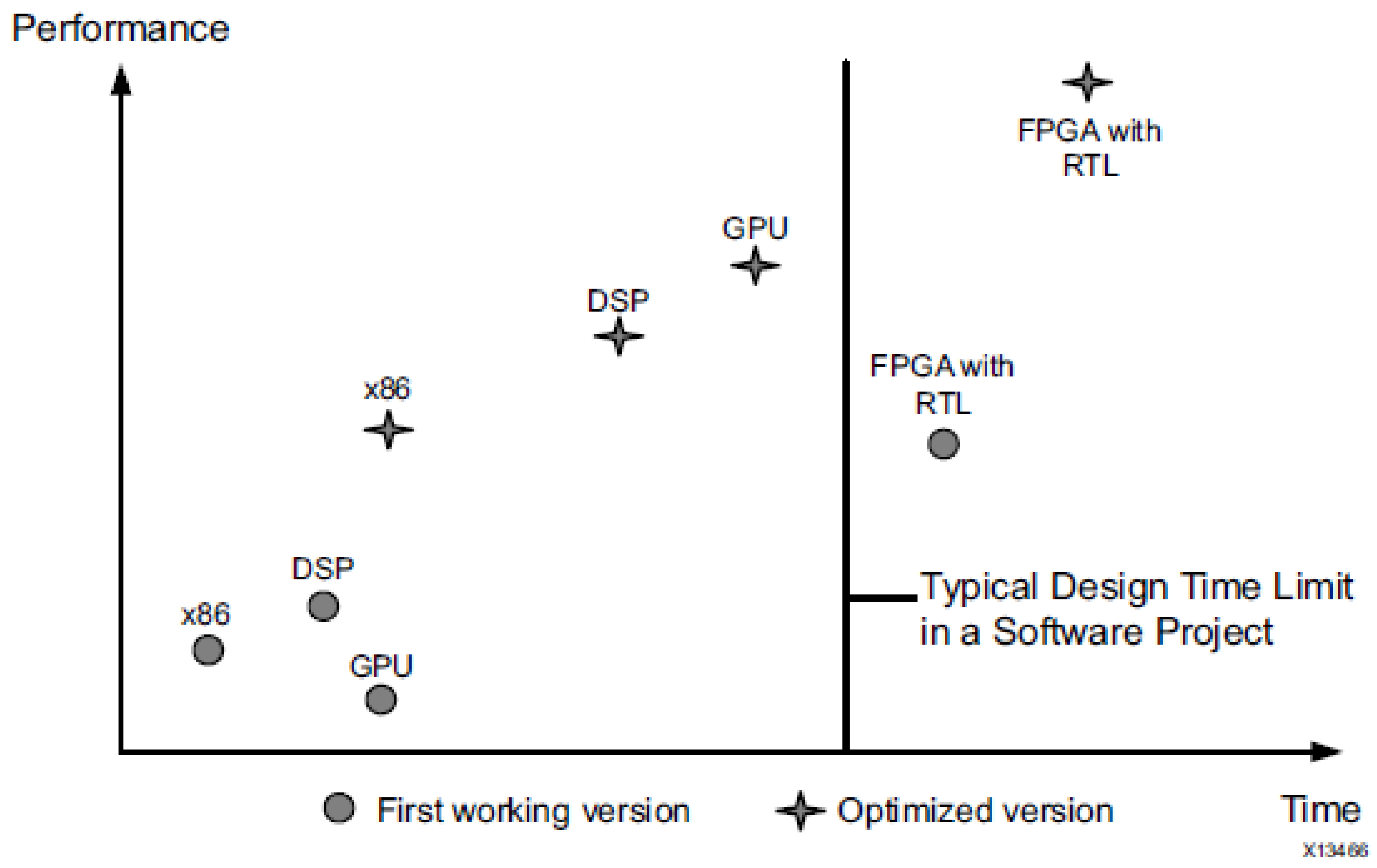
HUINS
Human Intelligent System

01. HLS(High-Level Synthesis)

- Xilinx에서 개발한 Tool
- C-level Design / Verify
  - 고급 언어(C/C++/System C)로 작성된 소스코드를 RTL 레벨로 변환
  - C-Level에서의 디자인 검증
- HDL 기반 설계 대비 생산성 향상
- 시스템 기준 → HW 가속으로 성능 향상



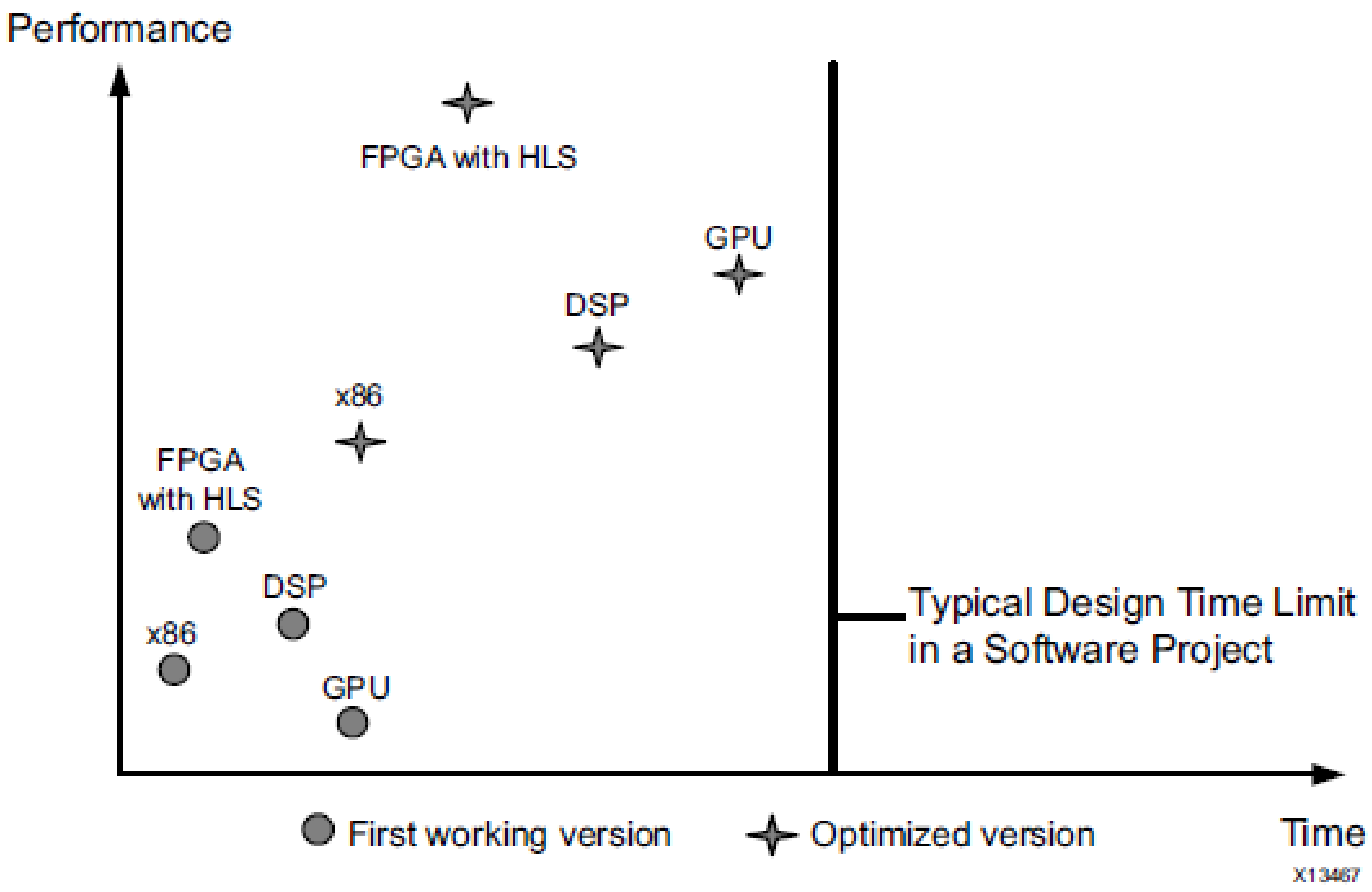
Why HLS?



시간 vs. 성능

HUINS
Human Intelligent System

Why HLS?



시간 vs. 성능

HUINS
Human Intelligent System

프로세서 vs. FPGA

프로세서에서의 프로그램 실행
→ 여러 개의 명령어의 집합

데이터 Read/Save?

z = a + b; —컴파일러(gcc) 변환→ ADD $R1, $R2, $R3 —레지스터 데이터 읽기/저장 명령어 생성→

LD	a, $R1
LD	b, $R2
ADD	$R1, $R2, $R3
ST	$R3, c

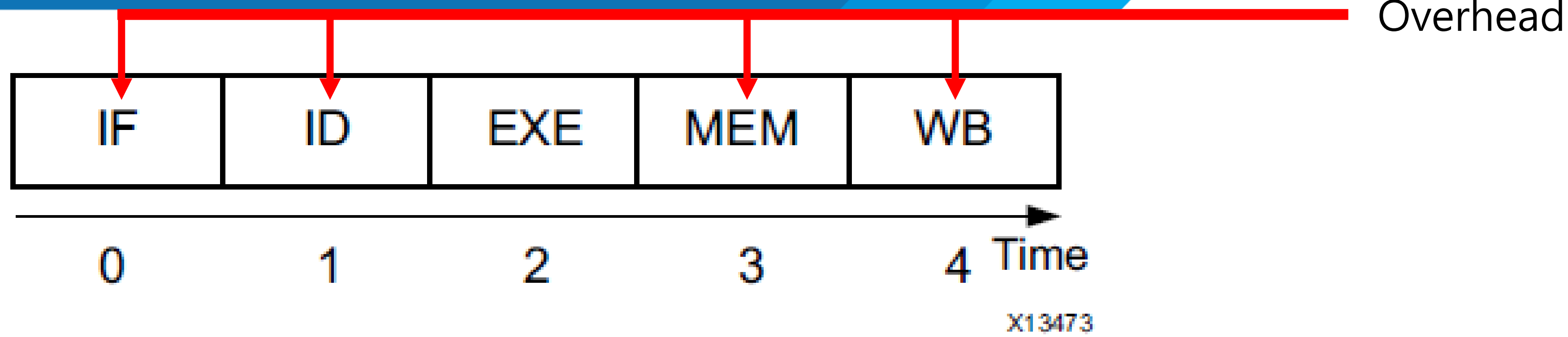
a, b, c Load에 필요한 클럭 수

캐시 내부 : ~100

DRAM : 수 백, 수 천

HUINS
Human Intelligent System

프로세서 vs. FPGA



프로세서에서의 명령어 실행 주기

1. Instruction fetch (IF)
2. Instruction decode (ID)
3. Execute (EXE)
4. Memory operations (MEM)
5. Write back (WB)

HUINS
Human Intelligent System

프로세서 vs. FPGA

다중 명령어 실행 유닛을 가진 프로세서

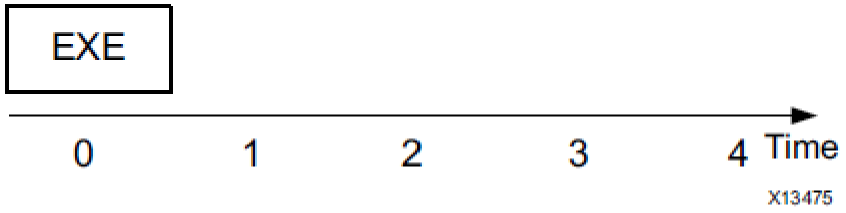
Instruction Pipelining

- 최적의 경우 한 클럭 사이클마다 하나의 EXE 실행

HUINS
Human Intelligent System

프로세서 vs. FPGA

FPGA(HLS)에서의 프로그램 실행
→단일 명령어 실행

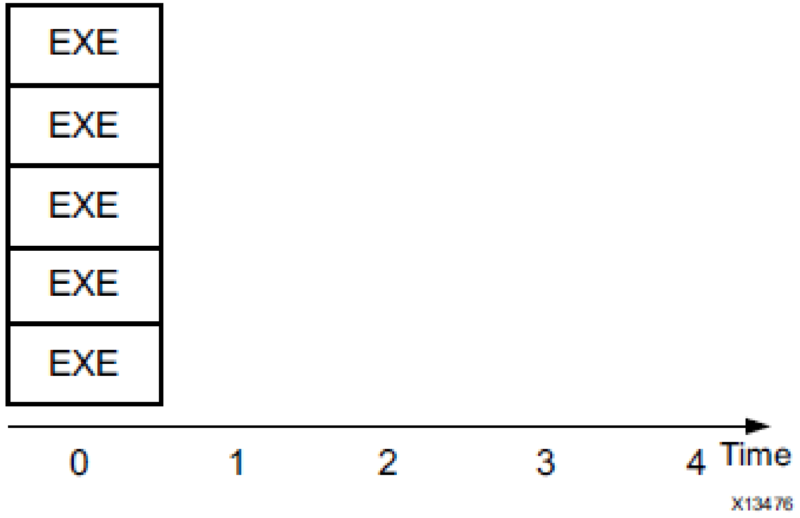


FPGA에서의 명령어 실행

동일한 계산 구조(ALU)가 아닌, target specific한 계산 회로 구현

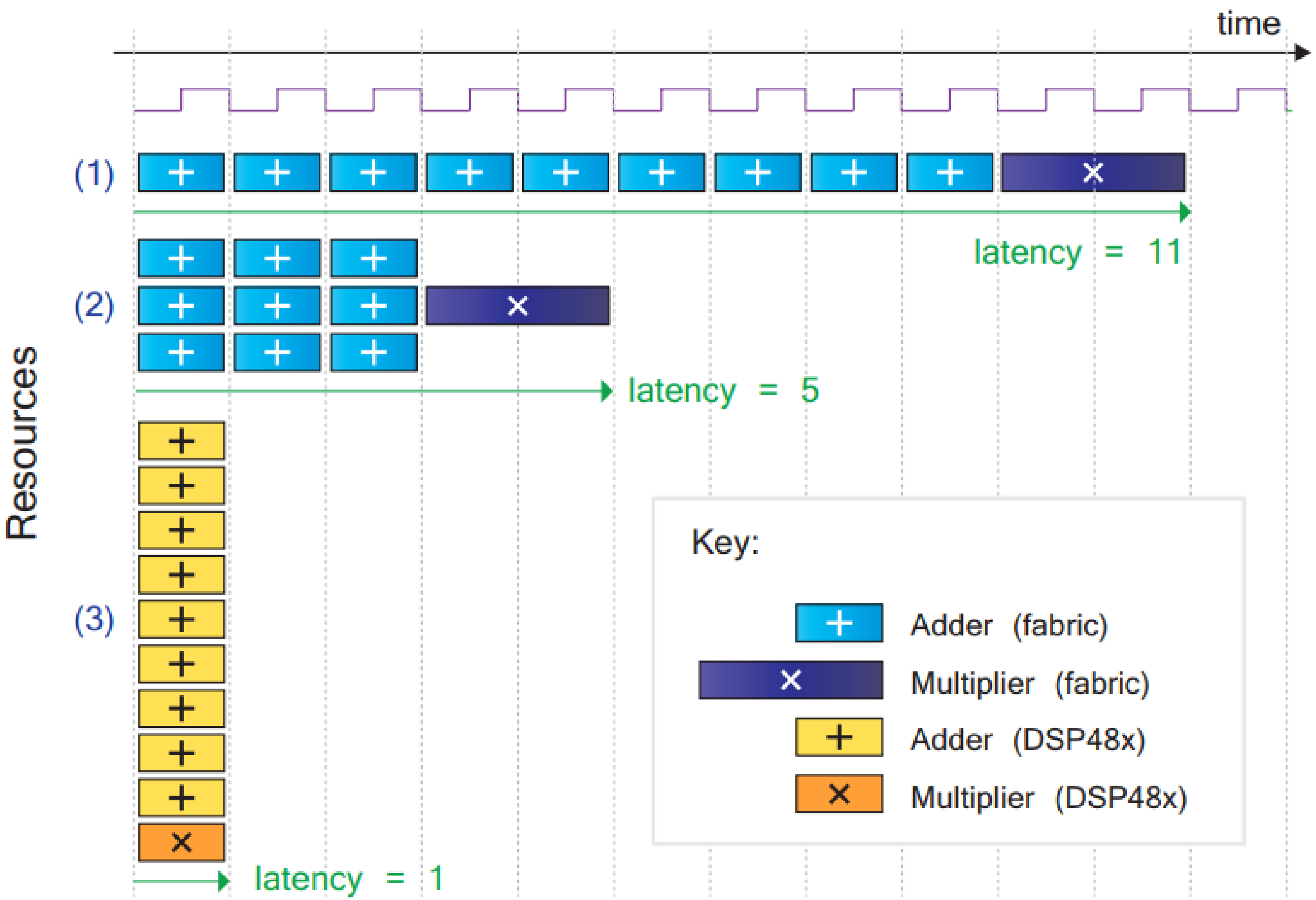
프로세서 vs. FPGA

FPGA(HLS)에서의 다중 명령어 실행
→오버헤드 없이 동시에 여러 명령어를 실행할 수 있음



FPGA에서의 다중 명령어 실행

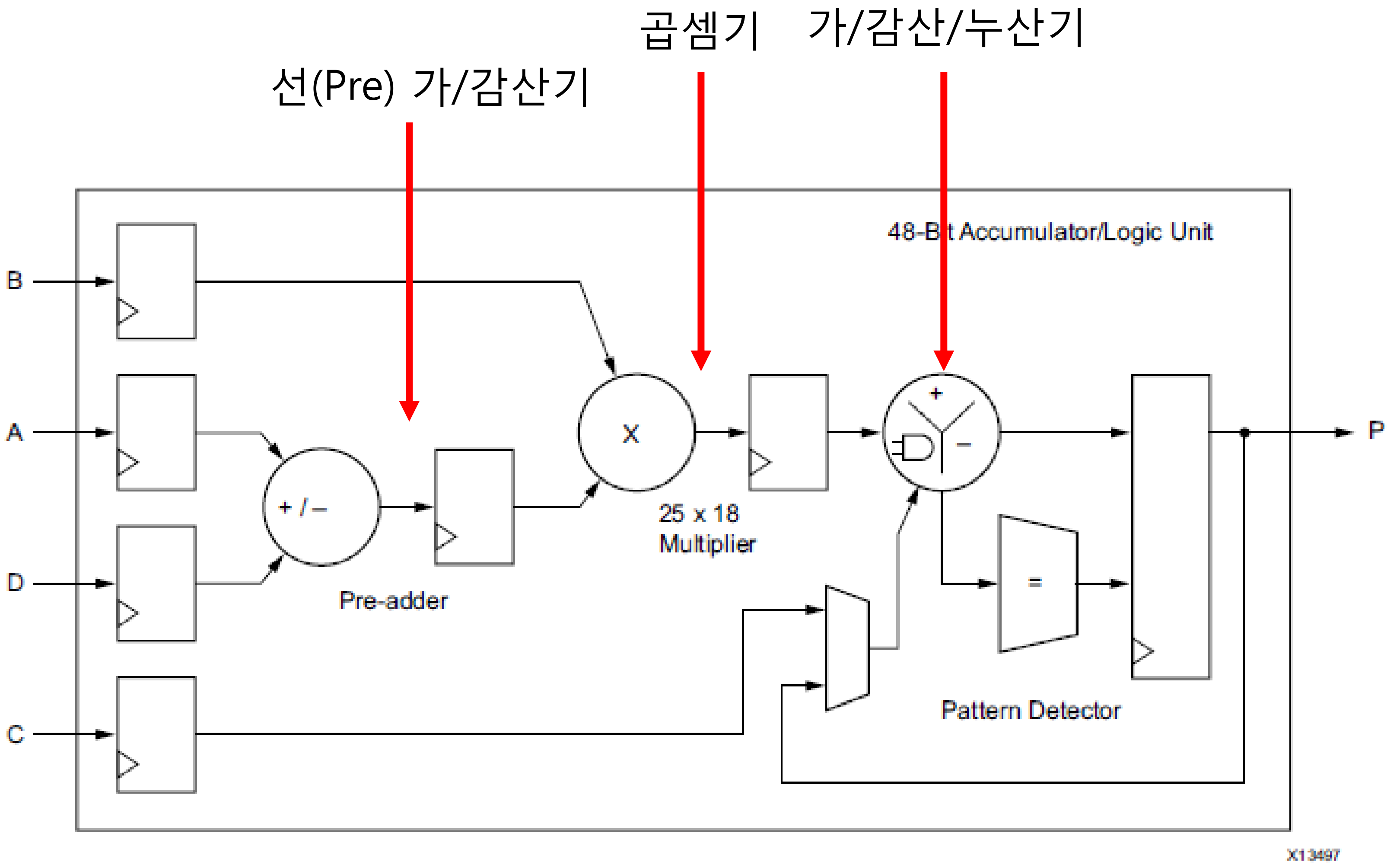
Parallelization



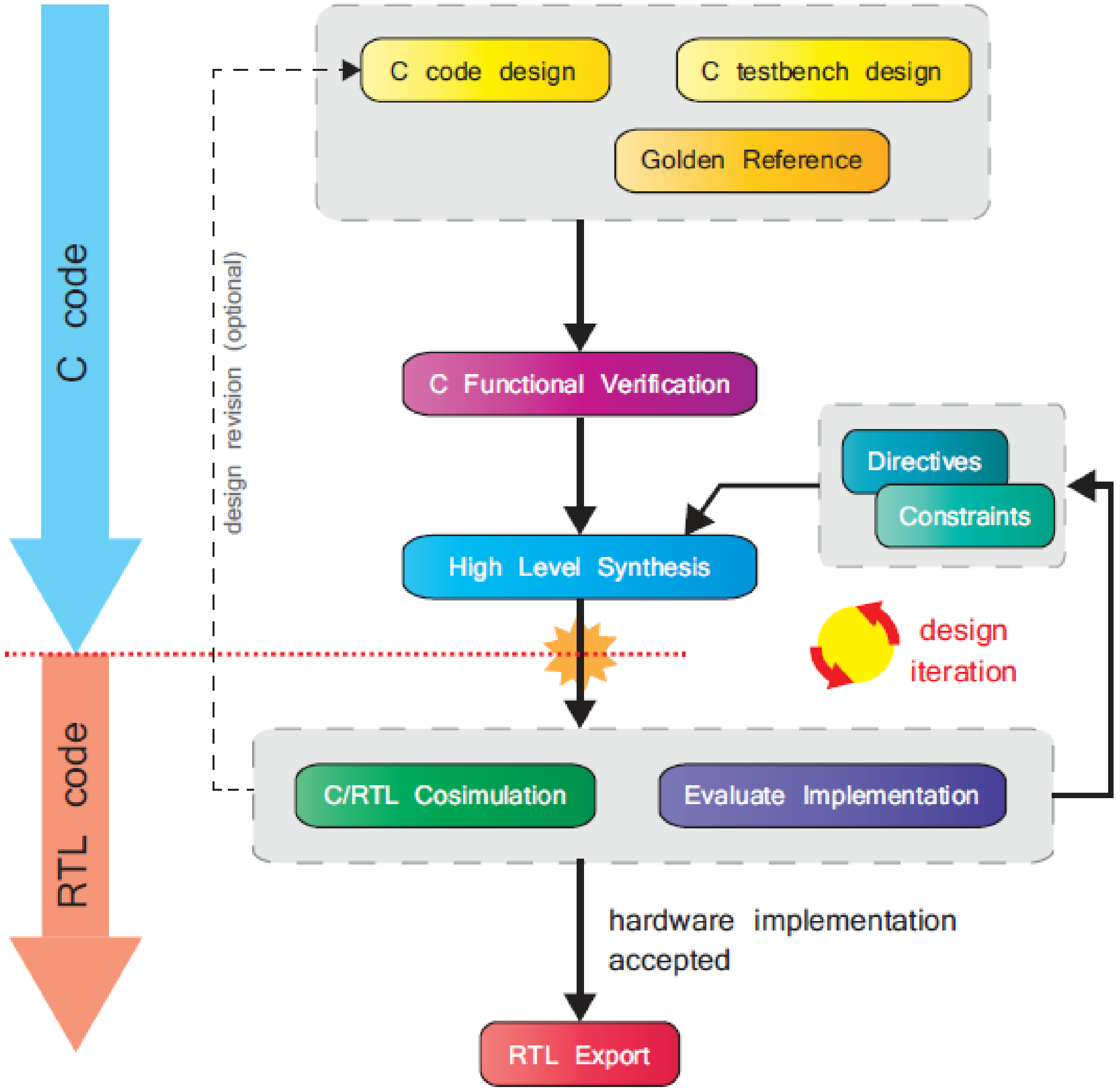
BRAM, DSP

- Block RAM
  - RAM
  - ROM
  - FIFO buffer

- DSP48E1(ALU)
  - 3가지 블록의 조합
  - $p = a \times (b + d) + c$
  - $p \ += a \times (b + d)$



02. HLS를 사용한 시스템 설계



- Inputs
  - C code Design
  - C Testbench Design
  - Golden Reference(출력 데이터)
  - Directives
  - Constraints
- High-Level Synthesis
  - Analysis, Processing C-based Code
  - Directives, Constraints 만족 확인

HLS Synthesis Basic Rules

- Scheduling
  - Operation의 순서, 병렬화 결정
  - Dependency, Clock cycle, FPGA 속도, User directives에 따라 결정됨
- Binding
  - 사용할 HW 리소스 결정(연산의 경우 내부 DSP48 블록 사용)
  - FPGA의 종류에 따라 결정됨
- Control logic extraction
  - 제어(loop 문, if-else 문, case 문)등을 FSM(Finite State Machine)으로 구현하기 위해 제어 로직을 추출

HUINS
Human Intelligent System

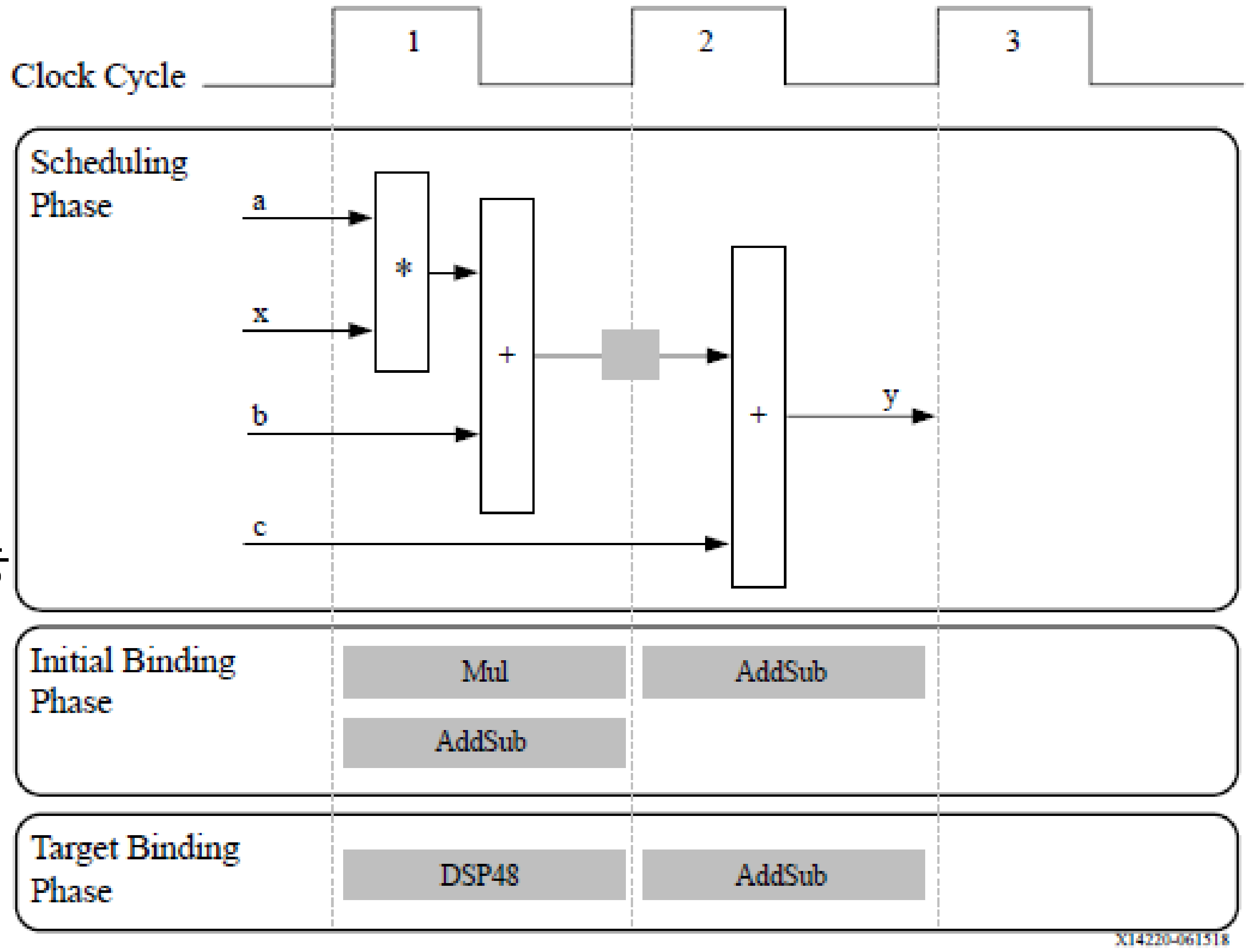
HLS Synthesis Basic Rules

- Function Arguments → I/O Port
- Function → RTL Block
  - 하나의 함수는 하나의 RTL 블록으로 구현됨
- 반복문(Loop)
  - RTL 블록 단위 반복
  - 반복 없이 병렬화
  - FSM 등을 이용한 파이프라인 중첩
- 어레이의 경우 Block RAM 내부에 저장

Scheduling, Binding EX.

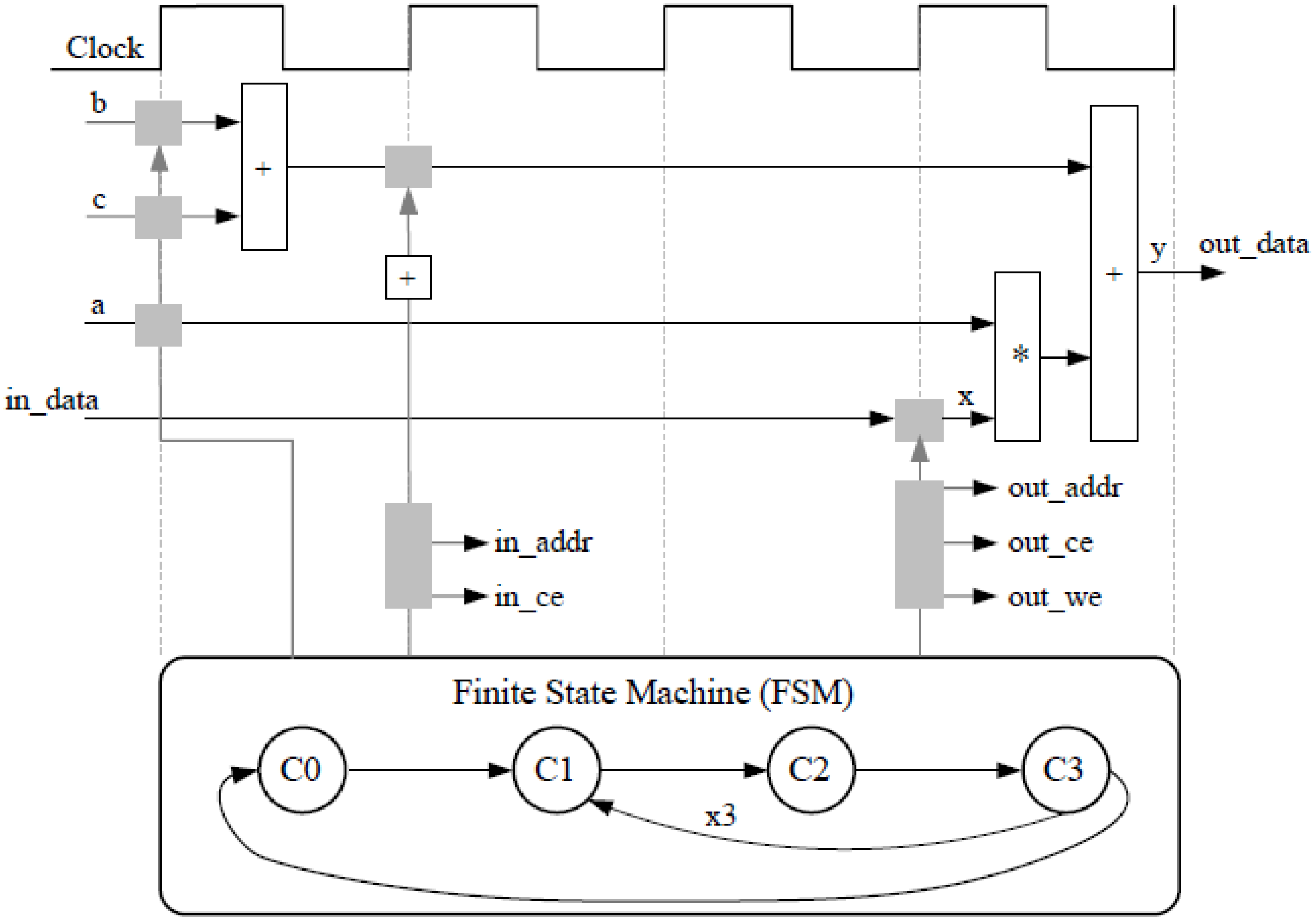
```
int foo(char x, char a, char b, char c){
    char y;
    y = x * a + b + c;
    return y;
}
```

- Scheduling Phase
  - 연산 선후관계에 따른 배치
  - (x * a + b)한 클럭 안에 계산 가능
- Binding Phase
  - *, + 로 연산 구분
  - FPGA 내부 DSP48 사용



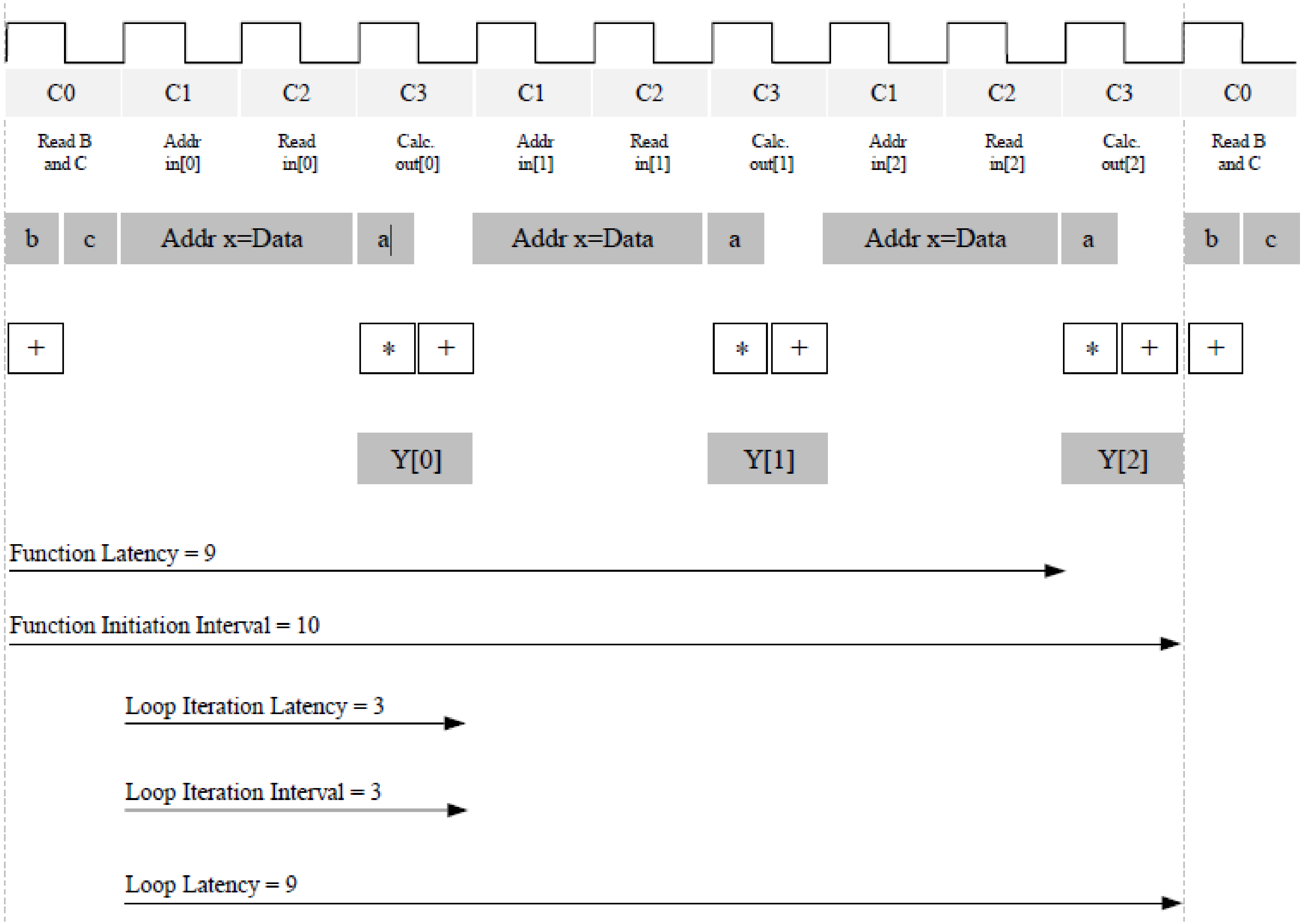
Control Logic

```
void foo(int in[3], char a, char b, char c, int out[3]) {
    int x,y;
    for(int i = 0; i < 3; i++) {
        x = in[i];
        y = a * x + b + c;
        out[i] = y;
    }
}
```



- 반복문
  - C0 - (C1,C2,C3) X3 – C0
  - C0: b + c
  - C1: in[n]의 Addr, Chip enable
  - C2: read from in[n], store at x[n]
  - C3: y = a * x + b + c
- Array: 모듈 외부에 BRAM 구현

Control Logic



Data Type

Type	Description	Number of Bits[a]	Range[b]
`char`	Representation of the basic character set.	8	-128 to 127
`signed char`		8	-128 to 127
`unsigned char`		8	0 to 255
`short int`	A reduced precision version of `int`, requiring less storage.	16	-32,768 to 32,767
`unsigned short int`		16	0 to 65,535
`int`	The basic integer data type.	32	-2,147,483,648 to 2,147,483,647
`unsigned int`		32	0 to 4,294,967,295
`long int`	In many cases the `long int` type will be the same length as `int`, i.e. 32 bits.	32	-2,147,483,648 to 2,147,483,647
`unsigned long int`		32	0 to 4,294,967,295
`long long int`	An extended precision integer type.	64	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
`unsigned long long int`		64	0 to 18,446,744,073,709,551,615
`float`	Single precision floating point (IEEE 754)	32	$-3.403e^{+38}$ to $3.403e^{+38}$
`double`	Double precision floating point (IEEE 754)	64	$-1.798e^{+308}$ to $1.798e^{+308}$

```
int foo(char x, char a, char b, char c)
```

- C언어의 Data Type은 고정
- Data bit, Sign, floating point

- S = A * B;
- A, B: 18-bit, S: 36-bit (RTL)
- A, B: 32-bit, S: 64-bit (C-Data type)

- DSP48E: 1 → 4 Slices

HUINS
Human Intelligent System

Data Type

Language	Integer Data Type	Description	Required Header
C	int*N* (e.g. `int7`)	signed integer of *N* bits precision	`#include "ap_cint.h"`
	uint*N* (e.g. `uint7`)	unsigned integer of *N* bits precision	
C++	ap_int<*N*> (e.g. `ap_int<7>`)	signed integer of *N* bits precision	`#include "ap_int.h"`
	ap_uint<*N*> (e.g. `ap_uint<7>`)	unsigned integer of *N* bits precision	

Arbitrary Data Type

Verilog: input wire data[17:0];

HLS-C: uint18 data;

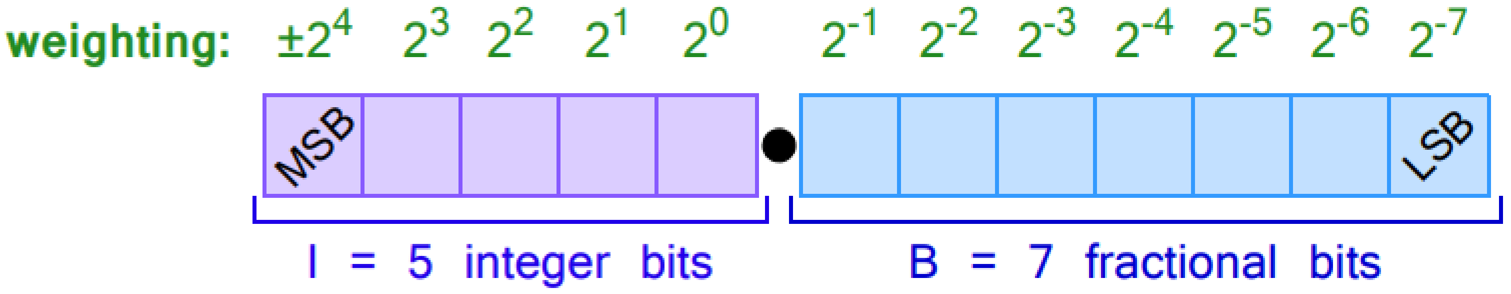
Data Type

```c
// C code example
#include “ap_cint.h”

void top_level_function (..)
{
      // declarations
        int6 small_signed;
        uint10 big_unsigned;
        int22 vbig_signed;
        ...
}
```

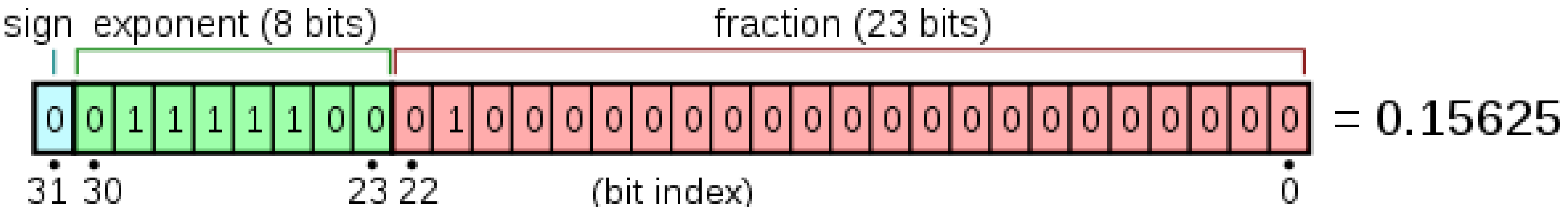
```cpp
// C++ code example
#include “ap_int.h”

void top_level_function (..)
{
      // declarations
        ap_int<6> small_signed;
        ap_uint<10> big_unsigned;
        ap_int<22> vbig_signed;
        ...
}
```


HUINS
Human Intelligent System

Data Type(floating point)



$$I = 5,\ B = 7,\ W = I + B = 12$$



Arbitrary Float Point

Language	Fixed Point Data Type	Description	Required Header
C++	ap_fixed<*W,I,Q,O,N*>	Signed fixed point number of *I* integer bits and *W-I* fractional bits.	`#include "ap_fixed.h"`
	ap_ufixed<*W,I,Q,O,N*>	Unsigned fixed point number of *I* integer bits and *W-I* fractional bits.	

```
// C++ code example
#include "ap_fixed.h"

void top_level_function (..)
{
    // declarations
     ap_ufixed<8,3> small_unsigned; // 3 int, 5 fract, defaults
     ap_fixed<10,4,AP_RND> big_signed; // round to + inf.
     ap_ufixed<10,4,AP_RND_ZERO> big_unsigned; // round to zero
     ap_fixed<21,10,AP_TRN,AP_SAT> vbig_signed; // trunc., satur.
     ap_ufixed<21,10,AP_RND_CONV> vbig_unsigned; // conv. round.
     ...
}
```

Arbitrary Data Type

Verilog: input wire data[1

HLS-C: uint18 data;

HUINS
Human Intelligent System

Pointers

- Array Access

```
int A[10];
int *pA;

pA = A;
```

- Pointer to External Memory

```
void foo(int *data_in,...)
{
   int item1, item2, item3;

  item1 = *data_in;
  item2 = *(data_in + 1);
  item3 = *(data_in + 2);
  ...
}
```

- Dynamic Memory Allocation
  → Not supported

```
int *A = malloc(10*sizeof(int));
```


HUINS
Human Intelligent System

I/O Port

```
void find_average_of_best_X (int *average, int samples[8], int X)

{
     // body of function (statements, sub-function calls, etc.)

}
```

- Port Name
- Direction
- Data Type, Dimension

HUINS
Human Intelligent System

Port Direction

C/C++ Function Argument	RTL Port Type
An argument which is read from and never written to	in
An argument which is written to and never read from	out
A value output by the function return statement	out
An argument which is both written to and read from	inout (bidirectional)

Port Interface Protocol Types

- ap: AXI Portocol
- ack: Acknowledgement
- vld: valid / ovld: out valid
- hs: Hand shaking(ack, vld)
- memory: array

Table 15.7: Protocol synthesis: supported types and defaults (S = supported; D = default) [18]

Argument Type	Variable			Pointer Variable			Array			Reference Variable		
	pass-by-value			pass-by-reference			pass-by-reference			pass-by-reference		
Interface Type[a]	**I**	**IO**	**O**	**I**	**IO**	**O**	**I**	**IO**	**O**	**I**	**IO**	**O**
ap_none	**D**	-	-	**D**	S	S	-	-	-	**D**	S	S
ap_stable	S	-	-	S	S	-	-	-	-	S	S	-
ap_ack	S	-	-	S	S	S	-	-	-	S	S	S
ap_vld	S	-	-	S	S	**D**	-	-	-	S	S	**D**
ap_ovld	-	-	-	-	**D**	S	-	-	-	-	**D**	S
ap_hs	S	-	-	S	S	S	S	-	S	S	S	S
ap_memory	-	-	-	-	-	-	**D**	**D**	**D**	-	-	-
bram	-	-	-	-	-	-	S	S	S	-	-	-
ap_fifo	-	-	-	S	-	S	S	-	S	S	-	S
ap_bus	-	-	-	S	S	S	S	S	S	S	S	S
axis	S	-	-	S	-	S	S	-	S	S	-	S
s_axilite	S	-	S	S	S	S	-	-	-	S	S	S
m_axi	-	-	-	S	S	S	S	S	S	S	S	S

a. Reading along the row: **I** = input port; **IO** = inout (bidirectional) port; **O** = output port.

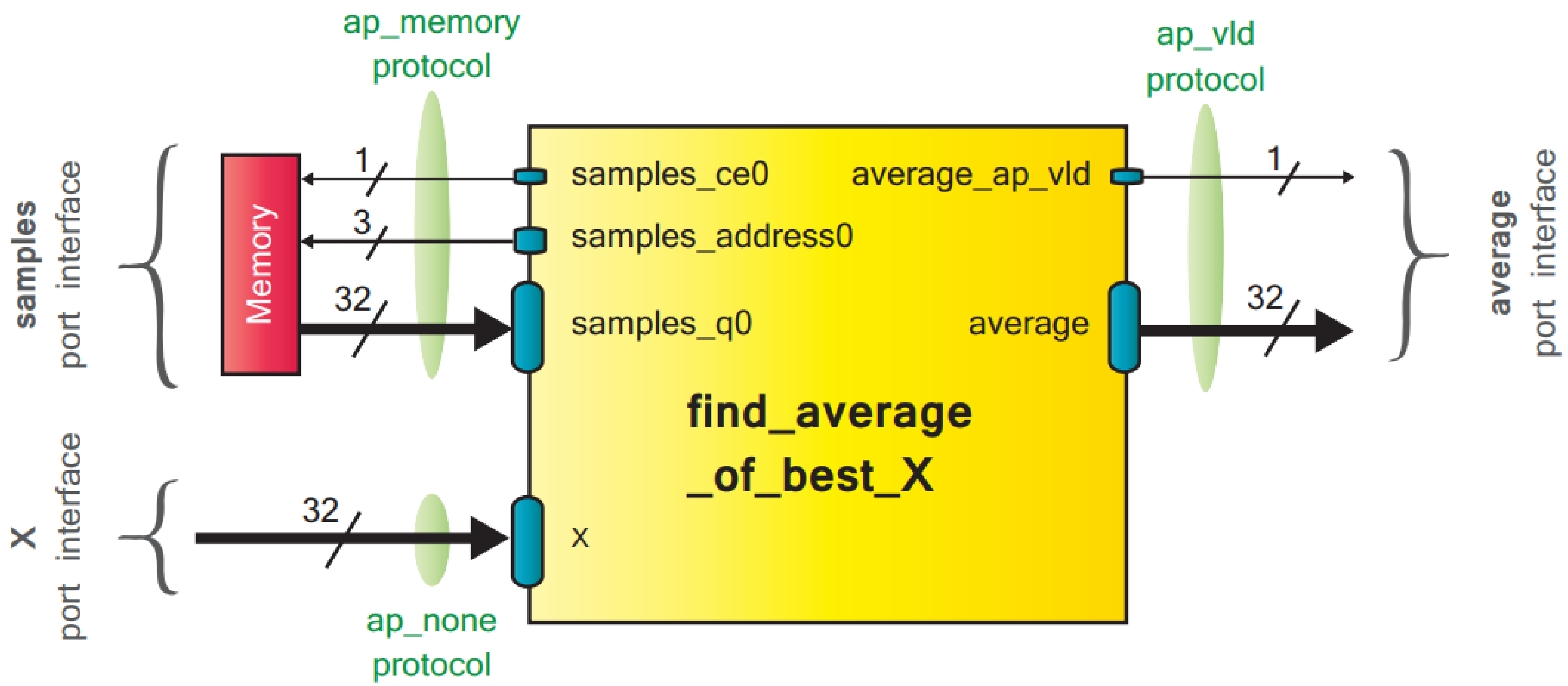
Port Interface

```
void find_average_of_best_X (int *average, int samples[8], int X)

{
    // body of function (statements, sub-function calls, etc.)

}
```

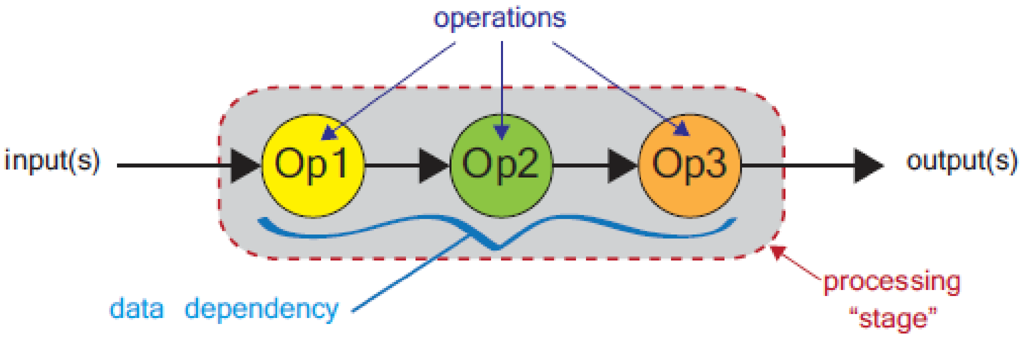
x : pass-by value → ap_none

samples[8] : array input → ap_memory

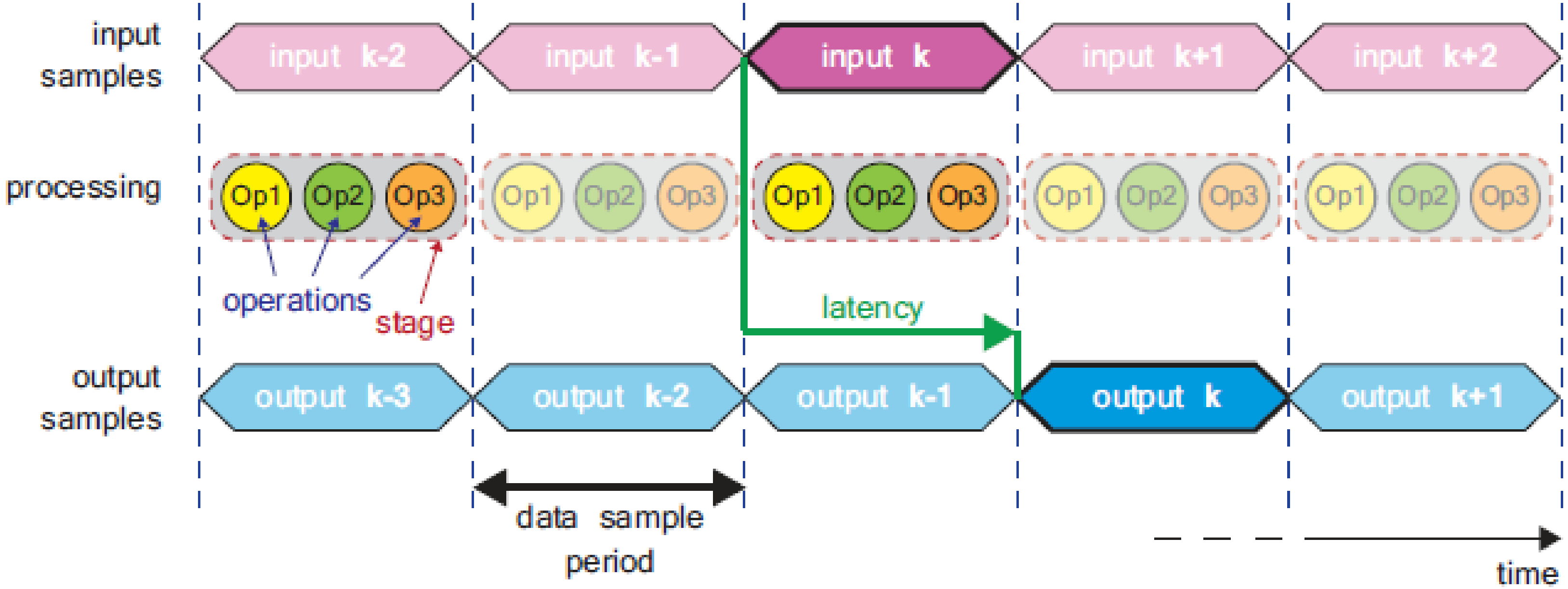
average : output pointer → ap_vld



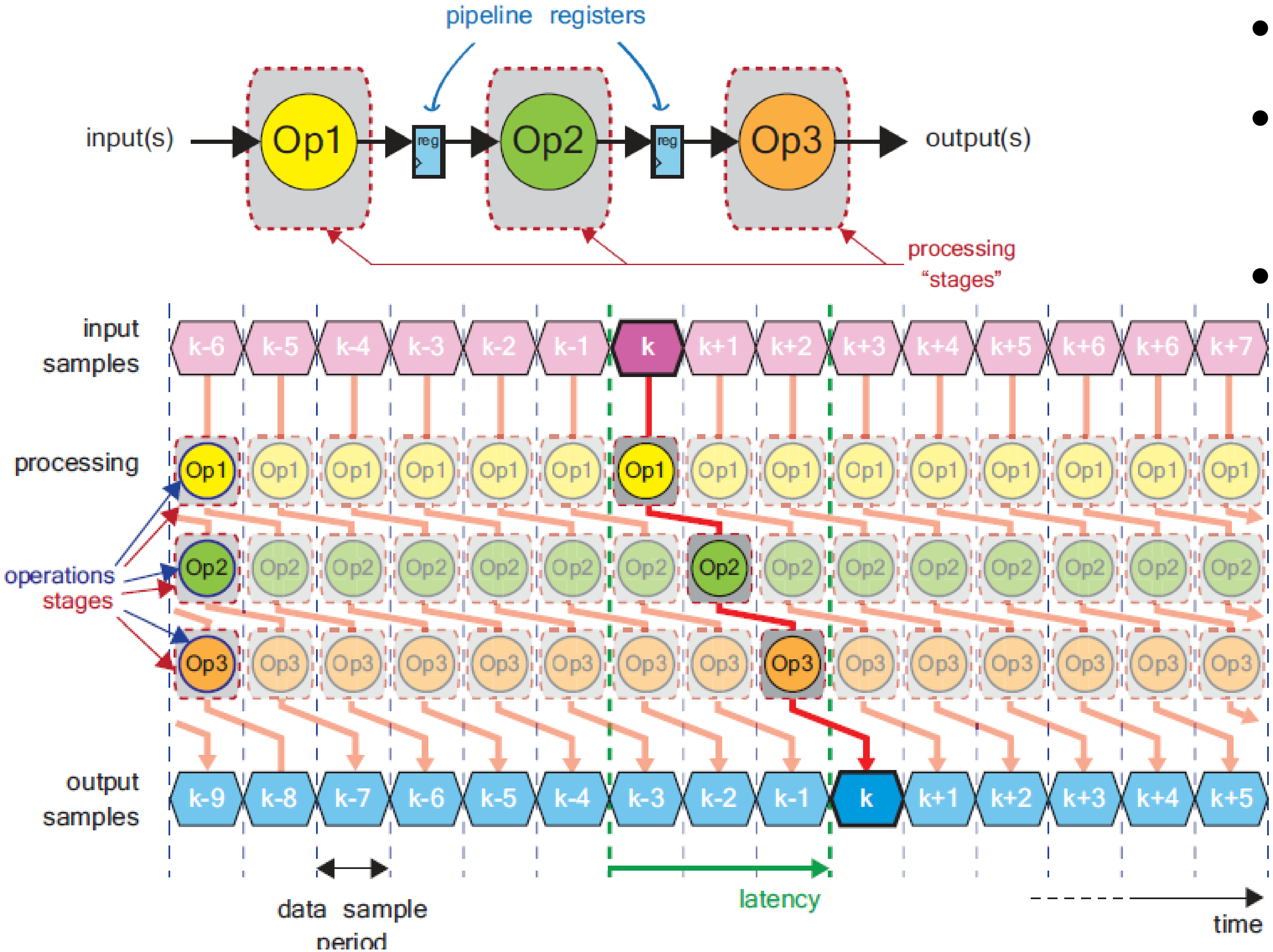
Pipelining



- If data dependency
  - Op1 → Op2 → Op3
  - Latency = T(Op1 + Op2 + Op3)
- Low Throughput



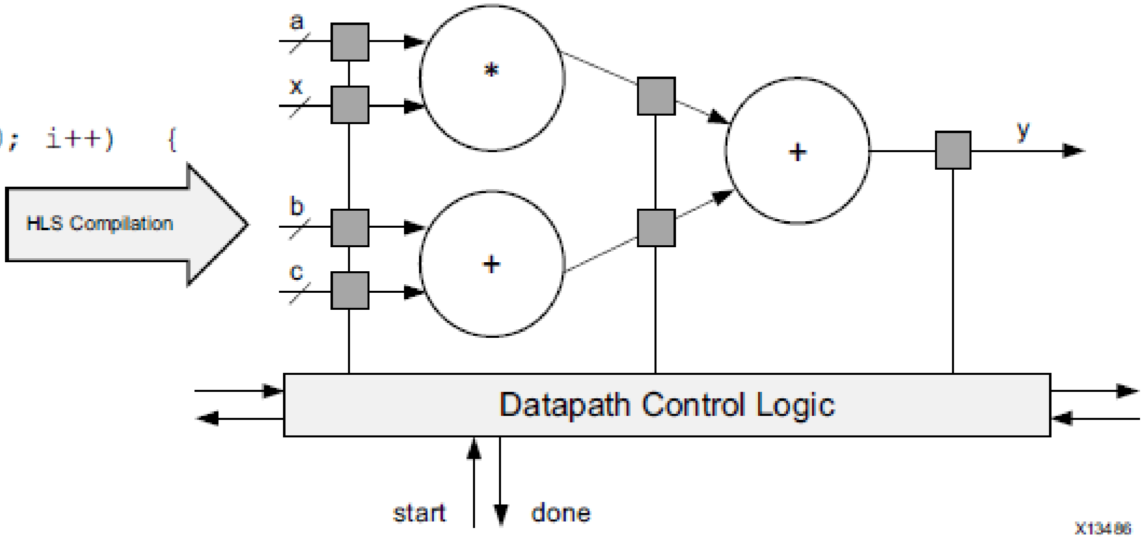
Pipelining



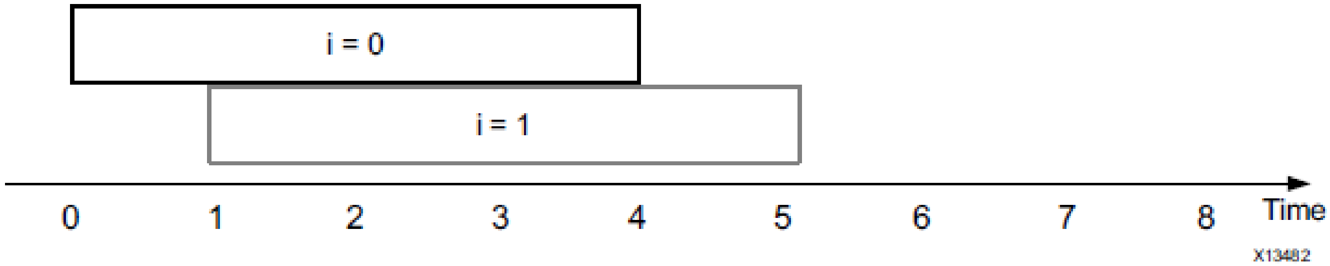
- 레지스터 추가
- 각각의 Operation 은 개별 Processing Stage
- Throughput 3 배

Control Algorithms - Loop

```
int a,b,c,x,y;
for(int i = 0; i < 20; i++)  {
  x = get();
  y = a*x + b + c;
  send(y);
}
```

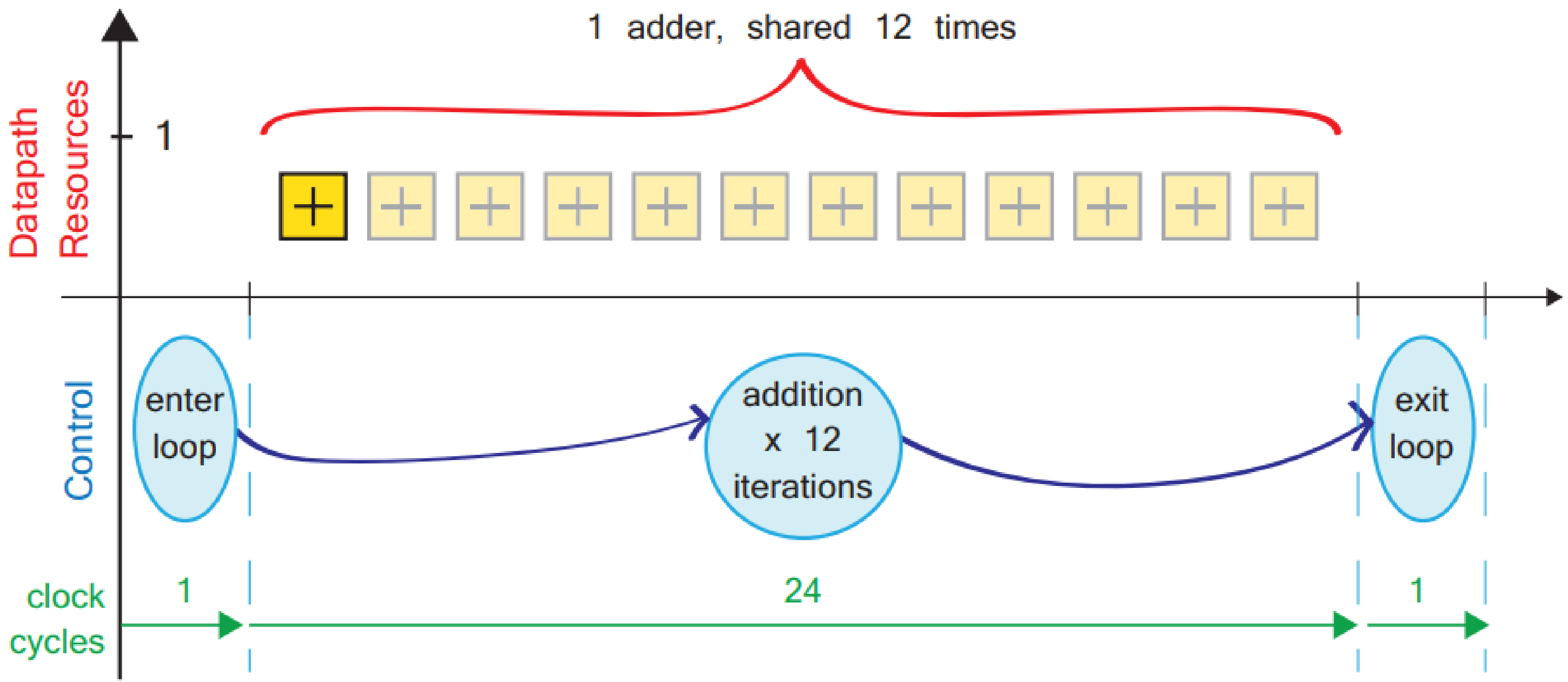


- FSM으로 구현
- Control Logic / 연산 구분
- 설정에 따라(Directives)
  - 루프 삭제(병렬 계산 처리)
  - 루프 Pipelining



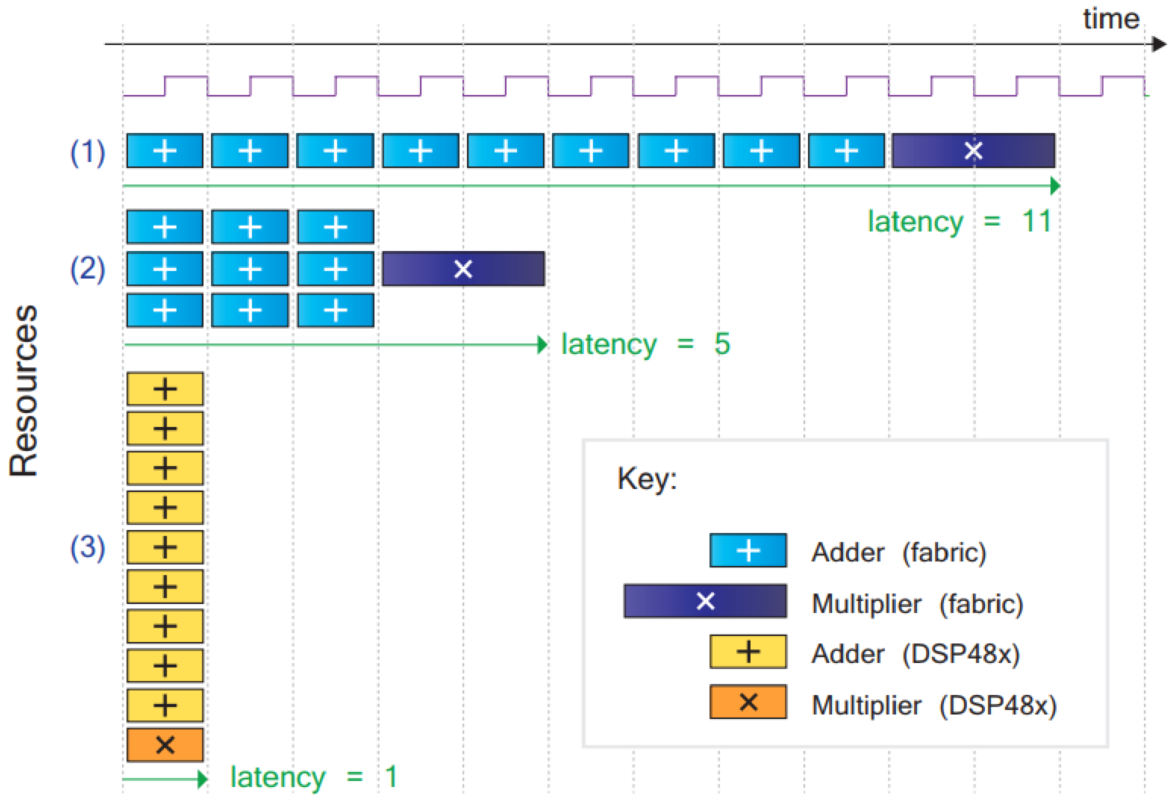
Loop

```
void add_array (short c[12], short a[12], short b[12])
{
  short j;                                 // loop variable

  add_loop: for (j=0;j<12;j++) {     // loop through elements (x12)
          c[j] = a[j] + b[j];          // addition operation
      }
}
```



Loop Optimization

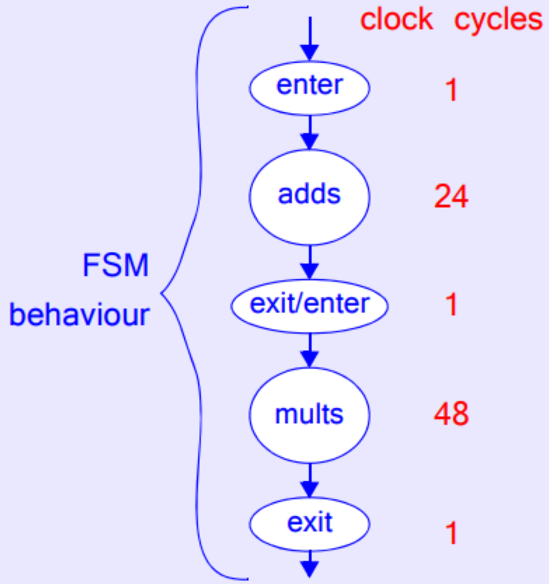
- Unrolling: (1) → (2)
- Merging
- Pipelining(Dependency related)



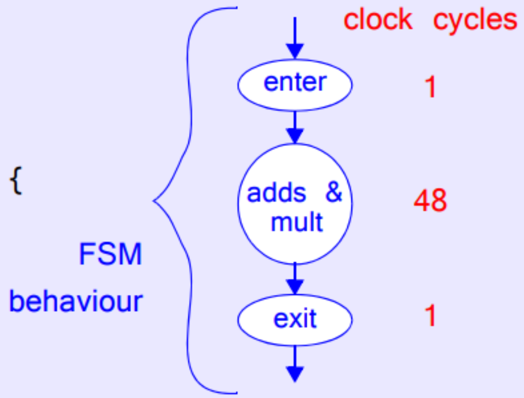
Loop Optimization

- Unrolling
- Merging
- Pipelining(Dependency related)

```
void add_mult (short c[12], short m[12], short a[12], short b[12])
{
  short j;

  add_loop: for (j=0;j<12;j++) {
        c[j] = a[j] + b[j];
    }

  mult_loop: for (j=0;j<12;j++) {
        m[j] = a[j] * b[j];
    }
}
```




```
void add_mult (short c[12], short m[12], short a[12], short b[12])
{
  short j;

  add_mult_loop: for (j=0;j<12;j++) {
        c[j] = a[j] + b[j];
        m[j] = a[j] * b[j];
    }
}
```



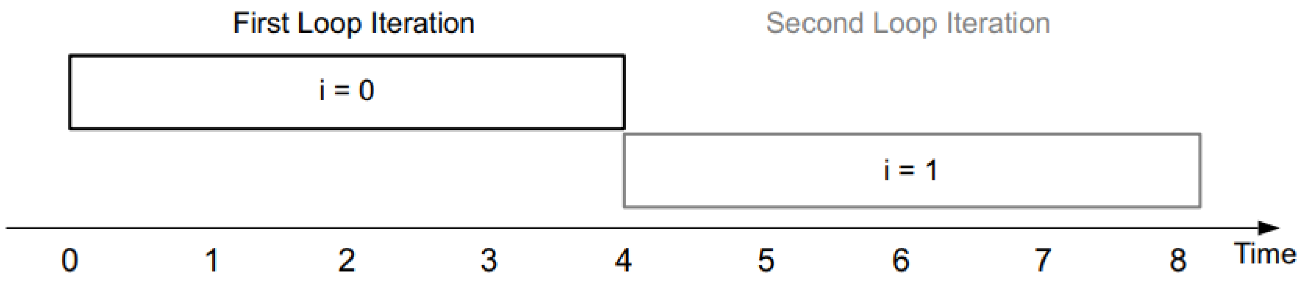
Loop Optimization

- Unrolling
- Merging
- Pipelining(Dependency related)

```
#pragma HLS PIPELINE II=1

for(i=0; i < 10; i++)
{
      A = A + (B[i] * C[i]);
}
```

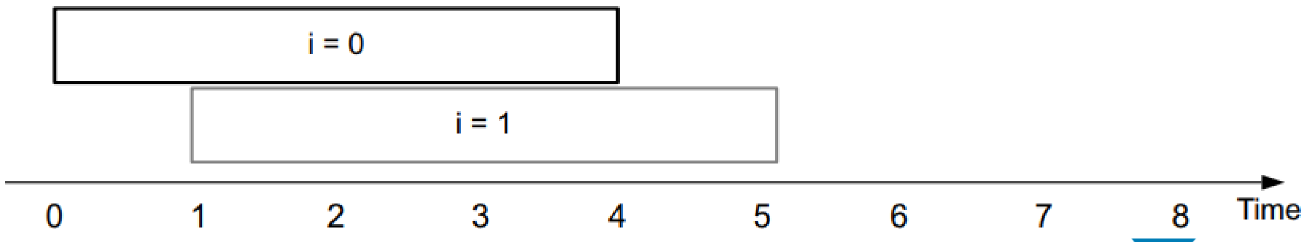
default

- Resource Optimized



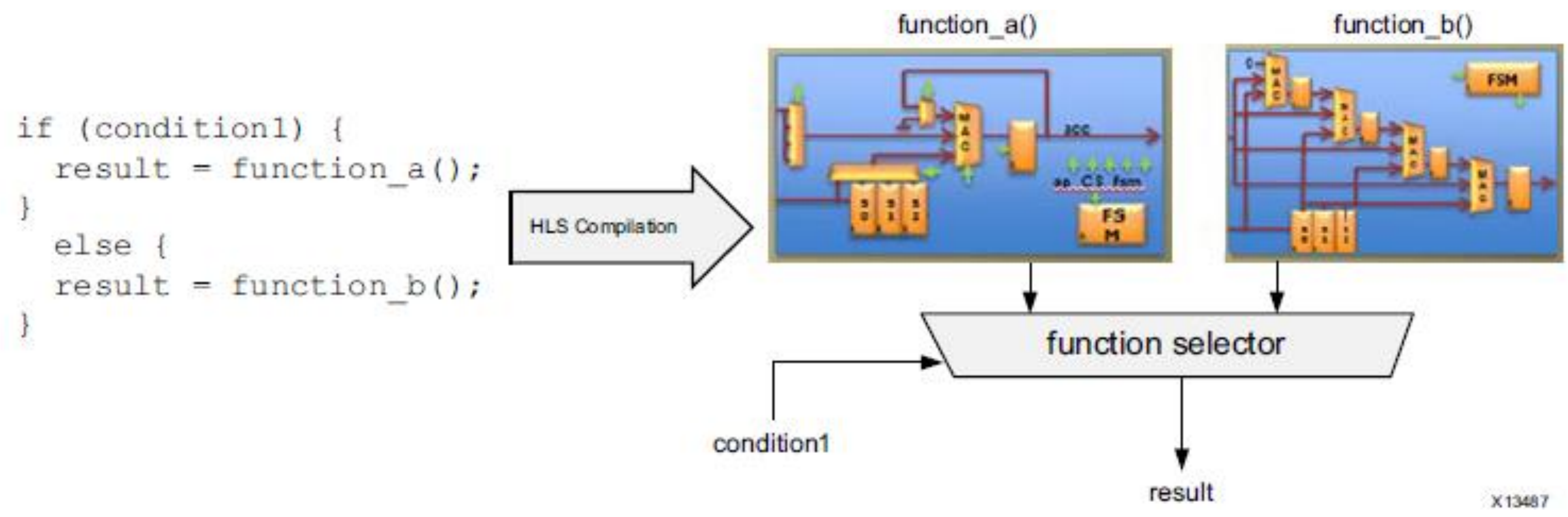
Pipelined

- loop initialization interval (II)



HUINS
Human Intelligent System

Control Algorithms - 조건문

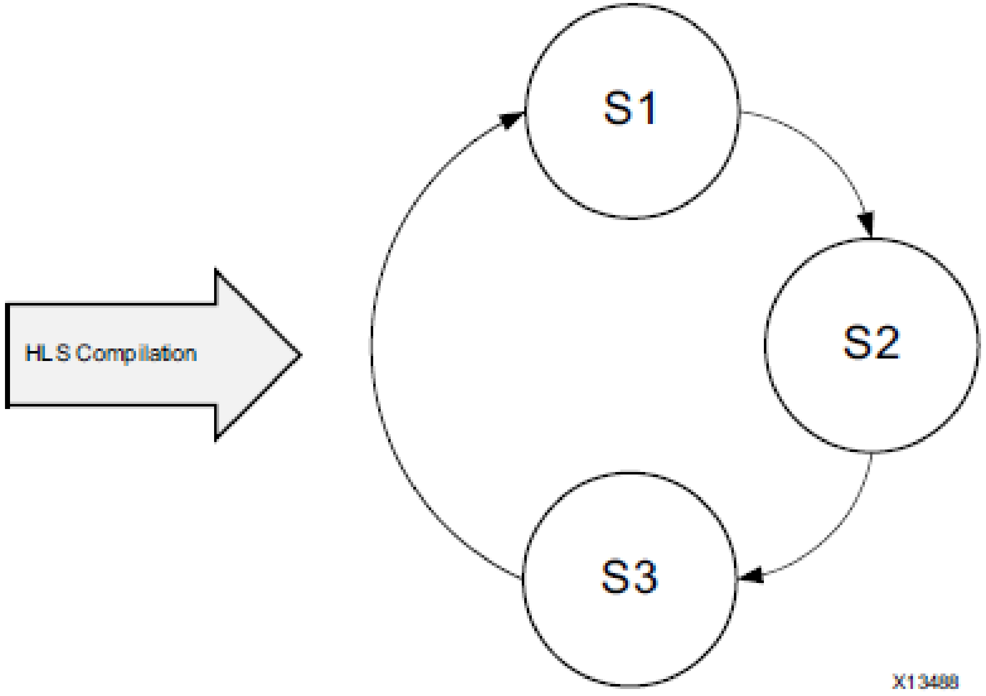


- 각 조건에 따른 함수(회로) 모두 구성
- 조건에 따라 함수 선택

HUINS
Human Intelligent System

Control Algorithms – Switch Case

```
switch (X){
  case S1: ... X = S2; break;
  case S2: ... X = S3; break;
  case S3: ... X = S1; break;
}
```

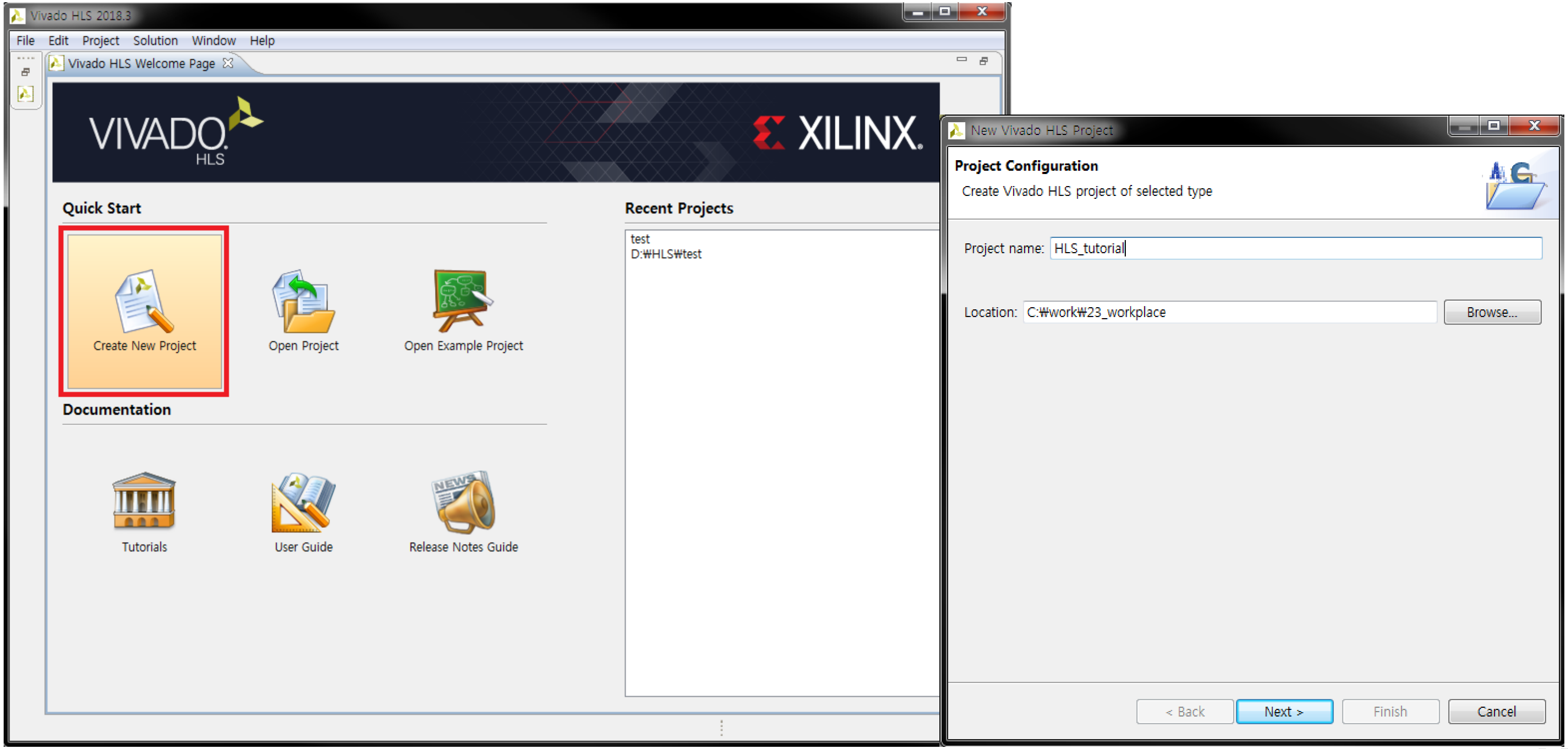


- FSM으로 구현

03. HLS를 이용한 FPGA 설계 실습

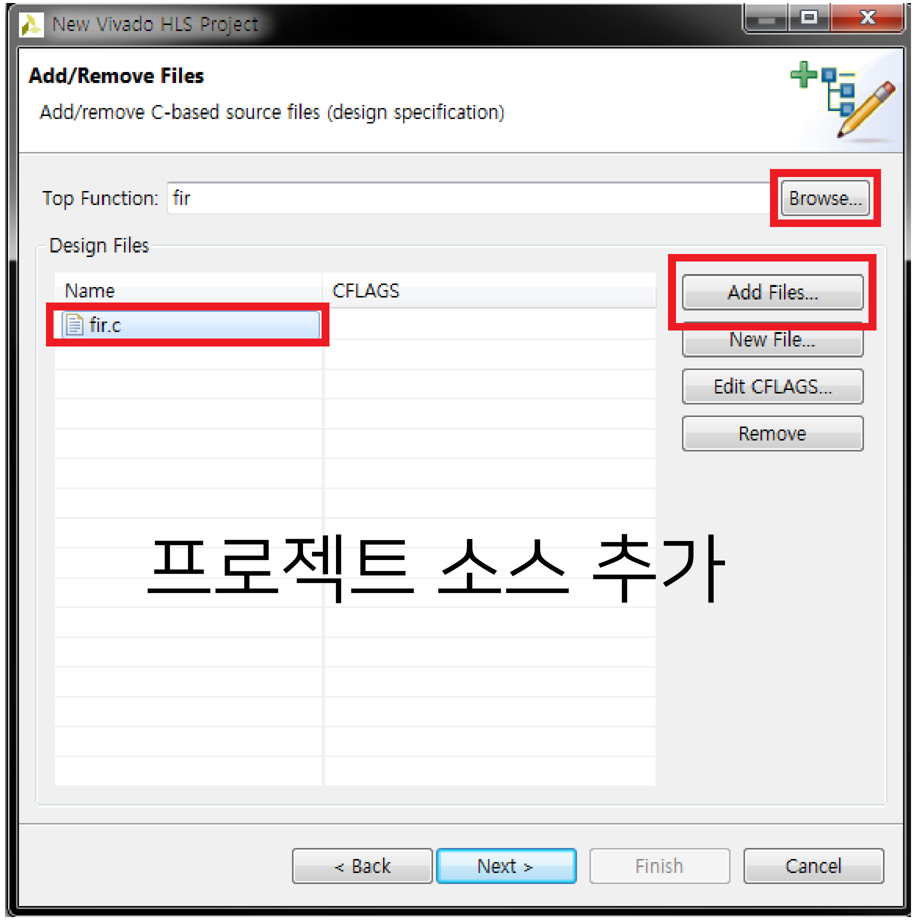
- Xilinx ug-871 Vivado High Level Synthesis Tutorial
- Source Code
- C-Level Test bench, Golden Ref
- HLS IP

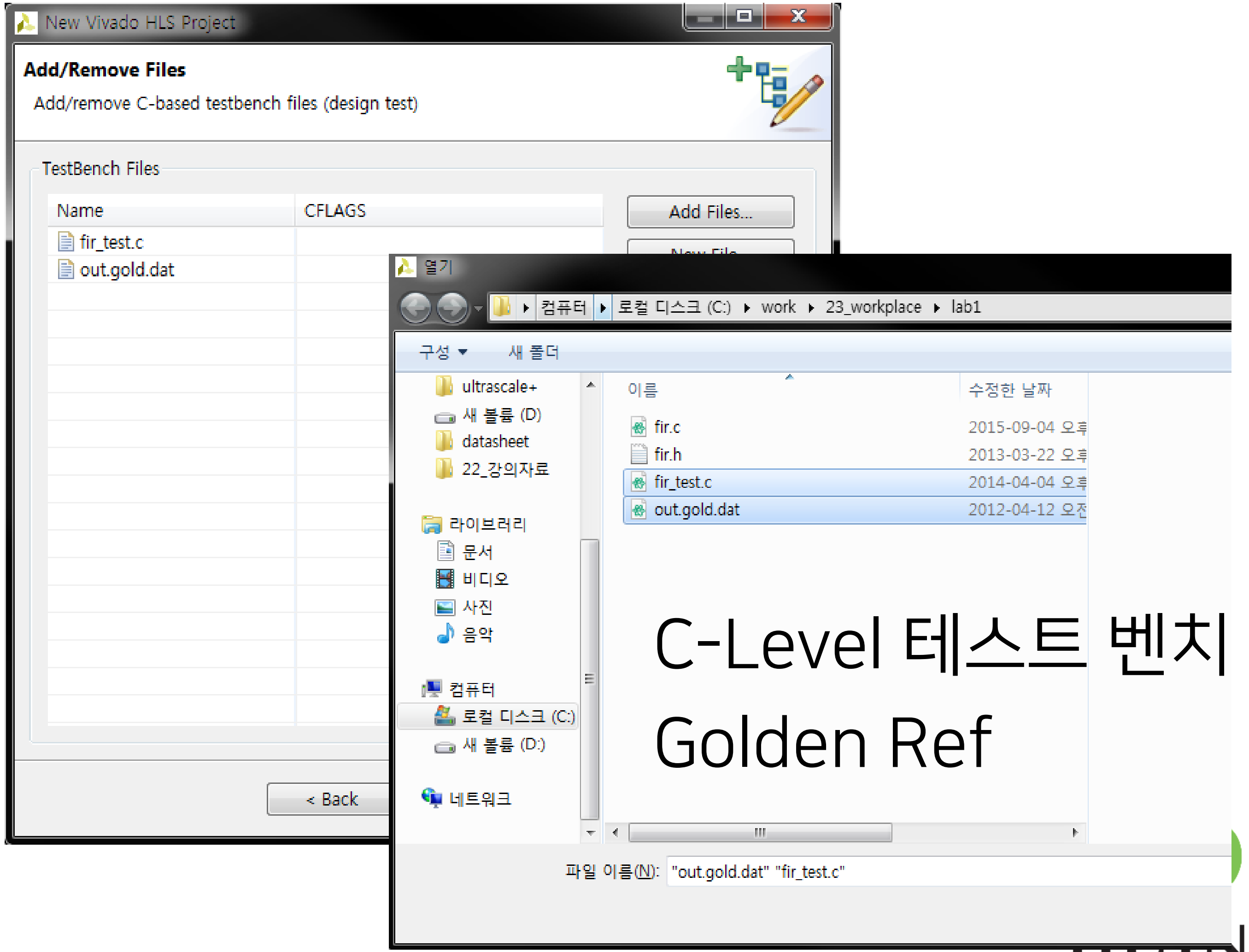
Vivado HLS Tutorial



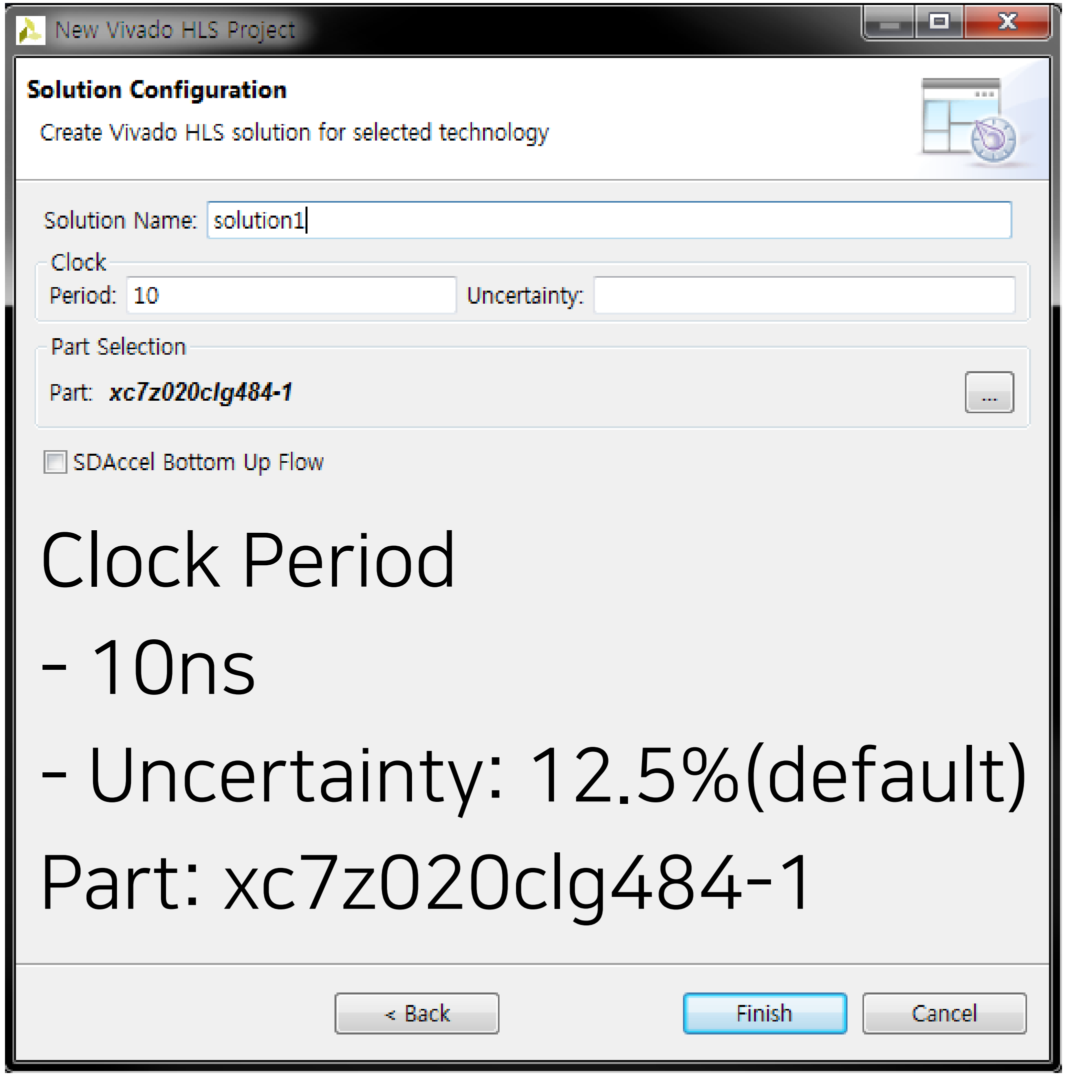
HUINS
Human Intelligent System

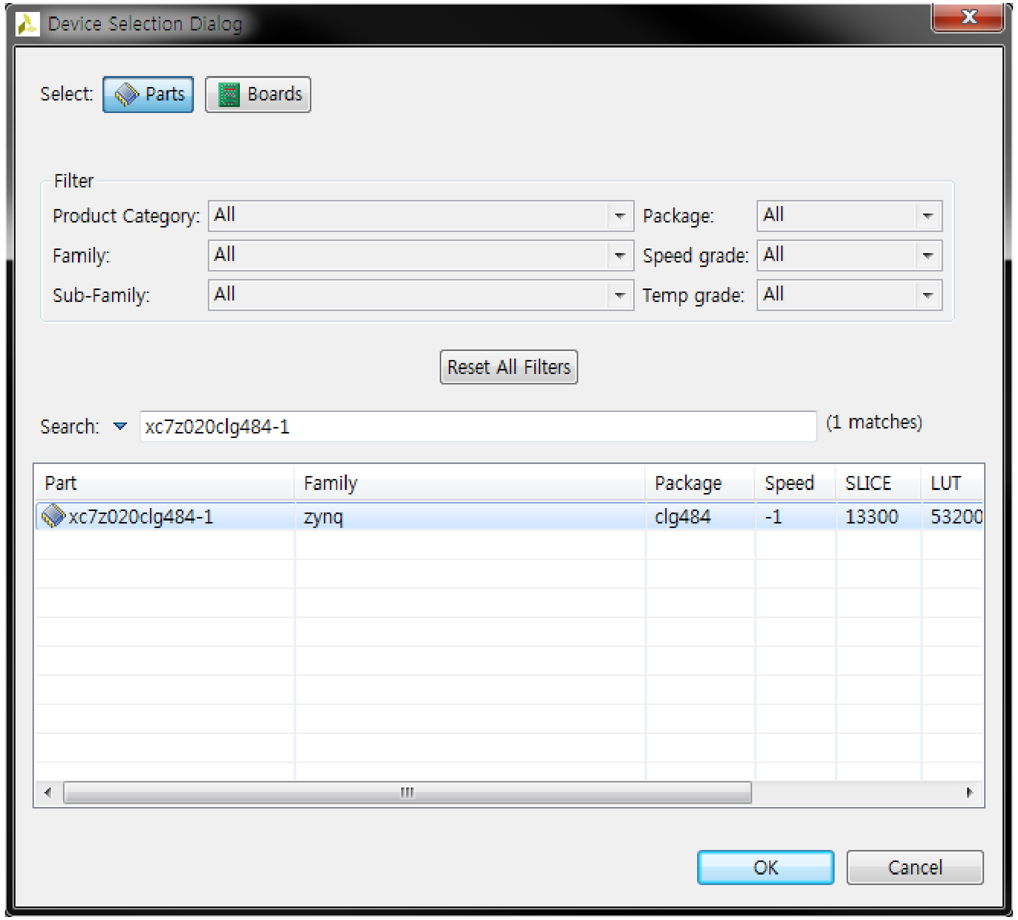
Vivado HLS Tutorial





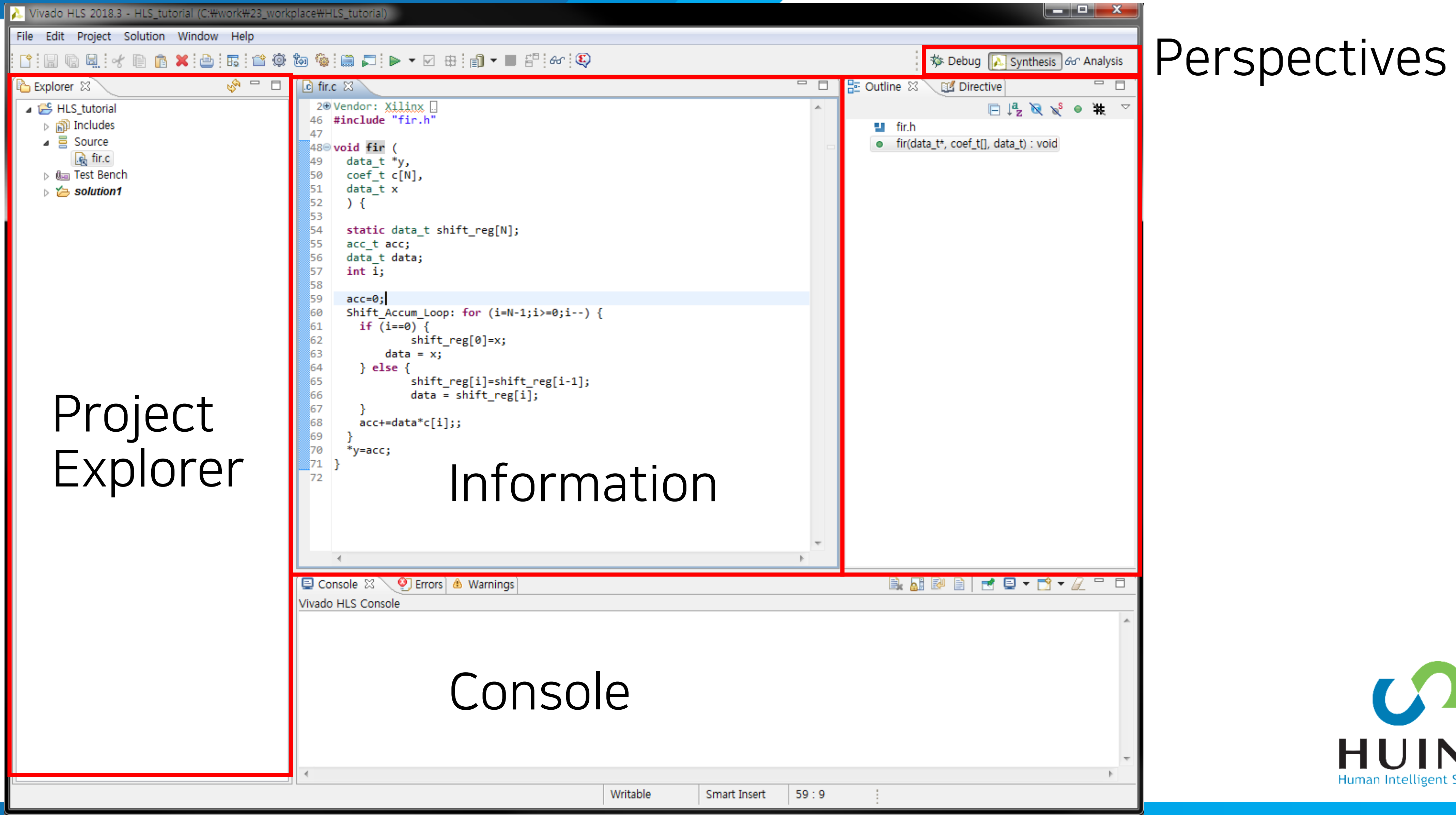
Vivado HLS Tutorial





HUINS
Human Intelligent System

Vivado HLS GUI

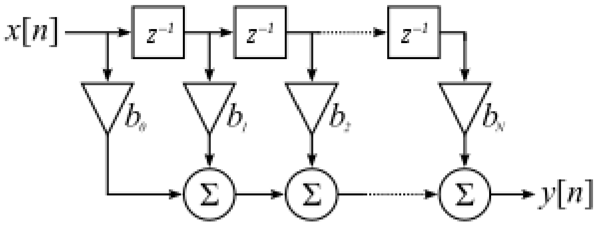


fir.c(FIR 필터 함수)

fir.c fir_test.c

```c
Vendor: Xilinx
#include "fir.h"

void fir (
  data_t *y,
  coef_t c[N],
  data_t x
  ) {

  static data_t shift_reg[N];
  acc_t acc;
  data_t data;
  int i;

  acc=0;
  Shift_Accum_Loop: for (i=N-1;i>=0;i--) {
    if (i==0) {
            shift_reg[0]=x;
        data = x;
    } else {
            shift_reg[i]=shift_reg[i-1];
            data = shift_reg[i];
    }
    acc+=data*c[i];;
  }
  *y=acc;
}
```

입력 신호 열 중에서 유한 샘플(N개)에 일정한 계수를 곱하여 더한 합을 말함(정보통신기술용어해설)



$$y[n] = b_0 x[n] + b_1 x[n-1] + \cdots + b_N x[n-N]$$
$$= \sum_{i=0}^{N} b_i \cdot x[n-i],$$

data = shift_reg[10,9,8,7,⋯, 1], x

acc += data * c[i]

Multiply & Add for each iteration

fir_test.c(test bench)

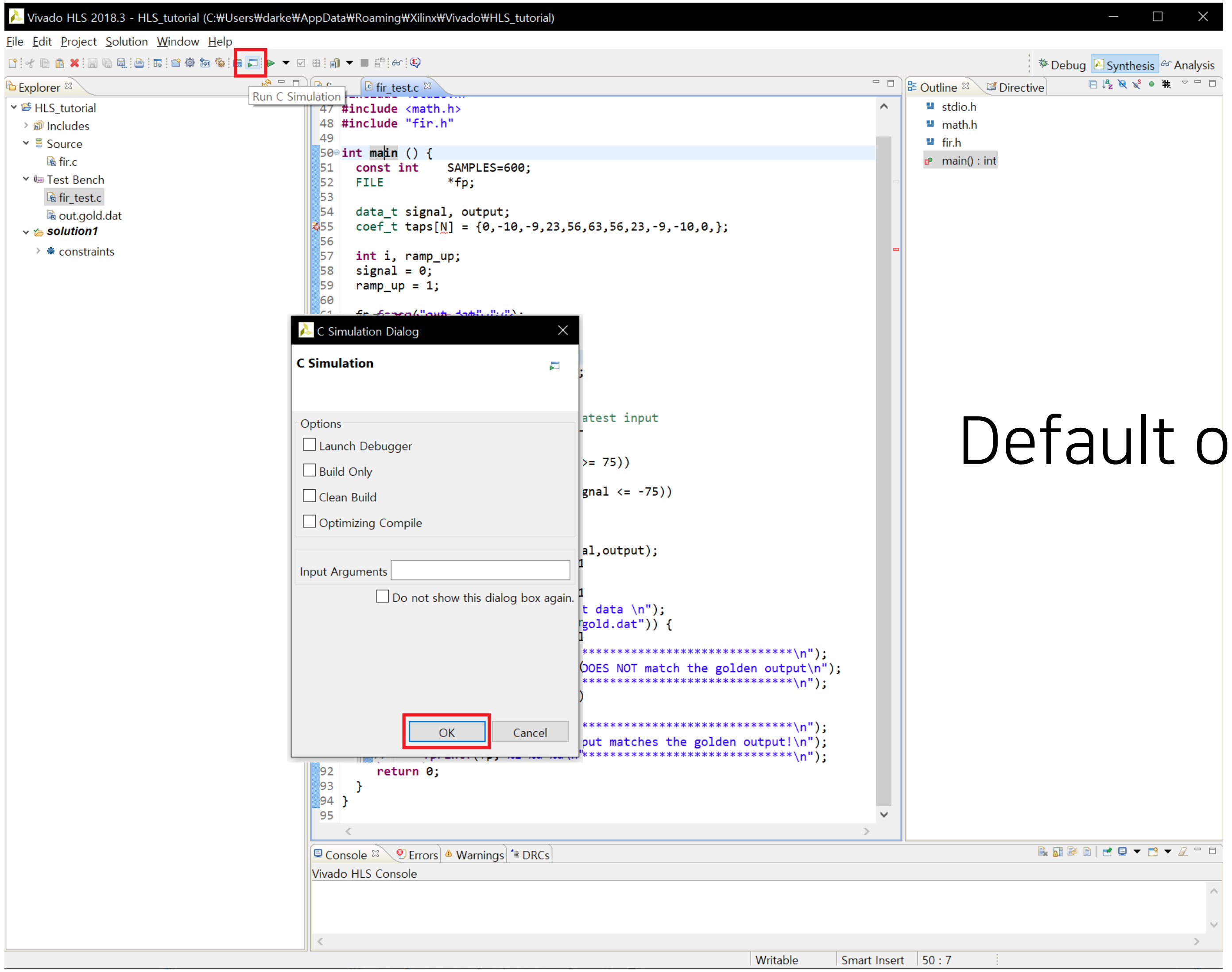
fir.c | fir_test.c

```c
#include <math.h>
#include "fir.h"

int main () {
  const int     SAMPLES=600;
  FILE          *fp;

  data_t signal, output;
  coef_t taps[N] = {0,-10,-9,23,56,63,56,23,-9,-10,0,};

  int i, ramp_up;
  signal = 0;
  ramp_up = 1;

  fp=fopen("out.dat","w");
  for (i=0;i<=SAMPLES;i++) {
    if (ramp_up == 1)
        signal = signal + 1;
    else
        signal = signal - 1;

    // Execute the function with latest input
    fir(&output,taps,signal);

    if ((ramp_up == 1) && (signal >= 75))
        ramp_up = 0;
    else if ((ramp_up == 0) && (signal <= -75))
        ramp_up = 1;

    // Save the results.
    fprintf(fp,"%i %d %d\n",i,signal,output);
  }
  fclose(fp);

  printf ("Comparing against output data \n");
  if (system("diff -w out.dat out.gold.dat")) {

    fprintf(stdout, "*******************************************\n");
    fprintf(stdout, "FAIL: Output DOES NOT match the golden output\n");
    fprintf(stdout, "*******************************************\n");
     return 1;
  } else {
    fprintf(stdout, "*******************************************\n");
    fprintf(stdout, "PASS: The output matches the golden output!\n");
    fprintf(stdout, "*******************************************\n");
     return 0;
  }
}
```

main() function

- calls fir(), saves output as out.dat
- Golden reference와 out.dat비교
  - Match 시 0 return
  - Not match 시 1 return

Launch simulation



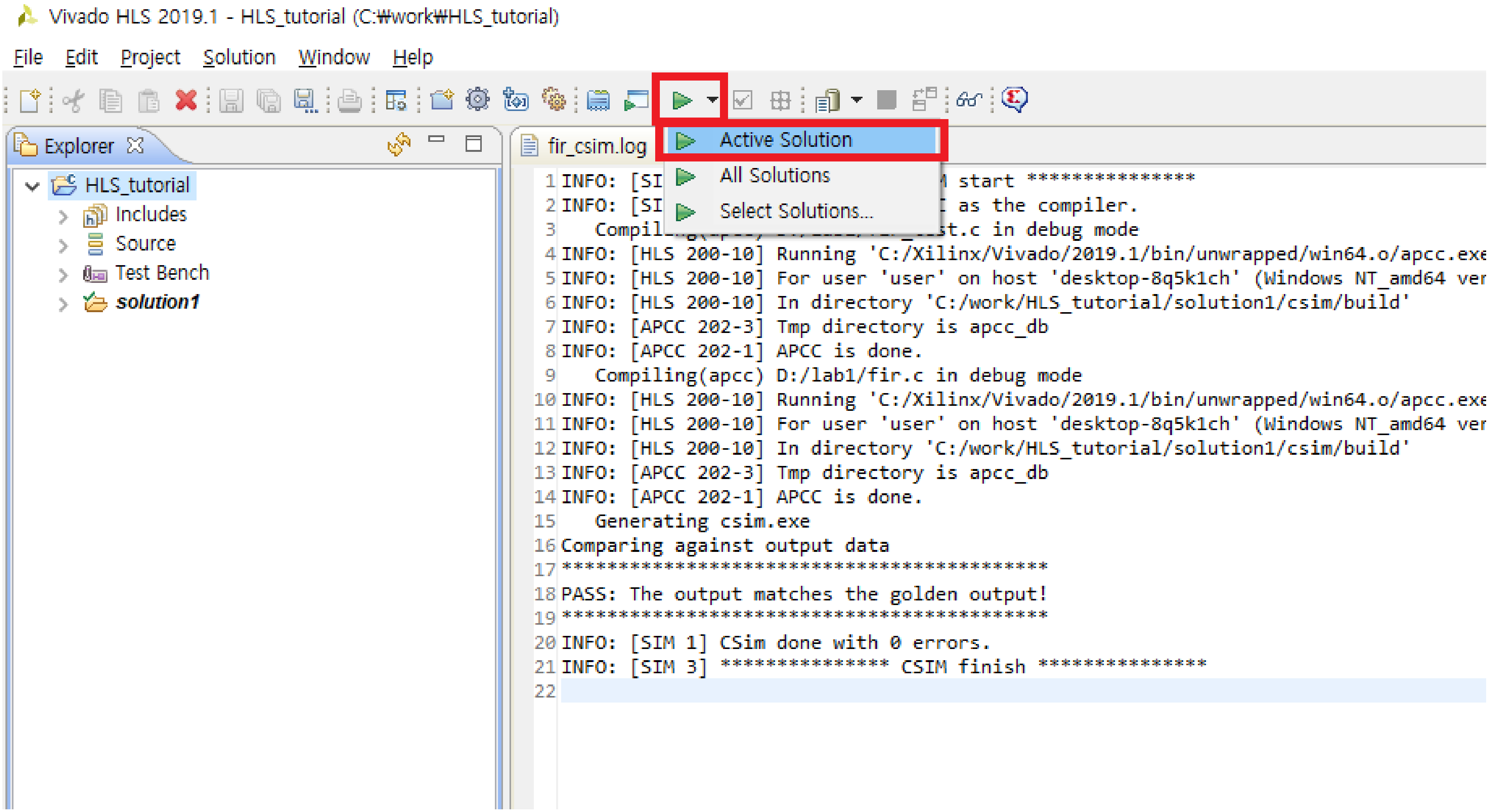
Default option으로 simulation launch

C-Simulation result

fir_csim.log

```
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
   Compiling(apcc) D:/lab1/fir_test.c in debug mode
INFO: [HLS 200-10] Running 'C:/Xilinx/Vivado/2019.1/bin/unwrapped/win64.o/apcc.exe'
INFO: [HLS 200-10] For user 'user' on host 'desktop-8q5k1ch' (Windows NT_amd64 version 6.2) on Thu J
INFO: [HLS 200-10] In directory 'C:/work/HLS_tutorial/solution1/csim/build'
INFO: [APCC 202-3] Tmp directory is apcc_db
INFO: [APCC 202-1] APCC is done.
   Compiling(apcc) D:/lab1/fir.c in debug mode
INFO: [HLS 200-10] Running 'C:/Xilinx/Vivado/2019.1/bin/unwrapped/win64.o/apcc.exe'
INFO: [HLS 200-10] For user 'user' on host 'desktop-8q5k1ch' (Windows NT_amd64 version 6.2) on Thu J
INFO: [HLS 200-10] In directory 'C:/work/HLS_tutorial/solution1/csim/build'
INFO: [APCC 202-3] Tmp directory is apcc_db
INFO: [APCC 202-1] APCC is done.
   Generating csim.exe
Comparing against output data
*******************************************
PASS: The output matches the golden output!
*******************************************
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
```

Synthesis to RTL



Synthesis Report

Synthesis(solution1)(fir_csynth.rpt)

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.510	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
56	56	56	56	none

Detail

Instance

N/A

Loop

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- Shift_Accum_Loop	55	55	5	-	-	11	no

Execution Timing, Latency

Resource

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	3	0	85	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	0	-	64	6	0
Multiplexer	-	-	-	116	-
Register	-	-	177	-	-
Total	0	3	241	207	0
Available	280	220	106400	53200	0
Utilization (%)	0	1	~0	~0	0

Detail

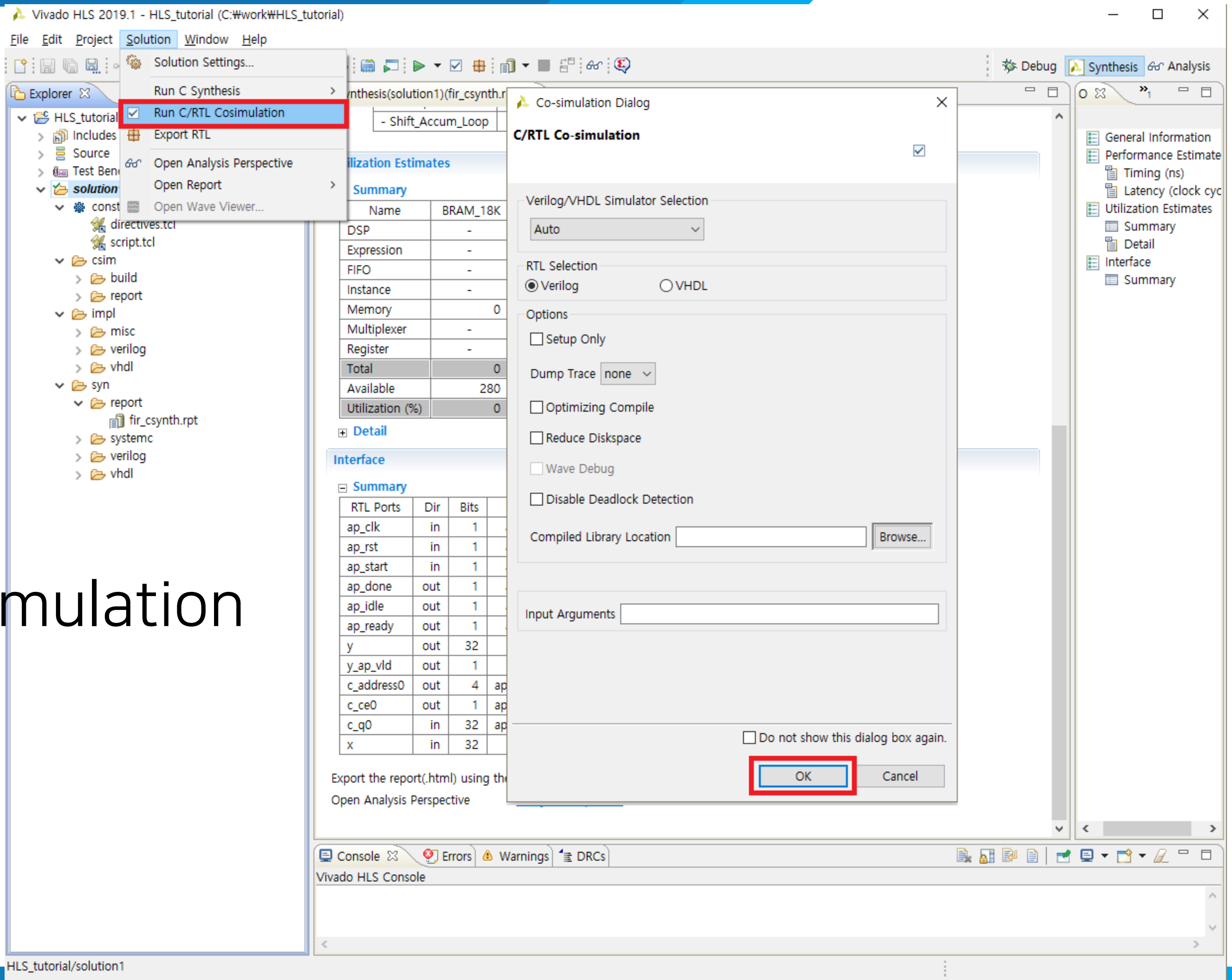
interface

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	fir	return value
ap_rst	in	1	ap_ctrl_hs	fir	return value
ap_start	in	1	ap_ctrl_hs	fir	return value
ap_done	out	1	ap_ctrl_hs	fir	return value
ap_idle	out	1	ap_ctrl_hs	fir	return value
ap_ready	out	1	ap_ctrl_hs	fir	return value
y	out	32	ap_vld	y	pointer
y_ap_vld	out	1	ap_vld	y	pointer
c_address0	out	4	ap_memory	c	array
c_ce0	out	1	ap_memory	c	array
c_q0	in	32	ap_memory	c	array
x	in	32	ap_none	x	scalar

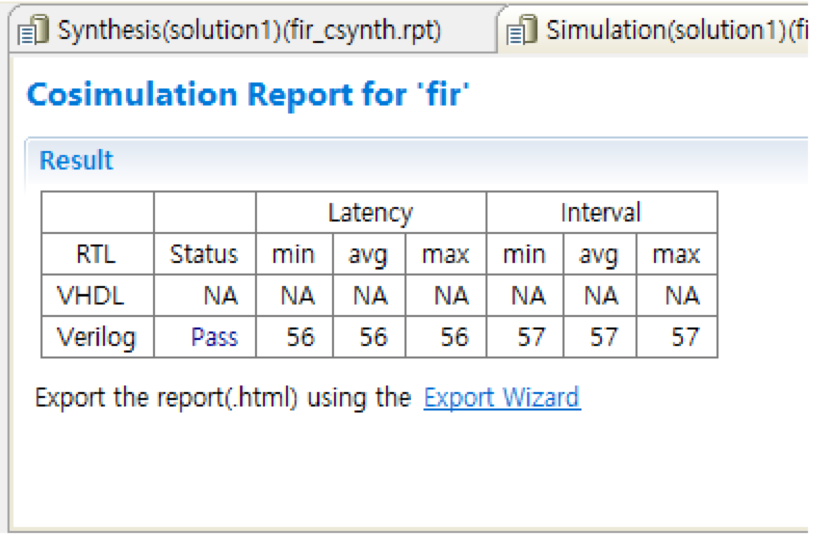
RTL Verification



C/RTL Co-simulation

HUINS
Human Intelligent System

Co-Simulation Result



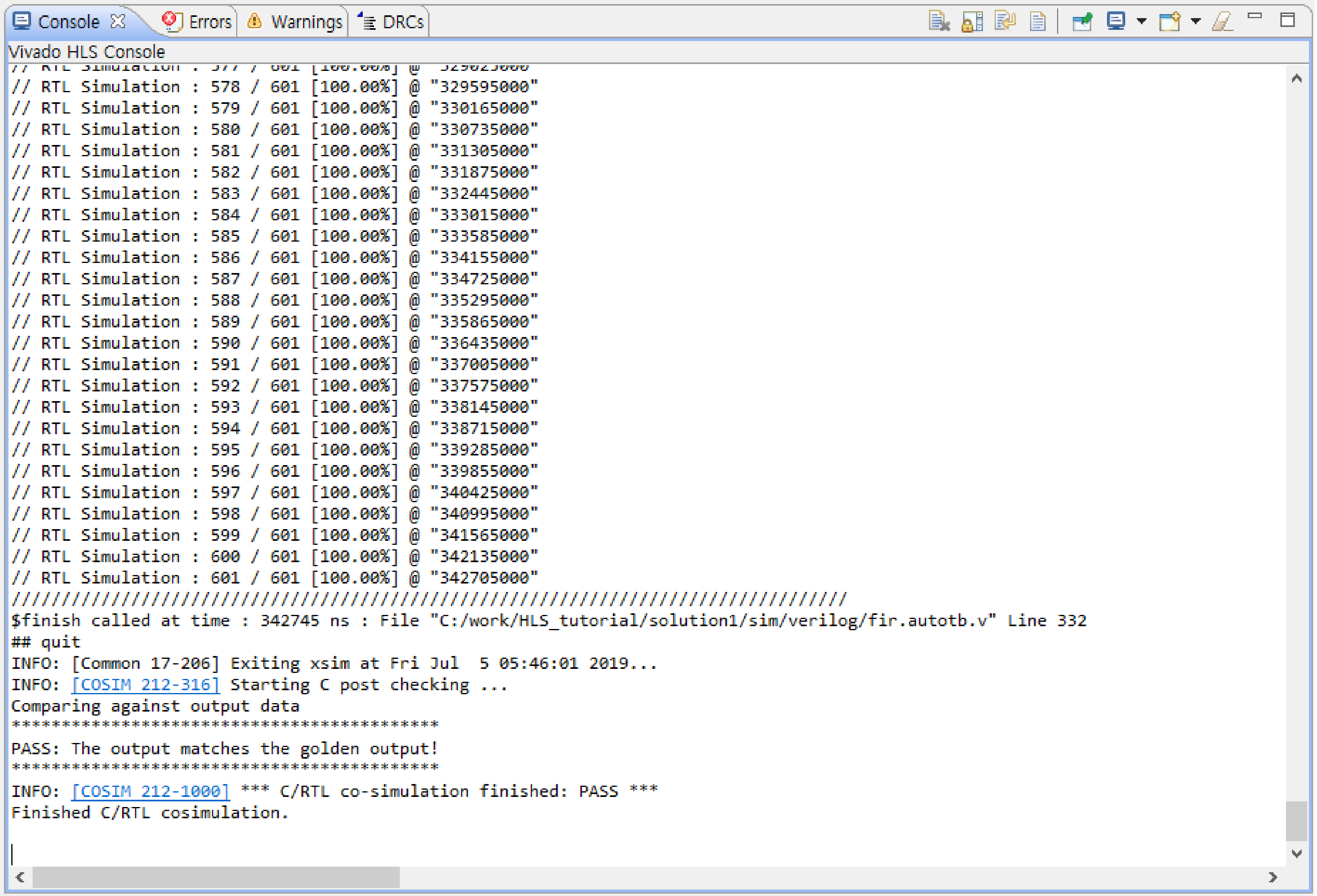
Synthesis(solution1)(fir_csynth.rpt) | Simulation(solution1)(fi

Cosimulation Report for 'fir'

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	56	56	56	57	57	57

Export the report(.html) using the Export Wizard



Console | Errors | Warnings | DRCs

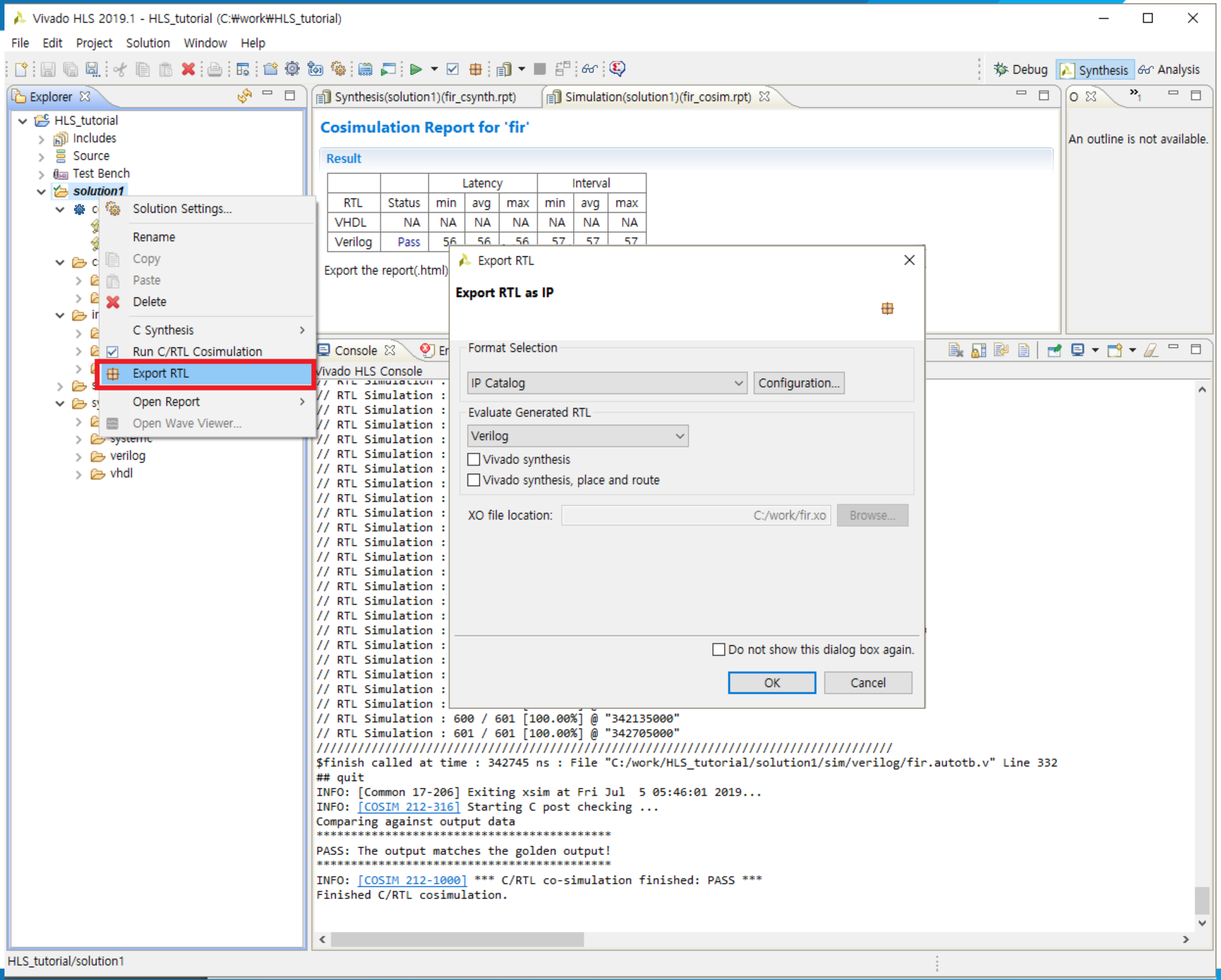
Vivado HLS Console

```
// RTL Simulation : 578 / 601 [100.00%] @ "329595000"
// RTL Simulation : 579 / 601 [100.00%] @ "330165000"
// RTL Simulation : 580 / 601 [100.00%] @ "330735000"
// RTL Simulation : 581 / 601 [100.00%] @ "331305000"
// RTL Simulation : 582 / 601 [100.00%] @ "331875000"
// RTL Simulation : 583 / 601 [100.00%] @ "332445000"
// RTL Simulation : 584 / 601 [100.00%] @ "333015000"
// RTL Simulation : 585 / 601 [100.00%] @ "333585000"
// RTL Simulation : 586 / 601 [100.00%] @ "334155000"
// RTL Simulation : 587 / 601 [100.00%] @ "334725000"
// RTL Simulation : 588 / 601 [100.00%] @ "335295000"
// RTL Simulation : 589 / 601 [100.00%] @ "335865000"
// RTL Simulation : 590 / 601 [100.00%] @ "336435000"
// RTL Simulation : 591 / 601 [100.00%] @ "337005000"
// RTL Simulation : 592 / 601 [100.00%] @ "337575000"
// RTL Simulation : 593 / 601 [100.00%] @ "338145000"
// RTL Simulation : 594 / 601 [100.00%] @ "338715000"
// RTL Simulation : 595 / 601 [100.00%] @ "339285000"
// RTL Simulation : 596 / 601 [100.00%] @ "339855000"
// RTL Simulation : 597 / 601 [100.00%] @ "340425000"
// RTL Simulation : 598 / 601 [100.00%] @ "340995000"
// RTL Simulation : 599 / 601 [100.00%] @ "341565000"
// RTL Simulation : 600 / 601 [100.00%] @ "342135000"
// RTL Simulation : 601 / 601 [100.00%] @ "342705000"
////////////////////////////////////////////////////////////////////////////////////
$finish called at time : 342745 ns : File "C:/work/HLS_tutorial/solution1/sim/verilog/fir.autotb.v" Line 332
## quit
INFO: [Common 17-206] Exiting xsim at Fri Jul  5 05:46:01 2019...
INFO: [COSIM 212-316] Starting C post checking ...
Comparing against output data
*******************************************
PASS: The output matches the golden output!
*******************************************
INFO: [COSIM 212-1000] *** C/RTL co-simulation finished: PASS ***
Finished C/RTL cosimulation.
```

C Test Bench

1. Creates Input Vector
2. Simulates RTL Design
3. Check RTL Output

IP - Creation



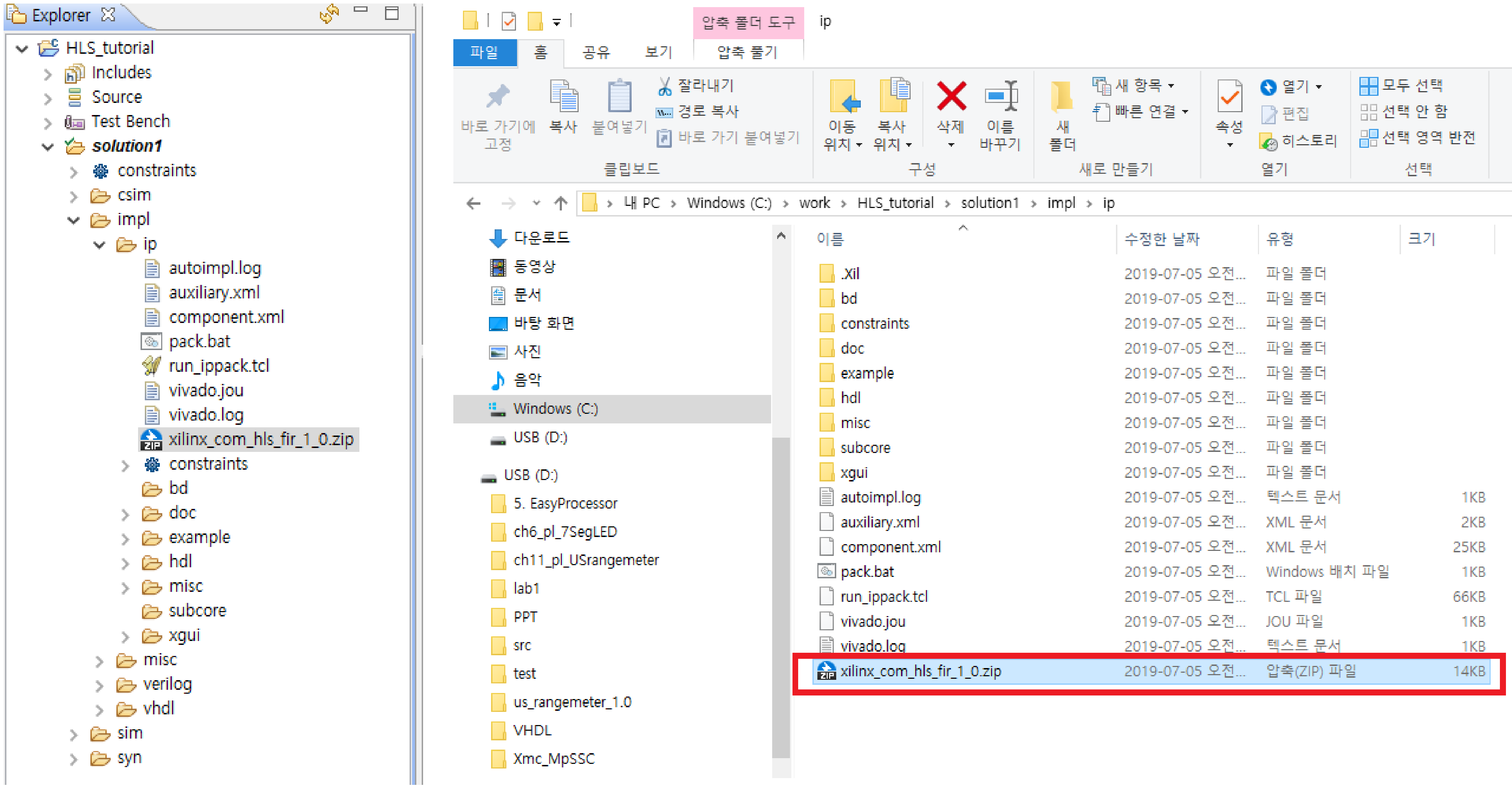
Export RTL as IP

Target Language 설정

Synthesis

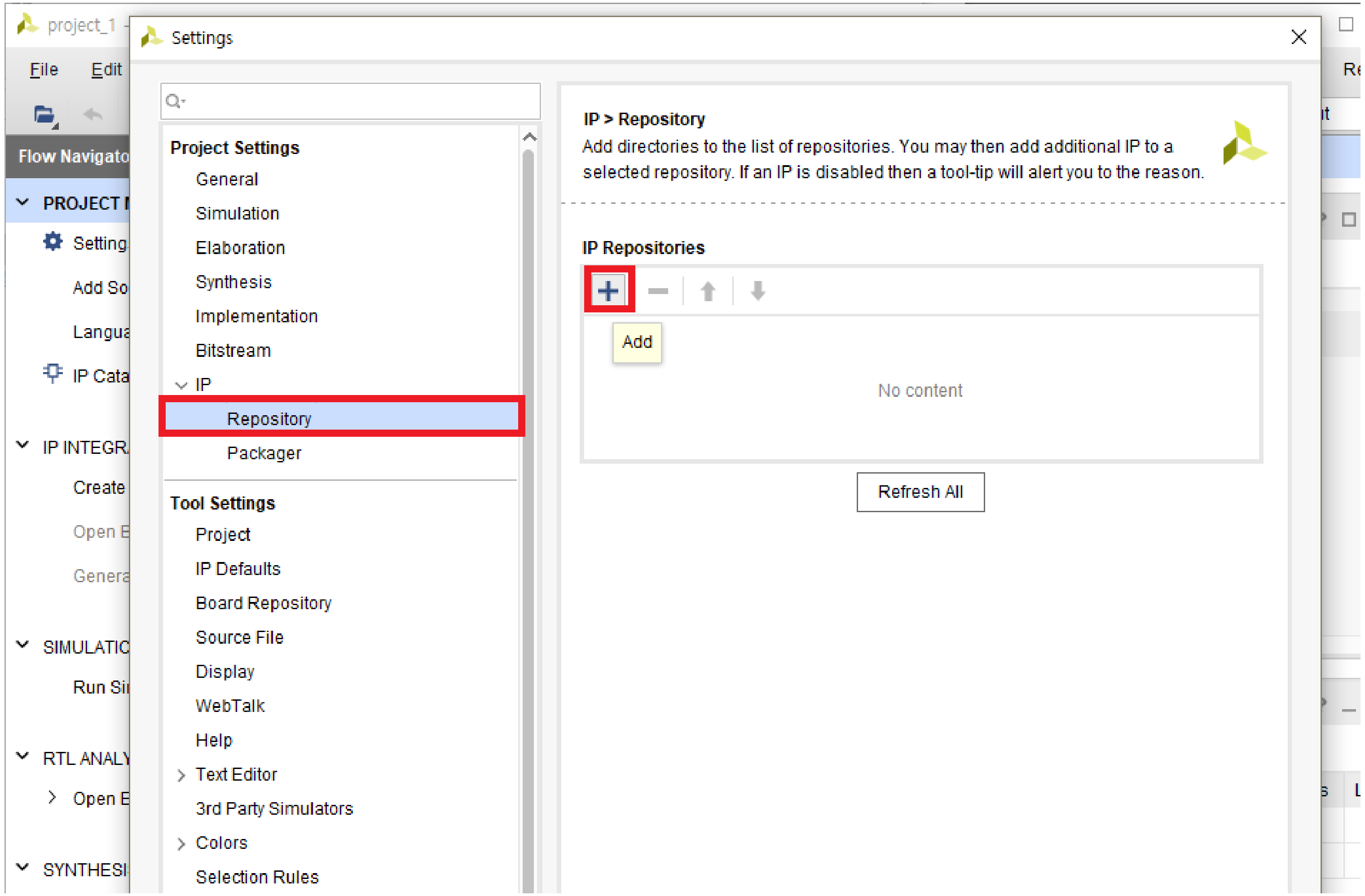
Implementation 여부 설정

Exported IP



HUINS
Human Intelligent System

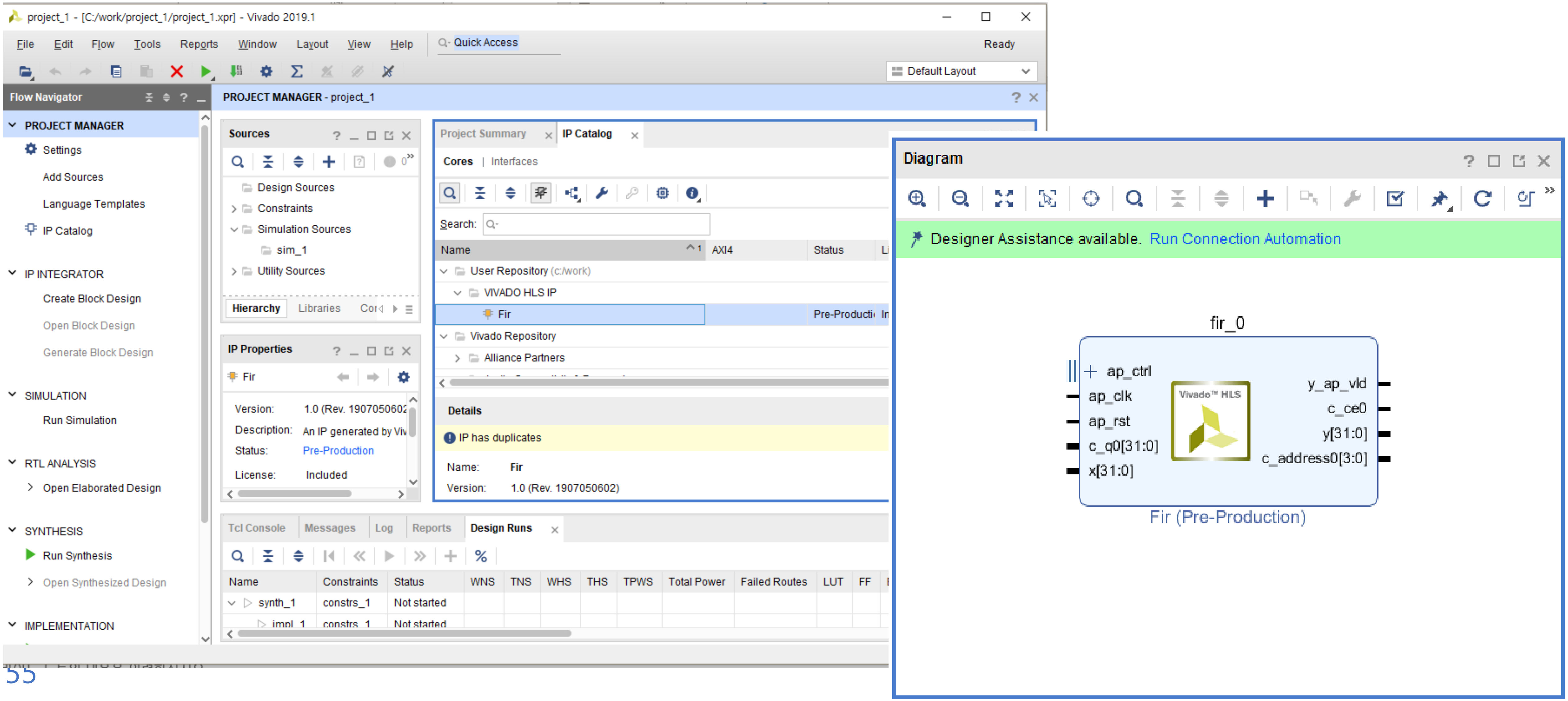
Use HLS in Vivado



Unzip HLS IP

Add IP Repository

Import in Vivado



Synthesis Report

Synthesis(solution1)(fir_csynth.rpt)

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.510	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
56	56	56	56	none

Detail

Instance

N/A

Loop

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- Shift_Accum_Loop	55	55	5	-	-	11	no

Execution Timing, Latency

Resource

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	3	0	85	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	0	-	64	6	0
Multiplexer	-	-	-	116	-
Register	-	-	177	-	-
Total	0	3	241	207	0
Available	280	220	106400	53200	0
Utilization (%)	0	1	~0	~0	0

Detail

interface

Interface

Summary

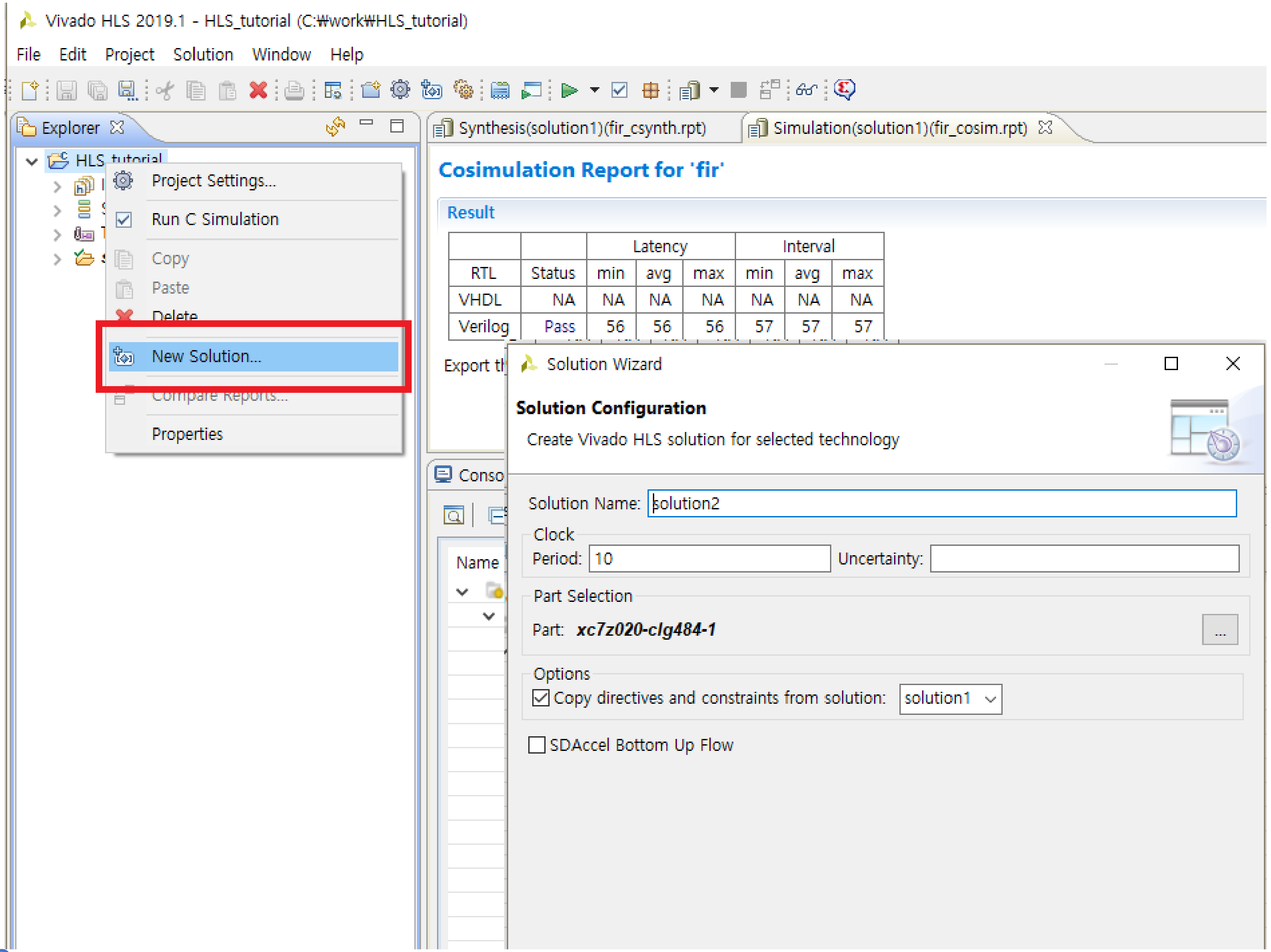
RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	fir	return value
ap_rst	in	1	ap_ctrl_hs	fir	return value
ap_start	in	1	ap_ctrl_hs	fir	return value
ap_done	out	1	ap_ctrl_hs	fir	return value
ap_idle	out	1	ap_ctrl_hs	fir	return value
ap_ready	out	1	ap_ctrl_hs	fir	return value
y	out	32	ap_vld	y	pointer
y_ap_vld	out	1	ap_vld	y	pointer
c_address0	out	4	ap_memory	c	array
c_ce0	out	1	ap_memory	c	array
c_q0	in	32	ap_memory	c	array
x	in	32	ap_none	x	scalar

Optimize Port to

Target

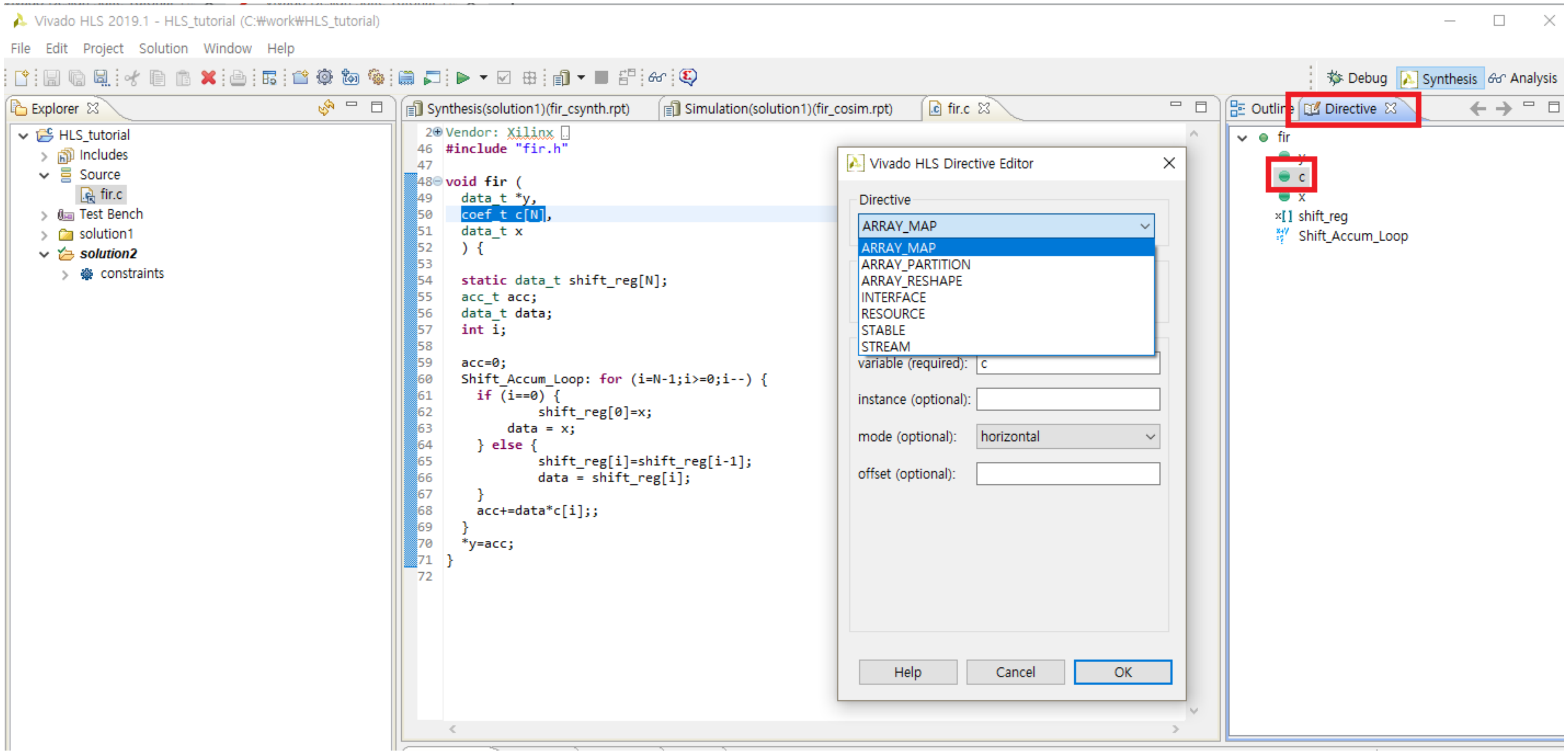
- Port C(input array): single port RAM access
- Port X(input data): input data valid signal
- Port Y(output data): output data valid signal

Create New Solution



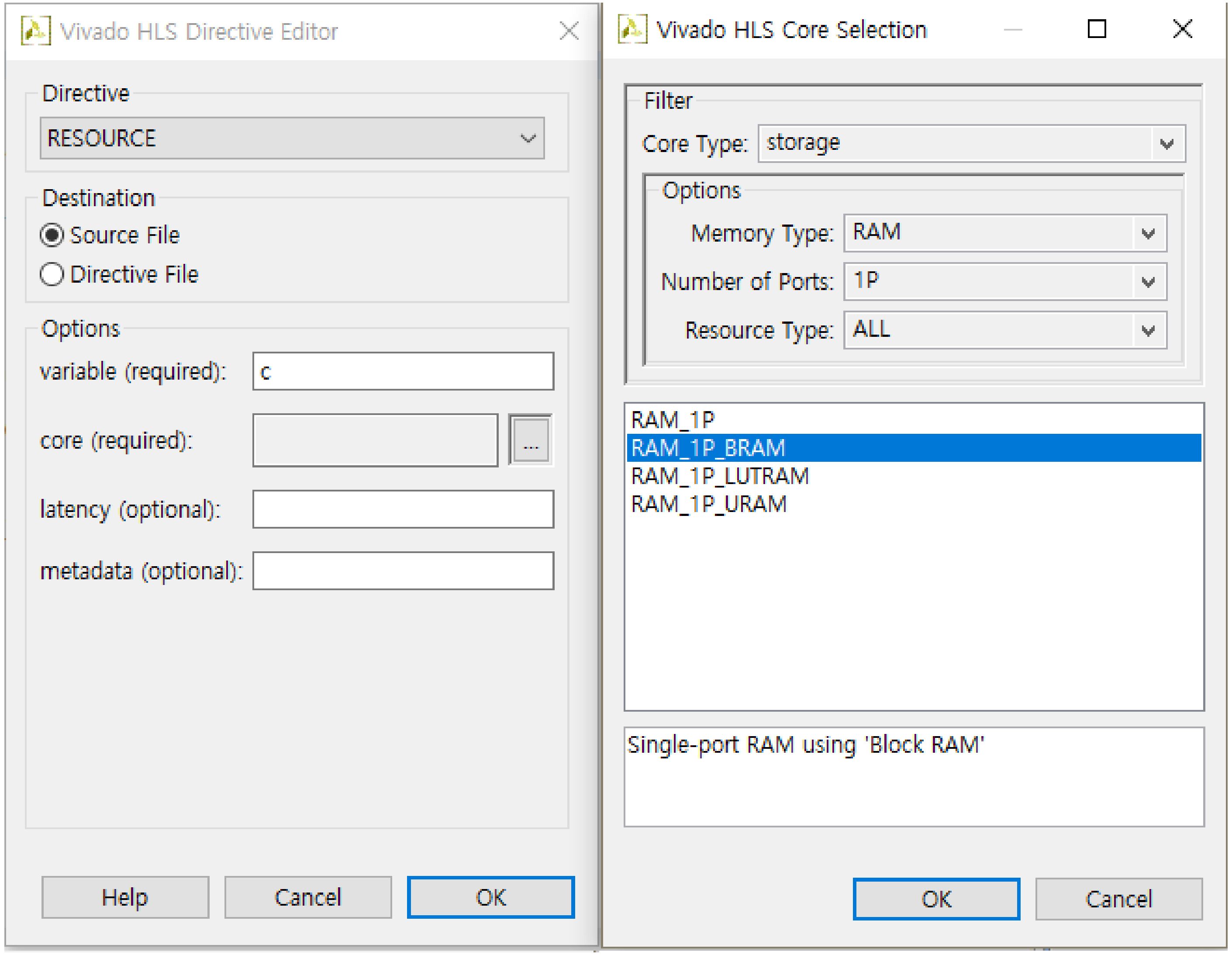
최적화를 통한 성능 향상이 목표이므로, 다른 것 변경하지 않고 새 솔루션 생성

Directives



Vivado HLS 2019.1 - HLS_tutorial (C:₩work₩HLS_tutorial)
File Edit Project Solution Window Help
Debug Synthesis Analysis
Explorer
HLS_tutorial
Includes
Source
fir.c
Test Bench
solution1
solution2
constraints
Synthesis(solution1)(fir_csynth.rpt)
Simulation(solution1)(fir_cosim.rpt)
fir.c
Vendor: Xilinx
#include "fir.h"
void fir (
data_t *y,
coef_t c[N],
data_t x
) {
static data_t shift_reg[N];
acc_t acc;
data_t data;
int i;
acc=0;
Shift_Accum_Loop: for (i=N-1;i>=0;i--) {
if (i==0) {
shift_reg[0]=x;
data = x;
} else {
shift_reg[i]=shift_reg[i-1];
data = shift_reg[i];
}
acc+=data*c[i];;
}
*y=acc;
}
Vivado HLS Directive Editor
Directive
ARRAY_MAP
ARRAY_MAP
ARRAY_PARTITION
ARRAY_RESHAPE
INTERFACE
RESOURCE
STABLE
STREAM
variable (required): c
instance (optional):
mode (optional): horizontal
offset (optional):
Help
Cancel
OK
Outline
Directive
fir
y
c
x
shift_reg
Shift_Accum_Loop

Directives for Port C



Port C: Single Port RAM

Directives as #pragma

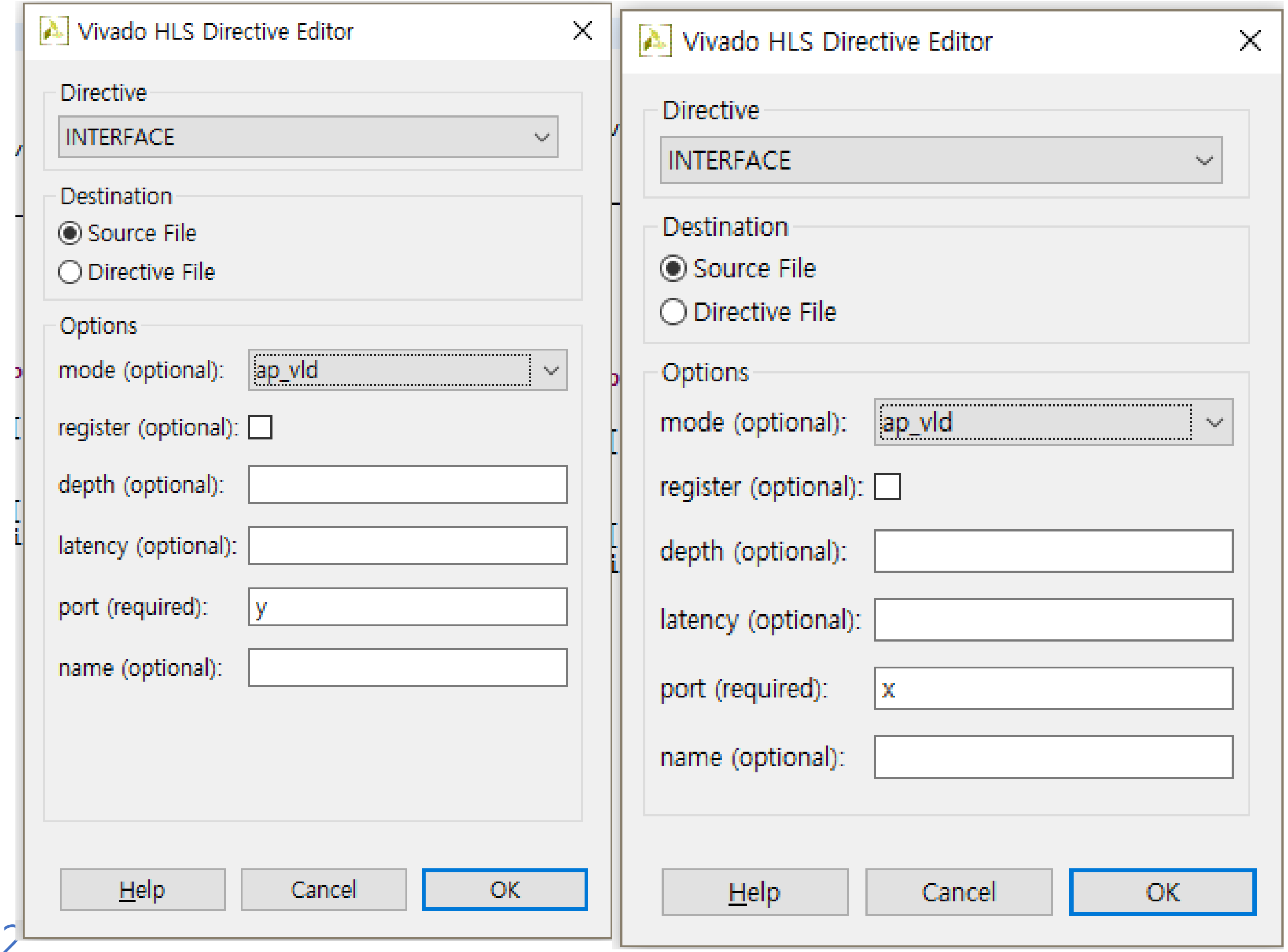
Synthesis(solution1)(fir_csynth.rpt) | Simulation(solution1)(fir_cosim.rpt) | *fir.c

```c
Vendor: Xilinx
#include "fir.h"

void fir (
  data_t *y,
  coef_t c[N],
  data_t x
  ) {
#pragma HLS RESOURCE variable=c core=RAM_1P_BRAM

  static data_t shift_reg[N];
  acc_t acc;
  data_t data;
  int i;

  acc=0;
  Shift_Accum_Loop: for (i=N-1;i>=0;i--) {
    if (i==0) {
          shift_reg[0]=x;
        data = x;
    } else {
          shift_reg[i]=shift_reg[i-1];
          data = shift_reg[i];
    }
    acc+=data*c[i];;
  }
  *y=acc;
}
```

Outline | Directive

- fir
  - y
  - c
    - # HLS RESOURCE variable=c core=RAM_1P_BRAM
  - x
  - shift_reg
  - Shift_Accum_Loop

Directives for X, Y



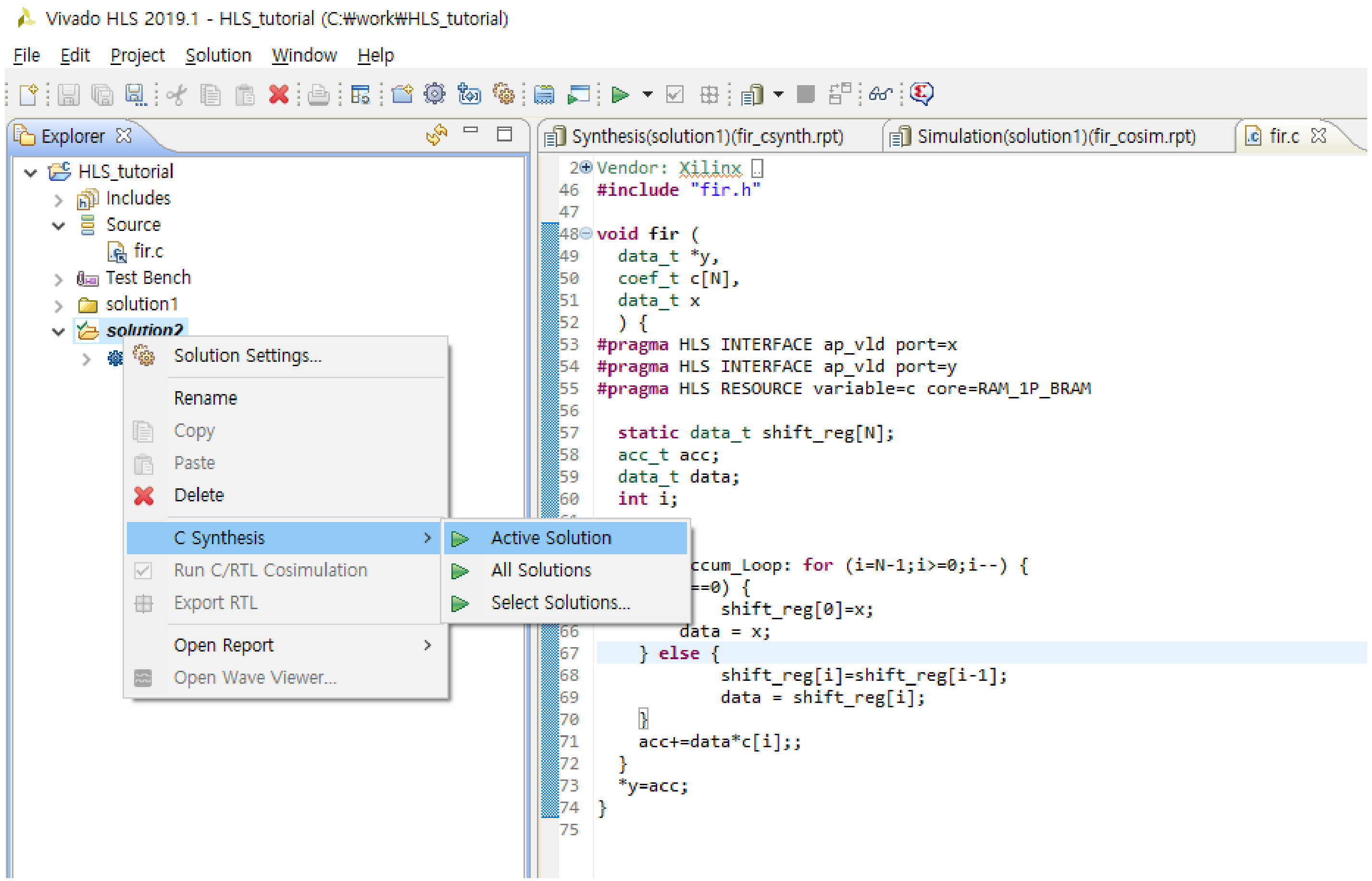
- Port X(input data): valid
- Port Y(output data): valid

Inserted #pragma

Synthesis(solution1)(fir_csynth.rpt) | Simulation(solution1)(fir_cosim.rpt) | *fir.c

```c
Vendor: Xilinx
#include "fir.h"

void fir (
  data_t *y,
  coef_t c[N],
  data_t x
  ) {
#pragma HLS INTERFACE ap_vld port=x
#pragma HLS INTERFACE ap_vld port=y
#pragma HLS RESOURCE variable=c core=RAM_1P_BRAM

  static data_t shift_reg[N];
  acc_t acc;
  data_t data;
  int i;

  acc=0;
  Shift_Accum_Loop: for (i=N-1;i>=0;i--) {
    if (i==0) {
            shift_reg[0]=x;
        data = x;
    } else {
            shift_reg[i]=shift_reg[i-1];
            data = shift_reg[i];
    }
    acc+=data*c[i];;
  }
  *y=acc;
}
```

C Synthesis(solution2)



Synthesis Report - Interface

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	fir	return value
ap_rst	in	1	ap_ctrl_hs	fir	return value
ap_start	in	1	ap_ctrl_hs	fir	return value
ap_done	out	1	ap_ctrl_hs	fir	return value
ap_idle	out	1	ap_ctrl_hs	fir	return value
ap_ready	out	1	ap_ctrl_hs	fir	return value
y	out	32	ap_vld	y	pointer
y_ap_vld	out	1	ap_vld	y	pointer
c_address0	out	4	ap_memory	c	array
c_ce0	out	1	ap_memory	c	array
c_q0	in	32	ap_memory	c	array
x	in	32	ap_none	x	scalar

Solution1

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	fir	return value
ap_rst	in	1	ap_ctrl_hs	fir	return value
ap_start	in	1	ap_ctrl_hs	fir	return value
ap_done	out	1	ap_ctrl_hs	fir	return value
ap_idle	out	1	ap_ctrl_hs	fir	return value
ap_ready	out	1	ap_ctrl_hs	fir	return value
y	out	32	ap_vld	y	pointer
y_ap_vld	out	1	ap_vld	y	pointer
c_address0	out	4	ap_memory	c	array
c_ce0	out	1	ap_memory	c	array
c_q0	in	32	ap_memory	c	array
x	in	32	ap_vld	x	scalar
x_ap_vld	in	1	ap_vld	x	scalar

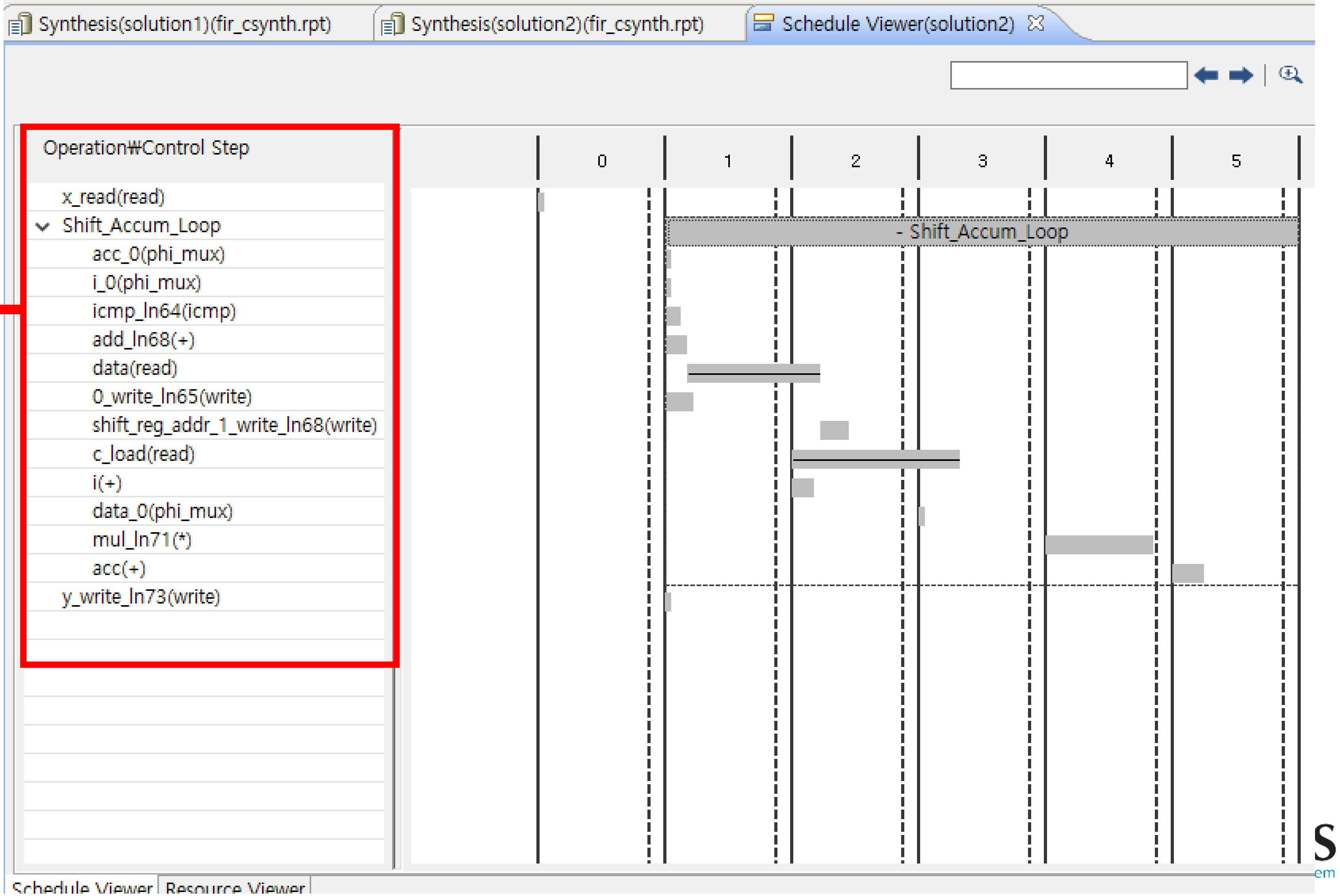
Solution2

HUINS
Human Intelligent System

Analyze

Export the report(.html) using the Export Wizard
Open Analysis Perspective Analysis Perspective

RTL Hierarchy module operation

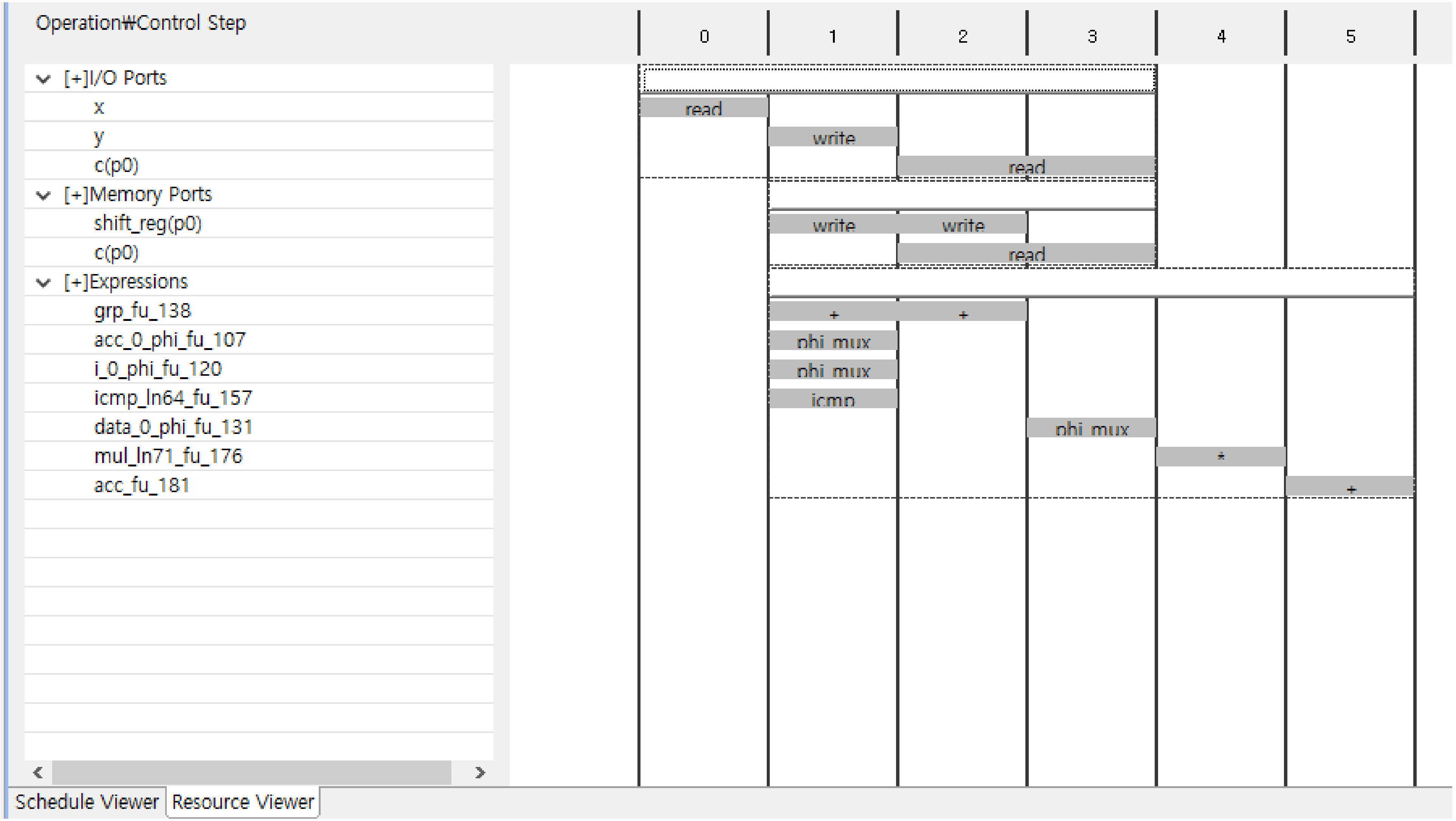


Operation Flow

x_read() – Shift_Accum_Loop operation

HUINS
Human Intelligent System

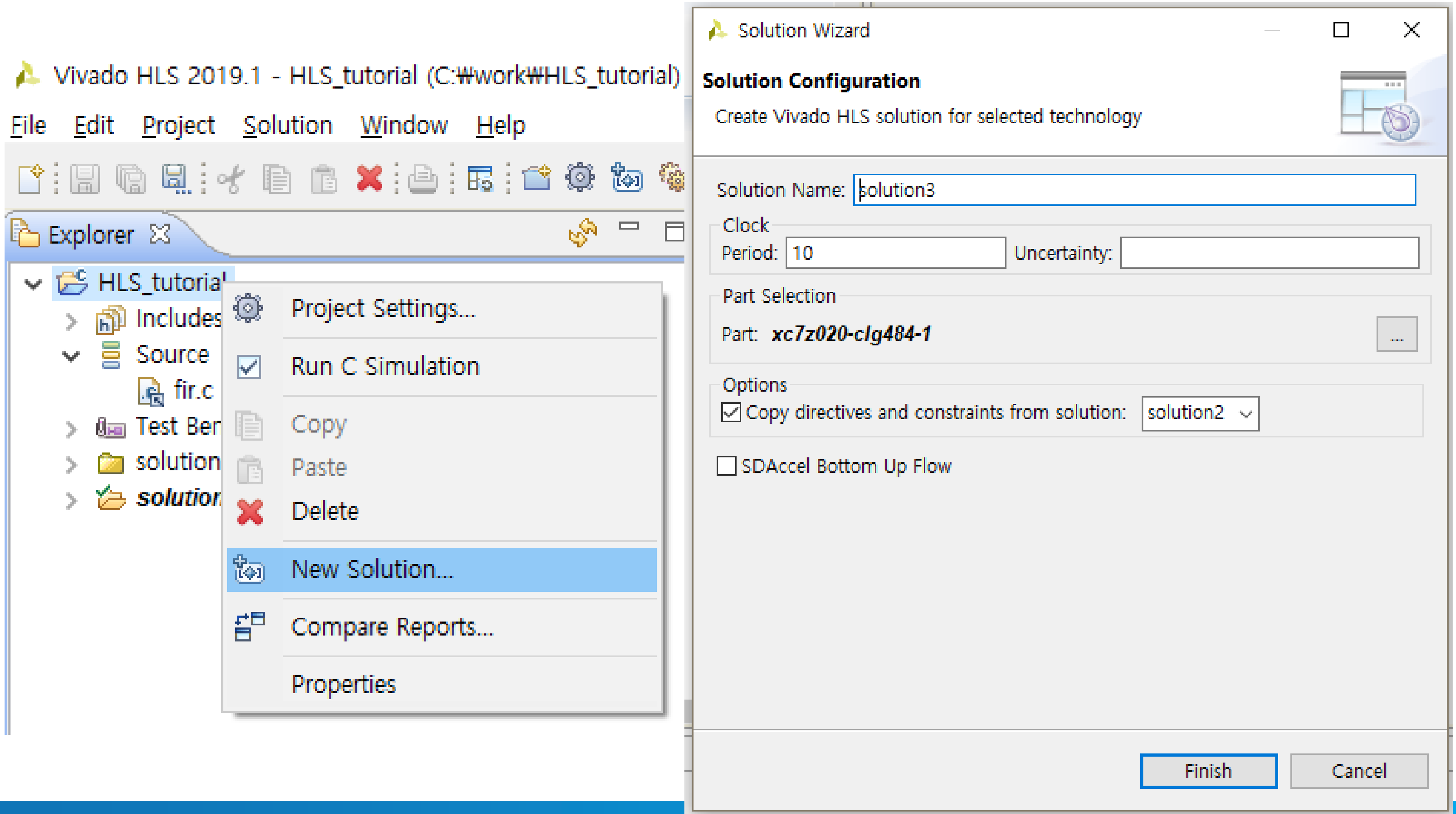
Analysis - Resource



Higher optimization

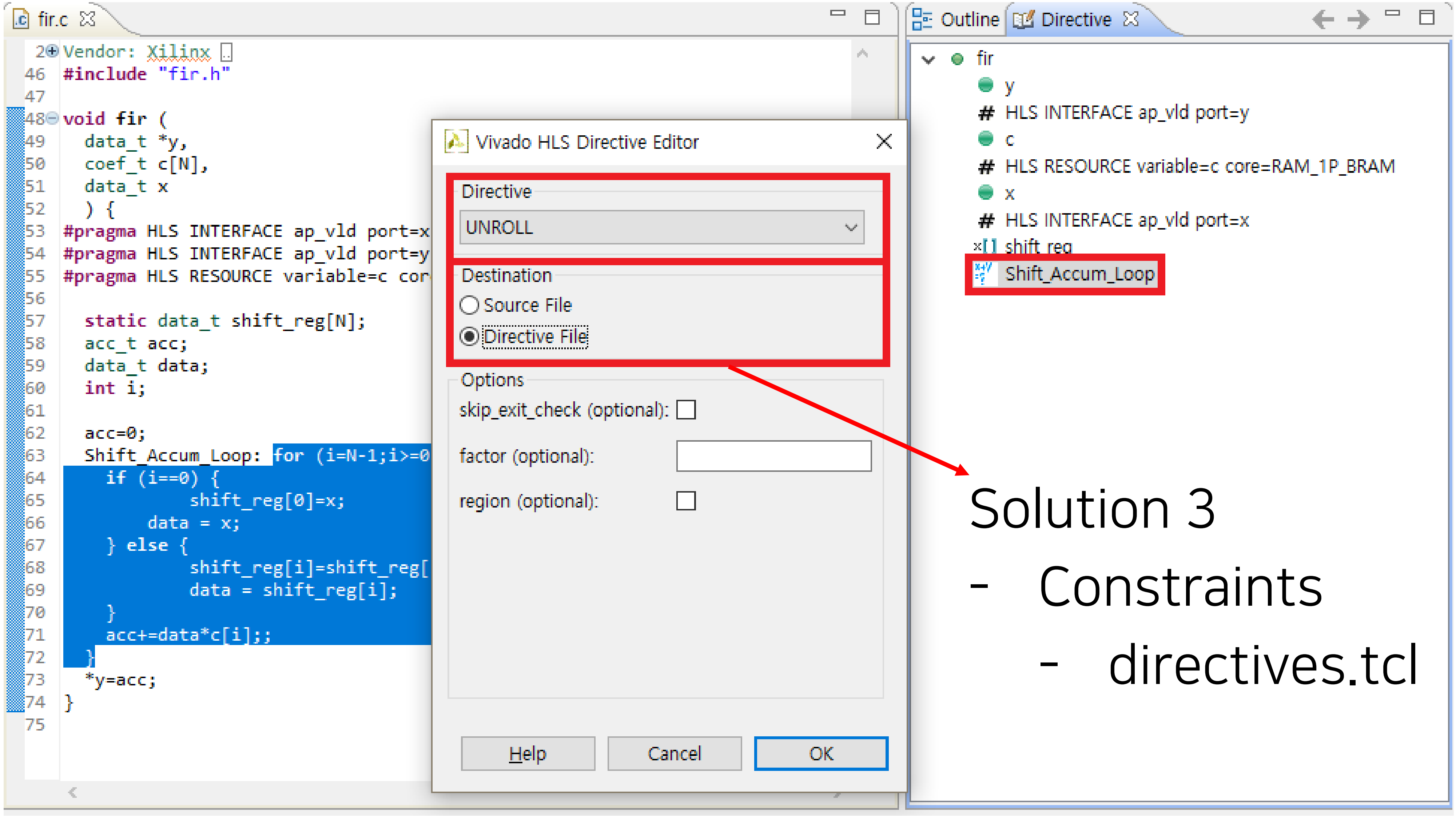
- 루프
  - Default → 루프 연산 수행
  - 하나의 루프를 재사용 → Dependency, 리소스 최적화
  - Unroll Loop(병렬화)
- block RAM, register
  - Array shift_reg[ ] 구현에 BRAM이 사용됨
  - Default: BRAM
  - Shift Register로 구현하기 위해 BRAM → 개별 레지스터로 분할

Create New Solution



HUINS
Human Intelligent System

Unroll the Loop

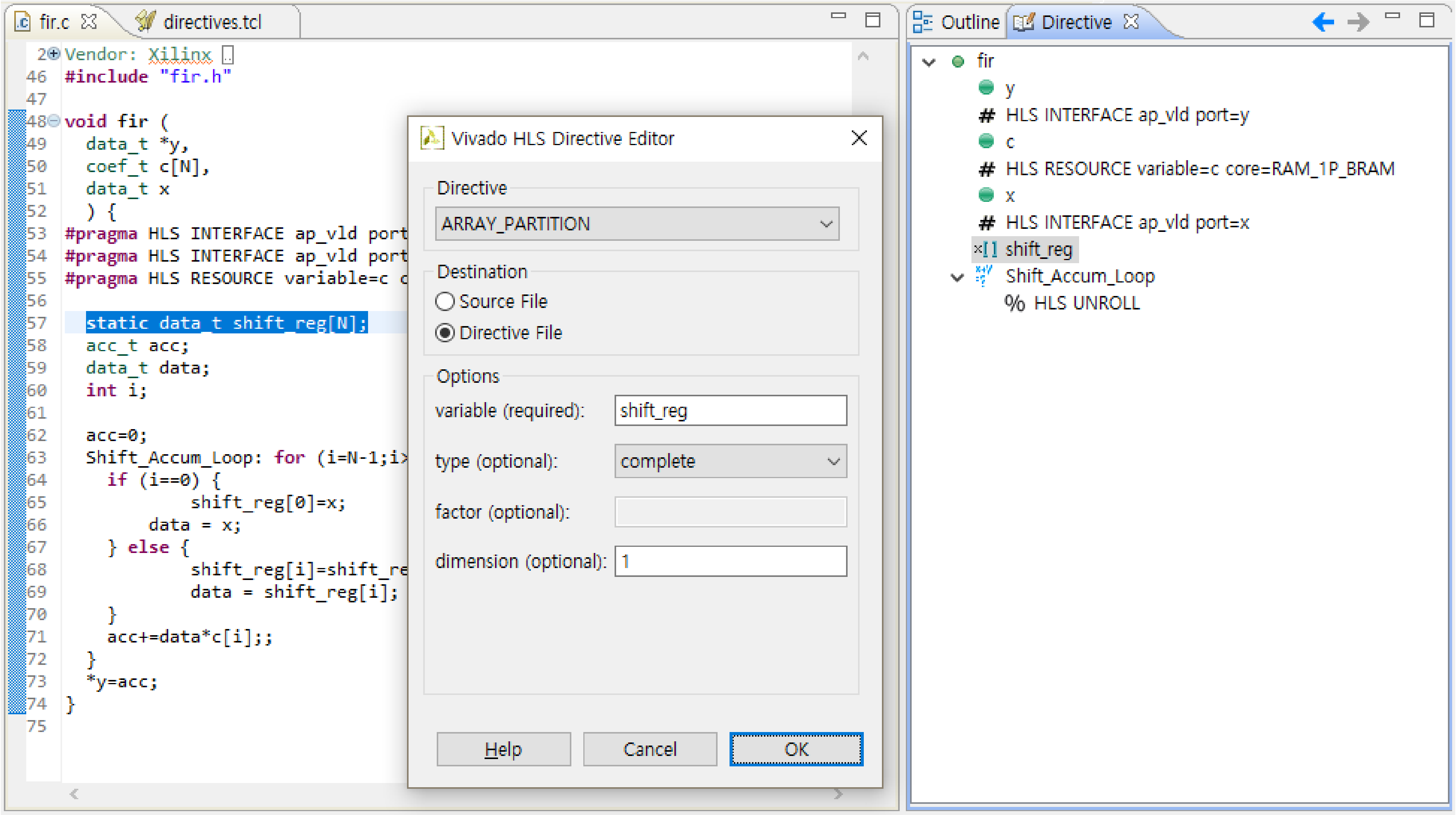


Solution 3

- Constraints
  - directives.tcl

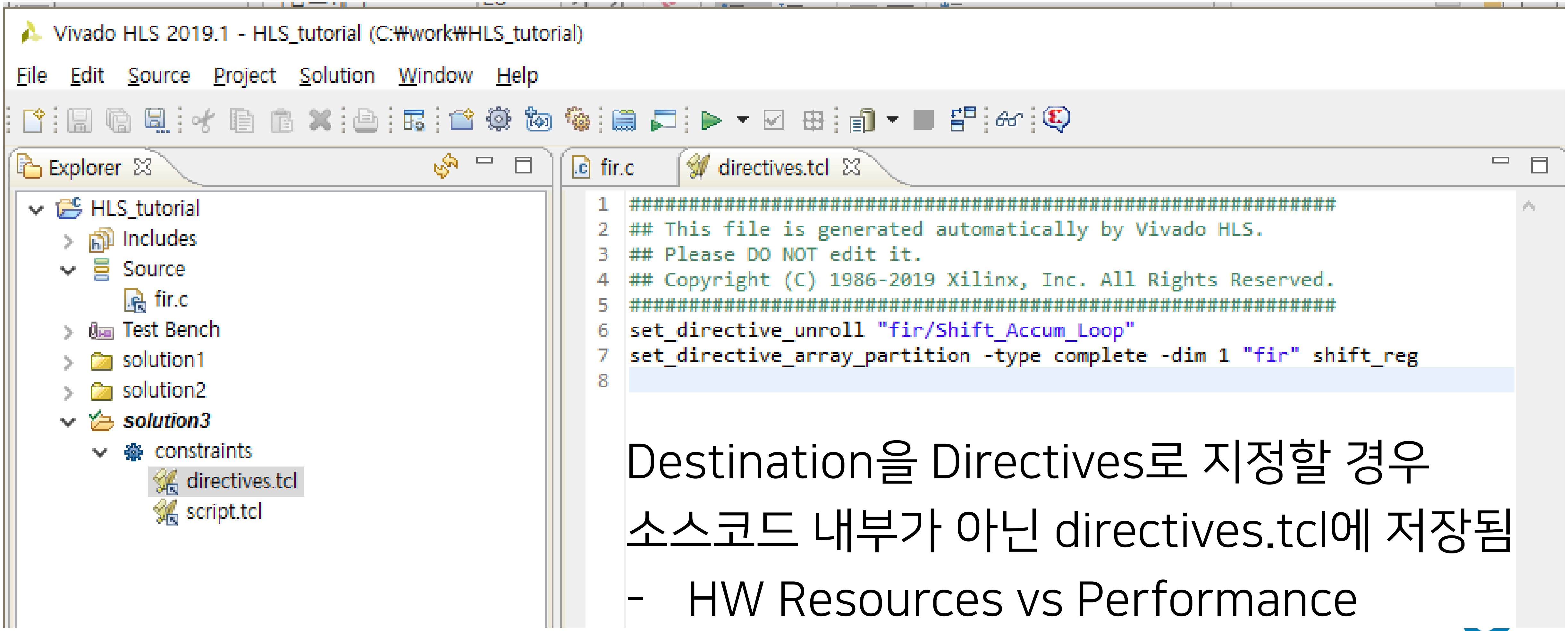
HUINS
Human Intelligent System

Partition Array



HUINS
Human Intelligent System

directives.tcl

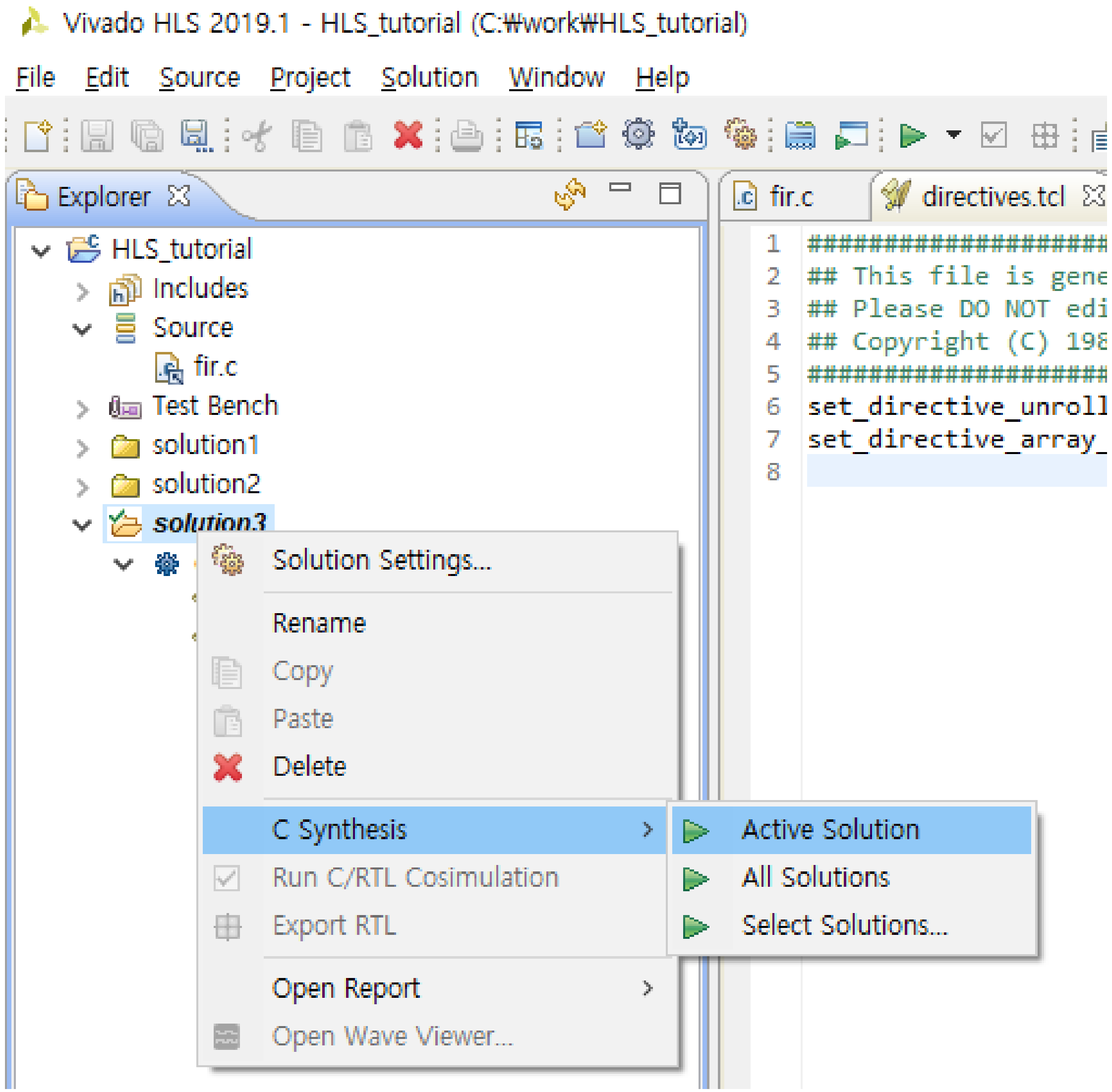


Destination을 Directives로 지정할 경우 소스코드 내부가 아닌 directives.tcl에 저장됨

- HW Resources vs Performance
- 여러 버전의 최적화 정도 비교 시 사용

HUINS Human Intelligent System

Synthesis


Vivado HLS 2019.1 - HLS_tutorial (C:₩work₩HLS_tutorial)
File Edit Source Project Solution Window Help
Explorer
HLS_tutorial
Includes
Source
fir.c
Test Bench
solution1
solution2
solution3
fir.c
directives.tcl
1 ####################
2 ## This file is gene
3 ## Please DO NOT edi
4 ## Copyright (C) 198
5 ####################
6 set_directive_unroll
7 set_directive_array_
8
Solution Settings...
Rename
Copy
Paste
Delete
C Synthesis
Run C/RTL Cosimulation
Export RTL
Open Report
Open Wave Viewer...
Active Solution
All Solutions
Select Solutions...

Synthesis Report

Performance Estimates

- Timing (ns)
  - Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.510	1.25

- Latency (clock cycles)
  - Summary

Latency		Interval		
min	max	min	max	Type
56	56	56	56	none

- Detail
  - Instance

N/A

Performance Estimates

- Timing (ns)
  - Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.742	1.25

- Latency (clock cycles)
  - Summary

Latency		Interval		
min	max	min	max	Type
14	14	14	14	none

- Detail
  - Instance
  - Loop

x4 faster

HUINS
Human Intelligent System

Summary

- HLS
  - C/C++ 작성된 SW를 HW IP로 변환, 테스트
  - Data Types, I/O Ports
  - HLS를 이용한 최적화

HUINS
Human Intelligent System

What's Next?

HLS

- C/C++ → HW IP 설계
- FPGA / ZYNQ 모두 사용

SDSoC(Software Defined System-On-a-Chip)

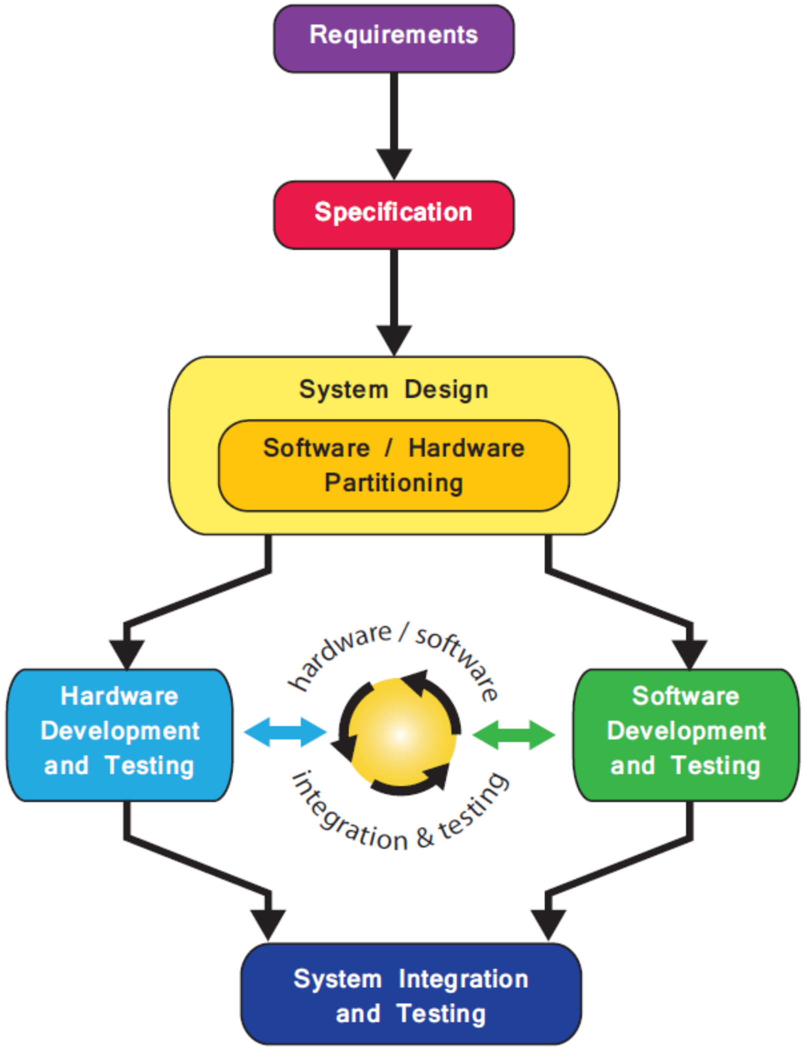
- Zynq용 HW Acceleration system
- System Profiling, SW/HW Partitioning, HLS, Compile 동시에

Zynq SoC 설계(HDL, HLS)

Xilinx VIVADO

VHDL/Verilog
HLS(SDSoC)

AXI HW IP



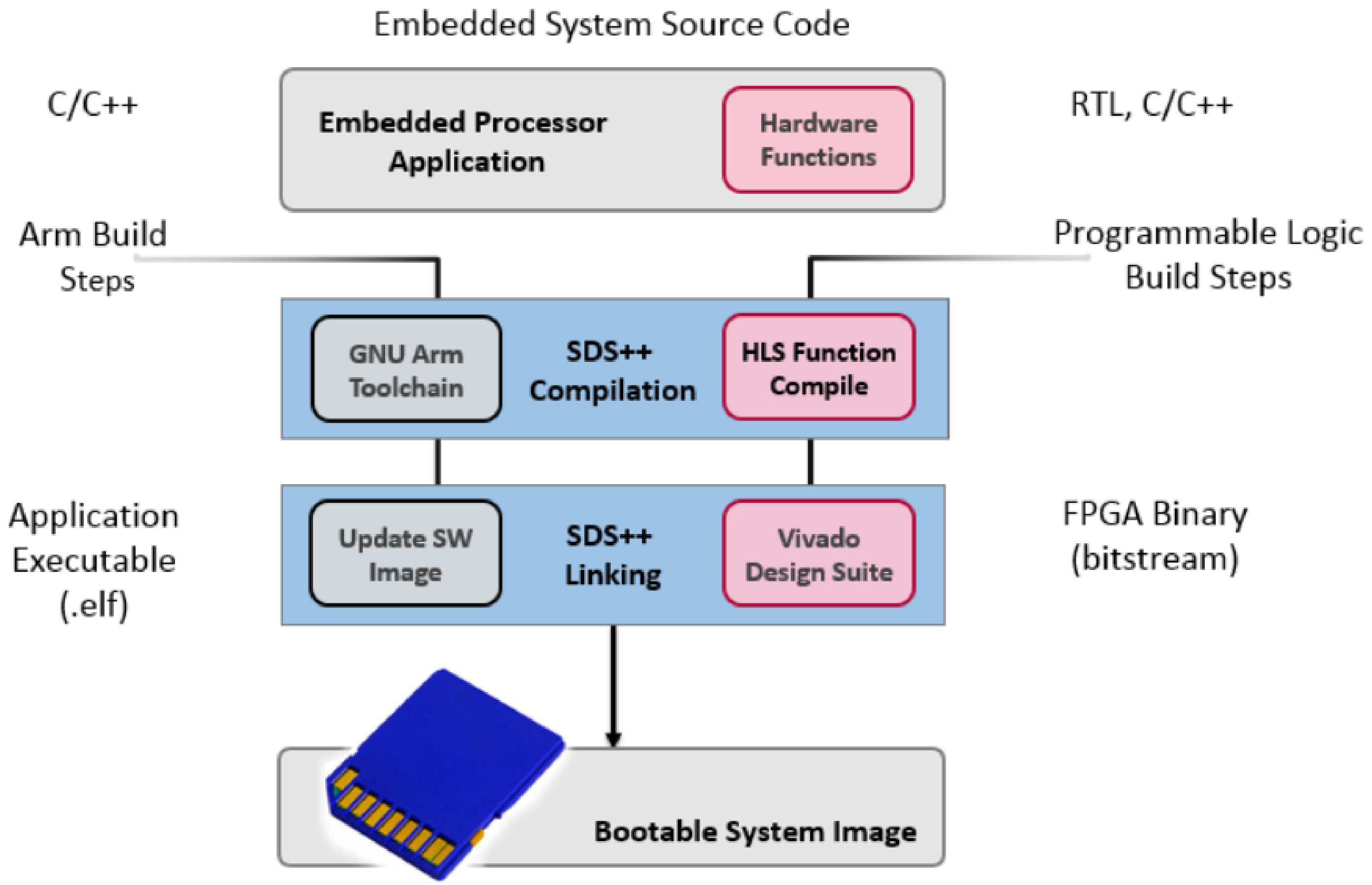
Xilinx SDK

Xilinx, ARM
라이브러리

C/C++ SW IP

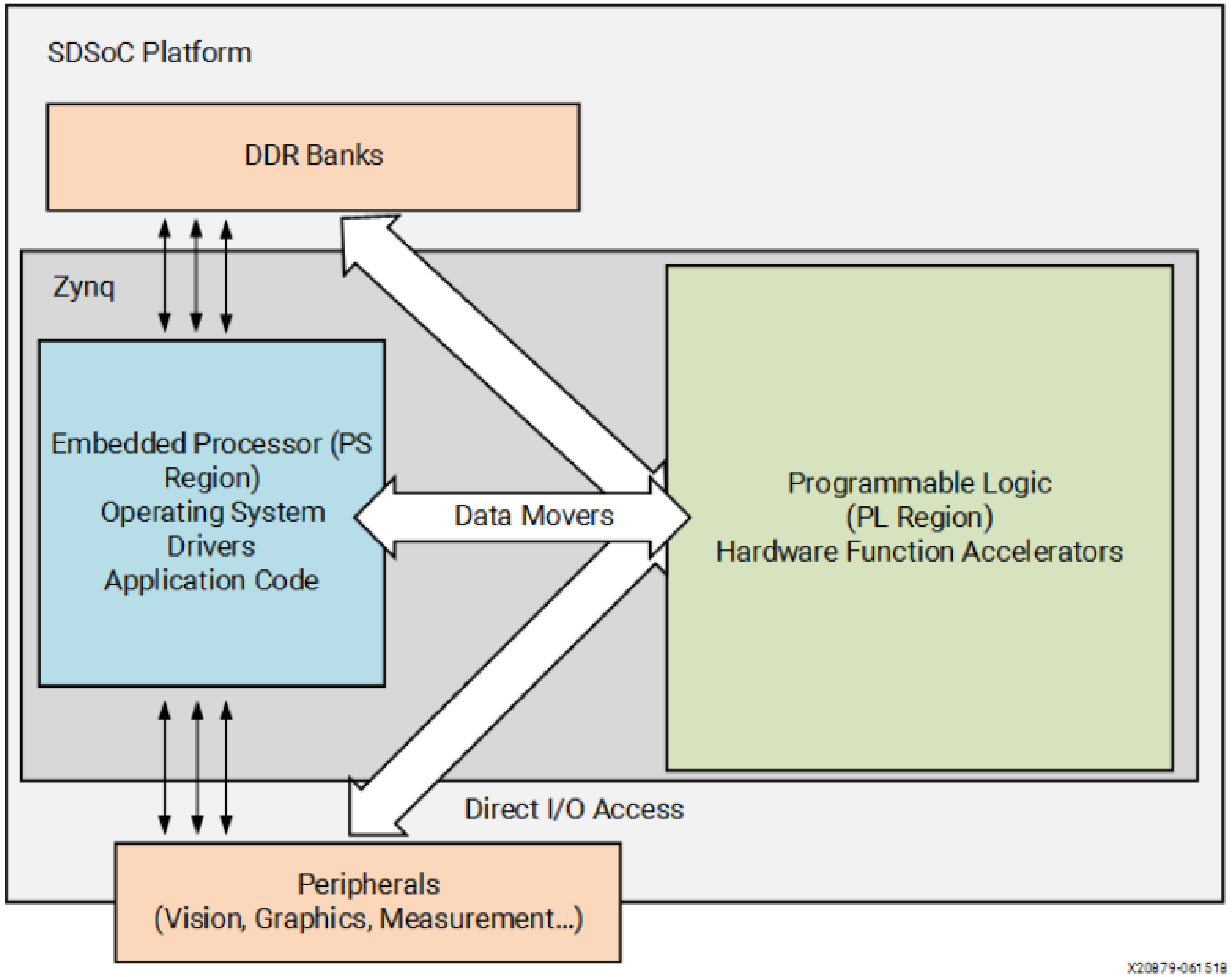
Bare Metal App
freeRTOS, Linux

SDSoC



HUINS
Human Intelligent System

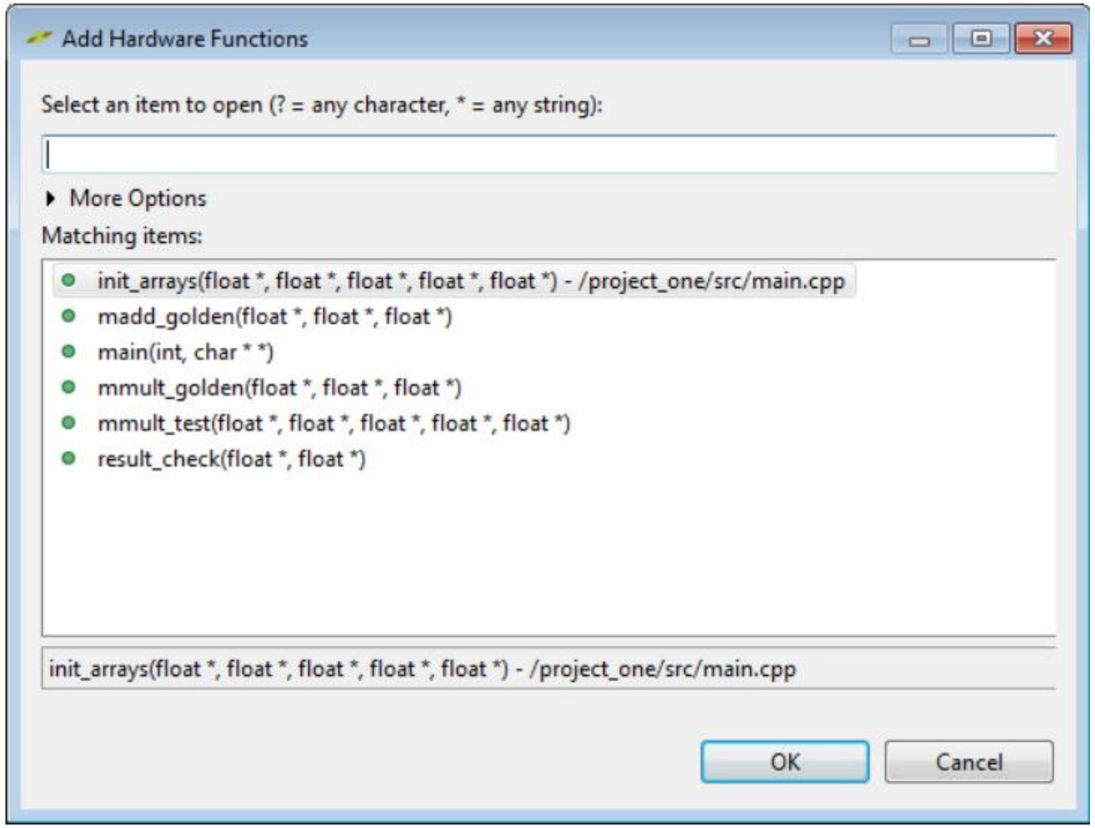
SDSoC System

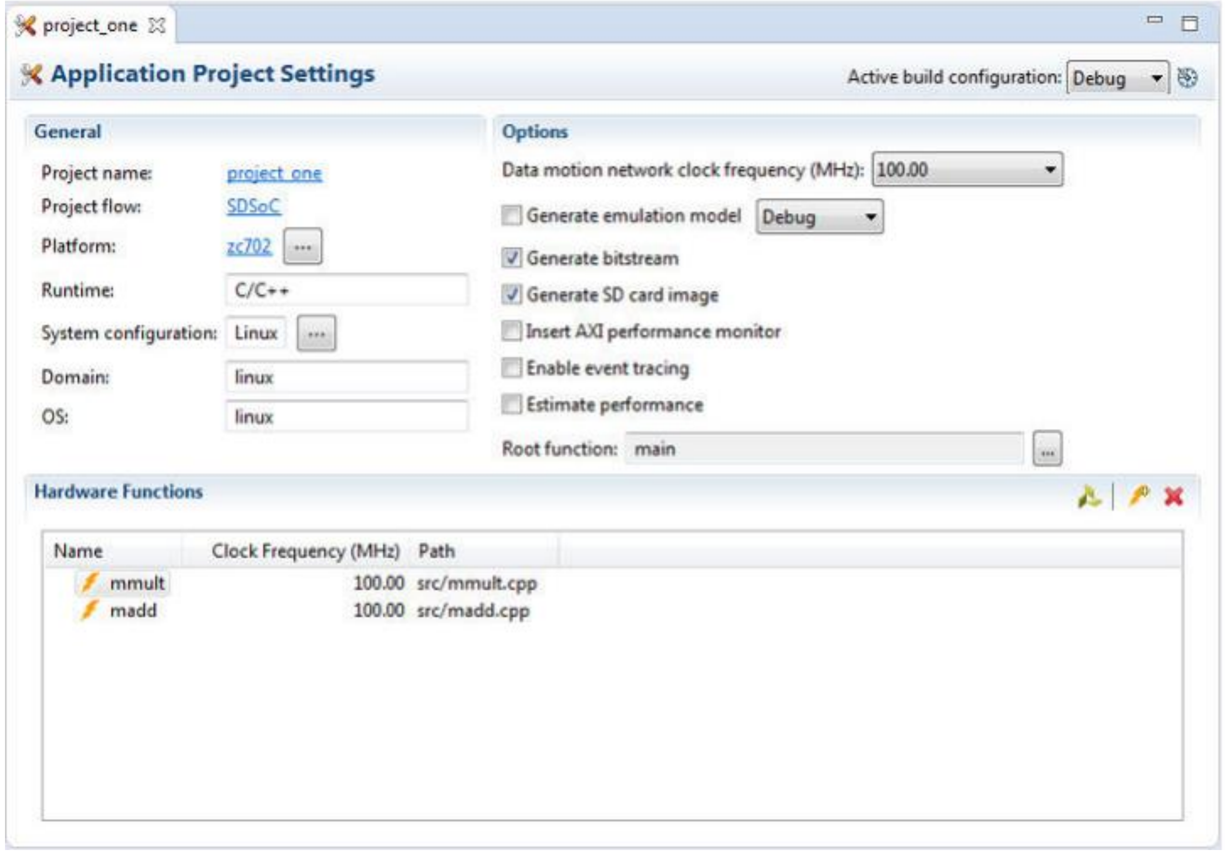


SDSoC System

- 기본 PS Configuration
  - DRAM
  - Peripherals
  - MIO
- PS <-> PL Port

SW/HW Partitioning in SDSoC

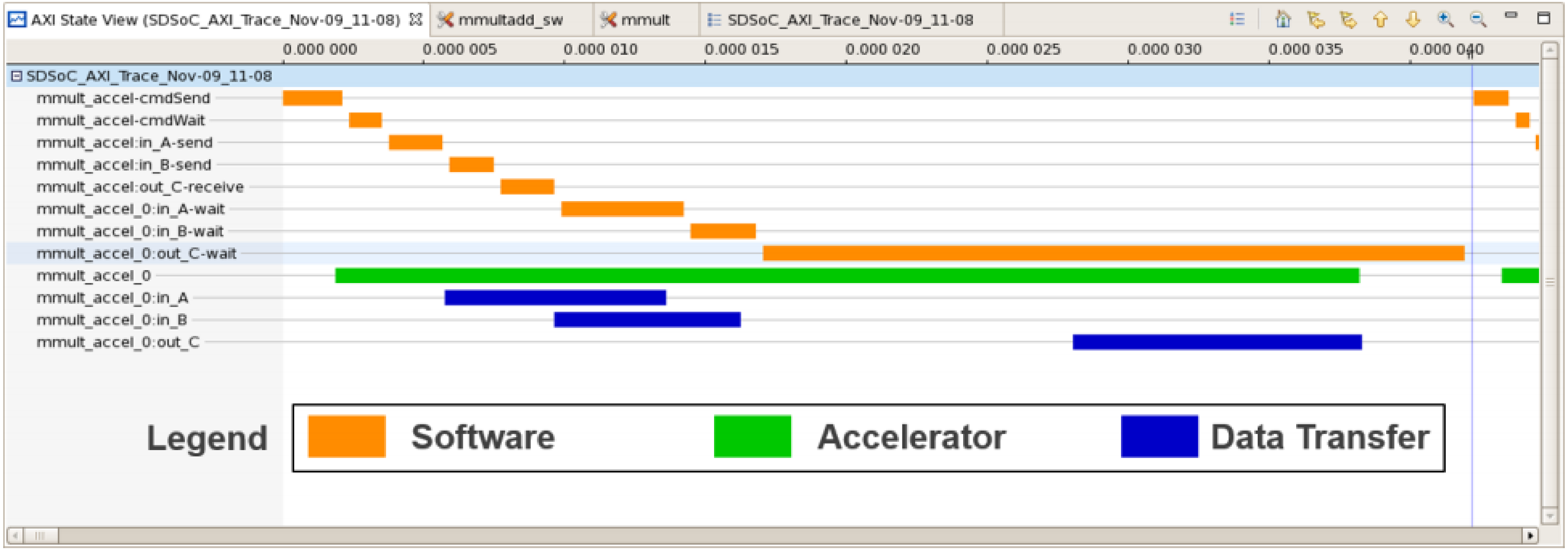




- 소스코드의 전체 함수 중 선택한 함수를 HW 변환

HUINS
Human Intelligent System

System Profiling in SDSoC



HUINS
Human Intelligent System

References

ZYNQ

- THE ZYNQ BOOK(http://www.zynqbook.com)
- ZYNQ Technical Reference Guide(UG-585)

HLS

- Vivado-high-level-synthesis(ug871)
- Vivado-high-level-synthesis(UG902)
- Vivado-Intro-FPGA-Design-HLS(UG998)

SDSoC

- SDSoC-optimization-guide(UG1235)