Automatic Interface Synthesis based on the Classification of Interface Protocols of IPs

ChangRyul Yun[1], DongSu Kang[2], YoungHwan Bae[3], HanJin Cho[3], KyoungSon Jhang[2]

[1]Agency for Defense Development, KOREA
[2]Dept. of Computer Engineering, ChungNam National University, KOREA
[3]Multimedia SoC Design Team, ETRI, KOREA

Outline

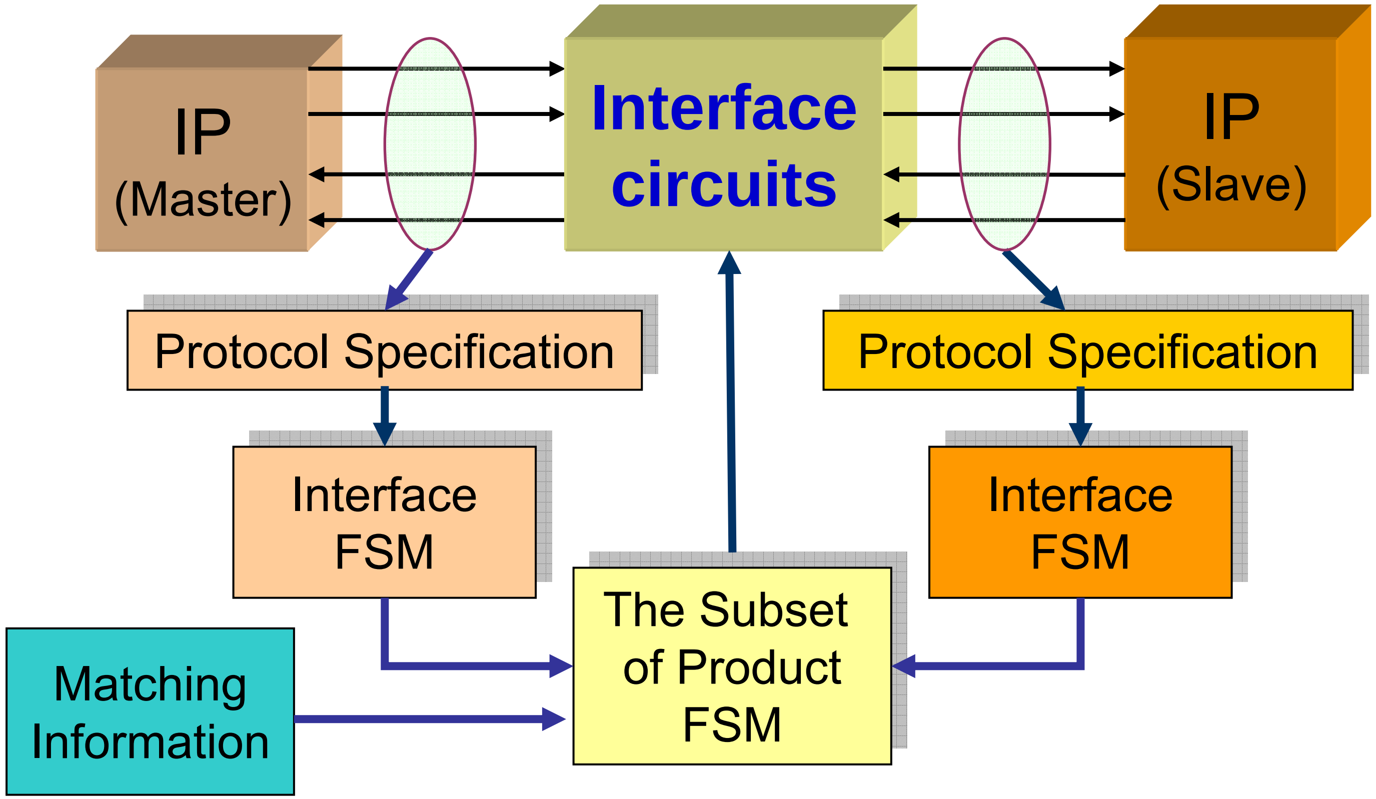
- Introduction & Motivation
  - Interface circuit structures
- Proposed Interface Synthesis Method
  - Interface Synthesis Flow
  - SIMPLE (Simplified Interface Protocol Description Language)
  - Classification of Interface Protocols of IPs
  - The Structures of interface Circuits
  - Interface Synthesis Algorithm
- Experiments
- Conclusion

Introduction

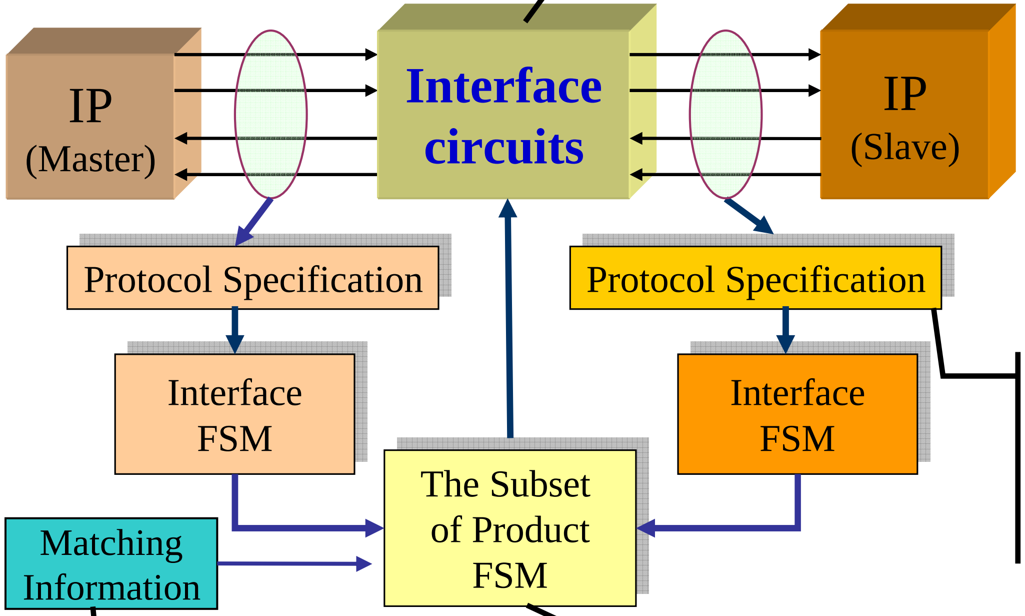
- IP-based Design Methodology in SoC
  - A methodology to respond to the demands of high performance, complex functionality, short time-to-market
  - One of key issues: The difficulty of IPs integration
    - IPs use various interface protocols
  - Interface circuits between IPs are necessary.
- Interface circuits
  - Time-consuming and error-prone task
  - Automatic interface circuit generation method is necessary.

Introduction

- Automatic Interface Synthesis Flow based on interface FSMs



Motivation



IP (Master)

Interface circuits

IP (Slave)

Protocol Specification

Protocol Specification

Interface FSM

Interface FSM

The Subset of Product FSM

Matching Information

Flexible and highly appropriate structure depending on IP characteristics and system level requirements.

Easy and efficient interface protocol description method of IPs.

A synthesis algorithm should consider various characteristics on generation of interface circuits.

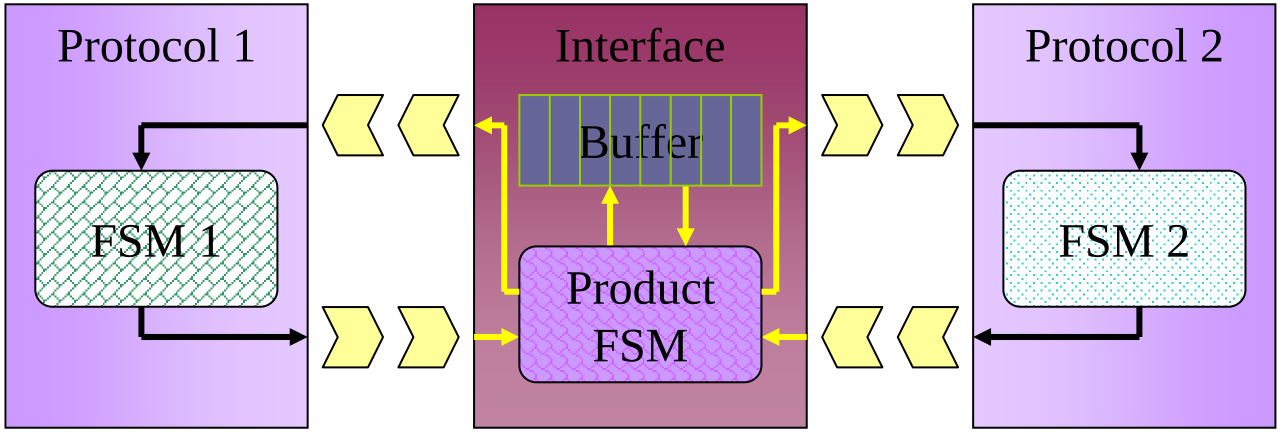
User constraints

Motivation

- The various characteristics of interface protocols of IPs
  - Address and data transfer characteristics
    - The number of addresses on burst
    - Shared signal or different signals for an address and data
  - No Address
  - Different clock frequencies
  - Different data widths
- These differences should be considered on an interface synthesis algorithm.

Product FSM Based Structure

- Product FSM based interface circuits
  - Consist of a product FSM and buffers
  - Allow the transfer of data without any clock cycle delay.

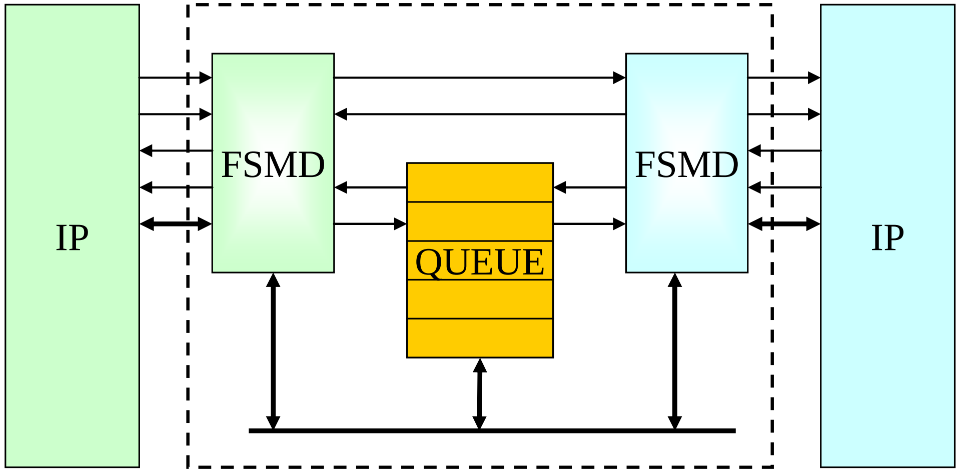


The interface circuit structure based on a product FSM[3]

[3] Vijay D'*silva*, S. Ramesh and Arcot Sowmya, "Bridge Over Troubled Wrappers: Automated Interface Synthesis," Proceedings of the 17th Int ernational Conference on VLSI Design 2004 Page(s):189 – 194

FSMD Based Structure

- FSMD(FSM+Data Path) based interface circuits
  - Consist of two FSMDs and buffers(Queue)
  - Every data transfer has at least two clock cycle.

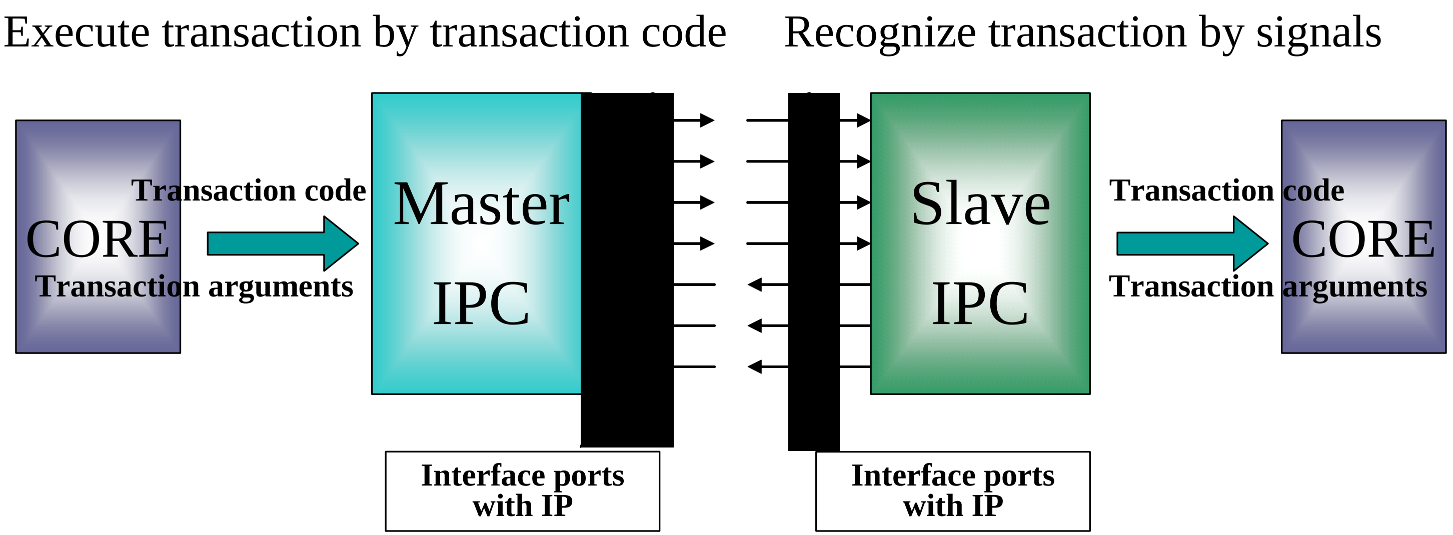


The interface circuit structure based on a product FSM[6]

[6] Dongwan Shin and Daniel Gajski, “Interface Synthesis from Protocol Specification,” Technical Report (CECS-02-13), April 12 2002, Center for Embedded Computer Systems University of California, Irvine

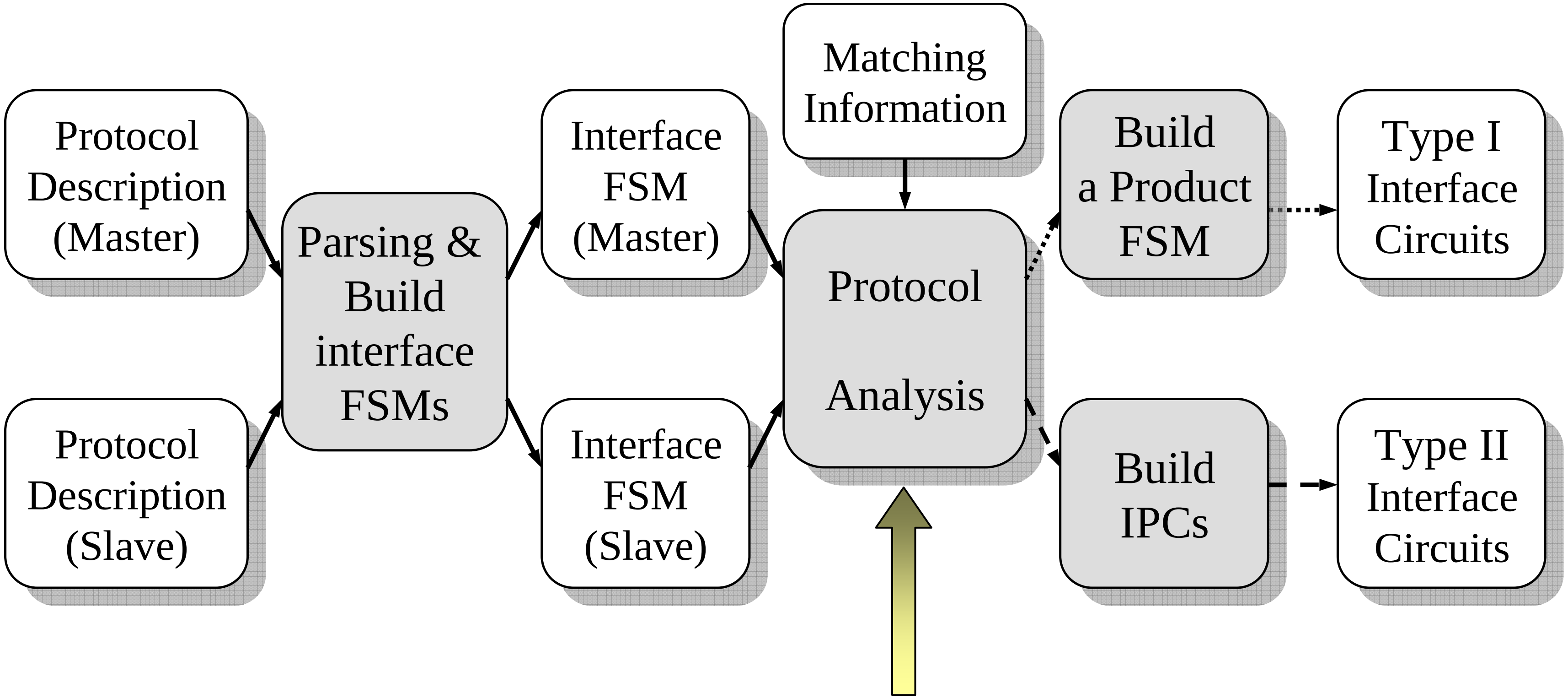
IPC Based Structure

- IPC (Interface Protocol Component) [10]
  - Protocol Convert
    - From transaction level to cycle level vice versa.
  - No internal buffers



[10] ChangRyul Yun, KyoungSon Jhang, "An Interface Protocol Component Modeling Language," Proceedings of the 15th ASIC/SOC Conference, Sept. 2002, Page(s): 456-460

Proposed Interface Synthesis Flow


Protocol Description (Master)
Protocol Description (Slave)
Parsing & Build interface FSMs
Interface FSM (Master)
Interface FSM (Slave)
Matching Information
Protocol Analysis
Build a Product FSM
Build IPCs
Type I Interface Circuits
Type II Interface Circuits

- Classify Interface Protocols
- Select a synthesis algorithm for interface structure depending on analysis results or user constraints

Interface FSM

- Protocol Description Methods
  - Waveform or timing diagram
  - Formal language
- SIMPLE (Proposed Description Method)
  - Simplified Interface Protocol Description Language[10]
  - Interface Protocol: Transfers + Parameters
    - Transfer: the behavior of ports on a cycle
    - Parameter
      - Address, data, transaction information and etc.
      - Used to match the different feature of IPs

[7] ChangRyul Yun, YoungHwan Bae, HanJin Cho, KyoungSon Jhang, “Automatic Synthesis of Interface Circuits from Simplified IP Interface Protocols,” Proceedings of the Eleventh Asia-Pacific Computer Systems Architecture Conference (ACSAC 2006), September 6-8th, Page(s):581-587

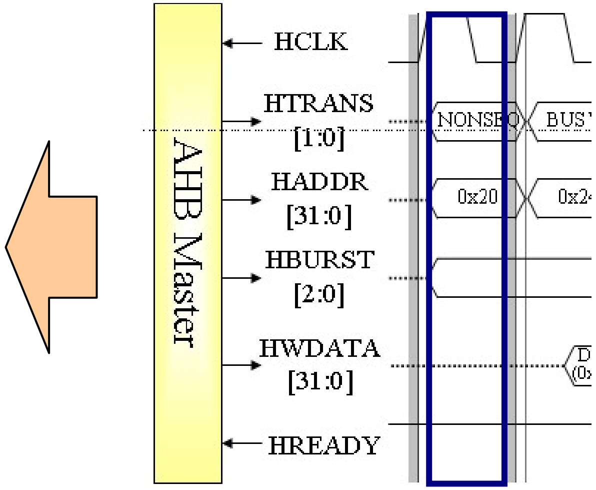
Interface FSM

- SIMPLE

```
Define $new_transaction, $write;              /* ⑤, ⑥ */
Interface AHB_Master {
   in  bit        HREADY;   /* Ports List */
              :
   initial      A;            /* ① Initial State */
   AMBA_MASTER: behavior   {
       int      count;      // User Defined Variable
       A:      if ( $new_transaction == 1 )         /* ⑤ */  {
                 HTRANS = "10";  // NONSEQ
                 HADDR = $address;           /* ④ */
                 HBURST = $transaction;      /* ② */
                 count = $transaction_length;     /* ③ */
                 if ($write == '1')      /* ⑥ */   {
                               HWRITE = '1';
                               if ( HREADY == '1')
                                           goto B;
                          :
               }
               else
                            goto A;
               }
```

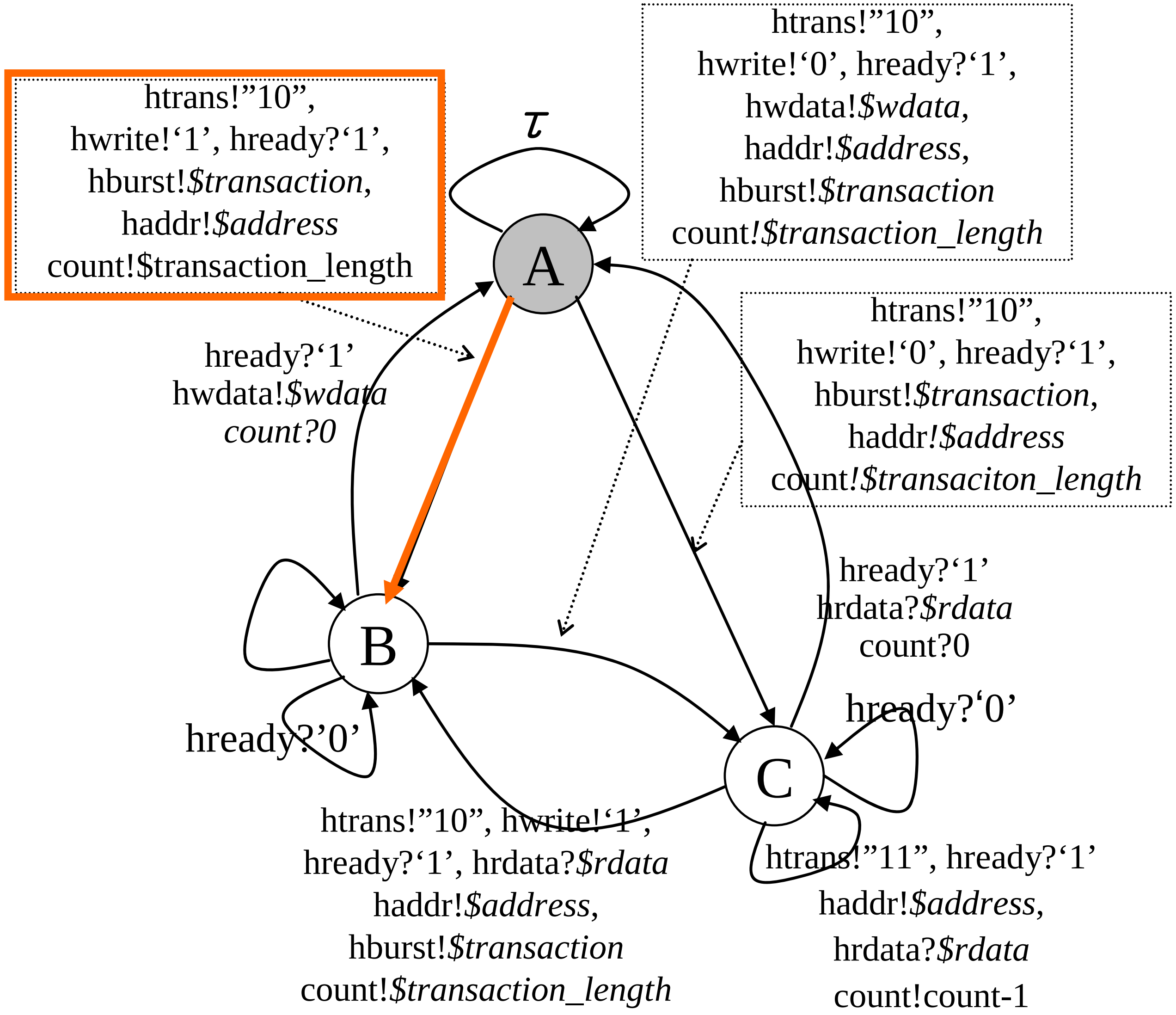
A part of the description

- Reserved Parameter
  - ②: transaction type
  - ③: transaction length
  - ④: address
- User Defined Parameter
  - ⑤, ⑥



AHB Master interface Protocol

Interface FSM



htrans!”10”,
hwrite!‘1’, hready?‘1’,
hburst!*$transaction*,
haddr!*$address*
count!$transaction_length

τ

htrans!”10”,
hwrite!‘0’, hready?‘1’,
hwdata!*$wdata*,
haddr!*$address*,
hburst!*$transaction*
count*!$transaction_length*

A

hready?‘1’
hwdata!*$wdata*
count?0

htrans!”10”,
hwrite!‘0’, hready?‘1’,
hburst!*$transaction*,
haddr*!$address*
count*!$transaciton_length*

hready?‘1’
hrdata?*$rdata*
count?0

B

hready?’0’

hready?‘0’

C

htrans!”10”, hwrite!‘1’,
hready?‘1’, hrdata?*$rdata*
haddr!*$address*,
hburst!*$transaction*
count!*$transaction_length*

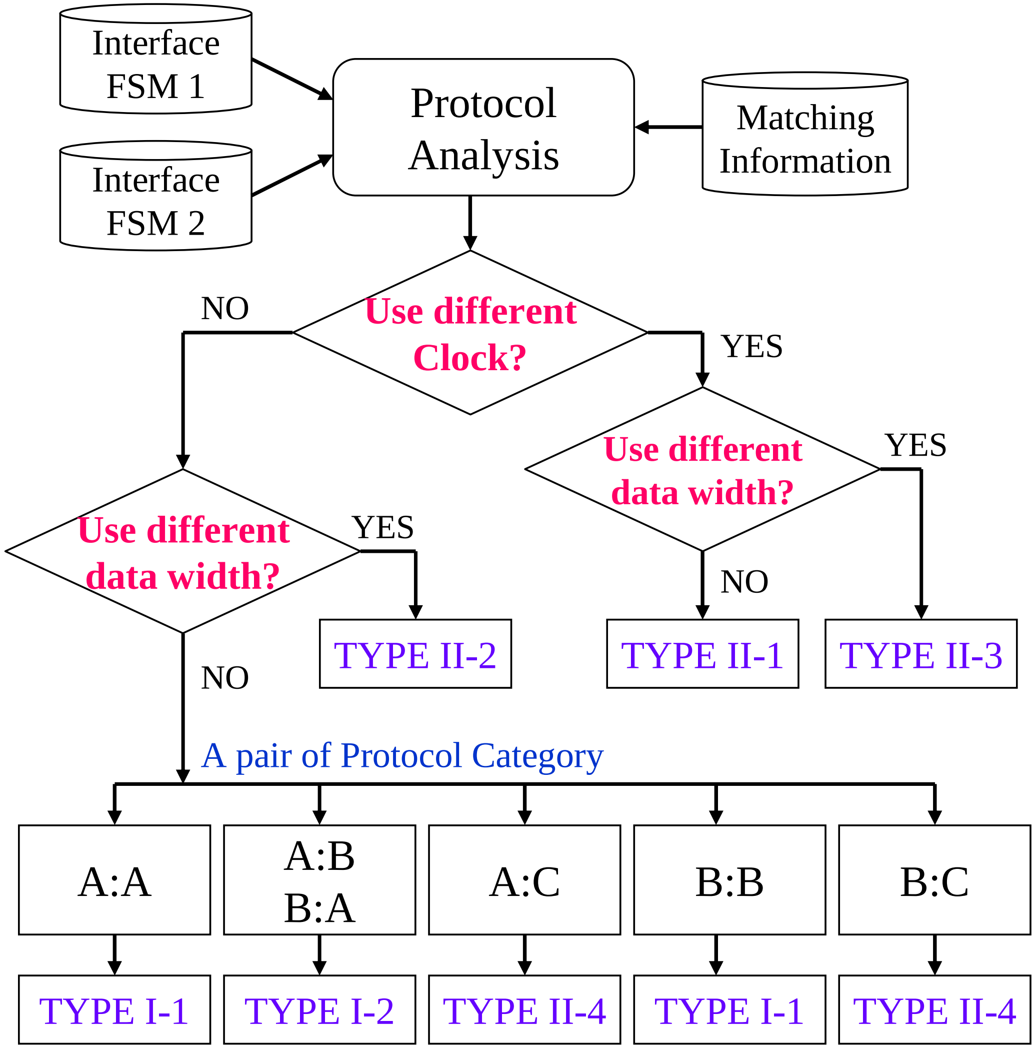
htrans!”11”, hready?‘1’
haddr!*$address*,
hrdata?*$rdata*
count!count-1

The categorization of Interface protocols

- Address & data Transfer Characteristics

Category	Feature	Examples
A	Separated ports for addresses, write data, and read data	AHB, OCN, VCI, OCP,PLB, etc.
B	Shared one port for addresses, write data, and read data	PCI, PCI-X, etc.
C	No address	DES, AES, Color converter, etc

Selection of Interface Circuit Structure



- Type I

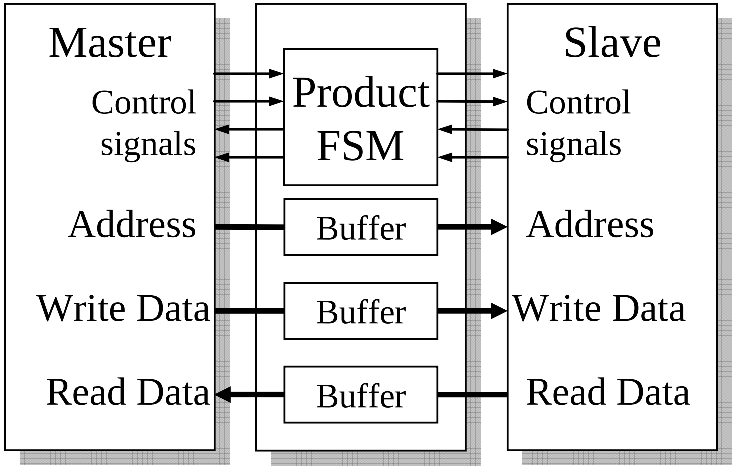
  Product FSM-Based Structure

- Type II

  IPC-Based Structure

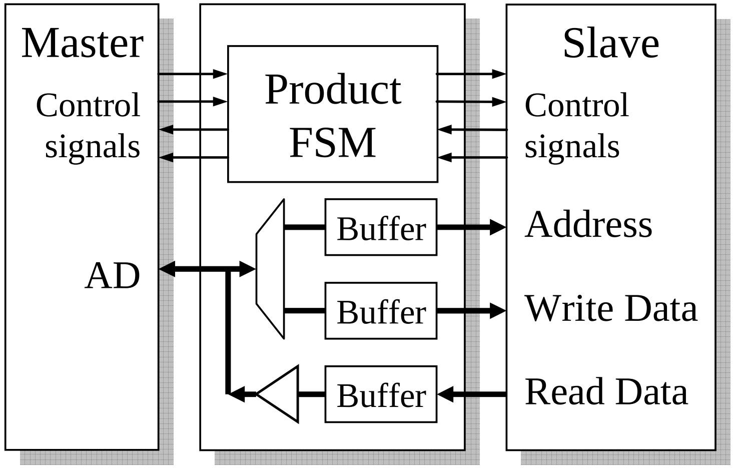
Type I

- Type I – 1



  - For a pair of Category A IP
  - A buffer can be employed for each port pair.
  - Data can be stored or bypassed to reduce transmission delay.

- Type I – 2



  - For a pair of Category A : Category B
  - Generate or ignore addresses

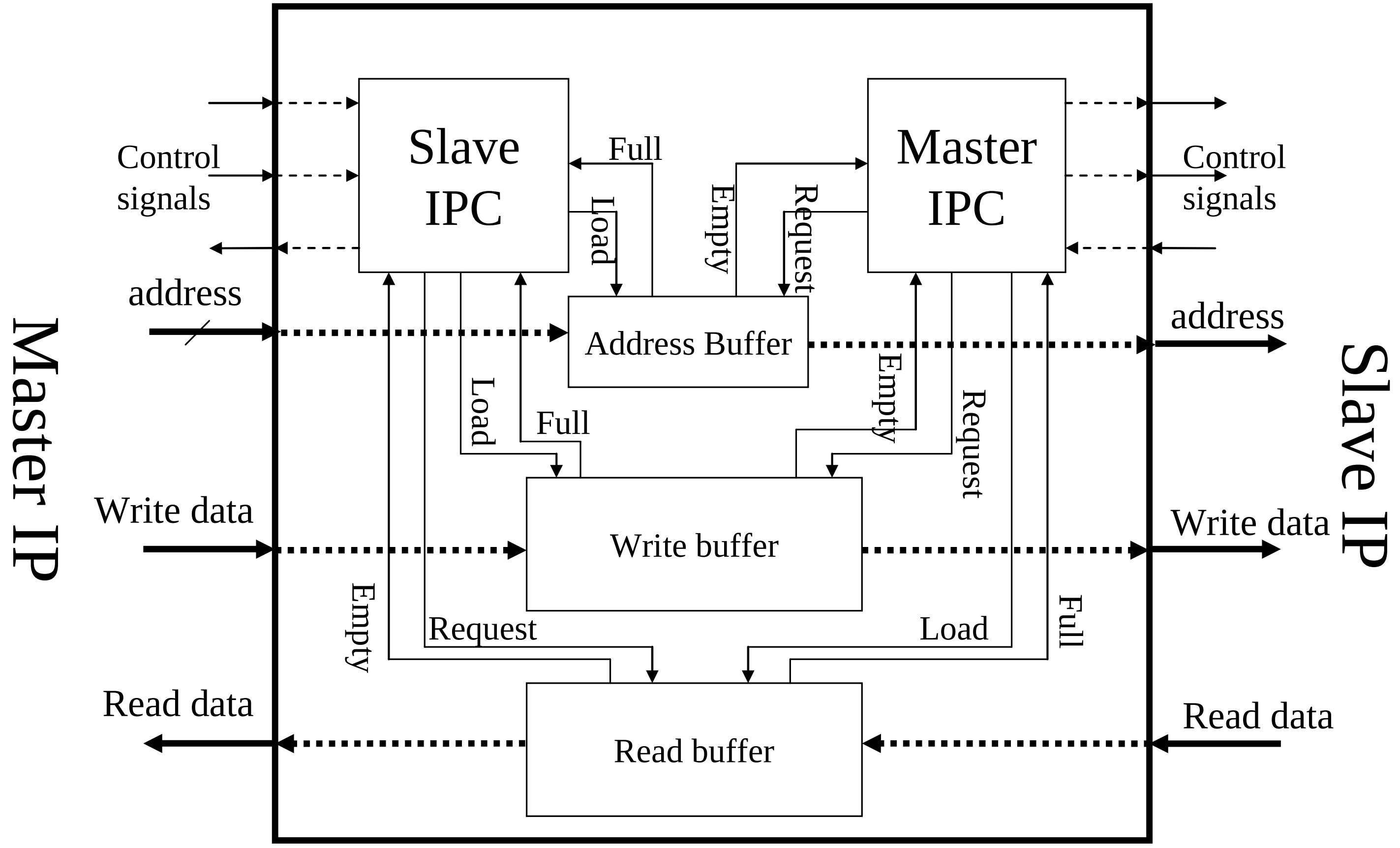
Type I - 1

- Matching Information for a pair of PCI(M):AHB(S)

```
[Clock Ratio] 1 : 1 //// Master IP, Slave IP
//// Buffer Size, Signal (master), MSB, LSB,
//// Signal (slave), MSB, LSB, $address or $wdata or $rdata
[Port Pair] 1 , FA , 31 , 0 , haddr , 31 , 0 , $address;
[Port Pair] 1 , FD , 31 , 0 , hwdata , 31 , 0 , $wdata;
// Burst Transaction Length, Signal(master), MSB, LSB,
// Values, Signal(slave), MSB, LSB, Values
[TM] 1 , FS , 3 , 2 , "00" , HBURST , 2 , 0 , "000";
[TM] 4 , FS , 3 , 2 , "01" , HBURST , 2 , 0 , "011";
[TM] 16 , FS , 3 , 2 , "11" , HBURST , 2 , 0 , "111";
```

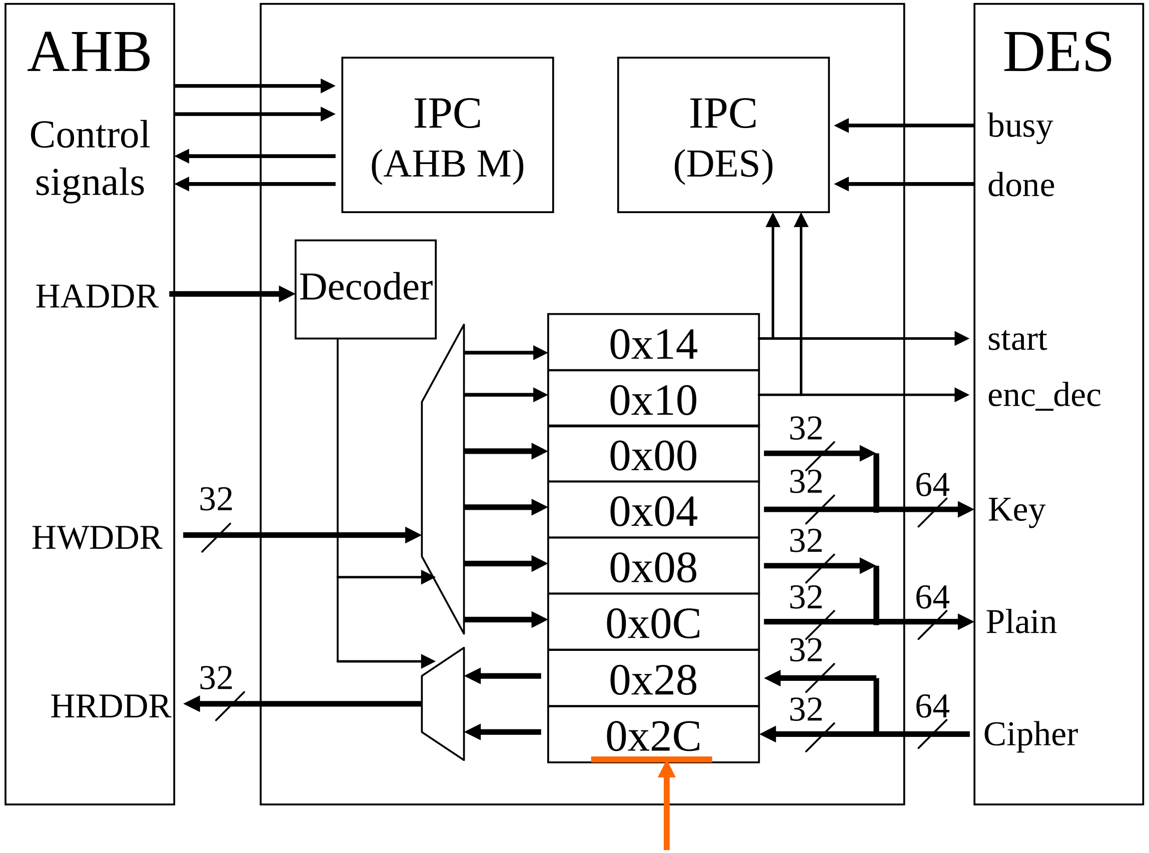
Type - II

- IPC Based Structure



Type - II

- Type II - 4



Buffer Address

Type II - 3

- Matching Information for a pair of AHB(M):DES(S)

```
// Buffer Size, Master Port Name, MSB, LSB, Slave Port Name, MSB, LSB,
// Address Port, MSB, LSB, address value, $Data or $Control ;
[Port Mapping] 1, HWDATA, 31, 0, KEY, 63, 32, HADDR, 7, 0, 0x00, $Data;
[Port Mapping] 1, HWDATA, 31, 0, KEY, 31, 0, HADDR, 7, 0, 0x04, $Data;
[Port Mapping] 1, HWDATA, 31, 0, Plain, 63, 32, HADDR, 7, 0, 0x08, $Data;
[Port Mapping] 1, HWDATA, 31, 0, Plain, 31, 0, HADDR, 7, 0, 0x0C, $Data;
[Port Mapping] 1, HWDATA, 31, 0, Cipher, 63, 32, HADDR, 7, 0, 0x28, $Data;
[Port Mapping] 1, HWDATA, 31, 0, Cipher, 31, 0, HADDR, 7, 0, 0x2C, $Data;
[Port Mapping] 1, HWDATA, 0, 0, START, 0, 0, HADDR, 7, 0, 0x14, $Control;
[Port Mapping] 1, HWDATA, 0, 0, enc_dec, 0, 0, HADDR, 7, 0, 0x14, $Control;
```

  - Graphical User Interface (GUI) or a script file can be used to construct the matching description.

Experiments

- The comparison between manual and automatic design

Type	Master : Slave	Area (Slices)		f_{max}		A / M	
		M	A	M	A	Area	f_{max}
I–1	AHB:OCN	57	103	234	181	1.8	0.8
	OCN:AHB	112	374	197	105	3.3	0.5
	BVCI:AHB	44	61	580	356	1.4	0.6
	AHB:APB	66	121	268	249	1.8	0.9
I–2	AHB:PCI	50	72	315	238	1.4	0.9
	PCI:AHB	82	212	258	144	2.6	0.6
II–1	AHB:PVCI (50Mhz, 32bit: 12.5Mhz, 32bit)	310	356	214	191	1.1	0.9
II–2	AHB:PVCI (50Mhz, 32bit: 50Mhz, 8bit)	244	316	225	208	1.3	0.9
II–3	AHB:PVCI (12.5Mhz, 32bit: 50Mhz, 8bit)	237	316	204	204	1.3	1.0
II-4	AHB:DES	134	145	225	197	1.1	0.9

Experiments

- Compare product FSM-based with IPC-based Structure

Master : Slave	Type I (Product FSM Based)						Type II (IPC Based)						Type I / Type II	
	# S	# T	Area	f_{max}	W	R	# S	# T	Area	f_{max}	W	R	Area	f_{max}
AHB:OCN	9	53	203	181	5	12	6	34	174	198	5	14	1.2	0.9
OCN:AHB	12	89	374	105	4	12	6	35	256	121	4	14	1.5	0.9
AHB:BVCI	18	48	324	219	5	8	6	20	276	228	5	10	1.2	1.0
BVCI:AHB	10	42	249	201	4	8	7	21	205	217	4	10	1.2	0.9
AHB:PVCI	10	39	252	204	5	8	6	19	267	212	5	10	0.9	1.0

Experiments

- The comparison with previous work[3]

Master:Slave	Clocks	Data Width	Previous Method		Our Method	
			# S	# T	# S	# T
AHB:PLB	1:1	1:1	7	12	6	12
OCP:AHB	1:1	1:1	8	18	9	20
AHB:PLB	1:2	1:1	12	15	6	8
OCP:PLB	1:1	1:3	17	32	6	8

#S : The number of States, #T : The number of transitions

[3] Vijay D'*silva*, S. Ramesh and Arcot Sowmya, "Bridge Over Troubled Wrappers: Automated Interface Synthesis," Proceedings of the 17th Int ernational Conference on VLSI Design 2004 Page(s):189 – 194

Conclusion

- Automatic Interface Synthesis based on protocol categorization
  - Chose an appropriate structure of interface circuits based on
    - Interface protocol category
    - Clock frequency
    - Data width
    - System level requirements (by user constraints)
  - Protocol Classification
    - Our Synthesis algorithm selects an interface circuit structure depending on the classification of interface protocols

Conclusion

- Product FSM-based structure
  - Less data transmission latency
  - Somewhat larger area and slower maximum operating frequency
- IPC-based structure
  - Less states and transitions than product FSM-based in case that IPs use different frequencies and/or data widths.
- The Performance
  - Comparable with performance of the manual designs

Thank You!