排序机FSMD方案collate

ca\_in

FSM/ASM

target\_addr

source\_addr

decode

data\_length

cmp0

reg0

doing

ready

ram\_rdata\_valid

reg1

cmp1

ram\_rdata

ram\_addr

ram\_wdata

reg2

cmp2

ram\_we

ram\_rd

ram\_ready

rst

clk

regn-1

cmpn-1

ca\_dout

fifo\_rdata

regn

fifo\_empty

fifo\_wdata

fifo\_rdreq

fifo\_wrreq

ca\_ out

cmp\_sel

mux\_sel

data\_out

mux\_decode

fifo\_clr

注：

1. 不包含寄存器自保持逻辑

2. 寄存器输出逻辑采用FSM发出的data\_out，从reg\_data\_out输出(写RAM时反序)

3. 不包含寄存器清零

4. 数据长度data\_length必须与n对齐（n=16，32，64）

5. 参数包括oredr\_reg\_num(即n)

6. 不包含寄存器使能reg\_en

排序机 (Collator)的存储器接口

source\_addr

FIFO

COLLATOR

RAM

DRAM

target\_addr

排序机 (Collator)控制器的STG

RAM突发读，BL=n（Avalon）

!doing

fifo\_clr=1

reg\_clr=1

\*

ready=1

count<length

!ram\_ready

ram\_ready

ram\_addr=source+count

ram\_rd=1

S3:count>=n

fifo\_wrreq=1

fifo\_wdata=ca\_out

doing

target=target\_addr

source=source\_addr

length=data\_length

fifo\_clr=0

reg\_clr=0

count=0

ready=0

RST

ram\_wdata=0

ram\_addr=0

ram\_we=0

ram\_rd=0

fifo\_wrreq=0

fifo\_rdreq=0

fifo\_wdata=0

fifo\_clr=1

reg\_clr=1

length=0

count=0

ready=0

target=0

source=0

reg\_data\_in=0

reg\_en=0

data\_out=0

S3:!ram\_rdata\_valid

reg\_en=0

count=0&&length=0

count=0

ram\_wr=0

reg\_clr=1

ready=1

data\_out=0

S3:ram\_rdata\_valid

reg\_en=1

ca\_in=ram\_data

count=count+1

count>= length

reg\_en=0

count=n

fifo\_wrreq=0

length=length-n

data\_out=1

count>0

S4:

reg\_en=0

S4: ram\_ready

ram\_addr=target+count

ram\_wr=1

ram\_wdata=ca\_dout

count=count-1

count=0&&length>0

count=0

ram\_wr=0

reg\_clr=1

data\_out=0

count>=length-1

ca\_in=fifo\_rdata

fifo\_rdreq=0

count=0

S4: !ram\_ready

ram\_wr=0

\*

fifo\_rdreq=1

count<length-1

ca\_in=fifo\_rdata

reg\_en=1

二叉树方案

fork1

f1

l1

r1

l0

r0

l1

r1

l2

r2

l3

r3

fork3

f3

l3

r3

l0

r0

l1

r1

l2

r2

l3

r3

fork0

f0

l0

r0

l0

r0

l1

r1

l2

r2

l3

r3

fork2

f2

l2

r2

l0

r0

l1

r1

l2

r2

l3

r3

result2

l0

r0

l1

r1

l2

r2

l3

r3

result1

l0

r0

l1

r1

l2

r2

l3

r3

result0

l0

r0

l1

r1

l2

r2

l3

r3

result3

l0

r0

l1

r1

l2

r2

l3

r3

1. fork和result可以是同一种模型，称为fork
2. fork为组合逻辑
3. fork将输入的逻辑值装配本地的树杈值后从左支和右支输出（逻辑值相加，逻辑长度+1）
4. 传递值包含逻辑值logic\_value（1和0）以及逻辑值长度logic\_length，根fork的输入逻辑长度为0
5. 结果result为每个符号的分配huffman码字
6. 排序过程逐步得到多路器的选择数值，排序结束得到二叉树和码字
7. 硬件要求设置最大树杈树max\_forks=N+1，N为符号数，N应该在硬件中指定。
8. 硬件设置最大码字长度max\_codeword\_length，设置传递值长度（Passing\_Value\_length）和定义，例如：

max\_codeword\_length=4; Passing\_Value\_length=6

lsb

msb

logic\_value=4bit

logic\_length=2bit

collate+binarytree

FSM/ASM

ca\_in

target\_addr

source\_addr

m0

decode

data\_length

cmp0

reg0

doing

ready

m1

ram\_rdata\_valid

reg1

cmp1

ram\_rdata

ram\_addr

m2

ram\_wdata

reg2

cmp2

ram\_we

ram\_rd

ram\_ready

rst

m

n-1

clk

regn-1

cmpn-1

ca\_dout

fifo\_rdata

regn

fifo\_empty

fifo\_wdata

fifo\_rdreq

fifo\_wrreq

mn

ca\_ out

cmp\_sel

mux\_sel

data\_out

mux\_decode

fifo\_clr

fork\_mux

bt\_out

s0

ind0

s\_mux

s1

ind1

s

n-2

indn-2

bt\_left

s

n-1

indn-1

bt\_right

RAM未作流水线/突发读

!ram\_ready

!doing

fifo\_clr=1

reg\_clr=1

\*

ram\_addr=source+count

ram\_rd=1

\*

ready=1

!ram\_rdata\_valid

doing

target=target\_addr

source=source\_addr

length=data\_length

fifo\_clr=0

reg\_clr=0

count=0

ready=0

RST

ram\_wdata=0

ram\_addr=0

ram\_we=0

ram\_rd=0

fifo\_wrreq=0

fifo\_rdreq=0

fifo\_wdata=0

fifo\_clr=1

reg\_clr=1

length=0

count=0

ready=0

target=0

source=0

reg\_data\_in=0

reg\_en=0

\*

reg\_data\_in=ram\_rdata

ram\_rd=0

reg\_en=1

count>=n-1&&length=0

count=0

ram\_wr=0

reg\_clr=1

ready=1

count<(n-1)

reg\_en=0

count<length-1

count=count+1

fifo\_wrreq=0

!ram\_ready

ram\_ready&&count<n-1

ram\_addr=target+count

ram\_wr=1

ram\_wdata=reg[count]

count=count+1

count>=(n-1)

fifo\_wdata=reg\_data\_out

fifo\_wrreq=1

reg\_en=0

count>=length-1

count=0

length=length-n

fifo\_wrreq=0

S6

reg\_en=0

count>=n-1&&length>0

count=0

ram\_wr=0

reg\_clr=1

count>=length-1

reg\_data\_in=fifo\_rdata

fifo\_rdreq=0

count=0

\*

fifo\_rdreq=1

count<length-1

reg\_data\_in=fifo\_rdata

reg\_en=1